NUS National University of Singapore | School of Computing

CS5340
Uncertainty Modeling in AI

Lecture 12:
Gaussian Processes

"Regression with Bayesian Non-parametrics"

Asst. Prof. Harold Soh

AY 2023/24

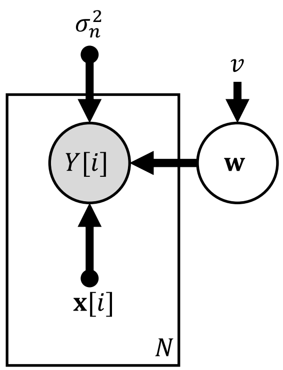
Semester 2

Course Schedule (Tentative)

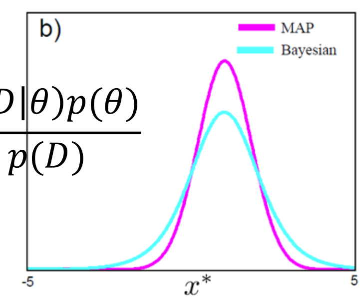
Week	Date	Lecture Topic	Tutorial
1	16 Jan	Introduction to Uncertainty Modeling + Probability Basics	~~Introduction~~
2	23 Jan	Simple Probabilistic Models	Introduction and Probability Basics
3	30 Jan	Bayesian networks (Directed graphical models)	More Basic Probability
4	6 Feb	Markov random Fields (Undirected graphical models)	DGM modelling and d-separation
5	13 Feb	Variable elimination and belief propagation	MRF ~~+ Sum/Max Product~~
6	20 Feb	Factor graphs	**Quiz 1**
-	-	**RECESS WEEK**	
7	5 Mar	Mixture Models and Expectation Maximization (EM)	Linear Gaussian Models
8	12 Mar	Hidden Markov Models (HMM)	Probabilistic PCA
9	19 Mar	Monte-Carlo Inference (Sampling)	Linear Gaussian Dynamical Systems
10	26 Mar	Variational Inference	MCMC + Langevin Dynamics
11	2 Apr	Inference and Decision-Making (optional)	Diffusion Models + Sequential VAEs
12	9 Apr	Gaussian Processes (optional)	**Quiz 2**
13	16 Apr	Closing Lecture	Project Presentations

CS5340 in a nutshell

CS5340 is about how to "**represent**" and "**reason**" with uncertainty in a computer.



$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$$



Representation: The *language* is probability and probabilistic graphical models (PGM).

The language is used to model problems.

Reasoning: We use learning and inference algorithms to answer questions.

e.g., Belief-propagation/sum-product, MCMC, and variational Bayes

Summary: Sum and Product Rules

- Sum rule:

$$p(x) = \int p(x,y)\,dy$$

$$p(x) = \sum_{y} p(x,y)$$

- Product/Chain rule:

$$p(x,y) = p(x|y)p(y)$$

Multivariate Normal Distribution

- Multivariate normal distribution describes a *D*-dimensional continuous variable $\boldsymbol{X}$, i.e. $\boldsymbol{x} \in \mathbb{R}^D$.
- *D*-dimensional mean $\boldsymbol{\mu} \in \mathbb{R}^D$, and $D \times D$ symmetrical positive definite covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}_+^{D \times D}$.

$$p(\boldsymbol{X} = \boldsymbol{a} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\{ -0.5(\boldsymbol{a} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{a} - \boldsymbol{\mu}) \}, \quad \boldsymbol{a} \in \mathbb{R}^D$$

Or

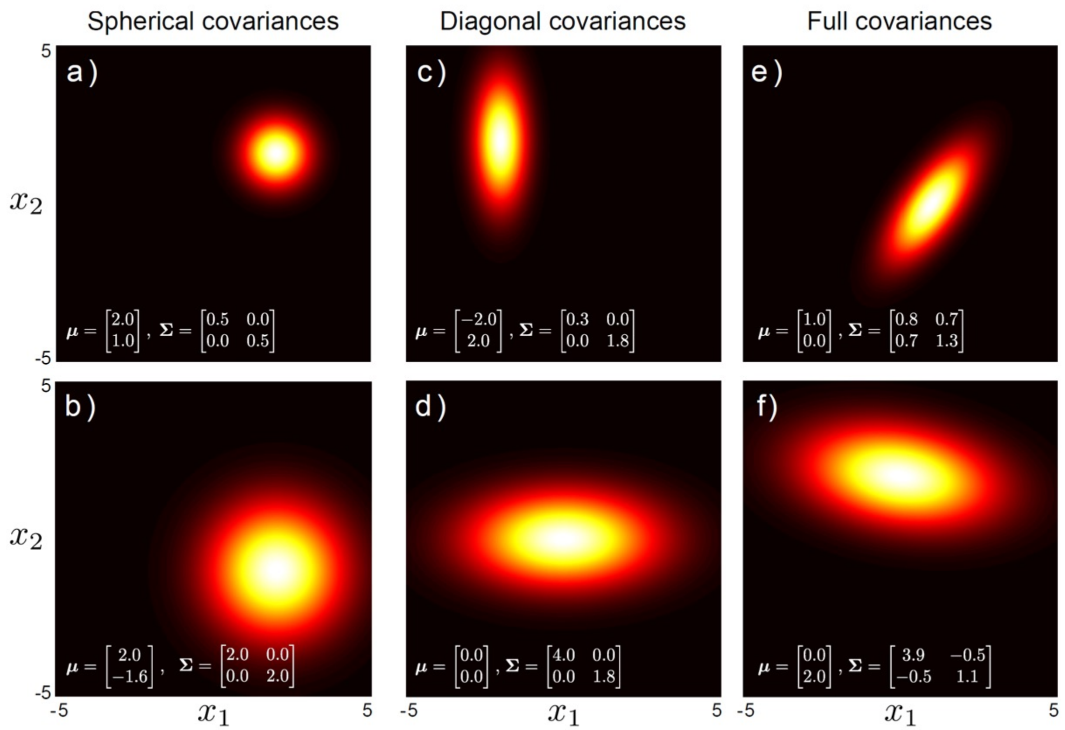
$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\{ -0.5(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \}$$

$$p(\boldsymbol{x}) = \mathrm{Norm}_{\boldsymbol{x}}[\boldsymbol{\mu}, \boldsymbol{\Sigma}]$$

Types of Covariance

- Covariance matrix has three forms: spherical, diagonal and full.

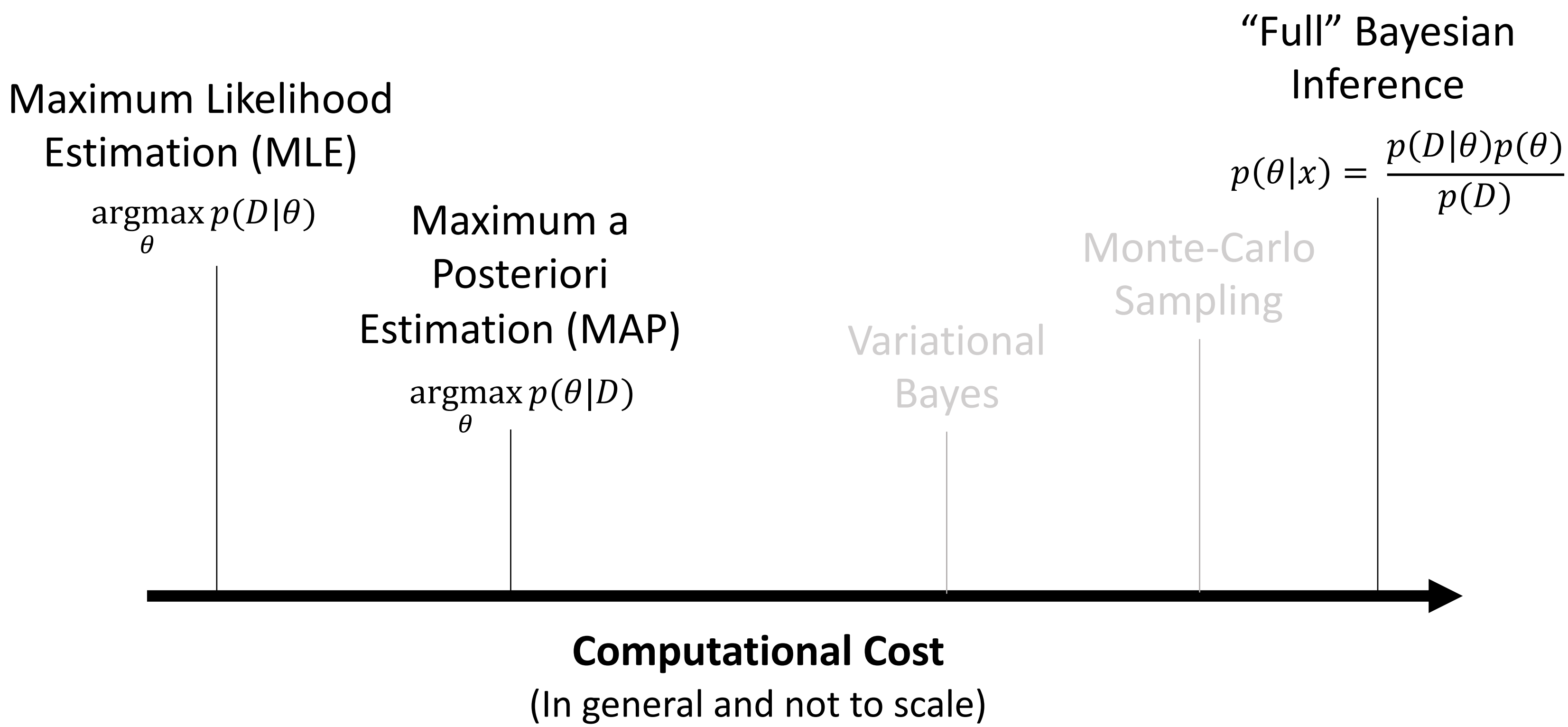
$$\mathbf{\Sigma}_{spher} = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix} \quad \mathbf{\Sigma}_{diag} = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} \quad \mathbf{\Sigma}_{full} = \begin{bmatrix} \sigma_{11}^2 & \sigma_{12}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 \end{bmatrix}$$



Images Source: “Computer Vision: Models, Learning, and Inference”, Simon Prince

Learning Parameters

- Common approaches to learn the unknown parameters θ from a set of given data $\mathcal{D} = \{x[1], \dots, x[N]\}$:



Learning Outcomes

- Explain the Gaussian Process and how it is used for Regression
- Describe the different covariance/kernel functions and what constitutes a valid covariance function.
- Describe the Empirical Bayesian procedure for learning hyperparameters

Acknowledgements

- Gaussian Processes for Machine Learning, Rasmussen and Williams, 2006. http://www.gaussianprocess.org

- Further Exploration:

  - A Visual Exploration of Gaussian Processes, Görtler, et al., Distill, 2019. https://distill.pub/2019/visual-exploration-gaussian-processes/

  - Kernel Cookbook, D. Duvenaud, https://www.cs.toronto.edu/~duvenaud/cookbook/

  - Gaussian Processes, Neil Lawrence, http://inverseprobability.com/talks/notes/gaussian-processes.html

NUS National University of Singapore | School of Computing

Motivating Example & Applications

Why do we need GPs?

(Non-linear) Regression

- Given $\mathcal{D} = \{(\mathbf{x}[1], y[1]), (\mathbf{x}[2], y[2]), \ldots, (\mathbf{x}[N], \mathbf{y}[N])\}$
- Want:
  - Function $y = f(\mathbf{x})$
  - Can predict $\mathrm{y}^* = f(\mathbf{x}^*)$ for new test point $\mathbf{x}^*$

Problem: How certain is the prediction $f(\mathbf{x}^*)$?

The Key Idea

- **Bayesian** framework
- **Prior** over possible **functions**
- **Infer Posterior** after seeing data.

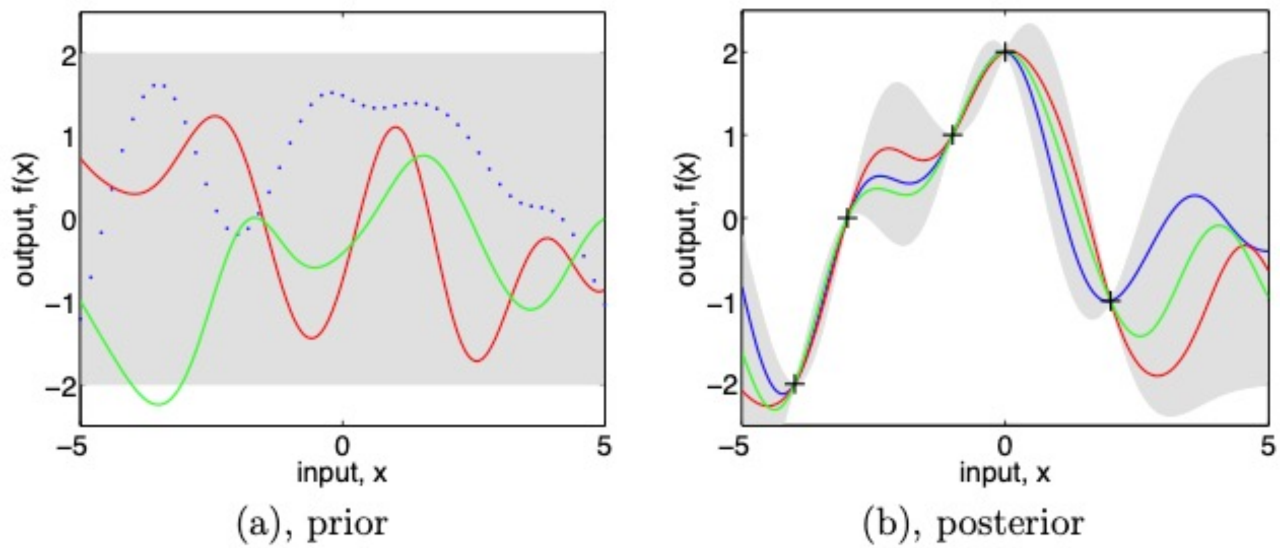
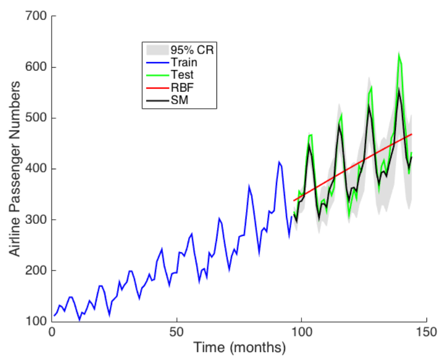


Image Credit: Rasmussen and Williams "Gaussian Processes for Machine Learning", Chp 2

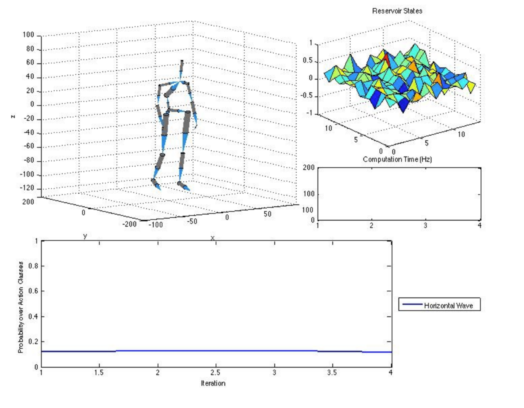
Example Applications

Airline Passenger Predictions



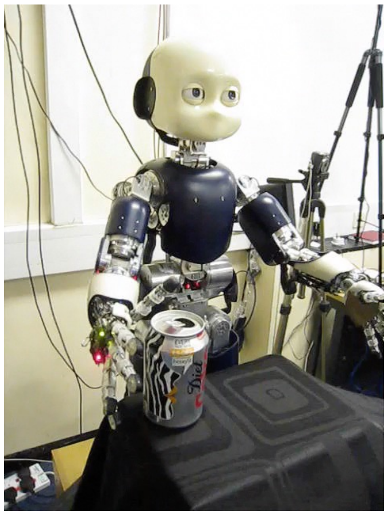
[Wilson and Adams, ICML 2013]. Image from: https://people.orie.cornell.edu/andrew/pattern/

Action Recognition



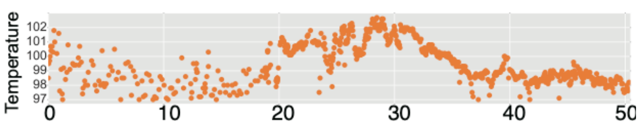
[Soh and Demiris, 2012]

Object Recognition By touch



[Soh and Demiris, 2015]

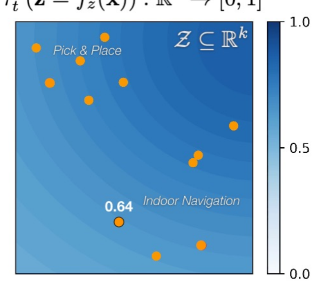
Medical Monitoring



[Cheng et al, 2018] Image:
https://arxiv.org/pdf/1703.09112.pdf
https://github.com/bee-hive/MedGP

Human Trust Modeling

$$\tau_t^a(\mathbf{z} = f_z(\mathbf{x})) : \mathbb{R}^k \to [0, 1]$$

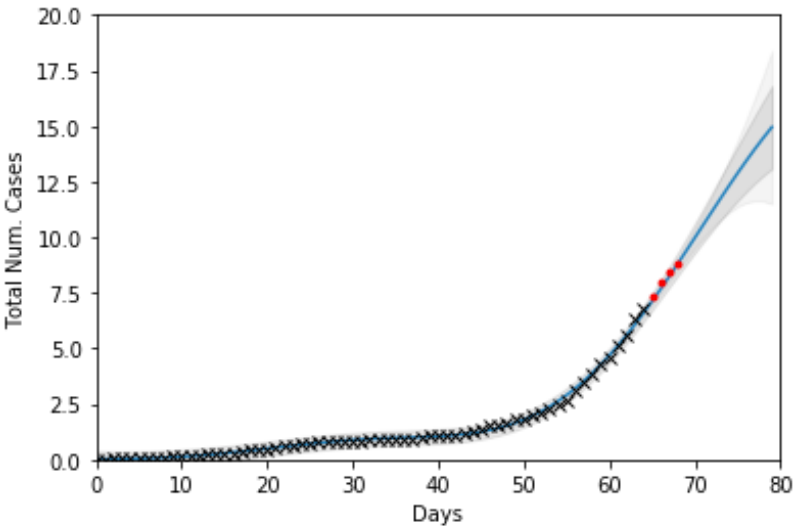


[Soh et al, 2018]

And many more…

GP Code

- Data from Kaggle: https://www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset/data#
- Code on class github



Benefits and Drawbacks of GPs

- **Pros:**
  - Conceptually simple and elegant
  - Interpretable
  - Posterior Predictive distributions $p(y^*|\mathbf{x}^*, \boldsymbol{f})$
  - Flexible, yet Prevents Overfitting
  - Model Selection
- **Cons:**
  - Computationally expensive, $O(N^3)$ for basic GP
  - Can be sensitive to choice of prior
    - Covariance/kernel function and hyperparameters can be difficult specify.

NUS National University of Singapore | School of Computing

The (Wonderful) Properties of Gaussians

Preliminaries

Key ideas

- Gaussians have **nice properties**!
- **“Closed”** under operations of interest
  - Affine transformations, Marginalization, Conditioning
- Probabilistic inference with Gaussians is usually **tractable** and **simple**
  - just linear algebra

Lecture 1: Multivariate Normal Distribution

- Multivariate normal distribution describes a *D*-dimensional continuous variable X, i.e. $\mathbf{x} \in \mathbb{R}^D$.
- *D*-dimensional mean $\boldsymbol{\mu} \in \mathbb{R}^D$, and $D \times D$ symmetrical positive semidefinite covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}_+^{D \times D}$.

$$p(\boldsymbol{X} = \boldsymbol{a} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\{ -0.5(\boldsymbol{a} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{a} - \boldsymbol{\mu}) \}, \quad \boldsymbol{a} \in \mathbb{R}^D$$

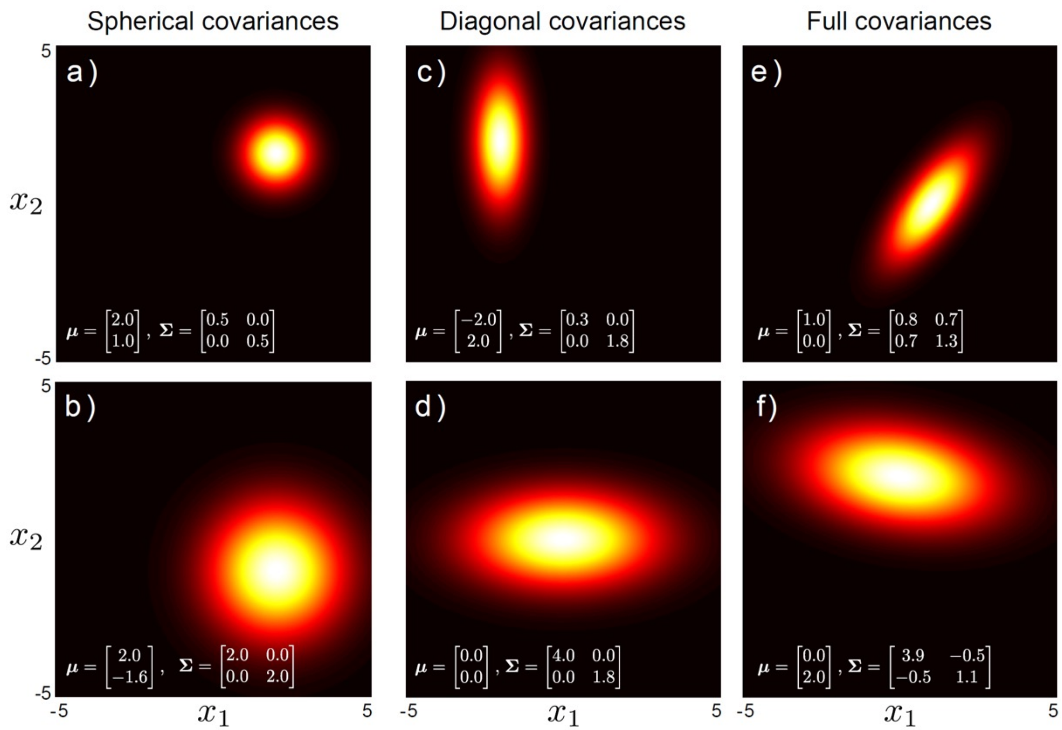
Or

$$\boxed{\begin{aligned} p(\boldsymbol{x}) &= \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\{ -0.5(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \} \\ p(\boldsymbol{x}) &= \mathrm{Norm}_{\boldsymbol{x}}[\boldsymbol{\mu}, \boldsymbol{\Sigma}] \end{aligned}}$$

Lecture 1: Types of Covariance

- Covariance matrix has three forms: spherical, diagonal and full.

$$\mathbf{\Sigma}_{spher} = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix} \quad \mathbf{\Sigma}_{diag} = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} \quad \mathbf{\Sigma}_{full} = \begin{bmatrix} \sigma_{11}^2 & \sigma_{12}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 \end{bmatrix}$$



Images Source: "Computer Vision: Models, Learning, and Inference", Simon Prince

Lecture 1: Sum and Product Rules

- Sum rule:

$$p(x) = \int p(x, y)\, dy$$

$$p(x) = \sum_y p(x, y)$$

- Product/Chain rule:

$$p(x, y) = p(x|y)p(y)$$

The Nice Properties of Gaussians

- Remains “**closed**” under the following operations:
  - Scaling
  - Adding a constant
  - Sum
  - Affine Transformations
  - Marginalization
  - Conditioning
- Applying any of the above operations leads to another **Gaussian**.

Scaling, Adding a Constant, & Sum

If $\mathbf{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a Gaussian random variable then:

- **Scaling:** $\alpha\mathbf{x} \sim N(\alpha\boldsymbol{\mu}, \alpha^2\boldsymbol{\Sigma})$
- **Adding a Constant:** $\mathbf{x} + \mathbf{a} \sim N(\boldsymbol{\mu} + \boldsymbol{a}, \boldsymbol{\Sigma})$

Sum: If $\mathbf{x} \sim N(\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}})$ and $\mathbf{y} \sim N(\boldsymbol{\mu}_{\mathbf{y}}, \boldsymbol{\Sigma}_{\mathbf{y}})$ are independent then

$$\mathbf{x} + \mathbf{y} \sim N(\boldsymbol{\mu}_{\mathbf{x}} + \boldsymbol{\mu}_{y}, \boldsymbol{\Sigma}_{\mathbf{x}} + \boldsymbol{\Sigma}_{\mathbf{y}})$$

Affine Transformation

If $\mathbf{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a Gaussian random variable

and $\mathbf{a}$ is a constant vector and $\mathbf{B}$ is a constant matrix

then:

$$\mathbf{a} + \mathbf{B}\mathbf{x} \sim N(\mathbf{a} + \mathbf{B}\boldsymbol{\mu}, \mathbf{B}\boldsymbol{\Sigma}\mathbf{B}^{\top})$$

Marginalization

Let $\mathbf{x}$ and $\mathbf{y}$ be jointly Gaussian random variables.

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \sim N\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^\top & \mathbf{B} \end{bmatrix}\right)$$

Then, the **marginal distribution** of $\mathbf{x}$

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{y}) dy$$

$$= N(\boldsymbol{\mu}_x, \mathbf{A})$$

Can **simply drop** the **irrelevant variables**.

Exercise: Proof follows from affine property. Consider a **B** that "selects" the appropriate variables.

Conditioning

Let $\mathbf{x}$ and $\mathbf{y}$ be jointly Gaussian random variables.

$$\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} \sim N\left(\begin{bmatrix}\boldsymbol{\mu}_x\\ \boldsymbol{\mu}_y\end{bmatrix}, \begin{bmatrix}\mathbf{A} & \mathbf{C}\\ \mathbf{C}^\top & \mathbf{B}\end{bmatrix}\right)$$

Then, the **conditional distribution** of $\mathbf{x}$

$$p(\mathbf{x}|\mathbf{y} = \widehat{\boldsymbol{y}}) =$$

$$N(\boldsymbol{\mu}_x + \mathbf{C}\mathbf{B}^{-1}(\widehat{\boldsymbol{y}} - \boldsymbol{\mu}_y), \mathbf{A} - \mathbf{C}\mathbf{B}^{-1}\mathbf{C}^\top)$$

The Nice Properties of Gaussians

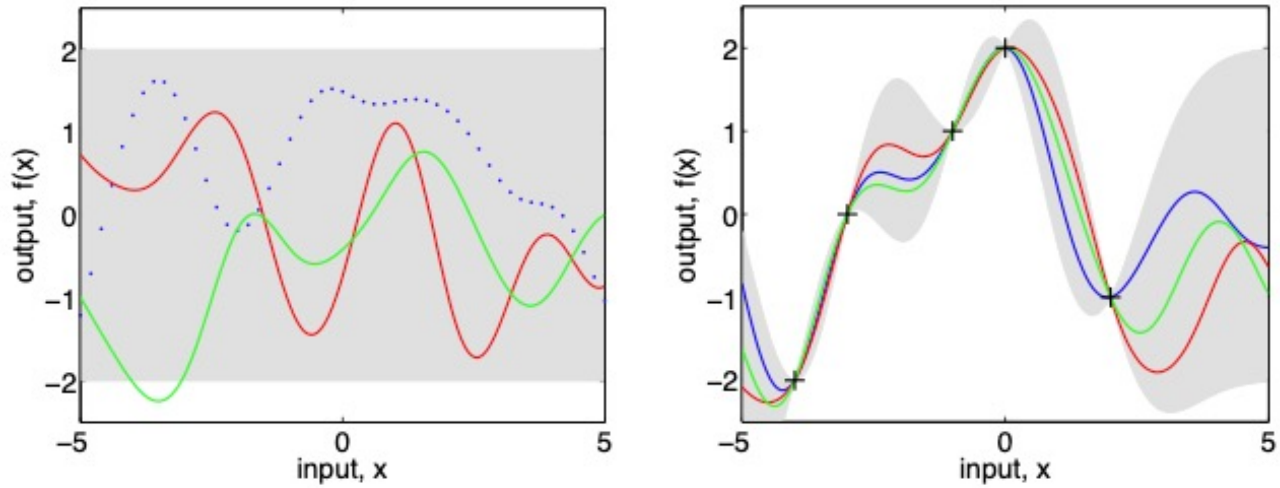
- Remains "**closed**" under the following operations:
  - Scaling
  - Adding a constant
  - Sum
  - Affine Transformations
  - Marginalization
  - Conditioning
- Applying any of the above operations leads to another **Gaussian**.

Gaussian Processes

Intuition and Formal Definitions

The Key Idea

- **Bayesian** framework
- **Prior** over possible **functions**
- **Infer Posterior** after seeing data.



(a), prior (b), posterior

Image Credit: Rasmussen and Williams "Gaussian Processes for Machine Learning", Chp 2

Intuition for GPs

- Two Approaches / Views
- **View 1:** Weight Space View
- **View 2:** Function Space View

(Non-linear) Regression

- Given $\mathcal{D} = \{(\mathbf{x}[1], y[1]), (\mathbf{x}[2], y[2]), \ldots, (\mathbf{x}[N], \mathbf{y}[N])\}$
- Want:
  - Function $y = f(\mathbf{x})$
  - Can predict $\mathrm{y}^* = f(\mathbf{x}^*)$ for new test point $\mathbf{x}^*$

$$\mathbf{X} = \begin{bmatrix} | & | & | \\ \mathbf{x}[0] & \mathbf{x}[1] & \mathbf{x}[N] \\ | & | & | \end{bmatrix}$$

- Notation:
  - **Design Matrix: $\mathbf{X}$**
  - **Targets $\mathbf{y}$**
  - **Data = $\mathcal{D} = (\mathbf{X}, \mathbf{y})$**

$$\mathbf{y} = \begin{bmatrix} y[1] \\ y[2] \\ \vdots \\ y[N] \end{bmatrix}$$

From L3: Linear Regression

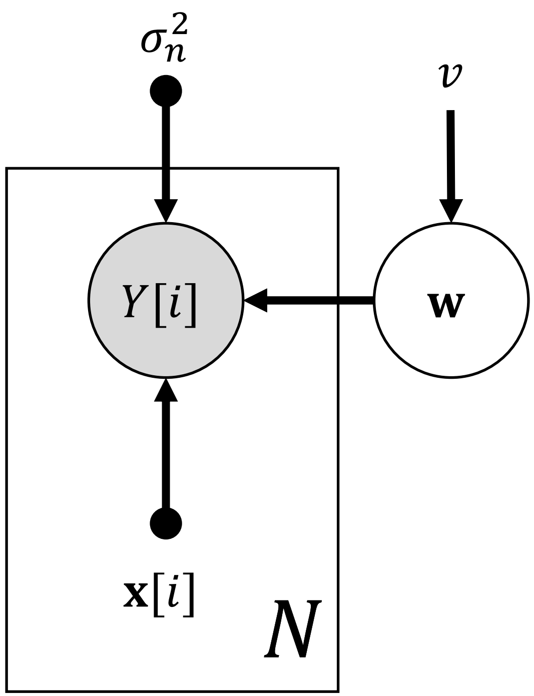
- Model for data point with index i:

$$Y[i] = \mathbf{w}^\top \mathbf{x}[i] + \epsilon[i]$$

where:

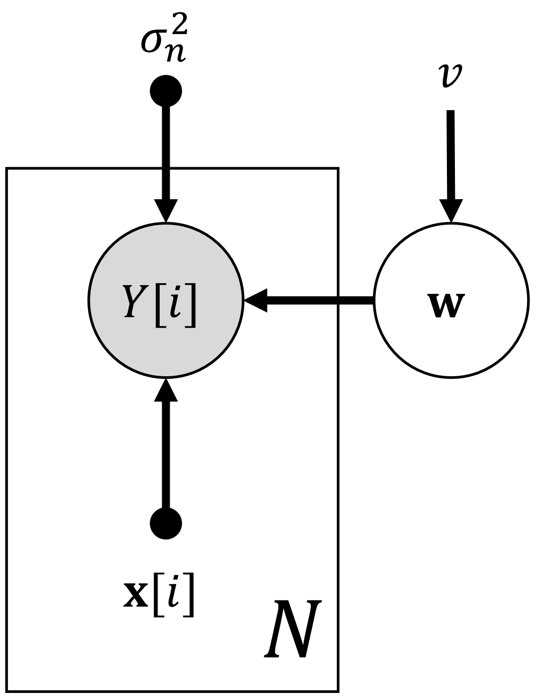
- $\mathbf{x}[i] = [\mathrm{x}[i]_1, \mathrm{x}[i]_2, \ldots, \mathrm{x}[i]_\mathrm{D}]^\top$ is a D-dimensional observed input vector
- $\mathbf{w} = [w_1, w_2, \ldots, w_D]^\top$ is a coefficient vector
- $\epsilon[i] \sim N(0, \sigma_n^2)$ is iid zero-mean Gaussian noise

From L3: DGM for Bayesian Linear Regression



- Model uncertainty over $\mathbf{w}$
- The coefficient vector $\mathbf{w}$ is now a random variable with a prior $p(\mathbf{w}|v) = N(\mathbf{0}, v\mathbf{I})$

From L3: DGM for Bayesian Linear Regression



- Write the factorization:

$$p(y[1], \dots, y[N], \mathbf{w}) = p(\mathbf{w}|v) \prod_{i}^{N} p(y[i] | \, \mathbf{w}^\top \mathbf{x}[i], \sigma_n^2)$$

Exercise: Assume we know σ_n^2, give the MAP solution for $\mathbf{w}$.

Bayesian Linear Regression

- We want the posterior:

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

- **Prior:** $p\left(\mathbf{w}\middle|\boldsymbol{\Sigma}_p\right) = N(\mathbf{0},\boldsymbol{\Sigma}_p)$ (Note: consider $\boldsymbol{\Sigma}_p$ instead of $v\mathbf{I}$)
- **Likelihood:** $\prod_i p(y[i]|\mathbf{X},\mathbf{w}) = \prod_i N(\mathbf{w}^\top\mathbf{x}[\mathrm{i}],\sigma_n^2)$

- **Posterior:**

$$p(\mathbf{w}|\mathbf{y},\mathbf{X}) = N(\mathbf{A}^{-1}\mathbf{X}\mathbf{y},\mathbf{A}^{-1})$$

$$\text{where } \mathbf{A} = \sigma_n^{-2}\mathbf{X}\mathbf{X}^\top + \boldsymbol{\Sigma}_p^{-1}$$

Bayesian Linear Regression

- **Posterior**:

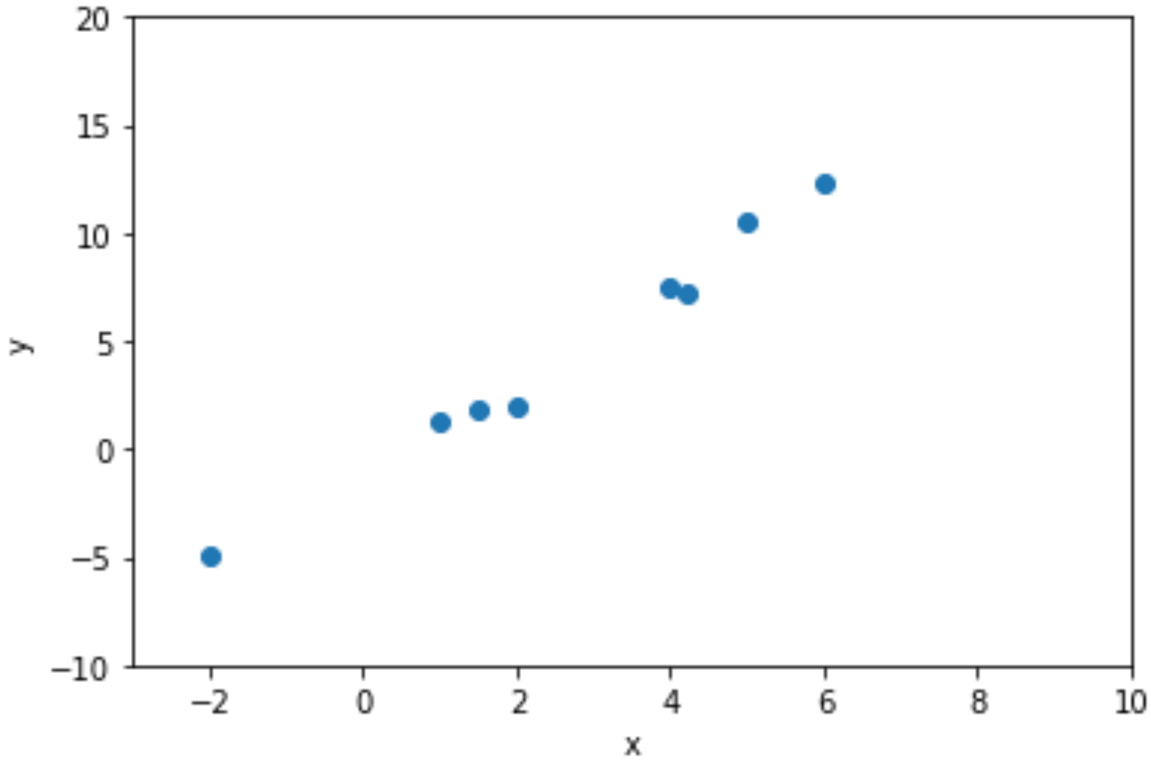
$$p(\mathbf{w}|\mathbf{y},\mathbf{X}) = N(\mathbf{A}^{-1}\mathbf{X}\mathbf{y},\mathbf{A}^{-1})$$

where $\mathbf{A} = \sigma_n^{-2}\mathbf{X}\mathbf{X}^\top + \mathbf{\Sigma}_p^{-1}$

- **Posterior Predictive Distribution**:

$$p(y^*|\mathbf{x}^*,\mathbf{X},\mathbf{y}) = \int p(y^*|\mathbf{x}^*,\mathbf{w})p(\mathbf{w}|\mathbf{y},\mathbf{X})d\mathbf{w}$$
$$= N(\sigma_n^{-2}\mathbf{x}^{*\top}\mathbf{A}^{-1}\mathbf{X}\mathbf{y}, \mathbf{x}^{*\top}\mathbf{A}^{-1}\mathbf{x}^*)$$

Visually: Posterior Predictive



From L3: Why Linear Regression?

- Basis function “trick”
- Let $\boldsymbol{\phi}(x)$ be some function that transforms x into another vector of “features”
- E.g.:
  - $\boldsymbol{\phi}(x) = [x, x^2, 1]^\top$
  - $\boldsymbol{\phi}(x) = [x^p, x^{p-1}, \dots\ x^2, x, 1]^\top$
- Then, applying the linear model, we get:
  - $Y[i] = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}[i]) \ + \epsilon[i]$
  - For the examples above, this is *polynomial regression*.
- $\boldsymbol{\phi}(x)$ can be more complex:
  - E.g.: $\boldsymbol{\phi}(\boldsymbol{x}[i]) = h(\mathbf{A}\mathbf{x}[i])$ where h is a nonlinear “activation function” (What is this?)

Bayesian Non-linear Regression

- Let $\boldsymbol{\Phi} = \Phi(\mathbf{X})$
  - apply the basis function to all input points $\mathbf{X}$
  - Also, $\boldsymbol{\phi}^* = \boldsymbol{\phi}(\mathbf{x}^*)$ (apply to test point)
- Then use the same reasoning to get:

$$p(y^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y}) = N(\sigma_n^{-2}\boldsymbol{\phi}^{*\top}\mathbf{A}^{-1}\boldsymbol{\Phi}\mathbf{y}, \boldsymbol{\phi}^{*\top}\mathbf{A}^{-1}\boldsymbol{\phi}^*)$$

where $\mathbf{A} = \sigma_n^{-2}\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \boldsymbol{\Sigma}_p^{-1}$

Compare with:

$$p(y^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y}) = N(\sigma_n^{-2}\mathbf{x}^{*\top}\mathbf{A}^{-1}\mathbf{X}\mathbf{y}, \mathbf{x}^{*\top}\mathbf{A}^{-1}\mathbf{x}^*)$$

Towards Kernels

- Define:
  - $\boldsymbol{\psi}(\mathbf{x}) = \boldsymbol{\Sigma}_p^{1/2}\boldsymbol{\phi}(\mathbf{x})$
  - $k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\psi}(\mathbf{x})^\top \boldsymbol{\psi}(\mathbf{x}')$
  - $k^* = k(\mathbf{x}^*, \mathbf{x}^*)$

Apply "kernel trick": compute inner product $k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\psi}(\mathbf{x})^\top \boldsymbol{\psi}(\mathbf{x}')$ by evaluating a kernel function

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}[1], \mathbf{x}[1]) & \cdots & k(\mathbf{x}[1], \mathbf{x}[N]) \\ k(\mathbf{x}[2], \mathbf{x}[1]) & & k(\mathbf{x}[2], \mathbf{x}[N]) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}[N], \mathbf{x}[1]) & \cdots & k(\mathbf{x}[\mathrm{N}], \mathbf{x}[N]) \end{bmatrix} \qquad \mathbf{k}^* = \begin{bmatrix} k(\mathbf{x}[1], \mathbf{x}^*) \\ k(\mathbf{x}[2], \mathbf{x}^*) \\ \vdots \\ k(\mathbf{x}[N], \mathbf{x}^*) \end{bmatrix}$$

Towards Kernels

Rewrite the posterior predictive:

$$p(y^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y}) = N\left(\sigma_n^{-2} \boldsymbol{\phi}^{*\top} \mathbf{A}^{-1} \boldsymbol{\Phi} \mathbf{y}, \boldsymbol{\phi}^{*\top} \mathbf{A}^{-1} \boldsymbol{\phi}^*\right)$$
$$= N(\boldsymbol{\mu}, \mathbf{V})$$

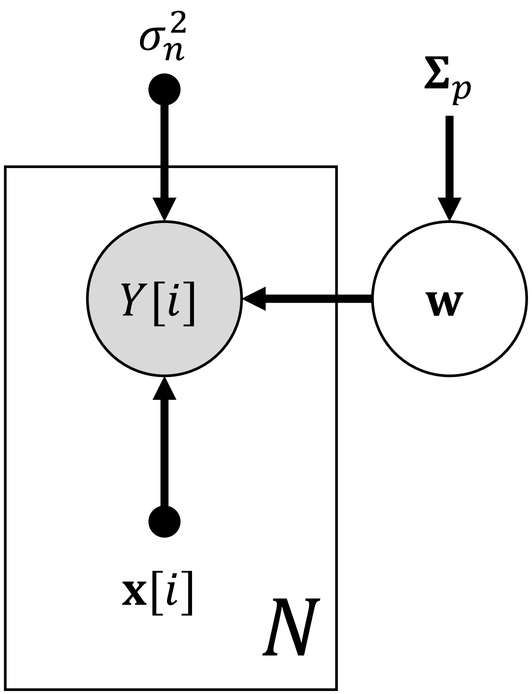
Where:

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

Apply "kernel trick": compute inner product $k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\psi}(\mathbf{x})^\top \boldsymbol{\psi}(\mathbf{x})$ by evaluating a kernel function

Recap: Weight Space view

Started with Bayesian Linear Regression



1. Since all Gaussian, posterior and posterior predictive computable in closed form.
2. Using basis function trick, perform non-linear regression.
3. Using kernel trick, obtain posterior predictive in terms of kernel evaluations

$$p(y^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y}) = N(\boldsymbol{\mu}, \mathbf{V})$$

where:

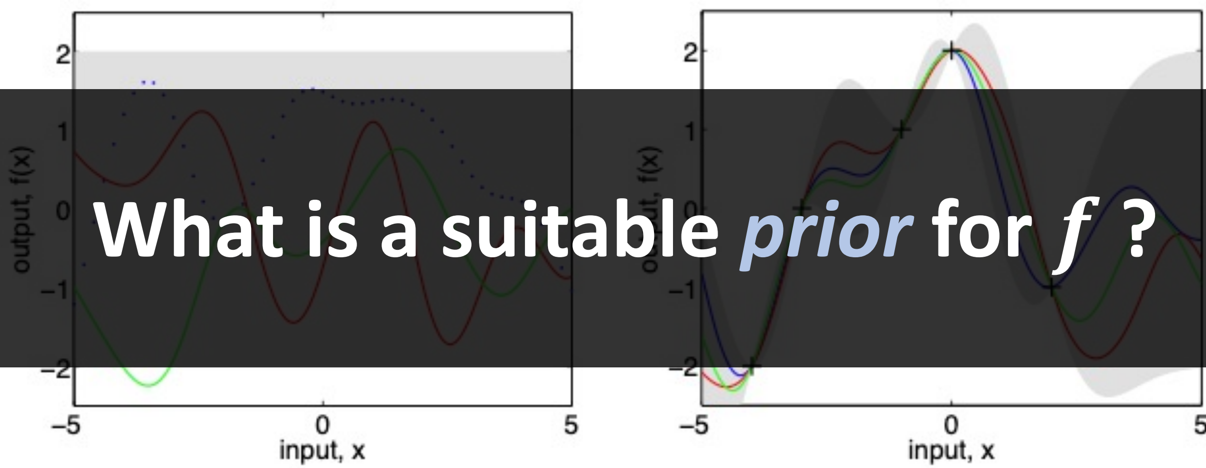
$$\boldsymbol{\mu} = \mathbf{k}^{*\top}(\mathbf{K} + \sigma_n^2 I)^{-1}\,\mathbf{y}$$
$$\mathbf{V} = k^* - \mathbf{k}^{*\top}(\mathbf{K} + \sigma_n^2 I)^{-1}\mathbf{k}^*$$

Intuition for GPs

- Two Approaches / Views
- **View 1:** Weight Space View
- **View 2:** Function Space View

Function space view: Key idea

- Consider the (unknown) function f
- Place a **prior** over f
- And perform **Bayesian inference** to get **posterior and posterior predictive.**



(a), prior (b), posterior

What is a suitable *prior* for f ?

Image Credit: Rasmussen and Williams "Gaussian Processes for Machine Learning", Chp 2

Gaussian Process (GP)

Definition: A **Gaussian process (GP)** is a **collection** of random variables, a **finite number** of which have **joint Gaussian distribution**

We write the GP as

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

where

Mean function: $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$

Covariance function:

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))$$

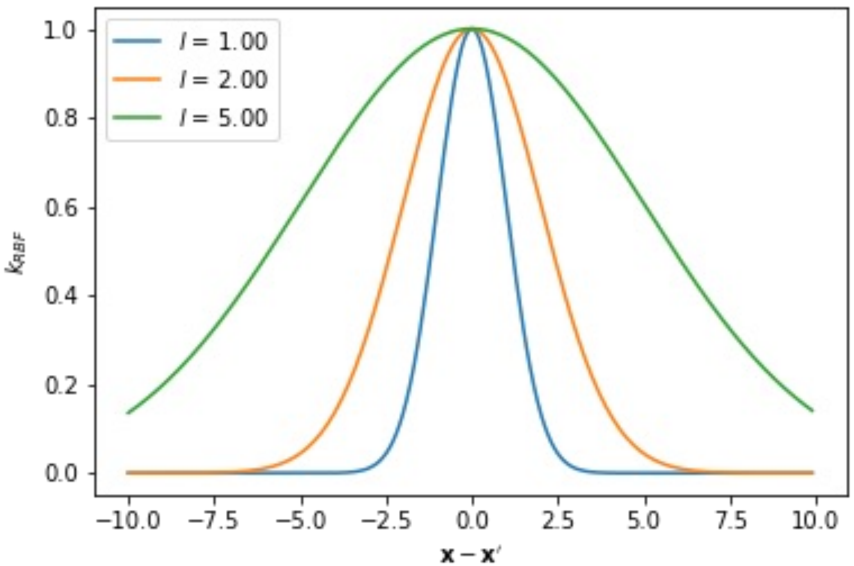
Ingredients of a GP

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

- **Mean function:** $m(\mathbf{x})$
  - *Usually* **set** to $m(\mathbf{x}) = \mathbf{0}$
- **Covariance function:** $k(\mathbf{x}, \mathbf{x}')$
  - The **main** ingredient
  - Popular example: **Squared Exponential** (SE) or **Radial Basis Function** (RBF)

Radial Basis Function Kernel

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2l^2}\right)$$

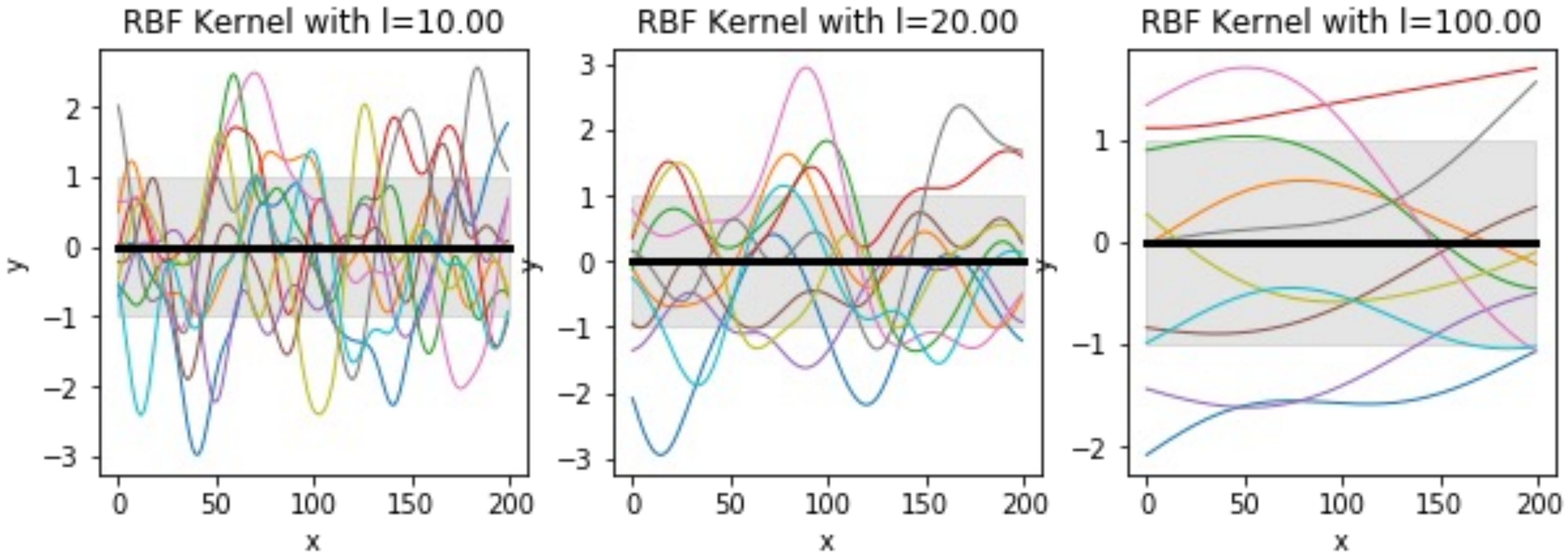


Sampling from the GP prior

- Generate a random vector

$$\mathbf{f}^* \sim N(0, k(\mathbf{X}^*, \mathbf{X}^*))$$

where $\mathbf{X}^*$ is a matrix of input locations



Bayesian Non-linear regression

- We **assume** that:

$$y = f(\mathbf{x}) + \epsilon$$

where $\epsilon \sim N(0, \sigma_n^2)$

- We could obtain a **posterior**:

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

- Often, we really want the **predictive distribution**

$$p(y^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y}) = \int p(y^*|\mathbf{x}^*, f)p(f|\mathbf{y}, \mathbf{X})df$$

Posterior Predictive (noise-free)

The joint distribution of $p(\mathbf{f}, \mathbf{f}^*)$ according to the GP prior is:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^* \end{bmatrix} \sim N\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{k}^* \\ \mathbf{k}^{*\top} & k^* \end{bmatrix}\right)$$

Then the **conditional**:

$$p(\mathbf{f}^* | \mathbf{X}^*, \mathbf{X}, \mathbf{f}) = N(\boldsymbol{\mu}, \mathbf{V})$$

where

$$\boldsymbol{\mu} = \mathbf{k}^{*\top} \mathbf{K}^{-1} \mathbf{y}$$
$$\mathbf{V} = k^* - \mathbf{k}^{*\top} \mathbf{K}^{-1} \mathbf{k}^*$$

Posterior Predictive (noisy)

Can only see **noisy** $\mathbf{y} = \mathbf{f} + \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma_n^2 \mathbf{I})$

Then, $p(\mathbf{y}, \mathbf{f}^*)$ according to the GP prior is:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}^* \end{bmatrix} \sim N\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma_n^2 \mathbf{I} & \mathbf{k}^* \\ \mathbf{k}^{*\top} & k^* \end{bmatrix}\right)$$

Then, the **conditional**:

$$p(\mathbf{f}^* | \mathbf{X}^*, \mathbf{X}, \mathbf{f}) = N(\boldsymbol{\mu}, \mathbf{V})$$

where

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

$$\mathbf{V} = k^* - \mathbf{k}^{*\top} (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{k}^*$$

From Weight Space View: Towards Kernels

Rewrite the posterior predictive:

$$p(y^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y}) = N\left(\sigma_n^{-2} \boldsymbol{\phi}^{*\top} \mathbf{A}^{-1} \boldsymbol{\Phi} \mathbf{y}, \boldsymbol{\phi}^{*\top} \mathbf{A}^{-1} \boldsymbol{\phi}^*\right)$$
$$= N(\boldsymbol{\mu}, \mathbf{V})$$

Where:

$$\boldsymbol{\mu} = \mathbf{k}^{*\top}(\mathbf{K} + \sigma_n^2 I)^{-1}\, \mathbf{y}$$
$$\mathbf{V} = k^* - \mathbf{k}^{*\top}(\mathbf{K} + \sigma_n^2 I)^{-1} \mathbf{k}^*$$

Apply "kernel trick": compute inner product $k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\psi}(\mathbf{x})^\top \boldsymbol{\psi}(\mathbf{x})$ by evaluating a kernel function

Posterior Predictive (noisy)

$$p(\mathbf{f}^*|\mathbf{X}^*, \mathbf{X}, \mathbf{f}) = N(\boldsymbol{\mu}, \mathbf{V})$$

where

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

$$\mathbf{V} = k^* - \mathbf{k}^{*\top}(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{k}^*$$

Posterior Predictive: **Mean**

- "***Linear* Predictor**" in terms of the **kernel functions**

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

$$= \mathbf{k}^{*\top}\boldsymbol{\alpha}$$

$$= \sum_{i=1}^{N} \alpha_i k(\mathbf{x}[i], \mathbf{x}^*)$$

Linear combination of N kernel functions at the training points

where $\boldsymbol{\alpha} = \underbrace{(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}}_{\text{Precision Matrix } \boldsymbol{\Lambda}}\mathbf{y}$

Precision Matrix $\boldsymbol{\Lambda}$

$\boldsymbol{\Lambda}_{ij} = 0$ iff X_i and X_j are **conditionally independent** given all other X.

Posterior Predictive: **Mean**

- "***Linear* Smoother**" in terms of the **targets**

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

$$= \boldsymbol{\beta}^{*\top}\boldsymbol{y}$$

$$= \sum_{i=1}^{N} \beta_i^* y_i$$

Linear combination of N targets

where $\boldsymbol{\beta}^* = \mathbf{k}^{*\top}(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}$

Posterior Predictive: **Variance**

$$\mathbf{V} = \underbrace{k^*}_{\text{Prior covariance}} - \underbrace{\mathbf{k}^{*\top}(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{k}^*}_{\text{Information from other points}}$$

- Does **not** depend on the targets $\mathbf{y}$.
- This variance is for $p(\mathbf{f}^*|\mathbf{X}^*, \mathbf{X}, \mathbf{f})$
  - Uncertainty about the true function values
- To get variance for targets $p(\boldsymbol{y}^*|\mathbf{X}^*, \mathbf{X}, \mathbf{f})$, add the noise:

$$\mathbf{V}_{\mathbf{y}^*} = \mathbf{V} + \sigma_n^2\mathbf{I}$$

Computational Complexity

$$p(\mathbf{f}^*|\mathbf{X}^*, \mathbf{X}, \mathbf{f}) = N(\boldsymbol{\mu}, \mathbf{V})$$

where

$$\boldsymbol{\mu} = \mathbf{k}^{*\top}(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\,\mathbf{y}$$
$$\mathbf{V} = k^* - \mathbf{k}^{*\top}(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{k}^*$$

Complexity is related to cost of inverting $(\mathbf{K} + \sigma_n^2\mathbf{I})$: $O(N^3)$

Also maintain kernel matrix of size $O(N^2)$

Bayesian Non-parametrics

- GPs are a specific example of **Bayesian non-parametric** models.
- Does **not mean no parameters or no assumptions**
- Number of parameters **grows** with data
  - Theoretically “infinite”
- Can represent **very complex functions**, but Bayesian so, naturally **can prevent overfitting**

- **See also:** Dirichlet Processes

Covariance Functions

Kernels, Stationary and non-stationary kernels,
Positive Semi-definite kernels

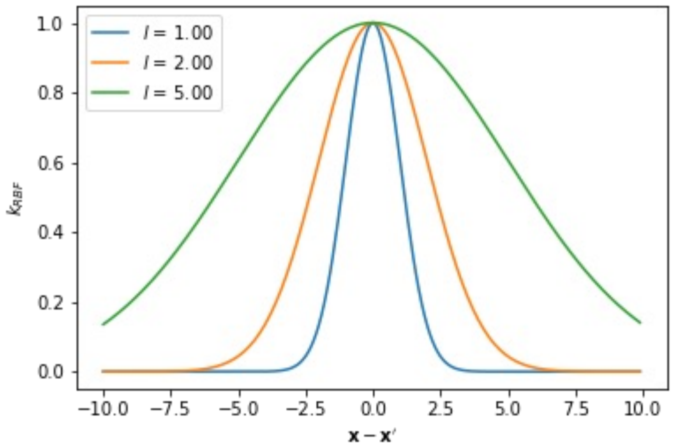
Ingredients of a GP

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

- **Mean function:** $m(\mathbf{x})$
  - *Usually* **set** to $m(\mathbf{x}) = \mathbf{0}$
- **Covariance function:** $k(\mathbf{x}, \mathbf{x}')$
  - The **main** ingredient
  - Popular example: **Squared Exponential** (SE) or **Radial Basis Function** (RBF)

Key Ideas

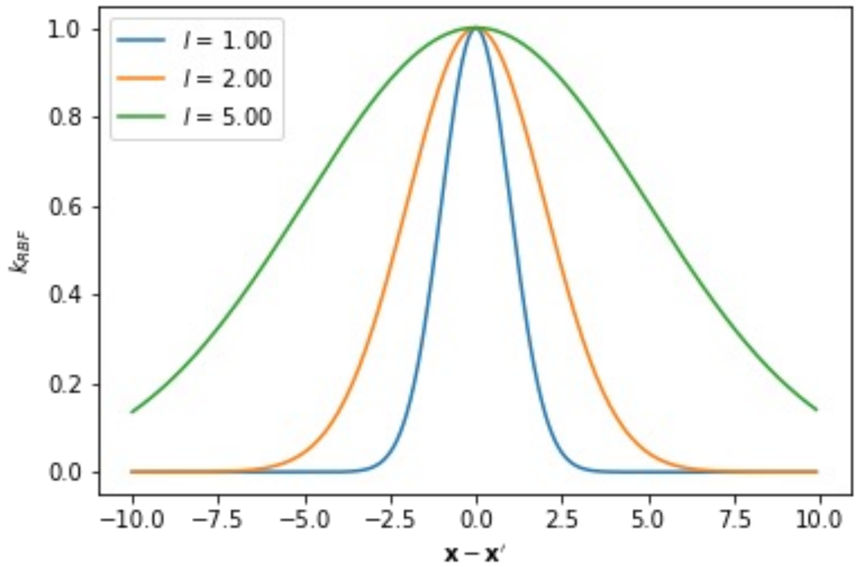
- Covariance functions or kernels encode **similarity**
- Examine some **properties** and **examples**
  - (Non)Stationary, (An)isotropic
- **Not all** similarity functions are **valid** covariance functions
  - Need to be Positive Semi-Definite (PSD)
- Making **new kernels**



Radial Basis Function Kernel

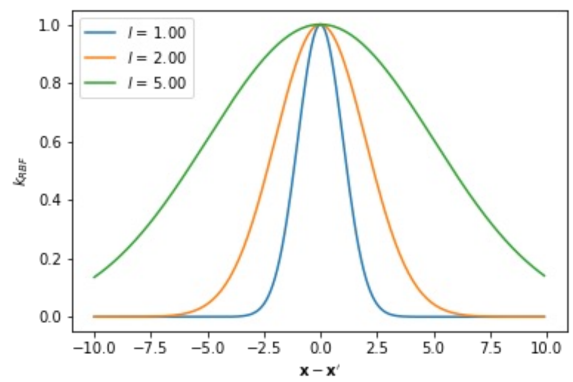
$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2l^2}\right)$$

l is a **hyperparameter**



Stationary Kernels

- A **stationary kernel** is a function of $\mathbf{x} - \mathbf{x}'$
  - **Invariant to translations** in input space
  - Example: Matern, Rational Quadratic, Exponential
- **Isotropic** if function of only $||\mathbf{x} - \mathbf{x}'||$
  - Invariant to **all rigid motions**.
    - E.g., rotation, reflection
  - Example: RBF



Anisotropic Stationary Kernel

Consider:

$$k(\mathbf{x}, \mathbf{x}') = \exp(-(\mathbf{x} - \mathbf{x}')^\top \mathbf{M}(\mathbf{x} - \mathbf{x}'))$$

where $\mathbf{M}$ is a PSD matrix

Examples:

- $\mathbf{M} = \mathbf{\Psi}$ where $\mathbf{\Psi} = \mathrm{diag}\left(\left[l_j^{-2}\right]_j^D\right)$:
  - Can specify lengthscale (feature importance) for each dimension.
  - With optimization: Automatic Relevance Determination (ARD)
- $\mathbf{M} = \mathbf{B}\mathbf{B}^\top$ where $\mathbf{B}$ is a $D \times k$ matrix
  - linear **dimensionality reduction**
- $\mathbf{M} = \mathbf{B}\mathbf{B}^\top + \mathbf{\Psi}$
  - **factor analysis** form

Non-Stationary: Dot-Product Kernel

- A **dot-product kernel** is a function of $\mathbf{x}^\top \mathbf{x}'$
  - **Invariant to rotations** (but **not translations**) of input space
  - Obtained by via linear regression
- Can generalize to:

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{C} \mathbf{x}'$$

where $\mathbf{C}$ is a general covariance matrix

Positive Semidefinite (PSD) kernels

- The covariance matrix K must be **positive semidefinite**.

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}[1], \mathbf{x}[1]) & \cdots & k(\mathbf{x}[1], \mathbf{x}[N]) \\ k(\mathbf{x}[2], \mathbf{x}[1]) & & k(\mathbf{x}[2], \mathbf{x}[N]) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}[N], \mathbf{x}[1]) & \cdots & k(\mathbf{x}[N], \mathbf{x}[N]) \end{bmatrix}$$

- A real *symmetric* matrix is **positive semidefinite** iff

$$\forall \mathbf{v} \in \mathbb{R}^d \;\; \mathbf{v}^\top \mathbf{K} \mathbf{v} \geq 0$$

- A kernel is PSD if it results in a **PSD matrix**

Is g a valid covariance function?

- A real symmetric matrix is **positive semidefinite** if

$$\forall \mathbf{v} \in \mathbb{R}^d \mathbf{v}^\top \mathbf{K}\mathbf{v} \geq 0$$

*Equivalently, if all eigenvalues of **K** are non-negative

- Example:
  - Consider $x, x' \in \mathbb{R}$, is $g(x, x') = x \cdot x'$ a PSD kernel?
  - **Yes** since $\mathbf{v}^\top \mathbf{K}\mathbf{v} = \mathbf{v}^\top \mathbf{x}\mathbf{x}^\top \mathbf{v} = (\mathbf{x}^\top \mathbf{v})^\top (\mathbf{x}^\top \mathbf{v}) \geq 0$

Making New Kernels

- Operations that **preserve PSD**
- **Scaling a kernel** by a non-negative constant $c \geq 0$
  - $k(\mathbf{x}, \mathbf{x}') = c k_1(\mathbf{x}, \mathbf{x}')$
- The **sum of two kernels** is a kernel
  - $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$
- The **product of two kernels** is a kernel
  - $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') k_2(\mathbf{x}, \mathbf{x}')$
- The **exponentiation of a kernel** is a kernel
  - $k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}'))$

Can apply these repeatedly, e.g.,

$$k_3(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$$
$$k_5(\mathbf{x}, \mathbf{x}') = k_4(\mathbf{x}, \mathbf{x}') \exp(k_3(\mathbf{x}, \mathbf{x}'))$$

Is g a valid covariance function?

- Can g be constructed from other valid kernels?
- Example:
  - Check: $g(\mathbf{x}, \mathbf{x}') = 2\exp\left((\mathbf{x}^\top \mathbf{x}')^3/b\right) \exp(-\|\mathbf{x} - \mathbf{x}'\|^2)$ where $b > 0$
  - **Yes** (Proof as Exercise)
    - **Hint:** consider the dot-product and RBF kernels. Can we obtain g from using operations on these kernels that preserve PSD?

Hyperparameter Learning

- **Q:** How can we learn the hyperparameters θ?
- The "right" way:

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

- **Problem:** Intractable in general
- So, approximate by maximizing the log marginal likelihood ("Empirical Bayes"):

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

Optimizing the Hyperparameters

The **log marginal likelihood**

$$\log p(\mathbf{y}|\mathbf{X},\theta) = \log \int p(\mathbf{y}|\mathbf{f},\mathbf{X},\theta)p(\mathbf{f}|\theta)d\mathbf{f}$$

Looks difficult, but due to the wonderful properties of Gaussians:

$$p(\mathbf{y}|\mathbf{X},\theta) = N(\mathbf{0},\mathbf{K}+\sigma_n^2\mathbf{I})$$

So,

$$\boxed{\begin{aligned}&\log p(\mathbf{y}|\mathbf{X},\theta)\\&= -\frac{1}{2}\log\det(\mathbf{K}+\sigma_n^2\mathbf{I}) - \frac{1}{2}\mathbf{y}^\top(\mathbf{K}+\sigma_n^2\mathbf{I})^{-1}\mathbf{y} - \frac{N}{2}\log 2\pi\end{aligned}}$$

Example: Automatic Relevance Determination

Recall the kernel:

$$k(\mathbf{x}, \mathbf{x}') = \exp(-(\mathbf{x} - \mathbf{x}')^\top \mathbf{M}(\mathbf{x} - \mathbf{x}'))$$

where $\mathbf{M}$ is a PSD matrix

Examples:

- $\mathbf{M} = \boldsymbol{\Psi}$ where $\boldsymbol{\Psi} = \mathrm{diag}\left(\left[l_j^{-2}\right]_j^D\right)$:
  - Can specify lengthscale (feature importance) for each dimension.
  - With optimization: Automatic Relevance Determination (ARD)

Further Exploration of Kernels

- Chapter 4 of “Gaussian Processes for Machine Learning”
  - E.g., kernels for strings

covariance function	expression	S	ND
constant	σ_0^2	$\surd$	
linear	$\sum_{d=1}^{D} \sigma_d^2 x_d x'_d$		
polynomial	$(\mathbf{x} \cdot \mathbf{x}' + \sigma_0^2)^p$		
squared exponential	$\exp(-\frac{r^2}{2\ell^2})$	$\surd$	$\surd$
Matérn	$\frac{1}{2^{\nu-1}\Gamma(\nu)} \Big(\frac{\sqrt{2\nu}}{\ell} r\Big)^{\nu} K_\nu \Big(\frac{\sqrt{2\nu}}{\ell} r\Big)$	$\surd$	$\surd$
exponential	$\exp(-\frac{r}{\ell})$	$\surd$	$\surd$
γ-exponential	$\exp\big(-(\frac{r}{\ell})^\gamma\big)$	$\surd$	$\surd$
rational quadratic	$(1 + \frac{r^2}{2\alpha\ell^2})^{-\alpha}$	$\surd$	$\surd$
neural network	$\sin^{-1}\Big(\frac{2\tilde{\mathbf{x}}^\top \Sigma \tilde{\mathbf{x}}'}{\sqrt{(1+2\tilde{\mathbf{x}}^\top \Sigma \tilde{\mathbf{x}})(1+2\tilde{\mathbf{x}}'^\top \Sigma \tilde{\mathbf{x}}')}}\Big)$		$\surd$

Image Credit: Rasmussen and Williams “Gaussian Processes for Machine Learning”, Chp 4