Tutorial 5 – Kernels and Gaussian Processes

CSC2541 Neural Net Training Dynamics – Winter 2022

Slides adapted from CSC2541 Scalable and Flexible Models of Uncertainty – Fall 2017

Stephan Rabanser stephan@cs.toronto.edu

University of Toronto
Department of Computer Science

Vector Institute for
Artificial Intelligence

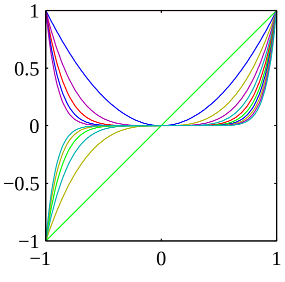
June 29, 2022

Kernel Methods

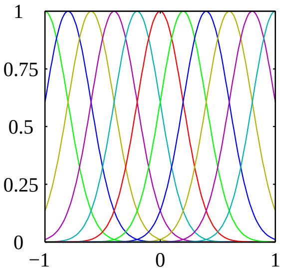
Recap: Basis Functions

- Basis functions allow us to use non-linear feature transformations.
- We can specify them by hand (examples below), or learn them automatically using a neural network.

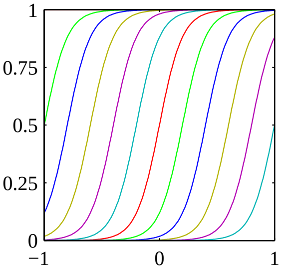
$$\phi_j(x) = x^j$$



$$\phi_j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2s^2}\right)$$



$$\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right)$$

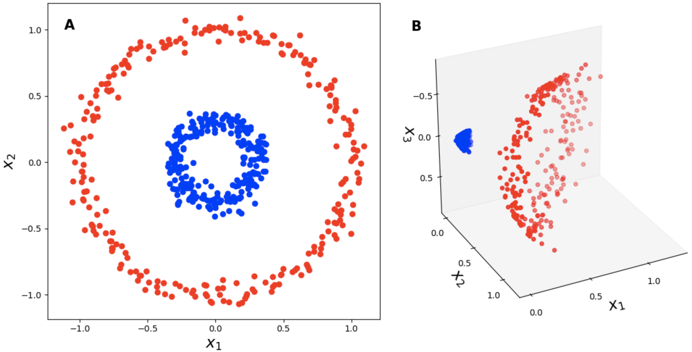


— Bishop, Pattern Recognition and Machine Learning

Recap: Basis Functions

- How is this useful? We can use linear methods on non-linear features to yield non-linear decision boundaries and regression curves.

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 x_2 \end{bmatrix}$$



— https://gregorygundersen.com/blog/2019/12/10/kernel-trick/

Kernels: Motivation

Generalized Linear Models (GLM)

- Fixed non-linear basis functions.
- Limited hypothesis space.
- Easy to optimize (convex).

Neural Network (NN)

- Adaptive non-linear basis functions.
- Rich hypothesis space.
- Hard to optimize (non-convex).

Towards Kernel Methods

- Feature space in GLM and NN needs to be explicitly constructed.
- Can we use a large (possibly infinite) set of fixed non-linear basis functions without explicitly constructing this space?
- Yes, by using kernel methods!

Kernel Methods

- Kernel methods are instance-based learners: they assign a weight θ_i to any training point $\mathbf{x}_i$.
- Predictions on new data points $\mathbf{x}'$ make use of a kernel function $\kappa(\cdot, \cdot)$ measuring the similarity of $\mathbf{x}'$ with all points $\mathbf{x}_i$ from the training set.
- Kernelized binary classification example:

$$\hat{y} = \text{sgn} \sum_{i=1}^{n} \theta_i y_i \kappa(\mathbf{x}_i, \mathbf{x}')$$

where

- $y \in \{-1, +1\}$ is the label assigned to a data point $\mathbf{x}$.
- θ_i is the weight for training example $\mathbf{x}_i$.
- $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the kernel function measuring similarity between $\mathbf{x}, \mathbf{x}' \in \mathbb{R}$.

The Kernel Trick

- Let $\phi(\cdot)$ be a set of not further specified basis functions mappings.
- Explicitly constructing a high-dimensional feature space is expensive.
- By using the kernel trick, we can implicitly perform operations in a high-dimensional feature space.
- In many algorithms, this feature space only appears as a dot product $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$ of input pairs $\mathbf{x}$, $\mathbf{x}'$.
- We define these dot products as the kernel function

$$\kappa(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$$

which can also be thought of as a similarity function between $\mathbf{x}$ and $\mathbf{x}'$.

Dual Representation

- Recall the regularized linear regression objective:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{n=1}^{N}(\boldsymbol{\theta}^\top \phi(\mathbf{x}_n) - y_n)^2 + \frac{\lambda}{2}\boldsymbol{\theta}^\top\boldsymbol{\theta}$$

- Finding optimal $\boldsymbol{\theta}$:

$$\nabla_\theta \mathcal{L}(\boldsymbol{\theta}) = \sum_{n=1}^{N}(\boldsymbol{\theta}^\top \phi(\mathbf{x}_n) - y_n)\phi(\mathbf{x}_n) + \lambda\boldsymbol{\theta} = 0$$

$$\boldsymbol{\theta} = -\frac{1}{\lambda}\sum_{n=1}^{N}\underbrace{(\boldsymbol{\theta}^\top \phi(\mathbf{x}_n) - y_n)}_{a_n}\phi(\mathbf{x}_n)$$

- The weights $\boldsymbol{\theta}$ can be written as a linear combination of the training examples:

$$\boldsymbol{\theta} = \sum_{n=1}^{N} a_n\phi(\mathbf{x}_n) \qquad \text{where } a = \left[a_1, \ldots, a_n\right] \text{ are called the dual parameters}$$

Dual Representation

- Substituting $\boldsymbol{\theta}$ back into linear regression $y(\mathbf{x}) = \boldsymbol{\theta}^\top \phi(\mathbf{x})$ yields:

$$\boldsymbol{\theta} = \sum_{n=1}^{N} a_n \phi(\mathbf{x}_n) \qquad y(\mathbf{x}) = \sum_{n=1}^{N} a_n \phi(\mathbf{x}_n)^\top \phi(\mathbf{x}) = \sum_{n=1}^{N} a_n \kappa(\mathbf{x}_n, \mathbf{x})$$

- The feature space only appears as a dot product.
- The kernel matrix, or gram matrix, $\mathbf{K} \in \mathbb{R}^{N \times N}$ collects kernel values in a symmetric positive semi-definite matrix for all data points (Mercer's theorem):

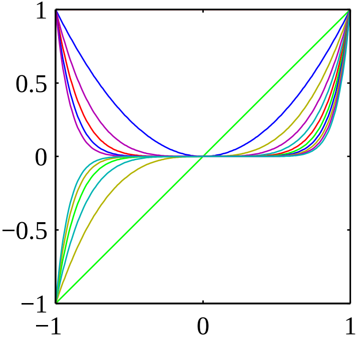
$$\mathbf{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$$

- If a kernel defines such a kernel matrix, then the kernel is valid.

Popular Kernels

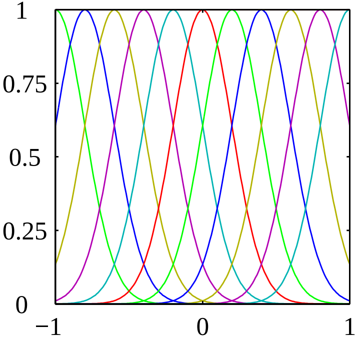
Polynomial Kernel

$$\kappa_{\mathrm{Pol}}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + c)^d$$



Squared Exponential Kernel

$$\kappa_{\mathrm{SE}}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{(\mathbf{x} - \mathbf{x}')^2}{2\ell^2}\right)$$



Kernel Composition Rules

Let $\kappa_1(\mathbf{x}, \mathbf{x}')$ and $\kappa_2(\mathbf{x}, \mathbf{x}')$ be valid kernels, then the following kernels are also valid:

- $\kappa(\mathbf{x}, \mathbf{x}') = c\kappa_1(\mathbf{x}, \mathbf{x}') \qquad \forall c > 0$
- $\kappa(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})\kappa_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}') \qquad \forall f$
- $\kappa(\mathbf{x}, \mathbf{x}') = g(\kappa_1(\mathbf{x}, \mathbf{x}'))$ $\qquad g$ is polynomial with coefficients ≥ 0.
- $\kappa(\mathbf{x}, \mathbf{x}') = \exp(\kappa_1(\mathbf{x}, \mathbf{x}'))$
- $\kappa(\mathbf{x}, \mathbf{x}') = \kappa_1(\mathbf{x}, \mathbf{x}') + \kappa_2(\mathbf{x}, \mathbf{x}')$ $\qquad$ kernel OR-ing
- $\kappa(\mathbf{x}, \mathbf{x}') = \kappa_1(\mathbf{x}, \mathbf{x}')\kappa_2(\mathbf{x}, \mathbf{x}')$ $\qquad$ kernel AND-ing
- $\kappa(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{A}\mathbf{x}'$ $\qquad$ $\mathbf{A}$ symmetric and p.s.d.

Check out the Kernel Cookbook:
`https://www.cs.toronto.edu/~duvenaud/cookbook/`

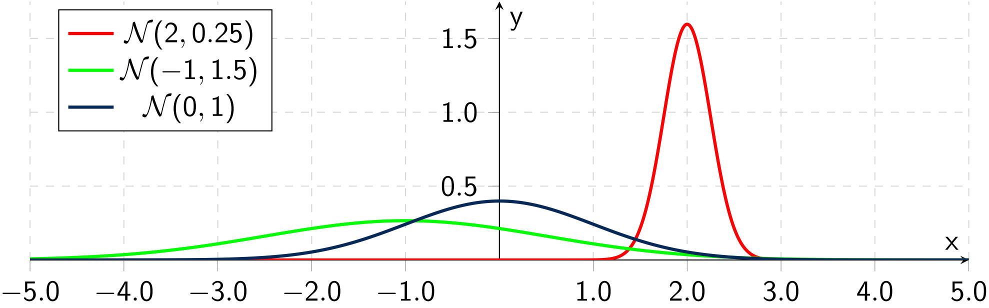
Gaussian Processes

Recap: Multivariate Gaussian

- Handy tool for Bayesian inference on real-valued variables
- General multivariate PDF:

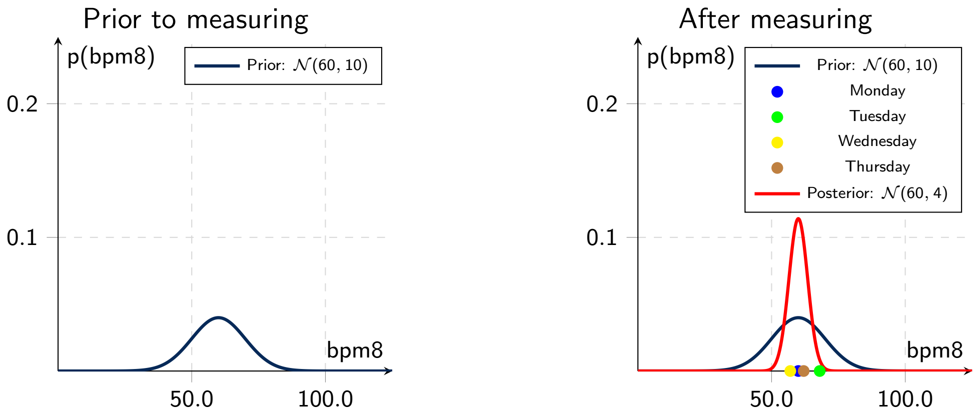
$$\mathbf{x} \sim \mathcal{N}_D(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

- Some examples of $D = 1$ Gaussians



Bayesian Parameter Estimation Example

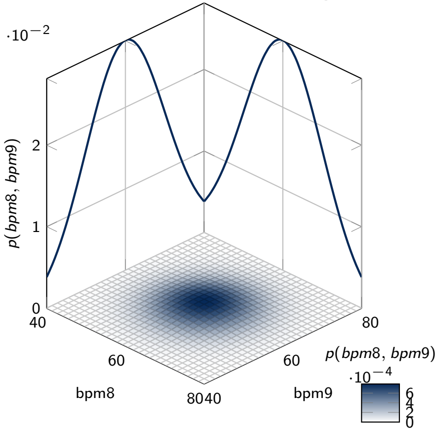
Measure your heart rate at 8am

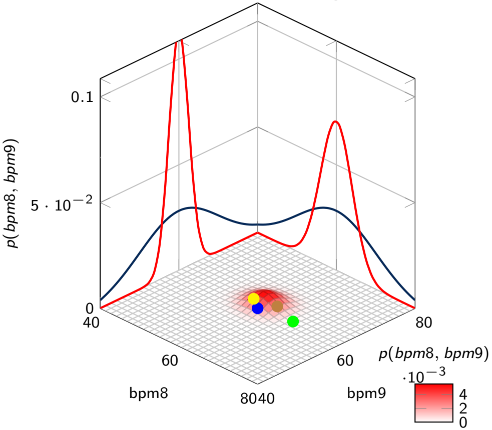


— Example from `http://videolectures.net/mlss2012_cunningham_gaussian_processes/`

Bayesian Parameter Estimation Example

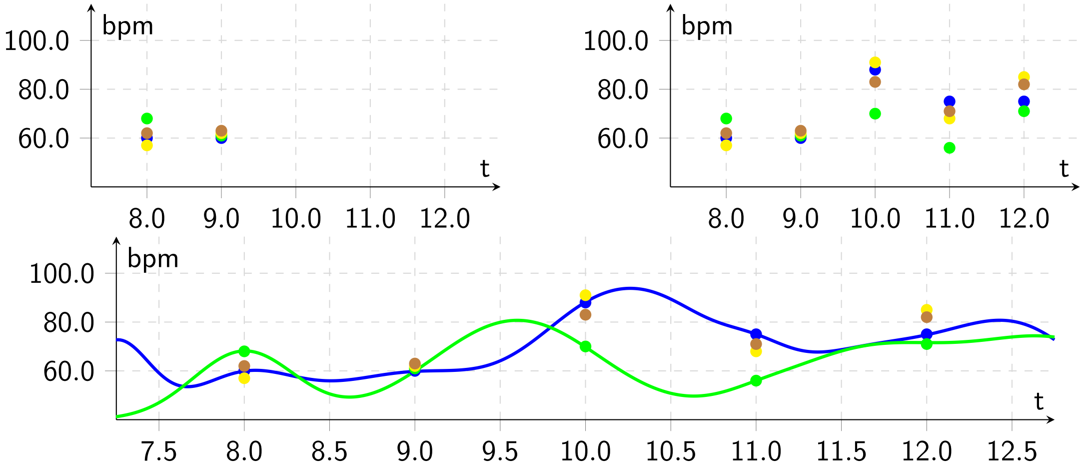
Measure your heart rate at 8am and 9am





Bayesian Parameter Estimation Example

Measuring your heart rate throughout the day



$\mathcal{GP}$ Definition

A Gaussian process describes a distribution over functions (infinitely long vectors).

- Notation: $f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), \kappa(\boldsymbol{x}, \boldsymbol{x}'))$
- Mean function: $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$
- Covariance function: $\kappa(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$

We have data points $\boldsymbol{X} = [\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_n^\top]^\top$ and are interested in their function values $f(\boldsymbol{X}) = (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n))^\top$.

A Gaussian process is a collection of random variables, any finite number of which have joint Gaussian distribution.

$f(\boldsymbol{x})$ is one such subset and has (prior) joint Gaussian distribution.

GP Mean and Covariance

The mean function m

- The mean function $m(\cdot)$ encodes the a-priori expectation of the function.
- $m(\mathbf{x})$ will dominate the inference result in case we have not yet observed data similar to $\mathbf{x}$.
- Typical choice: zero-centering the data: $m(\mathbf{x}) = 0$

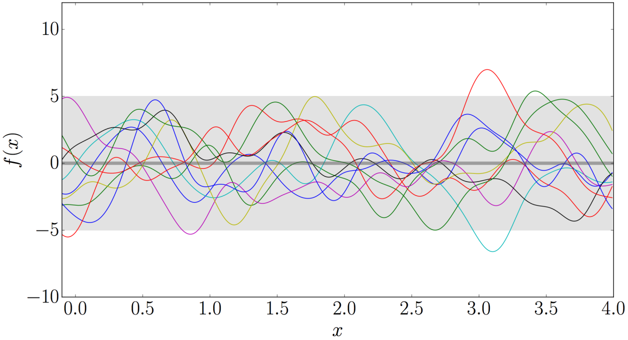
The covariance function κ

- $\kappa(\mathbf{x}, \mathbf{x}')$ measures similarity between $\mathbf{x}$ and $\mathbf{x}'$ $\rightarrow$ similar data points have similar function values.
- κ is a Mercer kernel.
- Typical choice: squared exponential kernel: $\kappa(\mathbf{x}, \mathbf{x}') = \sigma^2 e^{-\frac{(\mathbf{x}-\mathbf{x}')^\top(\mathbf{x}-\mathbf{x}')}{2\ell^2}}$ where σ defines the height and ℓ the width of the kernel.

Drawing Samples From the GP

Same procedure as for multivariate Gaussians:

1. Generate $\boldsymbol{u} \in \mathbb{R}^D$ by drawing d samples from $\mathcal{N}(\mathbf{0}, \mathbf{I}_D)$.
2. Perform Cholesky decomposition $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^\top$.
3. Compute $\boldsymbol{y} = \boldsymbol{\mu} + \boldsymbol{L}\boldsymbol{u}$ where $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.



The Joint Distribution

We have training data $\boldsymbol{X} \in \mathbb{R}^{N \times D}$, corresponding observations $\boldsymbol{y} = f(\boldsymbol{X})$, and test data points $\boldsymbol{X}_* \in \mathbb{R}^{N_* \times D}$ for which we want to infer function values $\boldsymbol{y}_* = f(\boldsymbol{X}_*)$. The GP defines the following joint distribution

$$p(\boldsymbol{y}, \boldsymbol{y}_* | \boldsymbol{X}, \boldsymbol{X}_*) = \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{y}_* \end{pmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(\boldsymbol{X}) \\ m(\boldsymbol{X}_*) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K} + \sigma_n^2 \boldsymbol{I} & \boldsymbol{K}_* \\ \boldsymbol{K}_*^\top & \boldsymbol{K}_{**} \end{bmatrix} \right)$$

where

$$\boldsymbol{K} = \kappa(\boldsymbol{X}, \boldsymbol{X}) \qquad \boldsymbol{K}_* = \kappa(\boldsymbol{X}, \boldsymbol{X}_*) \qquad \boldsymbol{K}_{**} = \kappa(\boldsymbol{X}_*, \boldsymbol{X}_*).$$

Typically, data points are corrupted by noise $\rightarrow$ our functions should not act as interpolators. We therefore assume

$$y_i = f(\boldsymbol{x}_i) + \epsilon \qquad \text{where} \qquad \epsilon \sim \mathcal{N}(0, \sigma_n^2).$$

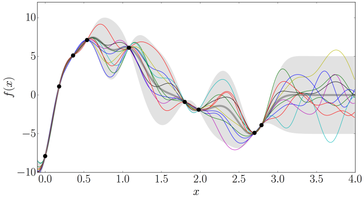
Inference with Gaussian Processes

Inferring an unknown function value and its covariance follows from conditioning multivariate Gaussians:

$$p(\boldsymbol{y}_* | \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{X}_*) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

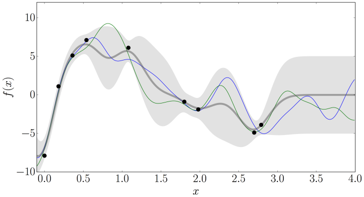
Non-noisy case

- $\boldsymbol{\mu} = m(\boldsymbol{X}_*) + \boldsymbol{K}_*^\top \boldsymbol{K}^{-1}(\boldsymbol{y} - m(\boldsymbol{X}))$
- $\boldsymbol{\Sigma} = \boldsymbol{K}_{**} - \boldsymbol{K}_*^\top \boldsymbol{K}^{-1} \boldsymbol{K}_*$



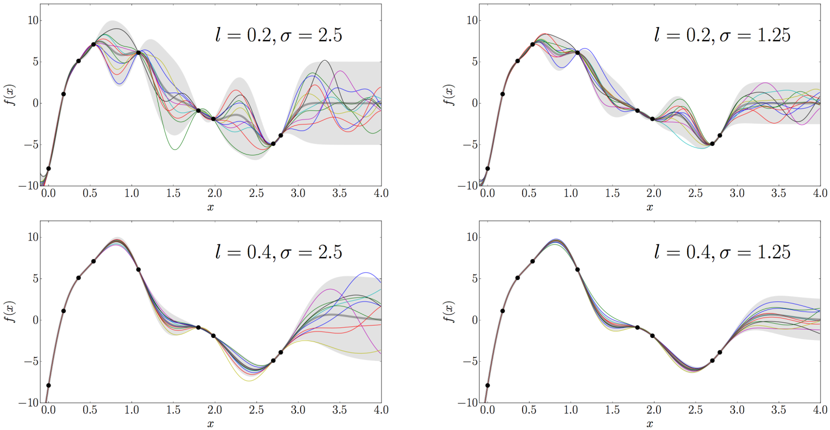
Noisy case

- $\boldsymbol{\mu} = m(\boldsymbol{X}_*) + \boldsymbol{K}_*^\top (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1}(\boldsymbol{y} - m(\boldsymbol{X}))$
- $\boldsymbol{\Sigma} = \boldsymbol{K}_{**} - \boldsymbol{K}_*^\top (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{K}_*$



Influence of Kernel Hyperparameters

$$\kappa(\mathbf{x}, \mathbf{x}') = \sigma^2 e^{-\frac{(\mathbf{x}-\mathbf{x}')^\top (\mathbf{x}-\mathbf{x}')}{2\ell^2}}$$



References I

Useful links

- https://distill.pub/2019/visual-exploration-gaussian-processes/
- https://thegradient.pub/gaussian-process-not-quite-for-dummies/
- http://www.infinitecuriosity.org/vizgp/
- https://mlg.eng.cam.ac.uk/tutorials/06/es.pdf
- https://xavierbourretsicotte.github.io/Kernel_feature_map.html
- https://cs229.stanford.edu/notes2021fall/cs229-notes3.pdf
- https://www.cs.toronto.edu/~hinton/csc2515/notes/gp_slides_fall08.pdf
- https://www.youtube.com/watch?v=nzSBvINmg28
- https://www.youtube.com/watch?v=exqpaqaPG2M

References II

Books

- Christopher M Bishop and Nasser M Nasrabadi, *Pattern recognition and machine learning*, vol. 4, Springer, 2006.
- Alex J Smola and Bernhard Schölkopf, *Learning with kernels*, vol. 4, Citeseer, 1998.
- Christopher K Williams and Carl Edward Rasmussen, *Gaussian processes for machine learning*, vol. 2, MIT press Cambridge, MA, 2006.

Backup

Connection Between GPs & Bayesian Parameter Estimation

Maximum Likelihood Estimation (MLE)

We can pick the model that maximizes the data likelihood without restrictions.

$$\arg\max_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta})$$

Maximum A-Posteriori Estimation (MAP)

We can incorporate prior information and regularize the model's prediction by introducing a prior $p(\boldsymbol{\theta})$ and reason about the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ using Bayes' rule.

$$\arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}} \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$$

Both MLE and MAP are point estimates of $\boldsymbol{\theta}$!

Connection Between GPs & Bayesian Parameter Estimation (cont'd)

Bayesian Model Averaging

Use the predictions of all potential models and weight each model's predictions by the posterior. This gives rise to Bayesian Linear Regression / Bayesian Neural Networks.

$$p(y|\boldsymbol{x}, \mathcal{D}) = \int_{\boldsymbol{\theta}} p(y|\boldsymbol{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta} = \int_{\boldsymbol{\theta}} p(y|\boldsymbol{x}, \boldsymbol{\theta}) \frac{p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta})}{\int_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}} d\boldsymbol{\theta}$$

Gaussian Process (GP)

Under the assumption that both the prior distribution $p(\boldsymbol{\theta})$ and the likelihood $p(\mathcal{D}|\boldsymbol{\theta})$ are Gaussian, then the posterior predictive distribution $p(y|\boldsymbol{x}, \mathcal{D})$ is also Gaussian. In this case, we can model the predictive distribution directly (i.e., non-parametrically) without explicitly performing model averaging.

$$p(y|\boldsymbol{x}, \mathcal{D}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \qquad y = f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), \kappa(\boldsymbol{x}, \boldsymbol{x}'))$$