Statistical Methods

Classification

(based on original lectures by Prof. Dr. N. Christlieb and Dr. Hans-G. Ludwig)

Dr. Yiannis Tsapras

ZAH – Heidelberg

Overview

- From parametric to nonparametric regression
  - From fixed functions (lines, polynomials) to flexible models without a preset shape
- Gaussian Processes (GPs): distributions over functions
  - Capture uncertainty, flexibility, and Bayesian reasoning
  - Use kernels to measure similarity between data points
- Hyperparameters
  - Model "settings" (e.g. smoothness, noise) - we'll see how to handle these
- Advantages & limitations
  - Powerful for small/medium data, but can be slow for very large sets
- Hands-on in R
  - Fit curves with uncertainty and practical concerns

From Parameters to Functions

- So far: we assumed a specific function (e.g. linear, polynomial) and estimated its parameters
- But what if we **don't know** the right functional form at all?
- New idea: treat the function itself as uncertain $\rightarrow$imagine many possible curves that could explain the data
- A **Gaussian Process** is a way to describe the probability of these possible curves
- Instead of one "best" curve, we'll get predictions **and** their uncertainty

What Are Gaussian Processes (GPs)?

- A GP is a **probability distribution over related functions**
  - think of these functions as *possible curves* that could explain the data
  - **not arbitrary functions**: they are similar in shape because their shape is determined by a **kernel** (covariance structure) – more on this later

- By definition, in a GP, any finite set of function values $\{f(x_1), \ldots, f(x_n)\}$ is **jointly Gaussian**
  - This is like extending the **multivariate Gaussian** you know to infinitely many points (i.e. infinite dimensions)
  - This is the key property that makes it a "Gaussian" process

- One GP describes **all points together**, not each point separately
  - That means: for any n inputs, $(f(x_1), \ldots, f(x_n))$ follows a multivariate normal with mean m and covariance k

- This lets us make smooth **predictions** with **quantified uncertainty** across the entire input domain

Why Use Gaussian Processes?

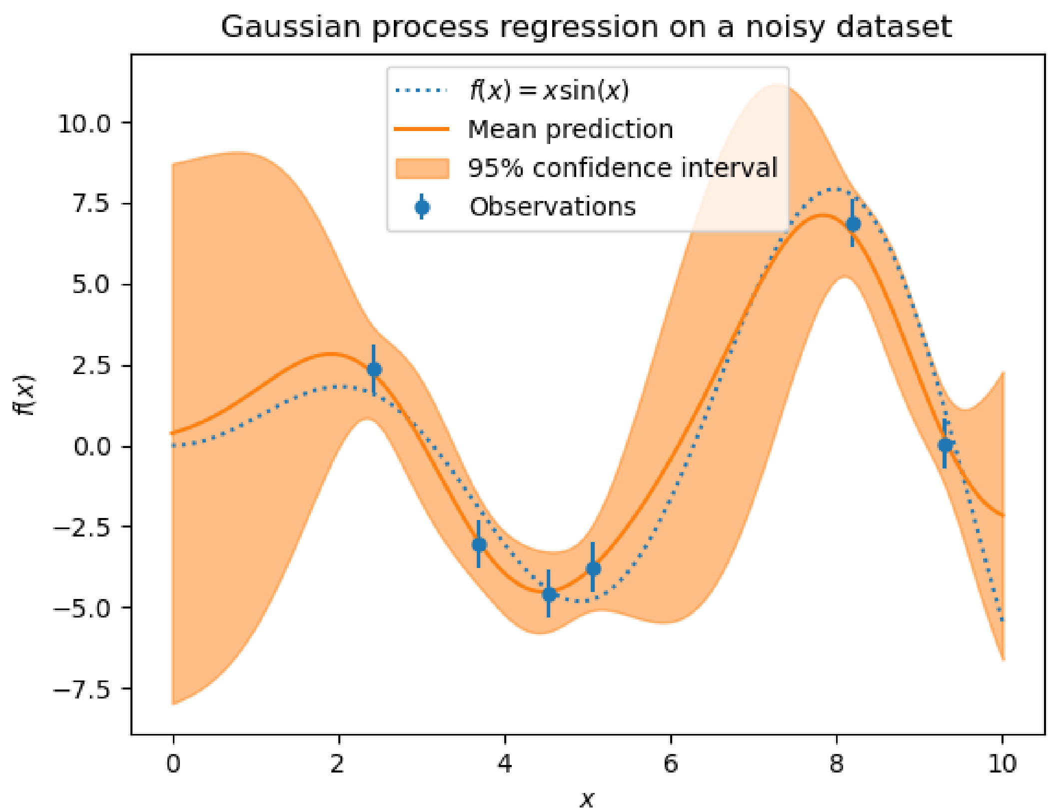
- GPs are a nonparametric supervised learning method used to solve regression and probabilistic classification problems

- **Flexibility**: GPs are **non-parametric** models (they do not assume a fixed functional form)
  - Instead, they "learn" the shape of the function/curve directly from the data
    - ⋆ we set only simple assumptions; details later
  - This flexibility allows them to capture a wide variety of patterns, from smooth trends to oscillatory behaviour

- **Built-in uncertainty quantification**: GPs don't give just point predictions
  - They return a distribution over possible functions that could fit the data
  - These come **with confidence intervals**, describing how uncertain the model is in any region.

- **Bayesian**: GPs are inherently Bayesian and interpretable
  - They **begin with a prior** over a space of functions and **update this prior** based on observed data to **form a posterior**

Fitting a curve to data with GPs



- $f(x)$ (dotted line) is the hidden function used to generate the data
  - In practice this function is unknown, we just have some data points (Blue) with uncertainties
  - Orange solid line is the GP prediction of the function shape
  - Shaded orange area shows the GPs uncertainties (95% CI)

How Do GPs Make Predictions?

- Start with a prior belief about possible functions (we'll later see this comes from a $kernel$)

- The data points act as **anchors**: they restrict the range of possible curves near the measurements

- Condition on the data: predictions near measurements are strongly influenced, far away less so

- For any new location x_* (not in the training set), the GP gives:
  - a predicted mean value $f(x_*)$
  - an uncertainty (variance) around that prediction

- The GP's output is a **distribution over functions**:
  - one mean curve (posterior mean) + uncertainty bands
    - ⋆ from this distribution, we can also sample many plausible curves

Example: 1D GP Regression in R (with uncertainty)

- Use the `mlegp` package to fit a Gaussian Process to $f(x) = x \sin x$
  - Download `day_10_example_GP_1_template.ipynb`
- Choose your own training inputs `x_train` (5-10 points).
  - Try leaving gaps between points
- Generate `y_train <- f(x_train) + rnorm(..., sd = noise_sd)`
- To fit the GP: `gp_model <- mlegp(X = matrix(x_train, ncol = 1), Z = y_train)`
- Predict on a grid: `pred <- predict.gp(gp_model, matrix(x_pred, ncol = 1), se.fit = TRUE)`
- Quick questions:
  - Move the points in `x_train` around. What do you notice?
  - What happens to your predictions beyond the training range?

What GPs Can and Cannot Do

- A GP does **not** recover the exact analytic function that generated the data
- Instead, it gives a **distribution of plausible functions** consistent with data + assumptions
- In practice, this is useful because we can:
  - interpolate between observations with quantified uncertainty
  - cautiously extrapolate nearby, with uncertainty growing as we move away
  - compare different **assumptions about function behaviour** (e.g. smooth vs. periodic) by testing which kernel explains the data better
- GPs are especially useful when data are **scarce or expensive** (e.g. costly experiments, simulations)
- They provide **calibrated uncertainty**, unlike many machine learning methods that only give point predictions
- Main limitation: **computational costs scale** as ($\mathcal{O}(n^3)$), so GPs are best for small/medium datasets

The GP Regression Recipe

- Step 1: Define a prior
  - Choose a mean function (often zero) and a **kernel**
  - The kernel specifies how **similar two input locations** x and x' are, and therefore how strongly their function values $f(x)$ and $f(x')$ are correlated
- Step 2: Add the observed data
  - The data anchor the possible functions near the measurements
- Step 3: Condition on the data
  - For any new location x_*, the GP gives a distribution for $f(x_*)$
- Step 4: Read off predictions
  - Posterior mean $\rightarrow$ best-guess curve
  - Posterior variance $\rightarrow$ uncertainty bands
- Outcome: smooth predictions with quantified uncertainty across the input domain

GP Regression: Two Key Equations

- Given training data (X, y) and a new point x_*:

$$\text{Posterior mean:} \quad \mu(x_*) = k(x_*, X)\,[K(X,X) + \sigma^2 I]^{-1} y$$

$$\text{Posterior variance:} \quad \sigma^2(x_*) = k(x_*, x_*) - k(x_*, X)\,[K(X,X) + \sigma^2 I]^{-1} k(X, x_*)$$

- $\mu(x_*)$: the GP's best guess (orange line)
  $\sigma^2(x_*)$: the GP's uncertainty (shaded band)

- You don't need to derive these, but you should know:
  - predictions are weighted averages of observed data
  - uncertainty shrinks near data, grows in gaps or far away

- **Notation:**
  - $k(x, x') =$ kernel function (covariance between inputs x and x')
  - $K(X, X) =$ covariance matrix of all training inputs
  - $I =$ identity matrix (noise added only on diagonal)

The Role of the Kernel (aka "covariance function")

- The **kernel** $k(x, x')$ is the key ingredient in a GP

- It specifies how similar two inputs x and x' are
  - $\rightarrow$ close inputs $\Rightarrow$ strongly correlated function values
  - $\rightarrow$ far inputs $\Rightarrow$ weakly correlated function values

- Different kernels **encode different assumptions** about the function's behaviour:
  - Squared Exponential (RBF): very smooth functions
  - Matérn: rougher, less smooth functions
  - Periodic: repeating patterns
  - Linear: captures global trends

- By **combining kernels** (adding or multiplying), we can model more complex structures in the data

The Kernel Zoo

- The **kernel** $k(x, x')$ controls how similar $f(x)$ and $f(x')$ are
  - close inputs $\Rightarrow$ high covariance; far inputs $\Rightarrow$ low covariance
- Common choices (1D examples; ℓ = length-scale, σ^2 = signal variance):
  - **Squared Exponential (RBF):** $k(r) = \sigma^2 \exp\big(-\frac{r^2}{2\ell^2}\big)$ (very smooth)
  - **Matérn-$\nu = 3/2$:** $k(r) = \sigma^2\big(1 + \frac{\sqrt{3}r}{\ell}\big)\exp\big(-\frac{\sqrt{3}r}{\ell}\big)$ (moderately rough)
  - **Matérn-$\nu = 5/2$:** $k(r) = \sigma^2\Big(1 + \frac{\sqrt{5}r}{\ell} + \frac{5r^2}{3\ell^2}\Big)\exp\big(-\frac{\sqrt{5}r}{\ell}\big)$
  - **Periodic:** $k(x, x') = \sigma^2 \exp\Big(-\frac{2\sin^2(\pi|x-x'|/p)}{\ell^2}\Big)$ (repeating patterns; period p)
  - **Linear:** $k(x, x') = \sigma_b^2 + \sigma_v^2\, x\, x'$ (global trend)
- Sums/products build richer structure: e.g. trend + seasonality + noise

Hands-on →`Kernel_Zoo.ipynb`

Gaussian Mixture models

- We saw that a GP is a **distribution over functions**
  - Used in regression or classification when we want to model smooth continuous functions with uncertainty

- A **Gaussian Mixture Model (GMM)** on the other hand, is a **distribution over data points**.
  - Assumes the observed data is generated from some mixture of multiple Gaussian distributions, each with its own mean and covariance
  - Typically used for **clustering analysis** and **density estimation**
  - Defined by the parameters of each component Gaussian, the Gaussian Mixture Model is:

$$p(x) = \sum_{k=1}^{K} \mathcal{N}\big(x \mid \mu_k, \Sigma_k\big)\, P_k$$

Gaussian Mixture models

- $\mathcal{N}$ is the Multivariate Gaussian density:

$$\mathcal{N}(x \mid \mu_k, \Sigma_k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\bigl(-\tfrac{1}{2}(x-\mu_k)^\top \Sigma_k^{-1}(x-\mu_k)\bigr)$$

  - $x \in \mathbb{R}^d$: observed data vector
  - K: number of Gaussian components
  - P_k: mixture weight of component k, with $P_k \geq 0$ and $\sum_{k=1}^{K} P_k = 1$
  - $\mu_k \in \mathbb{R}^d$: mean vector of component k
  - $\Sigma_k \in \mathbb{R}^{d \times d}$: covariance matrix of component k
  - $|\Sigma_k|$: determinant of the covariance matrix
  - Σ_k^{-1}: inverse of the covariance matrix
  - d: dimensionality of the data space

- Each data point is "softly" assigned to all clusters (not just one), with probabilities given by the mixture weights π_k

- GMMs give probabilistic cluster membership

Problem set-up for Gaussian mixture model

- Given: N independent data points $\boldsymbol{x}_n$ in M-dimensional space
  - Typically, M is small (e.g., 2-3 dimensions, such as RV and B-V color)
- Fitting Problem: Identify K multivariate Gaussian distributions that best describe the distribution of data points
  - Note: K (the number of Gaussian distributions) must be fixed in advance
  - The means and covariances of these Gaussian distributions are initially unknown
- Unsupervised Learning: It is not known beforehand which of the N data points belong to which of the K distributions
- Goal: Determine the N conditional probabilities $p_{nk} \equiv P(k|n)$ that point n belongs to distribution k
  - The matrix p_{nk} is known as the **responsibility matrix** (sometimes referred to as the mixing matrix)
- This responsibility matrix helps in determining how the data points are distributed among the K Gaussian distributions

Gaussian mixture model

- Things to estimate in GMM ...
  - $\vec{\mu}_k$: The mean vectors (centers) of the K multivariate Gaussians
  - $\mathbf{\Sigma}_k$: The K $M \times M$ covariance matrices of the Gaussians
  - The responsibility matrix $P(k|n)$: Probability that data point n belongs to Gaussian k

- The objective is to maximize the likelihood of the observed data:

$$\mathcal{L} = \prod_{n=1}^{N} P(\boldsymbol{x}_n)$$

Gaussian mixture model

- According to the law of total probability, the probability of each data point $P(\boldsymbol{x}_n)$ can be written as a sum over the K Gaussians:

$$P(\boldsymbol{x}_n) = \sum_k N(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)P(k)$$

  - Here, $N(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is the probability density function of a multivariate Gaussian distribution with mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$

- Typically, the EM (Expectation-Maximization) algorithm is used to maximize this likelihood

- The mixture weights p_{nk} can be computed as:

$$p_{nk} \equiv P(k|n) = \frac{N(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)P(k)}{P(\boldsymbol{x}_n)}$$

- This equation provides a recipe for calculating the likelihood $\mathcal{L}$ and the mixture weights p_{nk} given the data $\boldsymbol{x}_n$

Gaussian mixture model

$$p_{nk} \equiv P(k|n) = \frac{N(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)P(k)}{P(\boldsymbol{x}_n)}$$

- Problem: maximize $\mathcal{L}$ by varying the parameters $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, and $P(k)$
  (In a recent paper Hogg et al. worked with $N \approx 20\,000$, $M = 11$, $K = 256$)

- EM algorithm suprisingly simple and robust iterative procedure to estimate all the above parameters
  ($\rightarrow$ *Numerical Recipes* for more details of the method)

- However, there are two important considerations:
  - one must decide on the number of Gaussians K beforehand
  - as a non-linear maximization problem, the result may depend on the starting values chosen

Toying with Gaussian mixture models

- Exercise: This exercise is designed to give you hands-on experience with Gaussian mixture models and clustering analysis using real-world data
  - Step 1: Download the file `stars.dat` and the plotting routine `EMcluster.R`
  - Step 2: Load the `Mclust{mclust}` function in R
  - Step 3: Apply the `Mclust` function to the `stars.dat` dataset to explore clustering
- Explore a different dataset:
  - In R, explore available standard datasets using `library(help="datasets")`
  - Try applying `Mclust` to one of these datasets or search online for another dataset of interest
  - Select a dataset where you have some physical or contextual understanding to help interpret the results (does the grouping mean anything?)
- Interpreting Results:
  - Look at the number of clusters identified by the model and compare them with your expectations
  - Check summary statistics and plots to assess clustering quality