Gaussian Processes for Learning Nonlinear Functions

Piyush Rai

Topics in Probabilistic Modeling and Inference (CS698X)

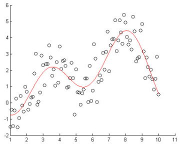
Feb 1, 2018

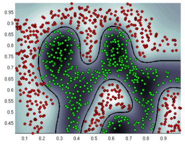
Linear Models

- Consider the problem of learning to map an input $\boldsymbol{x} \in \mathbb{R}^D$ to an output y
- Linear models use a weighted combination of input features (i.e., $\boldsymbol{w}^\top \boldsymbol{x}$) to generate y

$$\begin{aligned} p(y|\boldsymbol{w}, \boldsymbol{x}) &= \mathcal{N}(y|\boldsymbol{w}^\top \boldsymbol{x}, \beta^{-1}) && \text{(Linear Regression)} \\ p(y|\boldsymbol{w}, \boldsymbol{x}) &= [\sigma(\boldsymbol{w}^\top \boldsymbol{x})]^y [1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x})]^{1-y} && \text{(Logistic Regression)} \end{aligned}$$

- The weights $\boldsymbol{w}$ can be learned using MLE, MAP, or fully Bayesian inference
- However, linear models have limited expressive power. Unable to learn highly nonlinear patterns.


Nonlinear Regression


Nonlinear Classification

Learning Nonlinear Functions using Gaussian Process

- Assuming linear relationship between inputs and outputs, we had

$$\begin{aligned} p(y|\boldsymbol{w}, \boldsymbol{x}) &= \mathcal{N}(y|\boldsymbol{w}^\top \boldsymbol{x}, \beta^{-1}) \\ p(y|\boldsymbol{w}, \boldsymbol{x}) &= [\sigma(\boldsymbol{w}^\top \boldsymbol{x})]^y [1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x})]^{1-y} \end{aligned}$$

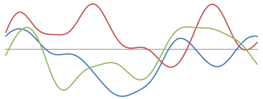
- Assume the input to output relationship to be modeled by a nonlinear function f

$$\begin{aligned} p(y|\boldsymbol{w}, \boldsymbol{x}) &= \mathcal{N}(y|f(\boldsymbol{x}), \beta^{-1}) \\ p(y|\boldsymbol{w}, \boldsymbol{x}) &= [\sigma(f(\boldsymbol{x}))]^y [1 - \sigma(f(\boldsymbol{x}))]^{1-y} \end{aligned}$$

- How can we define such a function f?
- How is f represented mathematically?
- Gaussian Process (GP) provides an answer to these questions.

What is Gaussian Process?

- A Gaussian Process, denoted as $\mathcal{GP}(\mu, \kappa)$, defines a distribution over functions
  - The GP is defined by mean function μ and covariance function κ
- Draw from a $\mathcal{GP}(\mu, \kappa)$ will give us a random function f (imagine it as an infinite dim. vector)



- Mean function μ models the "average" function f from $\mathcal{GP}(\mu, \kappa)$

$$\mu(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})]$$

- Cov. function κ models "shape/smoothness" of these functions
  - $\kappa(.,.)$ is a function that computes similarity between two inputs (just like a kernel function)
  - Note: $\kappa(.,.)$ needs to be positive definite (just like kernel functions)
- Can even learn μ and especially κ (makes GP very flexible to model, possibly nonlinear, functions)
- GP can therefore be used as a flexible prior distribution over functions

Gaussian Process Prior

- f is said to be drawn from a $\mathcal{GP}(\mu, \kappa)$ if its finite dim. version is the following joint Gaussian

$$\begin{bmatrix} f(\boldsymbol{x}_1) \\ f(\boldsymbol{x}_2) \\ \vdots \\ f(\boldsymbol{x}_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\boldsymbol{x}_1) \\ \mu(\boldsymbol{x}_2) \\ \vdots \\ \mu(\boldsymbol{x}_N) \end{bmatrix}, \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix} \right)$$

- The above means that f's output at any finite set of inputs is jointly Gaussian
- Also makes intuitive sense: If $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$ is large, we would expect $f(\boldsymbol{x}_n)$ and $f(\boldsymbol{x}_m)$ be the close
- We can also write the above more compactly as $\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$ where

$$\mathbf{f} = \begin{bmatrix} f(\boldsymbol{x}_1) \\ f(\boldsymbol{x}_2) \\ \vdots \\ f(\boldsymbol{x}_N) \end{bmatrix}, \boldsymbol{\mu} = \begin{bmatrix} \mu(\boldsymbol{x}_1) \\ \mu(\boldsymbol{x}_2) \\ \vdots \\ \mu(\boldsymbol{x}_N) \end{bmatrix}, \mathbf{K} = \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix}$$

  Note: $\mathbf{K}$ is also called the kernel matrix. $K_{nm} = \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m)$
- Note that $p(\mathbf{f}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$ can be seen as the finite-dimensional version of the GP prior over f
- If the mean function is zero, we will have $p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \mathbf{K})$

Connection with Linear Regression

- Let's first consider the (probabilistic) linear regression model

$$
\begin{aligned}
p(\boldsymbol{w}) &= \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu_0}, \Sigma_0) && \text{(Prior)} \\
p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) &= \mathcal{N}(\mathbf{X}\boldsymbol{w}, \beta^{-1}\mathbf{I}_N) && \text{(Likelihood w.r.t. } N \text{ obs.)} \\
p(\boldsymbol{y}|\mathbf{X}) &= \int p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})p(\boldsymbol{w})d\boldsymbol{w} = \mathcal{N}(\mathbf{X}\boldsymbol{\mu_0}, \beta^{-1}\mathbf{I}_N + \mathbf{X}\boldsymbol{\Sigma_0}\mathbf{X}^\top) && \text{(Marginal likelihood)} \\
p(\boldsymbol{y}|\mathbf{X}) &= \mathcal{N}(\mathbf{0}, \beta^{-1}\mathbf{I}_N + \mathbf{X}\mathbf{X}^\top) && \text{(if } \boldsymbol{\mu}_0 = 0 \text{ and } \boldsymbol{\Sigma}_0 = \mathbf{I}) \\
p(\boldsymbol{y}|\mathbf{X}) &= \mathcal{N}(\mathbf{0}, \mathbf{X}\mathbf{X}^\top) && \text{(if } \beta^{-1} = \infty\text{, i.e., zero noise)}
\end{aligned}
$$

- Thus the joint marginal distr. of $\boldsymbol{y}$ conditioned on $\mathbf{X}$ is the following multivariate Gaussian

$$
\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \boldsymbol{x}_1^\top \boldsymbol{x}_1 \dots \boldsymbol{x}_1^\top \boldsymbol{x}_N \\ \boldsymbol{x}_2^\top \boldsymbol{x}_1 \dots \boldsymbol{x}_2^\top \boldsymbol{x}_N \\ \vdots \quad \ddots \quad \vdots \\ \boldsymbol{x}_N^\top \boldsymbol{x}_1 \dots \boldsymbol{x}_N^\top \boldsymbol{x}_N \end{bmatrix} \right)
$$

- Thus a linear regression model also induces a jointly Gaussian marginal distribution over the responses (with a covariance matrix that consists of Euclidean similarities between points)

Gaussian Process Regression

GP Regression

- Training data: $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$. $\boldsymbol{x}_n \in \mathbb{R}^D$, $y_n \in \mathbb{R}$
- Assume the responses to be a noisy function of the inputs

$$y_n = f(\boldsymbol{x}_n) + \epsilon_n = f_n + \epsilon_n$$

- Assume a zero-mean Gaussian noise: $\epsilon_n \sim \mathcal{N}(\epsilon_n | 0, \sigma^2)$
- This implies the following likelihood model: $p(y_n | f_n) = \mathcal{N}(y_n | f_n, \sigma^2)$
- Denote $\boldsymbol{f} = [f_1, \ldots, f_N]$ and $\boldsymbol{y} = [y_1, \ldots, y_N]$. For i.i.d. responses, the joint likelihood will be

$$p(\boldsymbol{y}|\mathbf{f}) = \mathcal{N}(\boldsymbol{y}|\mathbf{f}, \sigma^2 \mathbf{I}_N)$$

- We now need a prior on the function f that enables us to model a nonlinear f
- Let's choose zero mean Gaussian Process prior $\mathcal{GP}(0, \kappa)$ on f, which is equivalent to

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

where $K_{nm} = \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m)$. For now, assume κ is a known function with fixed hyperparameters.

GP Regression

- The likelihood model: $p(\boldsymbol{y}|\mathbf{f}) = \mathcal{N}(\boldsymbol{y}|\mathbf{f}, \sigma^2\mathbf{I}_N)$. The prior distribution: $p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$
- The posterior $p(\mathbf{f}|\boldsymbol{y}) \propto p(\mathbf{f})p(\boldsymbol{y}|\mathbf{f})$, which will be another Gaussian (Exercise: Find its expression)
- What's the posterior predictive $p(y_*|\boldsymbol{x}_*, \boldsymbol{y}, \mathbf{X})$ or $p(y_*|\boldsymbol{y})$ (skipping $\mathbf{X}, \boldsymbol{x}_*$ from the notation)?

$$p(y_*|\boldsymbol{y}) = \int p(y_*|f_*)p(f_*|\boldsymbol{y})df_*$$

  where $p(f_*|\boldsymbol{y}) = \int p(f_*|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{y})d\boldsymbol{f}$ and note that $p(f_*|\boldsymbol{f})$ must be Gaussian for GP
- For this case (GP regression), we actually don't need to compute $p(y_*|\boldsymbol{y})$ using the above method
- Reason: The marginal distribution of the training data responses $\boldsymbol{y}$

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

- Using the same result, also note that the marginal distribution of y_* too must be

$$p(y_*) = \mathcal{N}(y_*|0, \kappa(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma^2)$$

GP Regression: Making Predictions

- Let's consider the joint distr. of N training responses $\boldsymbol{y}$ and test response y_*

$$p\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \mathbf{C}_{N+1}\right)$$

where the $(N+1) \times (N+1)$ matrix $\mathbf{C}_{N+1}$ is given by

$$\mathbf{C}_{N+1} = \begin{bmatrix} \mathbf{C}_N & \mathbf{k}_* \\ \mathbf{k}_*^\top & c \end{bmatrix}$$

and $\mathbf{k}_* = [\kappa(\boldsymbol{x}_*, \boldsymbol{x}_1), \ldots, \kappa(\boldsymbol{x}_*, \boldsymbol{x}_N)]^\top$, $c = \kappa(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma^2$

- The desired predictive posterior will be (using conditional from joint property of Gaussian)

$$\begin{aligned} p(y_*|\boldsymbol{y}) &= \mathcal{N}(y_*|\mu_*, \sigma_*^2) \\ \mu_* &= \mathbf{k}_*^\top \mathbf{C}_N^{-1} \boldsymbol{y} \\ \sigma_*^2 &= \kappa(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma^2 - \mathbf{k}_*^\top \mathbf{C}_N^{-1} \mathbf{k}_* \end{aligned}$$

GP Regression: Interpreting GP Predictions

- Let's look at the predictions made by GP regression

$$\begin{aligned} p(y_*|\boldsymbol{y}) &= \mathcal{N}(y_*|\mu_*, \sigma_*^2) \\ \mu_* &= \mathbf{k}_*{}^\top \mathbf{C}_N^{-1}\boldsymbol{y} \\ \sigma_*^2 &= k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma^2 - \mathbf{k}_*{}^\top \mathbf{C}_N^{-1}\mathbf{k}_* \end{aligned}$$

- Two interpretations for the mean prediction μ_*
  - A kernel SVM like interpretation

$$\mu_* = \mathbf{k}_*{}^\top \mathbf{C}_N^{-1}\boldsymbol{y} = \mathbf{k}_*{}^\top \boldsymbol{\alpha} = \sum_{n=1}^{N} k(\boldsymbol{x}_*, \boldsymbol{x}_n)\alpha_n$$

    where $\boldsymbol{\alpha}$ is akin to the weights of support vectors

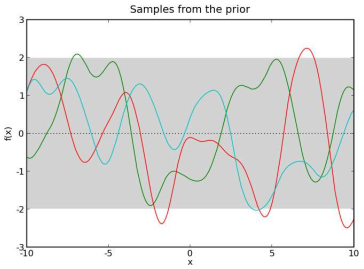
  - A nearest neighbors interpretation

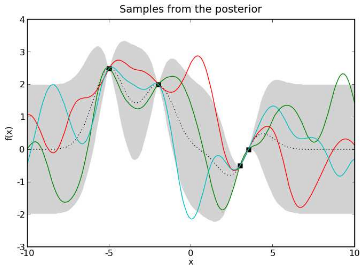
$$\mu_* = \mathbf{k}_*{}^\top \mathbf{C}_N^{-1}\boldsymbol{y} = \boldsymbol{w}^\top \boldsymbol{y} = \sum_{n=1}^{N} w_n y_n$$

    where $\boldsymbol{w}$ is akin to the weights of the neighbors

GP Regression: Pictorially

A GP with a squared-exponential kernel function





Left: Samples of f from the prior $\mathcal{GP}(0, \kappa)$, i.e, $\boldsymbol{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$

Right: Samples of f from the posterior of f after 4 observations

Picture courtesy: https://pythonhosted.org/infpy/gps.html

GP Regression: Learning Hyperparameters

- There are two hyperparameters in the GP regression model
  - Variance of the Gaussian noise σ^2
  - Assuming $\mu = 0$, the hyperparameters θ of the covariance/kernel function κ, e.g.,

$$
\begin{aligned}
\kappa(\mathbf{x}_n, \mathbf{x}_m) &= \exp\left(-\frac{||\mathbf{x}_n - \mathbf{x}_m||^2}{\gamma}\right) && \text{(RBF kernel)} \\
\kappa(\mathbf{x}_n, \mathbf{x}_m) &= \exp\left(-\sum_{d=1}^{D}\frac{(\mathbf{x}_{nd} - \mathbf{x}_{md})^2}{\gamma_d}\right) && \text{(ARD kernel)} \\
\kappa(\mathbf{x}_n, \mathbf{x}_m) &= \kappa_{\theta_1}(\mathbf{x}_n, \mathbf{x}_m) + \kappa_{\theta_2}(\mathbf{x}_n, \mathbf{x}_m) + \ldots + \kappa_{\theta_M}(\mathbf{x}_n, \mathbf{x}_m) && \text{(flexible composition of multiple kernels)}
\end{aligned}
$$

- Type-II MLE is a popular choice for learning these hyperparams, by maximizing marginal likelihood

$$p(\mathbf{y}|\sigma^2, \theta) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I}_N + \mathbf{K}_\theta)$$

- MLE-II for GP regression maximizes the log marginal likelihood w.r.t. the hyperparameters

$$\log p(\mathbf{y}|\sigma^2, \theta) = -\frac{1}{2}\log|\sigma^2\mathbf{I}_N + \mathbf{K}_\theta| - \frac{1}{2}\mathbf{y}^\top(\sigma^2\mathbf{I}_N + \mathbf{K}_\theta)^{-1}\mathbf{y} + \text{const}$$

GP Regression: Learning Hyperparameters

- The (log) marginal likelihood

$$\log p(\mathbf{y}|\sigma^2,\theta) = -\frac{1}{2}\log|\sigma^2\mathbf{I}_N + \mathbf{K}_\theta| - \frac{1}{2}\mathbf{y}^\top(\sigma^2\mathbf{I}_N + \mathbf{K}_\theta)^{-1}\mathbf{y} + \text{const}$$

- Defining $\mathbf{K}_y = \sigma^2\mathbf{I}_N + \mathbf{K}_\theta$ and taking derivative w.r.t. kernel hyperparams θ

$$\begin{aligned}\frac{\partial}{\partial\theta_j}\log p(\mathbf{y}|\sigma^2,\theta) &= -\frac{1}{2}\text{tr}\left(\mathbf{K}_y^{-1}\frac{\partial\mathbf{K}_y}{\partial\theta_j}\right) + \frac{1}{2}\mathbf{y}^\top\mathbf{K}_y^{-1}\frac{\partial\mathbf{K}_y}{\partial\theta_j}\mathbf{K}_y^{-1}\mathbf{y} \\ &= \frac{1}{2}\text{tr}\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \mathbf{K}_y^{-1})\frac{\partial\mathbf{K}_y}{\partial\theta_j}\right)\end{aligned}$$

  where θ_j is the j^{th} hyperparam. of the kernel, and $\boldsymbol{\alpha} = \mathbf{K}_y^{-1}\mathbf{y}$

- No closed form solution for θ_j. Gradient based methods can be used.
- Note: Computing $\mathbf{K}_y^{-1}$ itself takes $\mathcal{O}(N^3)$ time (faster approximations exist though). Then each gradient computation takes $\mathcal{O}(N^2)$ time
- Form of $\frac{\partial\mathbf{K}_y}{\partial\theta_j}$ depends on the covariance/kernel function κ
- Noise variance σ^2 can also be estimated likewise

GP Classification

- Now the likelihood $p(\boldsymbol{y}|\boldsymbol{f})$ will be Bernoulli: $p(y_n|f_n) = \text{Bernoulli}(\sigma(f_n))$
- The prior is still GP, therefore $p(\boldsymbol{f}) = \mathcal{N}(0, \mathbf{C}_N)$
- Posterior $p(\boldsymbol{f}|\boldsymbol{y})$ needs to be approximate due to lack of conjugacy (e.g., Laplace approx.)
- The posterior predictive $p(y_*|\boldsymbol{y})$ will be

$$p(y_*|\boldsymbol{y}) = \int p(y_*|f_*)p(f_*|\boldsymbol{y})df_*$$

  where $p(f_*|\boldsymbol{y}) = \int p(f_*|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{y})d\boldsymbol{f}$

- Due to lack of conjugacy, the posterior predictive needs to be approximated as well
- For binary classification with GP, we can use approximations used for logistic regression model