Gaussian Processes (Contd)

CS698X: Topics in Probabilistic Modeling and Inference

Piyush Rai

Recap: Gaussian Process

- A Gaussian Process (GP) defines a distribution over functions and is denoted as

$$\mathcal{GP}(\mu(.), \kappa(.,.))$$

- Assume training data with N input-output pairs

$$(x_1, f(x_1)), (x_2, f(x_2)), (x_3, f(x_3)), \ldots\ (x_N, f(x_N))$$

- Assuming f has a GP prior $\mathcal{GP}(\mu(.), \kappa(.,.))$

Can concisely write it as
$p(\mathbf{f}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$

$$p\left(\begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_N) \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \mu(x_1) \\ \mu(x_2) \\ \vdots \\ \mu(x_N) \end{bmatrix}, \begin{bmatrix} \kappa(x_1, x_1) & \cdots & \kappa(x_1, x_N) \\ \kappa(x_2, x_1) & & \kappa(x_2, x_N) \\ \vdots & \ddots & \vdots \\ \kappa(x_N, x_1) & \cdots & \kappa(x_N, x_N) \end{bmatrix}\right)$$

- Assuming $\mu(.) = \mathbf{0}$, the prediction $f_* = f(x_*)$ for a test input x_* follows

$$p(f_* | \boldsymbol{f}) = \mathcal{N}(\mathbf{k}_*^\top \mathbf{K}^{-1} \boldsymbol{f}, \kappa(x_*, x_*) - \mathbf{k}_*^\top \mathbf{K}^{-1} \mathbf{k}_*) = \mathcal{N}(\mu_*, \sigma_*^2)$$

- The mean of this predictive distribution: $\mu_* = \sum_{i=1}^N \beta_i f_i = \sum_{i=1}^N \alpha_i \kappa(x_i, x_*)$
- The above setting is "noiseless": Output is simply $f(x_n)$ with f modeled by a GP

GP for Noisy Setting

- Assume training data $(\boldsymbol{X}, \boldsymbol{y}) = \{(x_1, y_1), (x_2, y_2), (x_3, y_n), \dots\ (x_N, y_N)\}$
- Assume each y_n is obtained from $f_n = f(x_n)$ via an appropriate likelihood model, e.g.

$$p(y_n|f_n) = \mathcal{N}(y_n|f_n, \beta^{-1})$$
$$p(y_n|f_n) = \text{Bernoulli}(y_n|\sigma(f_n))$$
$$p(y_n|f_n) = \text{ExpFam}(y_n|f_n)$$

E.g., in the regression case, we can think of y_n as $y_n = f_n + \epsilon_n$ with $\epsilon_n \sim \mathcal{N}(0, \beta^{-1})$ being Gaussian noise

"Noise" simply refers to the fact that we are using a probability distribution to generate y_n and the probability distribution depends on a GP based "score" f_n

$N \times 1$ vector of the noisy outputs y_n's in training data

$N \times 1$ vector of the "scores" f_n's, each modeled by the GP

Note that in linear models, this score is simply $\boldsymbol{w}^\top \boldsymbol{x}_n$

- Now we have a likelihood model $p(\boldsymbol{y}|\boldsymbol{f})$ for these "noisy" outputs
- Our "prior" over the function f is still a GP and is given by $p(\boldsymbol{f}) = \mathcal{N}(\boldsymbol{f}|\boldsymbol{\mu}, \mathbf{K})$
- IMP: Prior $p(\boldsymbol{f})$ depends on training inputs $\boldsymbol{X}$ (via $\mathbf{K}$) but not on outputs $\boldsymbol{y}$
- We can now combine the prior and likelihood to compute
  - GP posterior $p(\boldsymbol{f}|\boldsymbol{y})$, marginal likelihood $p(\boldsymbol{y})$, PPD $p(y_*|\boldsymbol{y})$, etc
  - Note: For Gaussian lik. based regression, PPD can be computed without computing $p(\boldsymbol{f}|\boldsymbol{y})$

It is almost similar to the noiseless case (will see shortly)

GP for Noisy Setting: Regression (Gaussian Lik.)

- Assume each response modeled by a Gaussian likelihood $p(y_n|f_n) = \mathcal{N}(y_n|f_n, \beta^{-1})$
- Denoting $\boldsymbol{y} = [y_1, y_2, \dots, y_N]$ and $\boldsymbol{f} = [f_1, f_2, \dots, f_N]$ and i.i.d. responses

$$p(\boldsymbol{y}|\boldsymbol{f}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{f}, \beta^{-1}\mathbf{I})$$

$p(\boldsymbol{f}|\boldsymbol{y})$ for general likelihoods however will not be a Gaussian

Exercise: Derive by substituting the prior and likelihood expressions

- Assume a zero-mean GP prior $p(\boldsymbol{f}) = \mathcal{N}(\boldsymbol{f}|\mathbf{0}, \mathbf{K})$
- For Gaussian likelihood, the posterior $p(\boldsymbol{f}|\boldsymbol{y}) \propto p(\boldsymbol{f})\, p(\boldsymbol{y}|\boldsymbol{f})$ will be another Gaussian
- Posterior predictive $p(y_*|\boldsymbol{x}_*, \boldsymbol{y}, \boldsymbol{X})$ or $p(y_*|\, \boldsymbol{y})$ (skipping $\boldsymbol{X}$, $\boldsymbol{x}_*$ from the notation)

Note: This PPD result is general and holds for all likelihoods, not just Gaussian

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

and known hyperparams of the likelihood and GP prior

It's form will depend on the likelihood model

Always a Gaussian for GP

GP posterior (may or may (not be a Gaussian)

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

- For Gaussian likelihood, PPD can be computed without using this general method
  - The form will be almost identical to the noiseless case (we will see shortly)

GP for Noisy Setting: Regression (Gaussian Lik.)

- For Gaussian lik, we can get PPD $p(y_*|\boldsymbol{y})$ without computing the GP posterior $p(\boldsymbol{f}|\boldsymbol{y})$
- Note that, in this case, the marginal likelihood is also a Gaussian

A useful quantity when learning hyperparams of the GP covariance/kernel

$$p(\boldsymbol{y}) = \int p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f})d\boldsymbol{f} = \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \mathbf{K} + \beta^{-1}\mathbf{I}_N) = \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \mathbf{C}_\mathrm{N})$$

- The joint distribution of the training $\boldsymbol{y}$ and test response y_* is also a Gaussian

Note: All hyperparams assumed to be known

$$p\left(\begin{bmatrix}\boldsymbol{y}\\ y_*\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{y}\\ y_*\end{bmatrix} \middle| \begin{bmatrix}\mathbf{0}\\ 0\end{bmatrix}, \begin{bmatrix}\mathbf{C}_\mathrm{N} & \mathbf{k}_*\\ \mathbf{k}_*^\top & \kappa(x_*, x_*) + \beta^{-1}\end{bmatrix}\right)$$

Identical to the noiseless case except the additional β^{-1} term on the diagonal

- Using the above, we can easily obtain $p(y_*|\boldsymbol{y})$ using Gaussian properties

Weighted average of the training responses

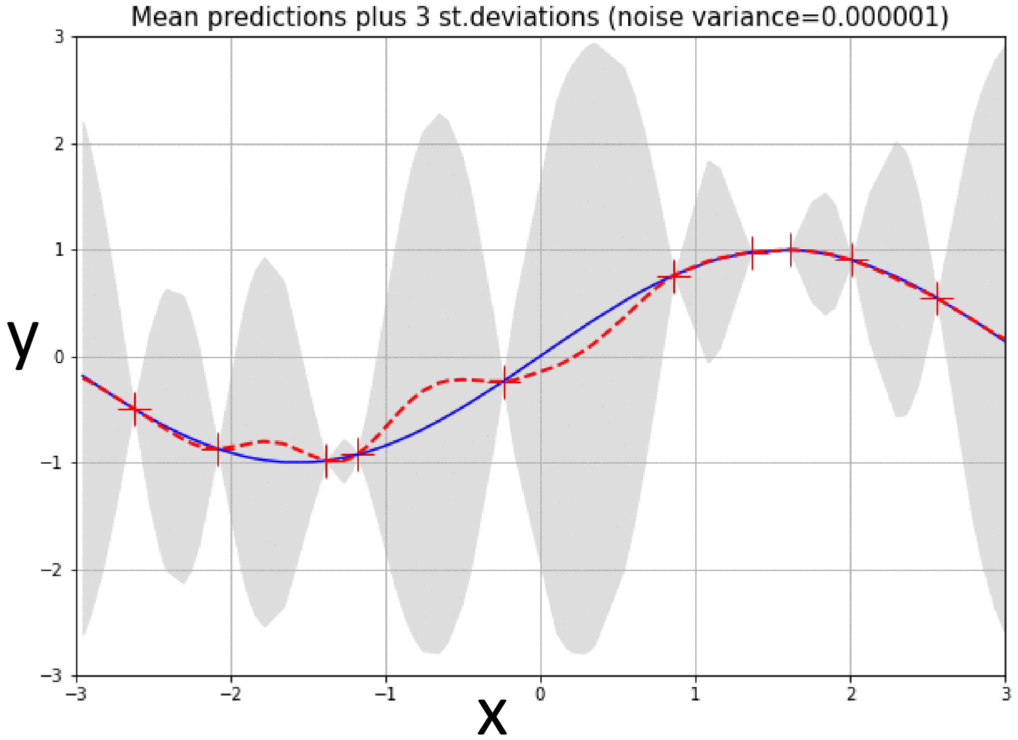
$$p(y_*|\boldsymbol{y}) = \mathcal{N}(\mathbf{k}_*^\top \mathbf{C}_N^{-1}\boldsymbol{y}, \kappa(x_*, x_*) - \mathbf{k}_*^\top \mathbf{C}_N^{-1}\mathbf{k}_* + \beta^{-1}) = \mathcal{N}(\mu_*, \sigma_*^2)$$

μ_* has a similar interpretation as in the noiseless case

- This is almost identical to the expression of $p(f_*|\boldsymbol{f})$ from the noiseless case except $\mathbf{K}$ there is replaced by $\mathbf{C}_\mathrm{N}$ and extra β^{-1} term in variance

GP Regression: An Illustration

- The figure below shows GP predictive mean and variance as noise variance changes



Blue curve: True function
Red point: Training inputs (noisy)
Red curve: Learned predictive mean
Shaded region: +/- 3 std-dev

- As expected, the predictive mean worsens and predictive variance increases as the noise variance increases

Figure courtesy: https://sandipandey.wixsite.com/simplydatascience/post/gaussian-process-regression-with-python

GP for Noisy Setting: Classification and GLM

- Binary classification: Now likelihood will be Bernoulli: $p(y_n|f_n) = \text{Bernoulli}(y_n|\sigma(f_n))$
- For multi-class $(K > 2)$ GP, $p(y_n|f_n)$ will be multinoulli and $\boldsymbol{f}_n$ will be a $K \times 1$ vector
- For GP based GLM, $p(y_n|f_n)$ will be some exp-family distribution
- The prior $p(\boldsymbol{f})$ will still be a GP. Assuming a zero-mean GP prior $p(\boldsymbol{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0},\mathbf{K})$
- The posterior predictive $p(y_*|\boldsymbol{y})$ can again be written as

$$
\begin{aligned}
p(y_*|\boldsymbol{y}) &= \int p(y_*|f_*)p(f_*|\boldsymbol{y})df_* \\
&= \int p(y_*|f_*)p(f_*|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{y})d\boldsymbol{f}df_*
\end{aligned}
$$

- This in general is not as easy to compute unlike the case of GP regression we saw
  - $p(f_*|\boldsymbol{f})$ is still not a problem (will be Gaussian due to the GP property)
  - GP posterior $p(\boldsymbol{f}|\boldsymbol{y}) \propto p(\boldsymbol{f})p(\boldsymbol{y}|\boldsymbol{f})$ will require approximation (Laplace, MCMC, variational, etc)
  - The overall integral will require approximation as well

Learning Hyperparameters in GP based Models

- Can learn the hyperparameters of the GP prior as well as of the likelihood model
- Assuming $\boldsymbol{\mu} = \mathbf{0}$, the hyperparams of GP are cov/kernel function hyperparams

$$\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\frac{||\boldsymbol{x}_n - \boldsymbol{x}_m||^2}{\gamma}\right) \quad \text{(RBF kernel)}$$

$$\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\sum_{d=1}^{D}\frac{(x_{nd} - x_{md})^2}{\gamma_d}\right) \quad \text{(ARD kernel)}$$

$$\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) = \kappa_{\theta_1}(\boldsymbol{x}_n, \boldsymbol{x}_m) + \kappa_{\theta_2}(\boldsymbol{x}_n, \boldsymbol{x}_m) + \ldots + \kappa_{\theta_M}(\boldsymbol{x}_n, \boldsymbol{x}_m) \quad \text{(flexible composition of multiple kernels)}$$

Can help in feature selection (irrelevant features will tend to have very large γ_d)

Different RBF kernel bandwidth γ_d for each feature

Ability to learn kernel hyperparams (without cross-valid) is another very appealing property of GP

- MLE-II is a popular choice for learning these hyperparams (otherwise MCMC, VI, etc)
- Denoting the covariance/kernel matrix as $\mathbf{K}_\theta$, for Gaussian likelihood case, the marg-lik

$$p(\boldsymbol{y}|\theta, \beta^{-1}) = \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \mathbf{K}_\theta + \beta^{-1}\mathbf{I}_N)$$

- This can be maximized to learn $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$
- For non-Gaussian likelihoods, the marg-lik itself will need to be approximated

Coming Up

- Some aspects of GPs
  - Scalability
  - Connections with neural nets
  - Some recent advances