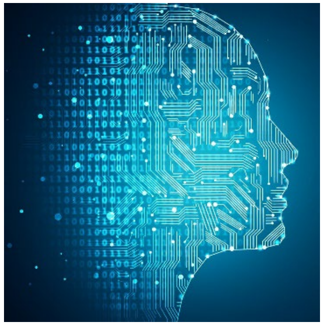
Machine Learning

Gaussian Processes

Kevin Moon (kevin.moon@usu.edu)

STAT/CS 5810/6655



Motivation

- Apply Gaussian random processes + Bayesian estimation theory to regression
- **Result**: probabilistic version of kernel ridge regression
  - Enables uncertainty quantification

- Let’s cover Bayesian estimation first

Bayesian Estimation

- Suppose we have parameters we want to estimate
  - E.g. regression coefficients or neural network parameters
- We observe $\boldsymbol{Z} \sim p(\boldsymbol{z}; \boldsymbol{\theta})$
- **Objective**: estimate $\boldsymbol{\theta}$ that best explains the observations
- In Bayesian estimation, we assume $\boldsymbol{\theta}$ is random with prior distribution $\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$
- The posterior distribution (by Bayes rule) is:

$$p(\boldsymbol{\theta}|\boldsymbol{z}) = \frac{p(\boldsymbol{z}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{z})}$$

Bayesian estimation

- The <u>posterior</u> distribution (by Bayes rule) is:

$$p(\boldsymbol{\theta}|\boldsymbol{z}) = \frac{p(\boldsymbol{z}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{z})}$$

- Bayesian parameter estimates are based on the posterior
  - Posterior mode: $\widehat{\boldsymbol{\theta}}(\boldsymbol{z}) = \arg\min_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\boldsymbol{z})$
  - Posterior mean: $\widehat{\boldsymbol{\theta}}(\boldsymbol{z}) = \mathbb{E}[\boldsymbol{\theta}|\boldsymbol{Z} = \boldsymbol{z}] = \int \boldsymbol{\theta} p(\boldsymbol{\theta}|\boldsymbol{z}) d\theta$

Random Processes

- In most Bayesian estimation problems, θ is finite dimensional
- In our case, the "parameter" is the unknown regression function f in
$$y = f(\boldsymbol{x}) + \epsilon$$
  - $\boldsymbol{x} \in \mathbb{R}^d, y \in \mathbb{R}, \epsilon \in \mathbb{R}$
- We'll use a *random process* as a prior for f
- Random process = a family of random variables indexed by a potentially uncountably infinite variable
  - $\mathbb{R}^d$ is the index in our case

Gaussian Processes

- Gaussian process = a collection of variables, where any finite number of them has a multivariate Gaussian distribution
- Gaussian process completely specified by:
  1. Mean function $m(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})]$
  2. Covariance function

$$k(\boldsymbol{x}, \boldsymbol{x}') = cov\big(f(\boldsymbol{x}), f(\boldsymbol{x}')\big)$$
$$= \mathbb{E}\Big[\big(f(\boldsymbol{x}) - m(\boldsymbol{x})\big)\big(f(\boldsymbol{x}') - m(\boldsymbol{x}')\big)\Big]$$

- Notation: $f(\boldsymbol{x}) \sim GP(m, k)$
- How could you plot a random function like this?

Gaussian Processes

- Restrictions on $k(\boldsymbol{x}, \boldsymbol{x}')$:
  1. Symmetric: $k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}', \boldsymbol{x})$
  2. Positive semi-definite. I.e. for any finite collection of points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathbb{R}^d$, the matrix formed by $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ must be PSD
- No restrictions on $m(\boldsymbol{x})$
- We'll need the *Gaussian conditioning property:*
  - If $a \in \mathbb{R}^p$ and $b \in \mathbb{R}^q$ and

$$\begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{m}_a \\ \boldsymbol{m}_b \end{bmatrix}, \begin{bmatrix} \Sigma_{\boldsymbol{aa}} & \Sigma_{\boldsymbol{ab}} \\ \Sigma_{\boldsymbol{ba}} & \Sigma_{\boldsymbol{bb}} \end{bmatrix} \right),$$

$$\Rightarrow \boldsymbol{a}|\boldsymbol{b} \sim \mathcal{N}\left(\boldsymbol{m}_a + \Sigma_{\boldsymbol{ab}} \Sigma_{\boldsymbol{bb}}^{-1} (\boldsymbol{b} - \boldsymbol{m}_b), \Sigma_{\boldsymbol{aa}} - \Sigma_{\boldsymbol{ab}} \Sigma_{\boldsymbol{bb}}^{-1} \Sigma_{\boldsymbol{ba}} \right)$$

Noise-free case

Given

- Training inputs $\boldsymbol{x}_1, \dots, \boldsymbol{x}_m$
- Training outputs $y_i = f(\boldsymbol{x}_i)$
- Test inputs $\tilde{\boldsymbol{x}}_1, \dots, \tilde{\boldsymbol{x}}_n$
- Mean function $m(\boldsymbol{x})$
- Covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$

Goal

- Predict the test outputs $f(\tilde{\boldsymbol{x}}_i)$

Noise-free case

<u>Given</u>

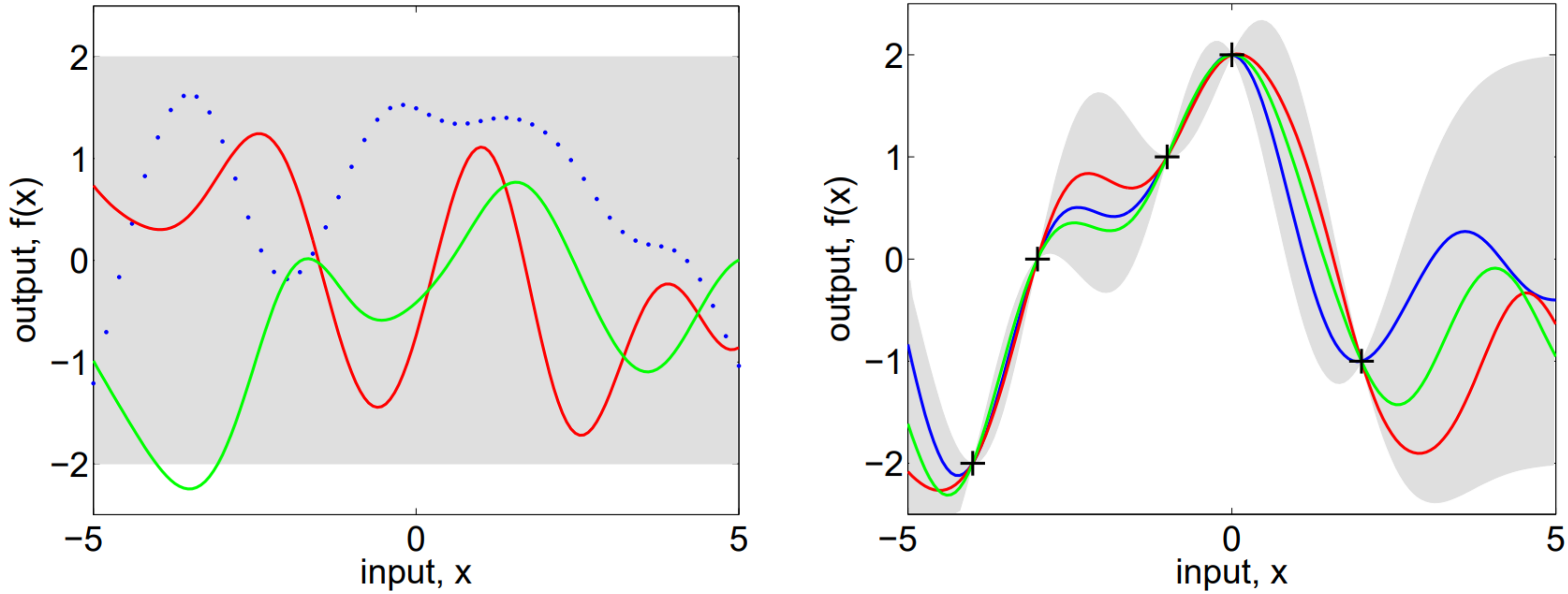
- Training inputs $\boldsymbol{x}_1, \dots, \boldsymbol{x}_m$
- Training outputs $y_i = f(\boldsymbol{x}_i)$
- Test inputs $\tilde{\boldsymbol{x}}_1, \dots, \tilde{\boldsymbol{x}}_n$
- Mean function $m(\boldsymbol{x})$ and covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$

$$\boldsymbol{f} = [f(\boldsymbol{x}_1) \quad \dots \quad f(\boldsymbol{x}_m)]^T$$
$$\tilde{\boldsymbol{f}} = [f(\tilde{\boldsymbol{x}}_1) \quad \dots \quad f(\tilde{\boldsymbol{x}}_n)]^T$$

- $\begin{bmatrix} \boldsymbol{f} \\ \tilde{\boldsymbol{f}} \end{bmatrix}$ is multivariate Gaussian distributed
- Compute the posterior distribution of $\tilde{\boldsymbol{f}}|\boldsymbol{f}$ using the Gaussian conditioning lemma

Noise free case: Example

- $m(x) = 0, k(x, x') = \sigma_g^2 \exp\left(-\frac{\|x - x'\|^2}{2\sigma_s^2}\right)$



(a), prior (b), posterior

- 3 samples from the prior and posterior with 95% CI
- What effect does σ_s^2 have on the functions?

Noisy case

Given

- Training inputs $\boldsymbol{x}_1, \dots, \boldsymbol{x}_m$
- Noisy training outputs $y_i = f(\boldsymbol{x}_i) + \epsilon_i$
- Test inputs $\tilde{\boldsymbol{x}}_1, \dots, \tilde{\boldsymbol{x}}_n$
- $m(\boldsymbol{x}), \quad k(\boldsymbol{x}, \boldsymbol{x}'), \sigma_\epsilon^2$

Assume

- $\epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$

Goal

- Predict $\tilde{f}_i = f(\tilde{\boldsymbol{x}}_i)$

Noisy case

Given

- Training inputs $\boldsymbol{x}_1, \dots, \boldsymbol{x}_m$
- Noisy training outputs $y_i = f(\boldsymbol{x}_i) + \epsilon_i$
- Test inputs $\widetilde{\boldsymbol{x}}_1, \dots, \widetilde{\boldsymbol{x}}_n$
- $m(\boldsymbol{x}),\ \ k(\boldsymbol{x}, \boldsymbol{x}'), \sigma_\epsilon^2$

- $\begin{bmatrix} \boldsymbol{y} \\ \widetilde{\boldsymbol{f}} \end{bmatrix}$ is multivariate Gaussian distributed
- Compute the posterior distribution of $\widetilde{\boldsymbol{f}} | \boldsymbol{y}$ using the Gaussian conditioning lemma

Notation

- $X = [\boldsymbol{x}_1 \quad \dots \quad \boldsymbol{x}_m] \in \mathbb{R}^{d\times m}$
- $\tilde{X} = [\tilde{\boldsymbol{x}}_1 \quad \dots \quad \tilde{\boldsymbol{x}}_n] \in \mathbb{R}^{d\times n}$
- $\boldsymbol{f} = [f(\boldsymbol{x}_1) \quad \dots \quad f(\boldsymbol{x}_m)]^T$
- $\tilde{\boldsymbol{f}} = [f(\tilde{\boldsymbol{x}}_1) \quad \dots \quad f(\tilde{\boldsymbol{x}}_n)]^T$
- $\boldsymbol{y} = [y_1 \quad \dots \quad y_m]^T$
- $\boldsymbol{m} = [m(\boldsymbol{x}_1) \quad \dots \quad m(\boldsymbol{x}_m)]^T$
- $\tilde{\boldsymbol{m}} = [m(\tilde{\boldsymbol{x}}_1) \quad \dots \quad m(\tilde{\boldsymbol{x}}_n)]^T$
- $K(X, X) = \left[k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)\right]_{i,j=1}^{m} \in \mathbb{R}^{m\times m}$
- $K\left(X, \tilde{X}\right) = \left[k\left(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_j\right)\right]_{i,j=1}^{m,n} \in \mathbb{R}^{m\times n}$
- $K\left(\tilde{X}, \tilde{X}\right) = \left[k\left(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j\right)\right]_{i,j=1}^{n} \in \mathbb{R}^{n\times n}$

Noisy case

$$\begin{bmatrix}\tilde{\boldsymbol{f}}\\ \boldsymbol{y}\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\widetilde{\boldsymbol{m}}\\ \boldsymbol{m}\end{bmatrix}, \begin{bmatrix}K(\tilde{X},\tilde{X}) & K(\tilde{X},X)\\ K(X,\tilde{X}) & K(X,X)+\sigma_\epsilon^2 I\end{bmatrix}\right)$$

By the Gaussian conditioning lemma, $\tilde{\boldsymbol{f}}|\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$

- $\boldsymbol{\mu} = \widetilde{\boldsymbol{m}} + K(X,\tilde{X})^T[K(X,X)+\sigma_\epsilon^2 I]^{-1}(\boldsymbol{y}-\boldsymbol{m})$
- $\Sigma = K(\tilde{X},\tilde{X}) - K(X,\tilde{X})^T[K(X,X)+\sigma_\epsilon^2 I]^{-1}K(X,\tilde{X})$
- Due to symmetry, the estimate of $\tilde{\boldsymbol{f}}$ is:

$$\hat{\tilde{\boldsymbol{f}}} = \widetilde{\boldsymbol{m}} + K(X,\tilde{X})^T[K(X,X)+\sigma_\epsilon^2 I]^{-1}(\boldsymbol{y}-\boldsymbol{m})$$

- This predictor is linear in $\boldsymbol{y}$ but nonlinear in $\boldsymbol{x}_i$ and $\widetilde{\boldsymbol{x}}_j$

Connection to KRR

- Assume from now on that $m(\boldsymbol{x}) = 0 \Rightarrow \boldsymbol{m}, \widetilde{\boldsymbol{m}} = \boldsymbol{0}$
- Denote $\boldsymbol{\alpha} = [K(X,X) + \sigma_\epsilon^2 I]^{-1}\boldsymbol{y}$
- Consider a single test point $\boldsymbol{x}$
- The GP prediction is

$$\hat{f}(\boldsymbol{x}) = K(X,\boldsymbol{x})\boldsymbol{\alpha} = \sum_{i=1}^{\boldsymbol{m}} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x})$$

- This is identical to the KRR estimate with $\lambda = \sigma_\epsilon^2$
- Have we gained anything?

Confidence bands

- GP predictor comes with a variance:

$$var\left(\hat{f}(\boldsymbol{x})\right)$$
$$= K(\boldsymbol{x}, \boldsymbol{x}\,) - K(X, \boldsymbol{x}\,)^T[K(X, X) + \sigma_\epsilon^2 I]^{-1} K(X, \boldsymbol{x}\,)$$

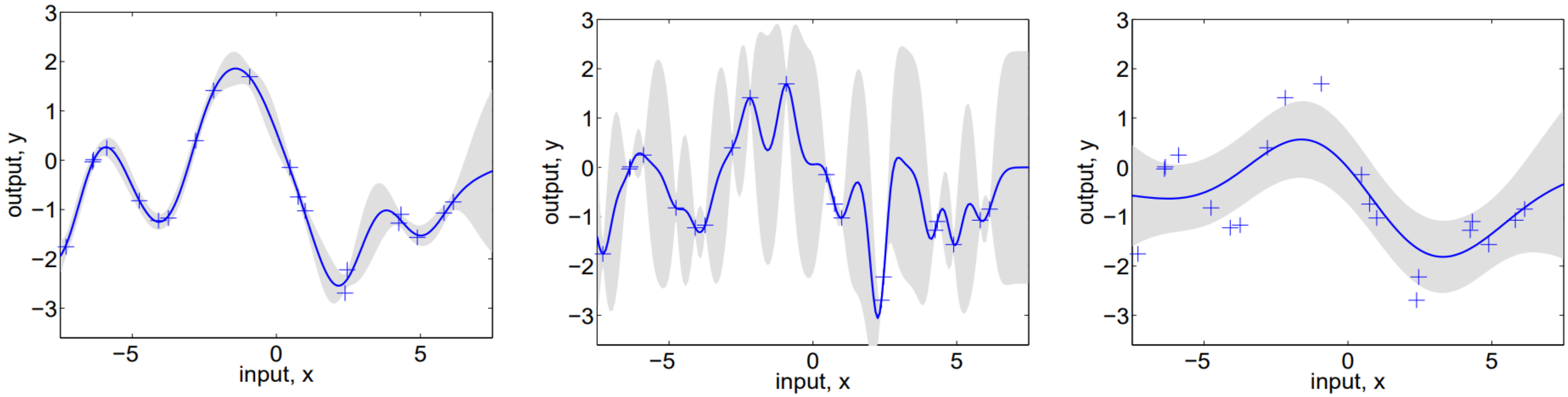
- Can place a confidence band around the estimate

Noisy case: Example

a) $\sigma_\epsilon^2 = 0.1, \sigma_s^2 = 1, \sigma_g^2 = 1$

b) $\sigma_\epsilon^2 = 0.00005, \sigma_s^2 = 0.3, \sigma_g^2 = 1.08$

c) $\sigma_\epsilon^2 = 0.89, \sigma_s^2 = 3, \sigma_g^2 = 1.16$



(a), $\ell = 1$ (b), $\ell = 0.3$ (c), $\ell = 3$

Conclusion

- Gaussian processes can be applied to obtain a Bayesian version of kernel ridge regression
- Can derive it via an alternative way: start with Bayesian linear regression, and then apply the kernel trick
  - See https://web.eecs.umich.edu/~cscott/past_courses/eecs545f16/32_gaussian_processes.pdf
- Gaussian processes can also be applied to classification

Further reading

- Rasmussen and Williams, Gaussian Processes for Machine Learning, 2006