Learning Nonlinear Functions via Gaussian Processes (2)

Piyush Rai

Probabilistic Machine Learning (CS772A)

October 24, 2017

Today's Plan

- Recap of GP and GP regression

Today's Plan

- Recap of GP and GP regression
- Hyperparameter estimation for GP based models

Today's Plan

- Recap of GP and GP regression
- Hyperparameter estimation for GP based models
- GP for problems other than regression

Today's Plan

- Recap of GP and GP regression
- Hyperparameter estimation for GP based models
- GP for problems other than regression
  - GP for nonlinear classification and nonlinear extension of GLM

Today's Plan

- Recap of GP and GP regression
- Hyperparameter estimation for GP based models
- GP for problems other than regression
  - GP for nonlinear classification and nonlinear extension of GLM
  - GP for nonlinear dimensionality reduction (Gaussian Process Latent Variable Models)

Today's Plan

- Recap of GP and GP regression
- Hyperparameter estimation for GP based models
- GP for problems other than regression
  - GP for nonlinear classification and nonlinear extension of GLM
  - GP for nonlinear dimensionality reduction (Gaussian Process Latent Variable Models)
  - Deep Gaussian Processes

Linear to Nonlinear Regression..

- For linear model $y_n = \boldsymbol{w}^\top \boldsymbol{x}_n$, marginal distribution of responses $\boldsymbol{y} = [y_1, \ldots, y_N]$

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

Linear to Nonlinear Regression..

- For linear model $y_n = \boldsymbol{w}^\top \boldsymbol{x}_n$, marginal distribution of responses $\boldsymbol{y} = [y_1, \ldots, y_N]$

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

- Replacing Euclidean similarity $\boldsymbol{x}_n^\top \boldsymbol{x}_m$ by $\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m)$ results in a nonlinear regression model

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

Linear to Nonlinear Regression..

- For linear model $y_n = \boldsymbol{w}^\top \boldsymbol{x}_n$, marginal distribution of responses $\boldsymbol{y} = [y_1, \ldots, y_N]$

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

- Replacing Euclidean similarity $\boldsymbol{x}_n^\top \boldsymbol{x}_m$ by $\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m)$ results in a nonlinear regression model

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

where the high-dim nonlinear mapping ϕ is an implicitly defined by a kernel function κ s.t.

$$\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) = \phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m)$$

Linear to Nonlinear Regression..

- For linear model $y_n = \boldsymbol{w}^\top \boldsymbol{x}_n$, marginal distribution of responses $\boldsymbol{y} = [y_1, \ldots, y_N]$

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

- Replacing Euclidean similarity $\boldsymbol{x}_n^\top \boldsymbol{x}_m$ by $\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m)$ results in a nonlinear regression model

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

  where the high-dim nonlinear mapping ϕ is an implicitly defined by a kernel function κ s.t.

$$\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) = \phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m)$$

- Note: Given the kernel function κ, we need not compute $\phi(\boldsymbol{x}_n)$ explicitly

Linear to Nonlinear Regression..

- For linear model $y_n = \boldsymbol{w}^\top \boldsymbol{x}_n$, marginal distribution of responses $\boldsymbol{y} = [y_1, \ldots, y_N]$

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

- Replacing Euclidean similarity $\boldsymbol{x}_n^\top \boldsymbol{x}_m$ by $\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m)$ results in a nonlinear regression model

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

  where the high-dim nonlinear mapping ϕ is an implicitly defined by a kernel function κ s.t.

$$\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) = \phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m)$$

- Note: Given the kernel function κ, we need not compute $\phi(\boldsymbol{x}_n)$ explicitly
- Straightforward to compute the response y_* for a new input $\boldsymbol{x}_*$ (simple Gaussian manipulations)

Gaussian Process

- A Gaussian Process, denoted as $\mathcal{GP}(\mu, \kappa)$, defines a distribution over functions

Gaussian Process

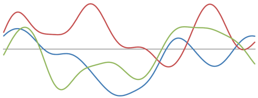
- A Gaussian Process, denoted as $\mathcal{GP}(\mu, \kappa)$, defines a distribution over functions
  - The GP is defined by mean function μ and covariance/kernel function κ

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/kernel function κ
  - If $\mu = 0$, then GP is defined solely using its covariance/kernel function

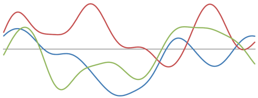
Gaussian Process

- A Gaussian Process, denoted as $\mathcal{GP}(\mu, \kappa)$, defines a distribution over functions
  - The GP is defined by mean function μ and covariance/kernel function κ
  - If $\mu = 0$, then GP is defined solely using its covariance/kernel function
- A draw from $\mathcal{GP}(\mu, \kappa)$ will give us a random function f (imagine it as an infinite dim. vector)



Gaussian Process

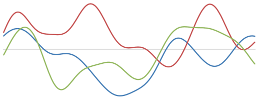
- A Gaussian Process, denoted as $\mathcal{GP}(\mu, \kappa)$, defines a distribution over functions
  - The GP is defined by mean function μ and covariance/kernel function κ
  - If $\mu = 0$, then GP is defined solely using its covariance/kernel function
- A draw from $\mathcal{GP}(\mu, \kappa)$ will give us a random function f (imagine it as an infinite dim. vector)



- GP covariance/kernel function controls the shape/smoothness of functions drawn from the GP

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/kernel function κ
  - If $\mu = 0$, then GP is defined solely using its covariance/kernel function
- A draw from $\mathcal{GP}(\mu, \kappa)$ will give us a random function f (imagine it as an infinite dim. vector)

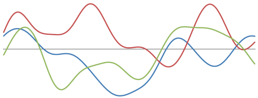


- GP covariance/kernel function controls the shape/smoothness of functions drawn from the GP
- **GP Definition:** A function f is said to be drawn from $\mathcal{GP}(\mu, \kappa)$ if

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

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/kernel function κ
  - If $\mu = 0$, then GP is defined solely using its covariance/kernel function
- A draw from $\mathcal{GP}(\mu, \kappa)$ will give us a random function f (imagine it as an infinite dim. vector)



- GP covariance/kernel function controls the shape/smoothness of functions drawn from the GP
- **GP Definition:** A function f is said to be drawn from $\mathcal{GP}(\mu, \kappa)$ if

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

.. i.e., the finite dimensional marginal of f is a multivariate Gaussian distribution

Gaussian Process

- Assume responses to be a nonlinear function of inputs (and corrupted by some Gaussian noise)

$$y_n = f(\boldsymbol{x}_n) + \epsilon_n = f_n + \epsilon_n \qquad \text{(assume } \epsilon_n \sim \mathcal{N}(\epsilon_n|0,\sigma^2)\text{)}$$

Gaussian Process

- Assume responses to be a nonlinear function of inputs (and corrupted by some Gaussian noise)

$$y_n = f(\boldsymbol{x}_n) + \epsilon_n = f_n + \epsilon_n \qquad \text{(assume } \epsilon_n \sim \mathcal{N}(\epsilon_n|0, \sigma^2))$$

- This implies the following likelihood model: $p(y_n|f_n) = \mathcal{N}(y_n|f_n, \sigma^2)$

Gaussian Process

- Assume responses to be a nonlinear function of inputs (and corrupted by some Gaussian noise)

$$y_n = f(\boldsymbol{x}_n) + \epsilon_n = f_n + \epsilon_n \qquad \text{(assume } \epsilon_n \sim \mathcal{N}(\epsilon_n|0,\sigma^2))$$

- 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 $\boldsymbol{y} = [y_1, \ldots, y_N]$.

Gaussian Process

- Assume responses to be a nonlinear function of inputs (and corrupted by some Gaussian noise)

$$y_n = f(\boldsymbol{x}_n) + \epsilon_n = f_n + \epsilon_n \qquad \text{(assume } \epsilon_n \sim \mathcal{N}(\epsilon_n|0,\sigma^2)\text{)}$$

- 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 $\boldsymbol{y} = [y_1,\ldots,y_N]$. Note $\boldsymbol{y}$ is observed, $\mathbf{f}$ is unobserved

Gaussian Process

- Assume responses to be a nonlinear function of inputs (and corrupted by some Gaussian noise)

$$y_n = f(\boldsymbol{x}_n) + \epsilon_n = f_n + \epsilon_n \qquad \text{(assume } \epsilon_n \sim \mathcal{N}(\epsilon_n|0, \sigma^2))$$

- 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 $\boldsymbol{y} = [y_1, \ldots, y_N]$. Note $\boldsymbol{y}$ is observed, $\mathbf{f}$ is unobserved
- For i.i.d. responses, the joint likelihood will be

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

Gaussian Process

- Assume responses to be a nonlinear function of inputs (and corrupted by some Gaussian noise)

$$y_n = f(\boldsymbol{x}_n) + \epsilon_n = f_n + \epsilon_n \qquad \text{(assume } \epsilon_n \sim \mathcal{N}(\epsilon_n|0, \sigma^2))$$

- 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 $\boldsymbol{y} = [y_1, \ldots, y_N]$. Note $\boldsymbol{y}$ is observed, $\mathbf{f}$ is unobserved
- 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 also need a prior on the function f

Gaussian Process

- Assume responses to be a nonlinear function of inputs (and corrupted by some Gaussian noise)

$$y_n = f(\boldsymbol{x}_n) + \epsilon_n = f_n + \epsilon_n \qquad \text{(assume } \epsilon_n \sim \mathcal{N}(\epsilon_n|0, \sigma^2))$$

- 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 $\boldsymbol{y} = [y_1, \ldots, y_N]$. Note $\boldsymbol{y}$ is observed, $\mathbf{f}$ is unobserved
- 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 also need a prior on the function f. Assuming a GP prior, i.e., $f \sim \mathcal{GP}(0, \kappa)$

Gaussian Process

- Assume responses to be a nonlinear function of inputs (and corrupted by some Gaussian noise)

$$y_n = f(\boldsymbol{x}_n) + \epsilon_n = f_n + \epsilon_n \qquad \text{(assume } \epsilon_n \sim \mathcal{N}(\epsilon_n|0, \sigma^2))$$

- 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 $\boldsymbol{y} = [y_1, \ldots, y_N]$. Note $\boldsymbol{y}$ is observed, $\mathbf{f}$ is unobserved
- 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 also need a prior on the function f. Assuming a GP prior, i.e., $f \sim \mathcal{GP}(0, \kappa)$, we'll have

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

  where $K_{nm} = \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m)$.

Gaussian Process

- Assume responses to be a nonlinear function of inputs (and corrupted by some Gaussian noise)
$$y_n = f(\boldsymbol{x}_n) + \epsilon_n = f_n + \epsilon_n \qquad \text{(assume } \epsilon_n \sim \mathcal{N}(\epsilon_n|0,\sigma^2))$$
- 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 $\boldsymbol{y} = [y_1,\ldots,y_N]$. Note $\boldsymbol{y}$ is observed, $\mathbf{f}$ is unobserved
- 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 also need a prior on the function f. Assuming a GP prior, i.e., $f \sim \mathcal{GP}(0,\kappa)$, we'll have
$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0},\mathbf{K})$$
where $K_{nm} = \kappa(\boldsymbol{x}_n,\boldsymbol{x}_m)$. For now, assume κ is a known cov. function with fixed hyperparams.

Gaussian Process

- Assume responses to be a nonlinear function of inputs (and corrupted by some Gaussian noise)
$$y_n = f(\boldsymbol{x}_n) + \epsilon_n = f_n + \epsilon_n \qquad \text{(assume } \epsilon_n \sim \mathcal{N}(\epsilon_n|0, \sigma^2)\text{)}$$
- 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 $\boldsymbol{y} = [y_1, \ldots, y_N]$. Note $\boldsymbol{y}$ is observed, $\mathbf{f}$ is unobserved
- 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 also need a prior on the function f. Assuming a GP prior, i.e., $f \sim \mathcal{GP}(0, \kappa)$, we'll have
$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$
where $K_{nm} = \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m)$. For now, assume κ is a known cov. function with fixed hyperparams.
- We can now compute things such as posterior $p(\mathbf{f}|\boldsymbol{y})$

Gaussian Process

- Assume responses to be a nonlinear function of inputs (and corrupted by some Gaussian noise)
$$y_n = f(\boldsymbol{x}_n) + \epsilon_n = f_n + \epsilon_n \qquad \text{(assume } \epsilon_n \sim \mathcal{N}(\epsilon_n|0,\sigma^2))$$
- 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 $\boldsymbol{y} = [y_1, \ldots, y_N]$. Note $\boldsymbol{y}$ is observed, $\mathbf{f}$ is unobserved
- 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 also need a prior on the function f. Assuming a GP prior, i.e., $f \sim \mathcal{GP}(0, \kappa)$, we'll have
$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$
where $K_{nm} = \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m)$. For now, assume κ is a known cov. function with fixed hyperparams.
- We can now compute things such as posterior $p(\mathbf{f}|\boldsymbol{y})$, posterior predictive $p(y_*|\boldsymbol{x}_*, \boldsymbol{y}, \mathbf{X})$

Gaussian Process

- Assume responses to be a nonlinear function of inputs (and corrupted by some Gaussian noise)
$$y_n = f(\boldsymbol{x}_n) + \epsilon_n = f_n + \epsilon_n \qquad \text{(assume } \epsilon_n \sim \mathcal{N}(\epsilon_n|0, \sigma^2))$$
- This implies the following likelihood model: $p(y_n|f_n) = \mathcal{N}(y_n|f_n, \sigma^2)$
- Denote $\boldsymbol{f} = [f_1, \dots, f_N]$ and $\boldsymbol{y} = [y_1, \dots, y_N]$. Note $\boldsymbol{y}$ is observed, $\mathbf{f}$ is unobserved
- 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 also need a prior on the function f. Assuming a GP prior, i.e., $f \sim \mathcal{GP}(0, \kappa)$, we'll have
$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$
where $K_{nm} = \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m)$. For now, assume κ is a known cov. function with fixed hyperparams.
- We can now compute things such as posterior $p(\mathbf{f}|\boldsymbol{y})$, posterior predictive $p(y_*|\boldsymbol{x}_*, \boldsymbol{y}, \mathbf{X})$, marginal likelihood $p(\boldsymbol{y}|\theta)$ where θ are model's hyperparams (e.g., of GP mean/covar function, noise var.)

Gaussian Process Inference

- General approach shown below (the practical shortcut for regression shown on next slide)
- The posterior over $\mathbf{f} = [f_1, f_2, \ldots, f_N]$ will be

$$p(\mathbf{f}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\mathbf{f})p(\mathbf{f})$$

Gaussian Process Inference

- General approach shown below (the practical shortcut for regression shown on next slide)
- The posterior over $\mathbf{f} = [f_1, f_2, \ldots, f_N]$ will be

$$p(\mathbf{f}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\mathbf{f})p(\mathbf{f})$$

- For regression, the above posterior will be Gaussian since $p(\boldsymbol{y}|\mathbf{f})$ and $p(\mathbf{f})$ are Gaussian (exercise)

Gaussian Process Inference

- General approach shown below (the practical shortcut for regression shown on next slide)
- The posterior over $\mathbf{f} = [f_1, f_2, \ldots, f_N]$ will be

$$p(\mathbf{f}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\mathbf{f})p(\mathbf{f})$$

- For regression, the above posterior will be Gaussian since $p(\boldsymbol{y}|\mathbf{f})$ and $p(\mathbf{f})$ are Gaussian (exercise)
- The posterior predictive distribution will be

$$p(y_*|\boldsymbol{x}_*, \boldsymbol{y}) = \int p(y_*|f_*)p(f_*|\boldsymbol{y})df_* \qquad (\text{where } f_* = f(\boldsymbol{x}_*))$$

- For regression $p(y_*|f_*) = \mathcal{N}(y_*|f_*, \sigma^2)$

Gaussian Process Inference

- General approach shown below (the practical shortcut for regression shown on next slide)
- The posterior over $\mathbf{f} = [f_1, f_2, \ldots, f_N]$ will be

$$p(\mathbf{f}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\mathbf{f})p(\mathbf{f})$$

- For regression, the above posterior will be Gaussian since $p(\boldsymbol{y}|\mathbf{f})$ and $p(\mathbf{f})$ are Gaussian (exercise)
- The posterior predictive distribution will be

$$p(y_*|\boldsymbol{x}_*, \boldsymbol{y}) = \int p(y_*|f_*)p(f_*|\boldsymbol{y})df_* \qquad \text{(where } f_* = f(\boldsymbol{x}_*)\text{)}$$

- For regression $p(y_*|f_*) = \mathcal{N}(y_*|f_*, \sigma^2)$. Also, $p(f_*|\boldsymbol{y}) = \int p(f_*|\mathbf{f})p(\mathbf{f}|\boldsymbol{y})d\mathbf{f}$

Gaussian Process Inference

- General approach shown below (the practical shortcut for regression shown on next slide)
- The posterior over $\mathbf{f} = [f_1, f_2, \ldots, f_N]$ will be

$$p(\mathbf{f}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\mathbf{f})p(\mathbf{f})$$

- For regression, the above posterior will be Gaussian since $p(\boldsymbol{y}|\mathbf{f})$ and $p(\mathbf{f})$ are Gaussian (exercise)
- The posterior predictive distribution will be

$$p(y_*|\boldsymbol{x}_*, \boldsymbol{y}) = \int p(y_*|f_*)p(f_*|\boldsymbol{y})df_* \qquad \text{(where } f_* = f(\boldsymbol{x}_*))$$

- For regression $p(y_*|f_*) = \mathcal{N}(y_*|f_*, \sigma^2)$. Also, $p(f_*|\boldsymbol{y}) = \int p(f_*|\mathbf{f})p(\mathbf{f}|\boldsymbol{y})d\mathbf{f}$. Since $p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$, the joint $p([\mathbf{f}, f_*])$ and conditional $p(f_*|\mathbf{f})$ will also be Gaussian

Gaussian Process Inference

- General approach shown below (the practical shortcut for regression shown on next slide)
- The posterior over $\mathbf{f} = [f_1, f_2, \ldots, f_N]$ will be

$$p(\mathbf{f}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\mathbf{f})p(\mathbf{f})$$

- For regression, the above posterior will be Gaussian since $p(\boldsymbol{y}|\mathbf{f})$ and $p(\mathbf{f})$ are Gaussian (exercise)
- The posterior predictive distribution will be

$$p(y_*|\boldsymbol{x}_*, \boldsymbol{y}) = \int p(y_*|f_*)p(f_*|\boldsymbol{y})df_* \qquad \text{(where } f_* = f(\boldsymbol{x}_*)\text{)}$$

- For regression $p(y_*|f_*) = \mathcal{N}(y_*|f_*, \sigma^2)$. Also, $p(f_*|\boldsymbol{y}) = \int p(f_*|\mathbf{f})p(\mathbf{f}|\boldsymbol{y})d\mathbf{f}$. Since $p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$, the joint $p([\mathbf{f}, f_*])$ and conditional $p(f_*|\mathbf{f})$ will also be Gaussian (always, not just for regression)

Gaussian Process Inference

- General approach shown below (the practical shortcut for regression shown on next slide)
- The posterior over $\mathbf{f} = [f_1, f_2, \ldots, f_N]$ will be

$$p(\mathbf{f}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\mathbf{f})p(\mathbf{f})$$

- For regression, the above posterior will be Gaussian since $p(\boldsymbol{y}|\mathbf{f})$ and $p(\mathbf{f})$ are Gaussian (exercise)
- The posterior predictive distribution will be

$$p(y_*|\boldsymbol{x}_*, \boldsymbol{y}) = \int p(y_*|f_*)p(f_*|\boldsymbol{y})df_* \qquad \text{(where } f_* = f(\boldsymbol{x}_*)\text{)}$$

- For regression $p(y_*|f_*) = \mathcal{N}(y_*|f_*, \sigma^2)$. Also, $p(f_*|\boldsymbol{y}) = \int p(f_*|\mathbf{f})p(\mathbf{f}|\boldsymbol{y})d\mathbf{f}$. Since $p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$, the joint $p([\mathbf{f}, f_*])$ and conditional $p(f_*|\mathbf{f})$ will also be Gaussian (always, not just for regression)
- Plugging-in $p(y_*|f_*)$ and $p(f_*|\boldsymbol{y})$, we can compute $p(y_*|\boldsymbol{x}_*, \boldsymbol{y})$, Very easy for regression case.

Gaussian Process Inference

- General approach shown below (the practical shortcut for regression shown on next slide)
- The posterior over $\mathbf{f} = [f_1, f_2, \ldots, f_N]$ will be

$$p(\mathbf{f}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\mathbf{f})p(\mathbf{f})$$

- For regression, the above posterior will be Gaussian since $p(\boldsymbol{y}|\mathbf{f})$ and $p(\mathbf{f})$ are Gaussian (exercise)
- The posterior predictive distribution will be

$$p(y_*|\boldsymbol{x}_*, \boldsymbol{y}) = \int p(y_*|f_*)p(f_*|\boldsymbol{y})df_* \qquad \text{(where } f_* = f(\boldsymbol{x}_*)\text{)}$$

- For regression $p(y_*|f_*) = \mathcal{N}(y_*|f_*, \sigma^2)$. Also, $p(f_*|\boldsymbol{y}) = \int p(f_*|\mathbf{f})p(\mathbf{f}|\boldsymbol{y})d\mathbf{f}$. Since $p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$, the joint $p([\mathbf{f}, f_*])$ and conditional $p(f_*|\mathbf{f})$ will also be Gaussian (always, not just for regression)
- Plugging-in $p(y_*|f_*)$ and $p(f_*|\boldsymbol{y})$, we can compute $p(y_*|\boldsymbol{x}_*, \boldsymbol{y})$, Very easy for regression case.
- Note: For regression case, we can actually compute the posterior predictive directly (shown next)

Gaussian Process Inference

- General approach shown below (the practical shortcut for regression shown on next slide)
- The posterior over $\mathbf{f} = [f_1, f_2, \ldots, f_N]$ will be

$$p(\mathbf{f}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\mathbf{f})p(\mathbf{f})$$

- For regression, the above posterior will be Gaussian since $p(\boldsymbol{y}|\mathbf{f})$ and $p(\mathbf{f})$ are Gaussian (exercise)
- The posterior predictive distribution will be

$$p(y_*|\boldsymbol{x}_*, \boldsymbol{y}) = \int p(y_*|f_*)p(f_*|\boldsymbol{y})df_* \qquad \text{(where } f_* = f(\boldsymbol{x}_*)\text{)}$$

- For regression $p(y_*|f_*) = \mathcal{N}(y_*|f_*, \sigma^2)$. Also, $p(f_*|\boldsymbol{y}) = \int p(f_*|\mathbf{f})p(\mathbf{f}|\boldsymbol{y})d\mathbf{f}$. Since $p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$, the joint $p([\mathbf{f}, f_*])$ and conditional $p(f_*|\mathbf{f})$ will also be Gaussian (always, not just for regression)
- Plugging-in $p(y_*|f_*)$ and $p(f_*|\boldsymbol{y})$, we can compute $p(y_*|\boldsymbol{x}_*, \boldsymbol{y})$, Very easy for regression case.
- Note: For regression case, we can actually compute the posterior predictive directly (shown next)
  - Still important to understand the above general procedure. For "hard" problems (e.g., when the above integrals are not tractable), we'd need to follow it to get the posterior, posterior predictive, etc.

Gaussian Process Inference (for Regression)

- For regression with fixed hyperparams, we don't even need to "infer" $\mathbf{f}$ (no need to find $p(\mathbf{f}|\boldsymbol{y})$)

Gaussian Process Inference (for Regression)

- For regression with fixed hyperparams, we don't even need to "infer" $\mathbf{f}$ (no need to find $p(\mathbf{f}|\boldsymbol{y})$)
- The marginal distribution of the N training responses $\boldsymbol{y} = [y_1, \ldots, y_N]$ will be

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

Gaussian Process Inference (for Regression)

- For regression with fixed hyperparams, we don't even need to "infer" $\mathbf{f}$ (no need to find $p(\mathbf{f}|\boldsymbol{y})$)
- The marginal distribution of the N training responses $\boldsymbol{y} = [y_1, \ldots, y_N]$ will be

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

- The joint distribution of $\boldsymbol{y}$ and the test input's 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

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

Gaussian Process Inference (for Regression)

- For regression with fixed hyperparams, we don't even need to "infer" $\mathbf{f}$ (no need to find $p(\mathbf{f}|\boldsymbol{y})$)
- The marginal distribution of the N training responses $\boldsymbol{y} = [y_1, \ldots, y_N]$ will be

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

- The joint distribution of $\boldsymbol{y}$ and the test input's 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 $\mathbf{C}_N = \mathbf{K} + \sigma^2\mathbf{I}_N$, $\mathbf{k}_* = [\kappa(\boldsymbol{x}_*, \boldsymbol{x}_1), \ldots, \kappa(\boldsymbol{x}_*, \boldsymbol{x}_N)]^\top$, $c = \kappa(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma^2$

Gaussian Process Inference (for Regression)

- For regression with fixed hyperparams, we don't even need to "infer" $\mathbf{f}$ (no need to find $p(\mathbf{f}|\boldsymbol{y})$)
- The marginal distribution of the N training responses $\boldsymbol{y} = [y_1, \ldots, y_N]$ will be

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

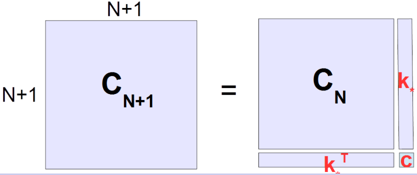
- The joint distribution of $\boldsymbol{y}$ and the test input's response y_*

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

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



- Given the jointly Gaussian distribution

$$p\left(\begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{C}_N & \mathbf{k}_* \\ \mathbf{k}_*{}^\top & c \end{bmatrix}\right)$$

Gaussian Process Inference (for Regression)

- Given the jointly Gaussian distribution

$$p\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{C}_N & \mathbf{k}_* \\ \mathbf{k}_*{}^\top & c \end{bmatrix}\right)$$

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

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

Gaussian Process Inference (for Regression)

- Given the jointly Gaussian distribution

$$p\left(\begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{C}_N & \mathbf{k}_* \\ \mathbf{k}_*^\top & c \end{bmatrix}\right)$$

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

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

- Note: Assuming $\mathbf{C}_N^{-1}$ and $\mathbf{C}_N^{-1}\mathbf{y}$ have been pre-computed, computing μ_* takes $\mathcal{O}(N)$ time and σ_*^2 takes $\mathcal{O}(N^2)$ time (scale in the number of training examples)

Gaussian Process Inference (for Regression)

- Given the jointly Gaussian distribution

$$p\left(\begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{C}_N & \mathbf{k}_* \\ \mathbf{k}_*^\top & c \end{bmatrix}\right)$$

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

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

- Note: Assuming $\mathbf{C}_N^{-1}$ and $\mathbf{C}_N^{-1}\mathbf{y}$ have been pre-computed, computing μ_* takes $\mathcal{O}(N)$ time and σ_*^2 takes $\mathcal{O}(N^2)$ time (scale in the number of training examples)
  - .. just like kernel SVM or nearest neighbor methods

Gaussian Process Inference (for Regression)

- Given the jointly Gaussian distribution

$$p\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{C}_N & \mathbf{k}_* \\ \mathbf{k}_*^\top & c \end{bmatrix}\right)$$

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

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

- Note: Assuming $\mathbf{C}_N^{-1}$ and $\mathbf{C}_N^{-1}\boldsymbol{y}$ have been pre-computed, computing μ_* takes $\mathcal{O}(N)$ time and σ_*^2 takes $\mathcal{O}(N^2)$ time (scale in the number of training examples)
  - .. just like kernel SVM or nearest neighbor methods
  - However, unlike kernel SVM or nearest neighbor methods, GP also gives us the variance σ_*^2

Gaussian Process Inference (for Regression)

- Given the jointly Gaussian distribution

$$p\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{C}_N & \mathbf{k}_* \\ \mathbf{k}_*^\top & c \end{bmatrix}\right)$$

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

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

- Note: Assuming $\mathbf{C}_N^{-1}$ and $\mathbf{C}_N^{-1}\boldsymbol{y}$ have been pre-computed, computing μ_* takes $\mathcal{O}(N)$ time and σ_*^2 takes $\mathcal{O}(N^2)$ time (scale in the number of training examples)
  - .. just like kernel SVM or nearest neighbor methods
  - However, unlike kernel SVM or nearest neighbor methods, GP also gives us the variance σ_*^2
- Note: A huge amount of work on scaling up GPs (recent methods can predict in $\mathcal{O}(1)$ time)

Gaussian Process for Regression: Interpreting the Predictions

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

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

Gaussian Process for Regression: Interpreting the 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_*

Gaussian Process for Regression: Interpreting the 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

Gaussian Process for Regression: Interpreting the 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

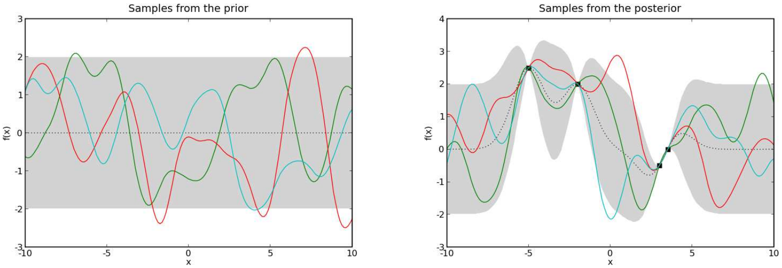
  - A nearest neighbors interpretation

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

    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 $\mathcal{GP}(0, \kappa)$
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

GP Regression: Learning Hyperparameters

- There are two hyperparameters in the GP regression model
  - Variance of the Gaussian noise σ^2

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.,

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

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.,

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

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

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

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

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

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.,

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

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

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

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

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

GP Regression: Learning Hyperparameters

- The (log) marginal likelihood

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

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 $\boldsymbol{\alpha} = \mathbf{K}_y^{-1}\mathbf{y}$

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.

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

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

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 for Problems other than Regression

- GPs are very widely applicable also to problems beyond regression

GP for Problems other than Regression

- GPs are very widely applicable also to problems beyond regression
- Supervised Learning: Binary/multiclass classification

GP for Problems other than Regression

- GPs are very widely applicable also to problems beyond regression
- Supervised Learning: Binary/multiclass classification
  - In fact, any GLM can be made nonlinear using GP

$$
\begin{aligned}
p(y_n|f_n) &= \text{GLM}(y_n|f_n) \qquad (\text{where } f_n = f(\boldsymbol{x}_n)) \\
p(\mathbf{f}) &= \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})
\end{aligned}
$$

GP for Problems other than Regression

- GPs are very widely applicable also to problems beyond regression
- Supervised Learning: Binary/multiclass classification
  - In fact, any GLM can be made nonlinear using GP

$$
\begin{aligned}
p(y_n|f_n) &= \text{GLM}(y_n|f_n) \qquad (\text{where } f_n = f(\boldsymbol{x}_n)) \\
p(\mathbf{f}) &= \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})
\end{aligned}
$$

  - For classification/GLMs, the likelihood $p(y_n|f_n)$ is not conjugate to the GP prior $p(\mathbf{f})$

GP for Problems other than Regression

- GPs are very widely applicable also to problems beyond regression
- Supervised Learning: Binary/multiclass classification
  - In fact, any GLM can be made nonlinear using GP

$$
\begin{aligned}
p(y_n|f_n) &= \text{GLM}(y_n|f_n) \qquad (\text{where } f_n = f(\boldsymbol{x}_n)) \\
p(\mathbf{f}) &= \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})
\end{aligned}
$$

  - For classification/GLMs, the likelihood $p(y_n|f_n)$ is not conjugate to the GP prior $p(\mathbf{f})$
  - Approximate inference methods such as VB, EP, MCMC, etc. need to be used

GP for Problems other than Regression

- GPs are very widely applicable also to problems beyond regression
- Supervised Learning: Binary/multiclass classification
  - In fact, any GLM can be made nonlinear using GP

$$
\begin{aligned}
p(y_n|f_n) &= \text{GLM}(y_n|f_n) \qquad (\text{where } f_n = f(\boldsymbol{x}_n)) \\
p(\mathbf{f}) &= \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})
\end{aligned}
$$

  - For classification/GLMs, the likelihood $p(y_n|f_n)$ is not conjugate to the GP prior $p(\mathbf{f})$
  - Approximate inference methods such as VB, EP, MCMC, etc. need to be used
  - A huge amount of work on such GP models

GP for Problems other than Regression

- GPs are very widely applicable also to problems beyond regression
- Supervised Learning: Binary/multiclass classification
  - In fact, any GLM can be made nonlinear using GP

$$
\begin{aligned}
p(y_n|f_n) &= \text{GLM}(y_n|f_n) \qquad (\text{where } f_n = f(\boldsymbol{x}_n)) \\
p(\mathbf{f}) &= \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})
\end{aligned}
$$

  - For classification/GLMs, the likelihood $p(y_n|f_n)$ is not conjugate to the GP prior $p(\mathbf{f})$
  - Approximate inference methods such as VB, EP, MCMC, etc. need to be used
  - A huge amount of work on such GP models
- Unsupervised Learning: Gaussian Process Latent Variable Models (for nonlinear dim-red)

$$\boldsymbol{x}_n = f(\boldsymbol{z}_n) \qquad (\text{where } f \text{ is a nonlinear mapping from latent } \boldsymbol{z}_n \text{ to observed } \boldsymbol{x}_n)$$

GP for Problems other than Regression

- GPs are very widely applicable also to problems beyond regression
- Supervised Learning: Binary/multiclass classification
  - In fact, any GLM can be made nonlinear using GP

$$
\begin{aligned}
p(y_n|f_n) &= \text{GLM}(y_n|f_n) \qquad (\text{where } f_n = f(\boldsymbol{x}_n)) \\
p(\mathbf{f}) &= \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})
\end{aligned}
$$

  - For classification/GLMs, the likelihood $p(y_n|f_n)$ is not conjugate to the GP prior $p(\mathbf{f})$
  - Approximate inference methods such as VB, EP, MCMC, etc. need to be used
  - A huge amount of work on such GP models
- Unsupervised Learning: Gaussian Process Latent Variable Models (for nonlinear dim-red)

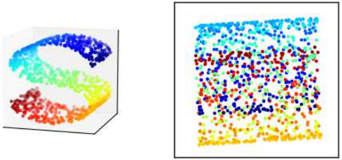
$$
\boldsymbol{x}_n = f(\boldsymbol{z}_n) \qquad (\text{where } f \text{ is a nonlinear mapping from latent } \boldsymbol{z}_n \text{ to observed } \boldsymbol{x}_n)
$$

- Designing deep learning models: Deep Gaussian Processes

GP for Unsupervised Learning (Nonlinear Dimensionality Reduction)

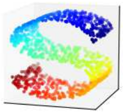
Why Nonlinear Dimensionality Reduction?

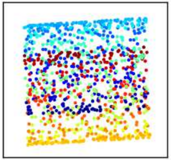
- Embeddings learned by PCA (left: original data, right: PCA)



Why Nonlinear Dimensionality Reduction?

- Embeddings learned by PCA (left: original data, right: PCA)

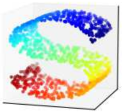


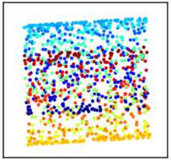


- Why PCA (or PPCA) doesn't work in such cases?

Why Nonlinear Dimensionality Reduction?

- Embeddings learned by PCA (left: original data, right: PCA)

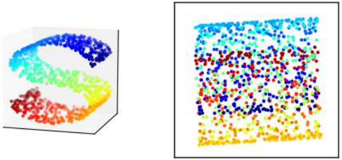




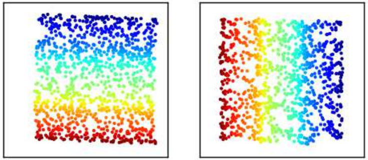
- Why PCA (or PPCA) doesn't work in such cases?
  - Uses Euclidean distances; learns linear projections which doesn't preserve nonlinear distances

Why Nonlinear Dimensionality Reduction?

- Embeddings learned by PCA (left: original data, right: PCA)



- Why PCA (or PPCA) doesn't work in such cases?
  - Uses Euclidean distances; learns linear projections which doesn't preserve nonlinear distances
- An ideal nonlinear embedding that we desire to learn for such data



Recap: Probabilistic PCA

- Given: $N \times D$ data matrix $\mathbf{X} = [\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top]^\top$, with $\boldsymbol{x}_n \in \mathbb{R}^D$
- Goal: Find a lower-dim. rep., an $N \times K$ matrix $\mathbf{Z} = [\boldsymbol{z}_1^\top, \ldots, \boldsymbol{z}_N^\top]^\top$, $\boldsymbol{z}_n \in \mathbb{R}^K$

Recap: Probabilistic PCA

- Given: $N \times D$ data matrix $\mathbf{X} = [\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top]^\top$, with $\boldsymbol{x}_n \in \mathbb{R}^D$
- Goal: Find a lower-dim. rep., an $N \times K$ matrix $\mathbf{Z} = [\boldsymbol{z}_1^\top, \ldots, \boldsymbol{z}_N^\top]^\top$, $\boldsymbol{z}_n \in \mathbb{R}^K$
- Assume the following generative model for each observation $\boldsymbol{x}_n$

$$\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n \qquad \text{with } \mathbf{W} \in \mathbb{R}^{D \times K},\ \epsilon_n \sim \mathcal{N}(0, \sigma^2)$$

Recap: Probabilistic PCA

- Given: $N \times D$ data matrix $\mathbf{X} = [\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top]^\top$, with $\boldsymbol{x}_n \in \mathbb{R}^D$
- Goal: Find a lower-dim. rep., an $N \times K$ matrix $\mathbf{Z} = [\boldsymbol{z}_1^\top, \ldots, \boldsymbol{z}_N^\top]^\top$, $\boldsymbol{z}_n \in \mathbb{R}^K$
- Assume the following generative model for each observation $\boldsymbol{x}_n$

$$\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n \qquad \text{with } \mathbf{W} \in \mathbb{R}^{D\times K},\ \epsilon_n \sim \mathcal{N}(0, \sigma^2)$$

- The conditional distribution

$$p(\boldsymbol{x}_n|\boldsymbol{z}_n, \mathbf{W}, \sigma^2) = \mathcal{N}(\mathbf{W}\boldsymbol{z}_n, \sigma^2\mathbf{I}_D)$$

Recap: Probabilistic PCA

- Given: $N \times D$ data matrix $\mathbf{X} = [\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top]^\top$, with $\boldsymbol{x}_n \in \mathbb{R}^D$
- Goal: Find a lower-dim. rep., an $N \times K$ matrix $\mathbf{Z} = [\boldsymbol{z}_1^\top, \ldots, \boldsymbol{z}_N^\top]^\top$, $\boldsymbol{z}_n \in \mathbb{R}^K$
- Assume the following generative model for each observation $\boldsymbol{x}_n$

$$\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n \qquad \text{with } \mathbf{W} \in \mathbb{R}^{D\times K},\ \epsilon_n \sim \mathcal{N}(0, \sigma^2)$$

- The conditional distribution

$$p(\boldsymbol{x}_n | \boldsymbol{z}_n, \mathbf{W}, \sigma^2) = \mathcal{N}(\mathbf{W}\boldsymbol{z}_n, \sigma^2 \mathbf{I}_D)$$

- Assume a Gaussian prior on $\boldsymbol{z}_n$: $p(\boldsymbol{z}_n) = \mathcal{N}(0, \mathbf{I}_K)$

Recap: Probabilistic PCA

- Given: $N \times D$ data matrix $\mathbf{X} = [\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top]^\top$, with $\boldsymbol{x}_n \in \mathbb{R}^D$
- Goal: Find a lower-dim. rep., an $N \times K$ matrix $\mathbf{Z} = [\boldsymbol{z}_1^\top, \ldots, \boldsymbol{z}_N^\top]^\top$, $\boldsymbol{z}_n \in \mathbb{R}^K$
- Assume the following generative model for each observation $\boldsymbol{x}_n$

$$\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n \qquad \text{with } \mathbf{W} \in \mathbb{R}^{D \times K},\ \epsilon_n \sim \mathcal{N}(0, \sigma^2)$$

- The conditional distribution

$$p(\boldsymbol{x}_n | \boldsymbol{z}_n, \mathbf{W}, \sigma^2) = \mathcal{N}(\mathbf{W}\boldsymbol{z}_n, \sigma^2 \mathbf{I}_D)$$

- Assume a Gaussian prior on $\boldsymbol{z}_n$: $p(\boldsymbol{z}_n) = \mathcal{N}(0, \mathbf{I}_K)$
- The marginal distribution of $\boldsymbol{x}_n$ (after integrating out latent variables $\boldsymbol{z}_n$)

$$\begin{aligned} p(\boldsymbol{x}_n | \mathbf{W}, \sigma^2) &= \mathcal{N}(\mathbf{0}, \mathbf{W}\mathbf{W}^\top + \sigma^2 \mathbf{I}_D) \\ p(\mathbf{X} | \mathbf{W}, \sigma^2) &= \prod_{n=1}^{N} p(\boldsymbol{x}_n | \mathbf{W}, \sigma^2) \end{aligned}$$

Recap: Probabilistic PCA

- Given: $N \times D$ data matrix $\mathbf{X} = [\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top]^\top$, with $\boldsymbol{x}_n \in \mathbb{R}^D$
- Goal: Find a lower-dim. rep., an $N \times K$ matrix $\mathbf{Z} = [\boldsymbol{z}_1^\top, \ldots, \boldsymbol{z}_N^\top]^\top$, $\boldsymbol{z}_n \in \mathbb{R}^K$
- Assume the following generative model for each observation $\boldsymbol{x}_n$

$$\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n \qquad \text{with } \mathbf{W} \in \mathbb{R}^{D\times K},\ \epsilon_n \sim \mathcal{N}(0, \sigma^2)$$

- The conditional distribution

$$p(\boldsymbol{x}_n|\boldsymbol{z}_n, \mathbf{W}, \sigma^2) = \mathcal{N}(\mathbf{W}\boldsymbol{z}_n, \sigma^2\mathbf{I}_D)$$

- Assume a Gaussian prior on $\boldsymbol{z}_n$: $p(\boldsymbol{z}_n) = \mathcal{N}(0, \mathbf{I}_K)$
- The marginal distribution of $\boldsymbol{x}_n$ (after integrating out latent variables $\boldsymbol{z}_n$)

$$\begin{aligned} p(\boldsymbol{x}_n|\mathbf{W}, \sigma^2) &= \mathcal{N}(\mathbf{0}, \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}_D) \\ p(\mathbf{X}|\mathbf{W}, \sigma^2) &= \prod_{n=1}^N p(\boldsymbol{x}_n|\mathbf{W}, \sigma^2) \end{aligned}$$

- Limitation: $\boldsymbol{z}_n$ to $\boldsymbol{x}_n$ mapping is linear (defined by $\mathbf{W}$)

Gaussian Process Latent Variable Model (GPLVM)

- Consider the same model $\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n$, $\mathbf{W} \in \mathbb{R}^{D \times K}$, $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$
- Assume a prior $p(\mathbf{W}) = \prod_{d=1}^{D} \mathcal{N}(\boldsymbol{w}_d | 0, \mathbf{I}_K)$ where $\boldsymbol{w}_d$ is the d^{th} row of $\mathbf{W}$

Gaussian Process Latent Variable Model (GPLVM)

- Consider the same model $\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n$, $\mathbf{W} \in \mathbb{R}^{D\times K}$, $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$
- Assume a prior $p(\mathbf{W}) = \prod_{d=1}^{D} \mathcal{N}(\boldsymbol{w}_d|0, \mathbf{I}_K)$ where $\boldsymbol{w}_d$ is the d^{th} row of $\mathbf{W}$
- Suppose we integrate out $\mathbf{W}$ instead of $\boldsymbol{z}_n$

$$p(\mathbf{X}|\mathbf{Z}, \sigma^2) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{X}_{:,d}|\mathbf{0}, \mathbf{Z}\mathbf{Z}^\top + \sigma^2\mathbf{I}_N) \quad \text{(product of } D\text{, } N\text{-dim Gaussians)}$$

Gaussian Process Latent Variable Model (GPLVM)

- Consider the same model $\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n$, $\mathbf{W} \in \mathbb{R}^{D \times K}$, $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$
- Assume a prior $p(\mathbf{W}) = \prod_{d=1}^{D} \mathcal{N}(\boldsymbol{w}_d | 0, \mathbf{I}_K)$ where $\boldsymbol{w}_d$ is the d^{th} row of $\mathbf{W}$
- Suppose we integrate out $\mathbf{W}$ instead of $\boldsymbol{z}_n$

$$
\begin{aligned}
p(\mathbf{X}|\mathbf{Z}, \sigma^2) &= \prod_{d=1}^{D} \mathcal{N}(\mathbf{X}_{:,d} | \mathbf{0}, \mathbf{Z}\mathbf{Z}^\top + \sigma^2 \mathbf{I}_N) \quad \text{(product of } D\text{, } N\text{-dim Gaussians)} \\
&= (2\pi)^{-DN/2} |\mathbf{K}_z|^{-D/2} \exp\left(-\frac{1}{2} \text{tr}(\mathbf{K}_z^{-1} \mathbf{X}\mathbf{X}^\top) \right) \quad \text{(verify)}
\end{aligned}
$$

where $\mathbf{K}_z = \mathbf{Z}\mathbf{Z}^\top + \sigma^2 \mathbf{I}$ and $\mathbf{X}_{:,d}$ is the d^{th} column of $N \times D$ data matrix $\mathbf{X}$

Gaussian Process Latent Variable Model (GPLVM)

- Consider the same model $\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n$, $\mathbf{W} \in \mathbb{R}^{D\times K}$, $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$
- Assume a prior $p(\mathbf{W}) = \prod_{d=1}^{D} \mathcal{N}(\boldsymbol{w}_d|0, \mathbf{I}_K)$ where $\boldsymbol{w}_d$ is the d^{th} row of $\mathbf{W}$
- Suppose we integrate out $\mathbf{W}$ instead of $\boldsymbol{z}_n$

$$
\begin{aligned}
p(\mathbf{X}|\mathbf{Z}, \sigma^2) &= \prod_{d=1}^{D} \mathcal{N}(\mathbf{X}_{:,d}|\mathbf{0}, \mathbf{Z}\mathbf{Z}^\top + \sigma^2\mathbf{I}_N) \quad \text{(product of } D\text{, } N\text{-dim Gaussians)} \\
&= (2\pi)^{-DN/2}|\mathbf{K}_z|^{-D/2} \exp\left(-\frac{1}{2}\text{tr}(\mathbf{K}_z^{-1}\mathbf{X}\mathbf{X}^\top)\right) \quad \text{(verify)}
\end{aligned}
$$

where $\mathbf{K}_z = \mathbf{Z}\mathbf{Z}^\top + \sigma^2\mathbf{I}$ and $\mathbf{X}_{:,d}$ is the d^{th} column of $N \times D$ data matrix $\mathbf{X}$

- Although expressed differently, note that this is still equivalent to PPCA

Gaussian Process Latent Variable Model (GPLVM)

- Consider the same model $\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n$, $\mathbf{W} \in \mathbb{R}^{D\times K}$, $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$
- Assume a prior $p(\mathbf{W}) = \prod_{d=1}^{D} \mathcal{N}(\boldsymbol{w}_d | 0, \mathbf{I}_K)$ where $\boldsymbol{w}_d$ is the d^{th} row of $\mathbf{W}$
- Suppose we integrate out $\mathbf{W}$ instead of $\boldsymbol{z}_n$

$$
\begin{aligned}
p(\mathbf{X}|\mathbf{Z}, \sigma^2) &= \prod_{d=1}^{D} \mathcal{N}(\mathbf{X}_{:,d} | \mathbf{0}, \mathbf{Z}\mathbf{Z}^\top + \sigma^2 \mathbf{I}_N) \quad \text{(product of } D\text{, } N\text{-dim Gaussians)} \\
&= (2\pi)^{-DN/2} |\mathbf{K}_z|^{-D/2} \exp\left(-\frac{1}{2}\text{tr}(\mathbf{K}_z^{-1}\mathbf{X}\mathbf{X}^\top)\right) \quad \text{(verify)}
\end{aligned}
$$

where $\mathbf{K}_z = \mathbf{Z}\mathbf{Z}^\top + \sigma^2\mathbf{I}$ and $\mathbf{X}_{:,d}$ is the d^{th} column of $N \times D$ data matrix $\mathbf{X}$

- Although expressed differently, note that this is still equivalent to PPCA
- What did we gain if it is still a linear model? Well, the above for can be "nonlinearized" easily. :)

Gaussian Process Latent Variable Model (GPLVM)

- Consider the same model $\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n$, $\mathbf{W} \in \mathbb{R}^{D \times K}$, $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$
- Assume a prior $p(\mathbf{W}) = \prod_{d=1}^{D} \mathcal{N}(\boldsymbol{w}_d | 0, \mathbf{I}_K)$ where $\boldsymbol{w}_d$ is the d^{th} row of $\mathbf{W}$
- Suppose we integrate out $\mathbf{W}$ instead of $\boldsymbol{z}_n$

$$
\begin{aligned}
p(\mathbf{X}|\mathbf{Z}, \sigma^2) &= \prod_{d=1}^{D} \mathcal{N}(\mathbf{X}_{:,d} | \mathbf{0}, \mathbf{Z}\mathbf{Z}^\top + \sigma^2 \mathbf{I}_N) \quad \text{(product of } D\text{, } N\text{-dim Gaussians)} \\
&= (2\pi)^{-DN/2} |\mathbf{K}_z|^{-D/2} \exp\left(-\frac{1}{2}\text{tr}(\mathbf{K}_z^{-1}\mathbf{X}\mathbf{X}^\top)\right) \quad \text{(verify)}
\end{aligned}
$$

where $\mathbf{K}_z = \mathbf{Z}\mathbf{Z}^\top + \sigma^2 \mathbf{I}$ and $\mathbf{X}_{:,d}$ is the d^{th} column of $N \times D$ data matrix $\mathbf{X}$

- Although expressed differently, note that this is still equivalent to PPCA
- What did we gain if it is still a linear model? Well, the above for can be "nonlinearized" easily. :)
  - How? By defining $\mathbf{K}_z = \mathbf{K} + \sigma^2 \mathbf{I}$ (with $\mathbf{K}$ being some appropriately defined kernel matrix over $\mathbf{Z}$)

Gaussian Process Latent Variable Model (GPLVM)

- With $\mathbf{K}_z = \mathbf{K} + \sigma^2\mathbf{I}$, we can write $p(\mathbf{X}|\mathbf{Z}, \sigma^2)$ as a product of D GPs ($\mathbf{Z}$ to $\mathbf{X}_{:,d}$, $d = 1, \ldots, D$)

$$p(\mathbf{X}|\mathbf{Z}, \sigma^2) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{X}_{:,d}|\mathbf{0}, \mathbf{K}_z)$$

Gaussian Process Latent Variable Model (GPLVM)

- With $\mathbf{K}_z = \mathbf{K} + \sigma^2\mathbf{I}$, we can write $p(\mathbf{X}|\mathbf{Z},\sigma^2)$ as a product of D GPs ($\mathbf{Z}$ to $\mathbf{X}_{:,d}$, $d = 1,\ldots,D$)

$$\begin{aligned} p(\mathbf{X}|\mathbf{Z},\sigma^2) &= \prod_{d=1}^{D} \mathcal{N}(\mathbf{X}_{:,d}|\mathbf{0},\mathbf{K}_z) \\ &= (2\pi)^{-DN/2}|\mathbf{K}_z|^{-D/2}\exp\left(-\frac{1}{2}\mathrm{tr}(\mathbf{K}_z^{-1}\mathbf{X}\mathbf{X}^\top)\right) \end{aligned}$$

Gaussian Process Latent Variable Model (GPLVM)

- With $\mathbf{K}_z = \mathbf{K} + \sigma^2\mathbf{I}$, we can write $p(\mathbf{X}|\mathbf{Z},\sigma^2)$ as a product of D GPs ($\mathbf{Z}$ to $\mathbf{X}_{:,d}$, $d = 1,\ldots,D$)

$$
\begin{aligned}
p(\mathbf{X}|\mathbf{Z},\sigma^2) &= \prod_{d=1}^{D} \mathcal{N}(\mathbf{X}_{:,d}|\mathbf{0},\mathbf{K}_z) \\
&= (2\pi)^{-DN/2}|\mathbf{K}_z|^{-D/2} \exp\left(-\frac{1}{2}\text{tr}(\mathbf{K}_z^{-1}\mathbf{X}\mathbf{X}^\top)\right)
\end{aligned}
$$

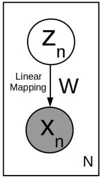
- Using $\mathbf{K}_z = \mathbf{Z}\mathbf{Z}^\top + \sigma^2\mathbf{I}$ and doing MLE will give the same solution for $\mathbf{Z}$ as probabilistic PCA (note that $\mathbf{Z}\mathbf{Z}^\top$ is a linear kernel over $\mathbf{Z}$, the low-dim rep of data)

Gaussian Process Latent Variable Model (GPLVM)

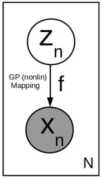
- With $\mathbf{K}_z = \mathbf{K} + \sigma^2\mathbf{I}$, we can write $p(\mathbf{X}|\mathbf{Z},\sigma^2)$ as a product of D GPs ($\mathbf{Z}$ to $\mathbf{X}_{:,d}$, $d = 1,\ldots,D$)

$$p(\mathbf{X}|\mathbf{Z},\sigma^2) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{X}_{:,d}|\mathbf{0},\mathbf{K}_z) = (2\pi)^{-DN/2}|\mathbf{K}_z|^{-D/2}\exp\left(-\frac{1}{2}\mathrm{tr}(\mathbf{K}_z^{-1}\mathbf{X}\mathbf{X}^\top)\right)$$

- Using $\mathbf{K}_z = \mathbf{Z}\mathbf{Z}^\top + \sigma^2\mathbf{I}$ and doing MLE will give the same solution for $\mathbf{Z}$ as probabilistic PCA (note that $\mathbf{Z}\mathbf{Z}^\top$ is a linear kernel over $\mathbf{Z}$, the low-dim rep of data)
- But with $\mathbf{K}_z = \mathbf{K} + \sigma^2\mathbf{I}$ (for some nonlinear covariance/kernel function) will give nonlinear dimensionality reduction



Probabilistic PCA



GPLVM

MLE for GPLVM

- Log-likelihood is given by

$$\mathcal{L} = -\frac{D}{2}\log|\mathbf{K}_z| - \frac{1}{2}\mathrm{tr}(\mathbf{K}_z^{-1}\mathbf{X}\mathbf{X}^\top)$$

where $\mathbf{K}_z = \mathbf{K} + \sigma^2\mathbf{I}$ and $\mathbf{K}$ denotes the kernel matrix of our low-dim rep. $\mathbf{Z}$

MLE for GPLVM

- Log-likelihood is given by
$$\mathcal{L} = -\frac{D}{2}\log|\mathbf{K}_z| - \frac{1}{2}\text{tr}(\mathbf{K}_z^{-1}\mathbf{X}\mathbf{X}^\top)$$
where $\mathbf{K}_z = \mathbf{K} + \sigma^2\mathbf{I}$ and $\mathbf{K}$ denotes the kernel matrix of our low-dim rep. $\mathbf{Z}$
- The goal is to estimate the $N \times K$ matrix $\mathbf{Z}$

MLE for GPLVM

- Log-likelihood is given by

$$\mathcal{L} = -\frac{D}{2}\log|\mathbf{K}_z| - \frac{1}{2}\text{tr}(\mathbf{K}_z^{-1}\mathbf{X}\mathbf{X}^\top)$$

  where $\mathbf{K}_z = \mathbf{K} + \sigma^2\mathbf{I}$ and $\mathbf{K}$ denotes the kernel matrix of our low-dim rep. $\mathbf{Z}$

- The goal is to estimate the $N \times K$ matrix $\mathbf{Z}$

- Can't find closed form estimate of $\mathbf{Z}$. Need to use gradient-based methods. The gradient will be

$$\frac{\partial \mathcal{L}}{\partial Z_{nk}} = \frac{\partial \mathcal{L}}{\partial \mathbf{K}_z}\frac{\partial \mathbf{K}_z}{\partial Z_{nk}}$$

  where $\frac{\partial \mathcal{L}}{\partial \mathbf{K}_z} = \mathbf{K}_z^{-1}\mathbf{X}\mathbf{X}^\top\mathbf{K}_z^{-1} - D\mathbf{K}_z^{-1}$ and $\frac{\partial \mathbf{K}_z}{\partial Z_{nk}}$ will depend on the kernel function used (note: hyperparameters of the kernel can also be learned just as we did it in the GP regression case)

MLE for GPLVM

- Log-likelihood is given by

$$\mathcal{L} = -\frac{D}{2}\log|\mathbf{K}_z| - \frac{1}{2}\text{tr}(\mathbf{K}_z^{-1}\mathbf{X}\mathbf{X}^\top)$$

  where $\mathbf{K}_z = \mathbf{K} + \sigma^2\mathbf{I}$ and $\mathbf{K}$ denotes the kernel matrix of our low-dim rep. $\mathbf{Z}$

- The goal is to estimate the $N \times K$ matrix $\mathbf{Z}$

- Can't find closed form estimate of $\mathbf{Z}$. Need to use gradient-based methods. The gradient will be

$$\frac{\partial \mathcal{L}}{\partial Z_{nk}} = \frac{\partial \mathcal{L}}{\partial \mathbf{K}_z}\frac{\partial \mathbf{K}_z}{\partial Z_{nk}}$$

  where $\frac{\partial \mathcal{L}}{\partial \mathbf{K}_z} = \mathbf{K}_z^{-1}\mathbf{X}\mathbf{X}^\top\mathbf{K}_z^{-1} - D\mathbf{K}_z^{-1}$ and $\frac{\partial \mathbf{K}_z}{\partial Z_{nk}}$ will depend on the kernel function used (note: hyperparameters of the kernel can also be learned just as we did it in the GP regression case)

- Can also impose a prior on $\mathbf{Z}$ and do MAP (or fully Bayesian) estimation

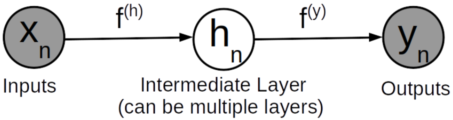
Deep Gaussian Processes

- GPs can be "stacked" to construct deep learning models (both supervised and unsupervised)

"Deep Gaussian Processes", Damianou and Lawrence (AISTATS 2013)

Deep Gaussian Processes

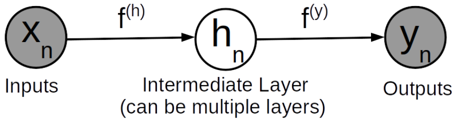
- GPs can be "stacked" to construct deep learning models (both supervised and unsupervised)
- An example of a deep supervised model (e.g., regression) using deep GP



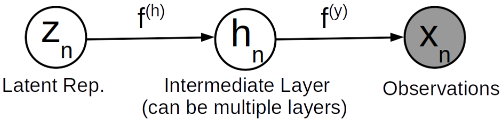
"Deep Gaussian Processes", Damianou and Lawrence (AISTATS 2013)

Deep Gaussian Processes

- GPs can be "stacked" to construct deep learning models (both supervised and unsupervised)
- An example of a deep supervised model (e.g., regression) using deep GP



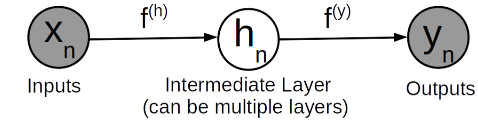
- An example of a deep unsupervised model (e.g., dim-red) using deep GP



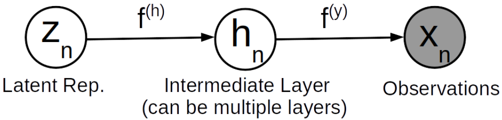
"Deep Gaussian Processes", Damianou and Lawrence (AISTATS 2013)

Deep Gaussian Processes

- GPs can be "stacked" to construct deep learning models (both supervised and unsupervised)
- An example of a deep supervised model (e.g., regression) using deep GP



- An example of a deep unsupervised model (e.g., dim-red) using deep GP



- With no intermediate layers, the above models reduce to the standard GP based models

"Deep Gaussian Processes", Damianou and Lawrence (AISTATS 2013)

Summary

- Gaussian Process is a very powerful framework nonlinear function learning
  - Applicable for both supervised and unsupervised learning

Summary

- Gaussian Process is a very powerful framework <u>nonlinear</u> function learning
  - Applicable for both supervised and unsupervised learning
- Can think of it as the Bayesian analogue of kernel methods (kernel SVM, kernel PCA, etc.)

Summary

- Gaussian Process is a very powerful framework <u>nonlinear</u> function learning
  - Applicable for both supervised and unsupervised learning
- Can think of it as the Bayesian analogue of kernel methods (kernel SVM, kernel PCA, etc.)
- Readily provides estimates of the variance in the predictions

Summary

- Gaussian Process is a very powerful framework nonlinear function learning
    - Applicable for both supervised and unsupervised learning
- Can think of it as the Bayesian analogue of kernel methods (kernel SVM, kernel PCA, etc.)
- Readily provides estimates of the variance in the predictions
- Can learn the kernel hyperparameters (big advantage)

Summary

- Gaussian Process is a very powerful framework <u>nonlinear</u> function learning
  - Applicable for both supervised and unsupervised learning
- Can think of it as the Bayesian analogue of kernel methods (kernel SVM, kernel PCA, etc.)
- Readily provides estimates of the variance in the predictions
- Can learn the kernel hyperparameters (big advantage)
- Can use kernels that can be compositions of many kernels

Summary

- Gaussian Process is a very powerful framework nonlinear function learning
  - Applicable for both supervised and unsupervised learning
- Can think of it as the Bayesian analogue of kernel methods (kernel SVM, kernel PCA, etc.)
- Readily provides estimates of the variance in the predictions
- Can learn the kernel hyperparameters (big advantage)
- Can use kernels that can be compositions of many kernels
- Many other applications - pretty much any problem that requires estimating an unknown function from (especially small amounts of) data, e.g., Bayesian Optimization, Active Learning, etc.

Some Useful Resources on Gaussian Process

- Some MATLAB/Python Packages: Useful to play with, build applications, extend existing models and inference algorithms for GPs (both regression and classification)
  - GPML: `http://www.gaussianprocess.org/gpml/code/matlab/doc/`
  - GPStuff: `http://research.cs.aalto.fi/pml/software/gpstuff/`
  - pyGPs: `https://github.com/marionmari/pyGPs`
  - GP in tensorflow: `https://github.com/GPflow/GPflow`
  - All these toolboxes allow using different types of likelihood function, different types of mean and covariance functions, and inference procedures, including hyperparameter estimation, etc.