Gaussian Processes for Learning Nonlinear Functions

Piyush Rai

Topics in Probabilistic Modeling and Inference (CS698X)

Jan 28, 2019
Announcements

- Discussion session on project topics/ideas: Tomorrow 7pm-8pm (KD-101)
Announcements

- Discussion session on project topics/ideas: Tomorrow 7pm-8pm (KD-101)
- Project proposals due on Feb 1
Announcements

- Discussion session on project topics/ideas: Tomorrow 7pm-8pm (KD-101)
- Project proposals due on Feb 1
- HW1 out now. Due on Feb 8, 11:59pm. Please start early.
Announcements

- Discussion session on project topics/ideas: Tomorrow 7pm-8pm (KD-101)
- Project proposals due on Feb 1
- HW1 out now. Due on Feb 8, 11:59pm. Please start early.
- Quiz 1 on Jan 31, 7pm-8pm (RM-101)
Recap: Bayesian Generative Classification

Recall generative classification $p(y = k | x) = \frac{p(y=k)p(x|y=k)}{\sum_{k=1}^{K} p(y=k)p(x|y=k)}$

Equation (1) is the posterior predictive distribution of test output $y^*$ given input $x^*$.

Note that we have done posterior averaging for all the parameters.

In contrast, for gen. class with MLE/MAP, $p(y^* = k | y)$ and $p(x^* | X(y))$.
Recall generative classification \( p(y = k|x) = \frac{p(y=k)p(x|y=k)}{\sum_{k=1}^{K} p(y=k)p(x|y=k)} \). Prediction rule for a test input \( \mathbf{x}_* \)

\[
p(y_* = k|\mathbf{x}_*, \mathbf{X}, \mathbf{y})
\]
Recall generative classification $p(y = k|x) = \frac{p(y=k)p(x|y=k)}{\sum_{k=1}^{K} p(y=k)p(x|y=k)}$. Prediction rule for a test input $x_*$:

$$p(y_* = k|x_*, X, y) = \frac{p(y_* = k|X, y)p(x_*|y_* = k, X, y)}{\sum_{k=1}^{K} p(y_* = k|X, y)p(x_*|y_* = k, X, y)}$$
Recap: Bayesian Generative Classification

Recall generative classification $p(y = k | x) = \frac{p(y=k)p(x|y=k)}{\sum_{k=1}^{K} p(y=k)p(x|y=k)}$. Prediction rule for a test input $x_*$

$$
p(y_* = k | x_*, X, y) = \frac{p(y_* = k | X, y)p(x_*|y_* = k, X, y)}{\sum_{k=1}^{K} p(y_* = k | X, y)p(x_*|y_* = k, X, y)}
= \frac{p(y_* = k | y)p(x_*|X^{(k)})}{\sum_{k=1}^{K} p(y_* = k | y)p(x_*|X^{(k)})}
$$

(1)

Note: $X^{(k)}$ denotes training inputs with $y = k$. Here $p(y_* = k | y) = \int p(y_* = k | \pi)p(\pi | y)d\pi$ (we did this; recall dice roll example).

Here $p(x_*|X^{(k)}) = \int p(x_*|\theta_k)p(\theta_k|X^{(k)})d\theta_k$ (post. predictive dist. of input $x_*$ under class $k$).

Eq (1) is the posterior predictive distribution of test output $y_*$ given input $x_*$. Note that we have done posterior averaging for all the parameters.

In contrast, for gen. class with MLE/MAP, $p(y_* = k | y) \approx \pi_k$ and $p(x_*|X^{(k)}) \approx p(x_*|\theta_k)$.
Recall generative classification $p(y = k| x) = \frac{p(y = k)p(x| y = k)}{\sum_{k=1}^{K} p(y = k)p(x| y = k)}$. Prediction rule for a test input $x_*$

$$p(y_* = k | x_*, X, y) = \frac{p(y_* = k | X, y)p(x_*| y_* = k, X, y)}{\sum_{k=1}^{K} p(y_* = k | X, y)p(x_*| y_* = k, X, y)}$$

$$= \frac{p(y_* = k | y)p(x_* | X^{(k)})}{\sum_{k=1}^{K} p(y_* = k | y)p(x_* | X^{(k)})}$$

Note: $X^{(k)}$ denotes training inputs with $y = k$
Recap: Bayesian Generative Classification

- Recall generative classification \( p(y = k|x) = \frac{p(y=k)p(x|y=k)}{\sum_{k=1}^{K} p(y=k)p(x|y=k)} \). Prediction rule for a test input \( x^* \):

\[
p(y^* = k|x^*, X, y) = \frac{p(y^* = k|X, y)p(x^*|y^* = k, X, y)}{\sum_{k=1}^{K} p(y^* = k|X, y)p(x^*|y^* = k, X, y)} \]

\[
= \frac{p(y^* = k|y)p(x^*|X^{(k)})}{\sum_{k=1}^{K} p(y^* = k|y)p(x^*|X^{(k)})} \tag{1}
\]

- Note: \( X^{(k)} \) denotes training inputs with \( y = k \)

- Here \( p(y^* = k|y) = \int p(y^*|\pi)p(\pi|y)d\pi \) (we did this; recall dice roll example)
Recap: Bayesian Generative Classification

- Recall generative classification $p(y = k | x) = \frac{p(y=k)p(x|y=k)}{\sum_{k=1}^{K} p(y=k)p(x|y=k)}$. Prediction rule for a test input $x_*$.

\[ p(y_* = k | x_*, X, y) = \frac{p(y_* = k | X, y)p(x_* | y_* = k, X, y)}{\sum_{k=1}^{K} p(y_* = k | X, y)p(x_* | y_* = k, X, y)} = \frac{p(y_* = k | y)p(x_* | X^{(k)})}{\sum_{k=1}^{K} p(y_* = k | y)p(x_* | X^{(k)})} \]  

(1)

- Note: $X^{(k)}$ denotes training inputs with $y = k$

- Here $p(y_* = k | y) = \int p(y_* | \pi)p(\pi | y)d\pi$ (we did this; recall dice roll example)

- Here $p(x_* | X^{(k)}) = \int p(x_* | \theta_k)p(\theta_k | X^{(k)})d\theta_k$ (post. predictive dist. of input $x_*$ under class $k$)
Recap: Bayesian Generative Classification

- Recall generative classification $p(y = k | x) = \frac{p(y=k)p(x|y=k)}{\sum_{k=1}^{K} p(y=k)p(x|y=k)}$. Prediction rule for a test input $x^*$

  $$p(y_* = k | x^*, X, y) = \frac{p(y_* = k | X, y)p(x_*|y_* = k, X, y)}{\sum_{k=1}^{K} p(y_* = k | X, y)p(x_*|y_* = k, X, y)}$$

  $$= \frac{p(y_* = k | y)p(x_*|X^{(k)})}{\sum_{k=1}^{K} p(y_* = k | y)p(x_*|X^{(k)})} \quad (1)$$

- Note: $X^{(k)}$ denotes training inputs with $y = k$

- Here $p(y_* = k | y) = \int p(y_* | \pi)p(\pi | y)d\pi$ (we did this; recall dice roll example)

- Here $p(x_*|X^{(k)}) = \int p(x_* | \theta_k)p(\theta_k | X^{(k)})d\theta_k$ (post. predictive dist. of input $x_*$ under class $k$)

- Eq (1) is the posterior predictive distribution of test output $y_*$ given input $x_*$
Recap: Bayesian Generative Classification

- Recall generative classification $p(y = k|x) = \frac{p(y=k)p(x|y=k)}{\sum_{k=1}^{K} p(y=k)p(x|y=k)}$. Prediction rule for a test input $x_*$

$$p(y_* = k|x_*, X, y) = \frac{p(y_* = k|X, y)p(x_*|y_* = k, X, y)}{\sum_{k=1}^{K} p(y_* = k|X, y)p(x_*|y_* = k, X, y)}$$

$$= \frac{p(y_* = k|y)p(x_*|X^{(k)})}{\sum_{k=1}^{K} p(y_* = k|y)p(x_*|X^{(k)})}$$

(1)

- Note: $X^{(k)}$ denotes training inputs with $y = k$

- Here $p(y_* = k|y) = \int p(y_*|\pi)p(\pi|y)d\pi$ (we did this; recall dice roll example)

- Here $p(x_*|X^{(k)}) = \int p(x_*|\theta_k)p(\theta_k|X^{(k)})d\theta_k$ (post. predictive dist. of input $x_*$ under class $k$)

- Eq (1) is the posterior predictive distribution of test output $y_*$ given input $x_*$

  - Note that we have done posterior averaging for all the parameters
**Recap: Bayesian Generative Classification**

- Recall generative classification \( p(y = k| x) = \frac{p(y=k)p(x|y=k)}{\sum_{k=1}^{K} p(y=k)p(x|y=k)} \). Prediction rule for a test input \( x_\ast \)
  \[
  p(y_\ast = k| x_\ast , X, y) = \frac{p(y_\ast = k| X, y)p(x_\ast | y_\ast = k, X, y)}{\sum_{k=1}^{K} p(y_\ast = k| X, y)p(x_\ast | y_\ast = k, X, y)} 
  \]
  \[
  = \frac{p(y_\ast = k| y)p(x_\ast | X(k))}{\sum_{k=1}^{K} p(y_\ast = k| y)p(x_\ast | X(k))} \tag{1}
  \]
  
  - Note: \( X(k) \) denotes training inputs with \( y = k \)
  
  - Here \( p(y_\ast = k| y) = \int p(y_\ast | \pi)p(\pi | y)d\pi \) (we did this; recall dice roll example)
  
  - Here \( p(x_\ast | X(k)) = \int p(x_\ast | \theta_k)p(\theta_k | X(k))d\theta_k \) (post. predictive dist. of input \( x_\ast \) under class \( k \))
  
  - Eq (1) is the posterior predictive distribution of test output \( y_\ast \) given input \( x_\ast \)
  
  - Note that we have done posterior averaging for all the parameters
Recap: Bayesian Generative Classification

Recall generative classification \( p(y = k | x) = \frac{p(y=k)p(x|y=k)}{\sum_{k=1}^{K} p(y=k)p(x|y=k)} \). Prediction rule for a test input \( x^* \)

\[
p(y^* = k | x^*, X, y) = \frac{p(y^* = k | X, y)p(x^* | y^* = k, X, y)}{\sum_{k=1}^{K} p(y^* = k | X, y)p(x^* | y^* = k, X, y)} = \frac{p(y^* = k | y)p(x^* | X^{(k)})}{\sum_{k=1}^{K} p(y^* = k | y)p(x^* | X^{(k)})} \tag{1}
\]

Note: \( X^{(k)} \) denotes training inputs with \( y = k \)

Here \( p(y^* = k | y) = \int p(y^* | \pi)p(\pi | y) d\pi \) (we did this; recall dice roll example)

Here \( p(x^* | X^{(k)}) = \int p(x^* | \theta_k)p(\theta_k | X^{(k)}) d\theta_k \) (post. predictive dist. of input \( x^* \) under class \( k \))

Eq (1) is the posterior predictive distribution of test output \( y^* \) given input \( x^* \)

Note that we have done posterior averaging for all the parameters

In contrast, for gen. class with MLE/MAP, \( p(y^* = k | y) \approx \pi_k \) and \( p(x^* | X^{(k)}) \approx p(x^* | \theta_k) \)
Gaussian Processes (GP)

\[(GP = \text{Bayesian Modeling} + \text{Kernel Methods})\]

\[(\text{Goal: learning } \textbf{nonlinear} \text{ discriminative models } p(y|x))\]
Consider the problem of learning to map an input $x \in \mathbb{R}^D$ to an output $y$. Linear models use a weighted combination of input features (i.e., $w^\top x$) to generate $y$.

- **Linear Regression**: $p(y | w^\top x, x) = \mathcal{N}(y | w^\top x, \beta^{-1})$
- **Logistic Regression**: $p(y | w, x) = \sigma(w^\top x)$ where $\sigma(z) = \frac{1}{1 + e^{-z}}$
- **Generalized Linear Model**: $p(y | w, x) = \text{ExpFam}(w^\top x)$

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

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

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

$$p(y|w, x) = \mathcal{N}(y|w^\top x, \beta^{-1})$$ (Linear Regression)
Consider the problem of learning to map an input $x \in \mathbb{R}^D$ to an output $y$

Linear models use a weighted combination of input features (i.e., $w^T x$) to generate $y$

$$p(y|w, x) = \mathcal{N}(y|w^T x, \beta^{-1})$$  \hspace{1cm} \text{(Linear Regression)}

$$p(y|w, x) = [\sigma(w^T x)]^y[1 - \sigma(w^T x)]^{1-y}$$  \hspace{1cm} \text{(Logistic Regression)}
Consider the problem of learning to map an input $x \in \mathbb{R}^D$ to an output $y$

Linear models use a weighted combination of input features (i.e., $w^\top x$) to generate $y$

$$p(y|w, x) = \mathcal{N}(y|w^\top x, \beta^{-1}) \quad \text{(Linear Regression)}$$

$$p(y|w, x) = [\sigma(w^\top x)]^y[1 - \sigma(w^\top x)]^{1-y} \quad \text{(Logistic Regression)}$$

$$p(y|w, x) = \text{ExpFam}(w^\top x) \quad \text{(Generalized Linear Model)}$$
Linear Models

- Consider the problem of learning to map an input \( x \in \mathbb{R}^D \) to an output \( y \)

- Linear models use a weighted combination of input features (i.e., \( w^T x \)) to generate \( y \)

\[
p(y|w, x) = \mathcal{N}(y|w^T x, \beta^{-1}) \quad \text{(Linear Regression)}
\]
\[
p(y|w, x) = [\sigma(w^T x)]^y[1 - \sigma(w^T x)]^{1-y} \quad \text{(Logistic Regression)}
\]
\[
p(y|w, x) = \text{ExpFam}(w^T x) \quad \text{(Generalized Linear Model)}
\]

- The weights \( w \) can be learned using MLE, MAP, or fully Bayesian inference
Consider the problem of learning to map an input $x \in \mathbb{R}^D$ to an output $y$.

Linear models use a weighted combination of input features (i.e., $w^\top x$) to generate $y$:

$$p(y|w, x) = \mathcal{N}(y|w^\top x, \beta^{-1})$$  \hspace{1cm} (Linear Regression)

$$p(y|w, x) = [\sigma(w^\top x)]^y [1 - \sigma(w^\top x)]^{1-y}$$  \hspace{1cm} (Logistic Regression)

$$p(y|w, x) = \text{ExpFam}(w^\top x)$$  \hspace{1cm} (Generalized Linear Model)

The weights $w$ can be learned using MLE, MAP, or fully Bayesian inference.

However, linear models have limited expressive power. Unable to learn highly nonlinear patterns.

![Nonlinear Regression](image1.png)  ![Nonlinear Classification](image2.png)
Assume the input to output relationship to be modeled by a nonlinear function $f$.
Modeling Nonlinear Functions

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

$$p(y|f, x) = \mathcal{N}(y|f(x), \beta^{-1})$$
Assume the input to output relationship to be modeled by a nonlinear function $f$

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

$$p(y|f, x) = [\sigma(f(x))]^y[1 - \sigma(f(x))]^{1-y}$$
Modeling Nonlinear Functions

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

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

$$
p(y|f, x) = [\sigma(f(x))]^y[1 - \sigma(f(x))]^{1-y}
$$

$$
p(y|f, x) = \text{ExpFam}(f(x))
$$
Assume the input to output relationship to be modeled by a nonlinear function $f$

$$p(y|f, x) = \mathcal{N}(y|f(x), \beta^{-1})$$
$$p(y|f, x) = [\sigma(f(x))]^y[1 - \sigma(f(x))]^{1-y}$$
$$p(y|f, x) = \text{ExpFam}(f(x))$$

How can we define such a function nonlinear $f$?
Assume the input to output relationship to be modeled by a nonlinear function $f$

\[
p(y|f, x) = \mathcal{N}(y|f(x), \beta^{-1})
\]

\[
p(y|f, x) = [\sigma(f(x))]^y[1 - \sigma(f(x))]^{1-y}
\]

\[
p(y|f, x) = \text{ExpFam}(f(x))
\]

How can we define such a function nonlinear $f$?

Note: We not only want nonlinearity but also all benefits of probabilistic/Bayesian modeling
Modeling Nonlinear Functions

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

$$p(y|f, x) = \mathcal{N}(y|f(x), \beta^{-1})$$
$$p(y|f, x) = [\sigma(f(x))]^y [1 - \sigma(f(x))]^{1-y}$$
$$p(y|f, x) = \text{ExpFam}(f(x))$$

- How can we define such a function nonlinear $f$?

- Note: We not only want nonlinearity but also all benefits of probabilistic/Bayesian modeling
  - Must be able to get uncertainty estimates in the function and its predictions
Modeling Nonlinear Functions

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

\[
p(y|f, x) = \mathcal{N}(y|f(x), \beta^{-1})
\]

\[
p(y|f, x) = [\sigma(f(x))]^y [1 - \sigma(f(x))]^{1-y}
\]

\[
p(y|f, x) = \text{ExpFam}(f(x))
\]

- How can we define such a function nonlinear $f$?

- Note: We not only want nonlinearity but also all benefits of probabilistic/Bayesian modeling
  - Must be able to get uncertainty estimates in the function and its predictions

- Usually done in one of the following ways

- Ad-hoc: Define nonlinear features $\phi(x) + \text{train Bayesian linear model (} f(x) = w^\top \phi(x)\text{)}$: HW1)
- Ad-hoc: Train a neural net to extract features $\phi(x) + \text{train Bayesian linear model}$
- Gaussian Processes for Learning Nonlinear Functions
- Prob. Mod. & Inference - CS698X (Piyush Rai, IITK)
Modeling Nonlinear Functions

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

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

  $$
p(y|f, x) = [\sigma(f(x))]^y[1 - \sigma(f(x))]^{1-y}
  $$

  $$
p(y|f, x) = \text{ExpFam}(f(x))
  $$

- How can we define such a function nonlinear $f$?

- Note: We not only want nonlinearity but also all benefits of probabilistic/Bayesian modeling
  - Must be able to get uncertainty estimates in the function and its predictions

- Usually done in one of the following ways
  - Ad-hoc: Define nonlinear features $\phi(x) +$ train Bayesian linear model ($f(x) = \mathbf{w}^T \phi(x)$): HW1
  - Ad-hoc: Train a neural net to extract features $\phi(x) +$ train Bayesian linear model
  - Bayesian Neural Networks (later this semester)
  - Gaussian Processes (a Bayesian approach to kernel based nonlinear learning; today)
Modeling Nonlinear Functions

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

$$
\begin{align*}
p(y|f, x) &= \mathcal{N}(y|f(x), \beta^{-1}) \\
p(y|f, x) &= \left[\sigma(f(x))\right]^y[1 - \sigma(f(x))]^{1-y} \\
p(y|f, x) &= \text{ExpFam}(f(x))
\end{align*}
$$

- How can we define such a function nonlinear $f$?

- Note: We not only want nonlinearity but also all benefits of probabilistic/Bayesian modeling
  - Must be able to get uncertainty estimates in the function and its predictions

- Usually done in one of the following ways
  - Ad-hoc: Define nonlinear features $\phi(x)$ + train Bayesian linear model ($f(x) = w^T \phi(x)$): HW1
  - Ad-hoc: Train a neural net to extract features $\phi(x)$ + train Bayesian linear model
Assume the input to output relationship to be modeled by a nonlinear function \( f \)

\[
p(y|f, x) = \mathcal{N}(y|f(x), \beta^{-1})
\]

\[
p(y|f, x) = [\sigma(f(x))]^y[1 - \sigma(f(x))]^{1-y}
\]

\[
p(y|f, x) = \text{ExpFam}(f(x))
\]

How can we define such a function nonlinear \( f \)?

Note: We not only want nonlinearity but also all benefits of probabilistic/Bayesian modeling

Must be able to get uncertainty estimates in the function and its predictions

Usually done in one of the following ways

Ad-hoc: Define nonlinear features \( \phi(x) \) + train Bayesian linear model (\( f(x) = w^T \phi(x) \)): HW1)

Ad-hoc: Train a neural net to extract features \( \phi(x) \) + train Bayesian linear model

Bayesian Neural Networks (later this semester)
Assume the input to output relationship to be modeled by a nonlinear function $f$

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

$$
p(y|f(x)) = [\sigma(f(x))]^{y} [1 - \sigma(f(x))]^{1-y}
$$

$$
p(y|f(x)) = \text{ExpFam}(f(x))
$$

How can we define such a function nonlinear $f$?

Note: We not only want nonlinearity but also all benefits of probabilistic/Bayesian modeling

- Must be able to get uncertainty estimates in the function and its predictions

Usually done in one of the following ways

- Ad-hoc: Define nonlinear features $\phi(x)$ + train Bayesian linear model ($f(x) = \mathbf{w}^T \phi(x)$): HW1)
- Ad-hoc: Train a neural net to extract features $\phi(x)$ + train Bayesian linear model
- Bayesian Neural Networks (later this semester)
- Gaussian Processes (a Bayesian approach to kernel based nonlinear learning; today)
What is Gaussian Process?

- A Gaussian Process, denoted as $GP(\mu, \kappa)$, defines a distribution over functions.
  - The GP is defined by mean function $\mu$ and covariance/kernel function $\kappa$.
What is Gaussian Process?

- A Gaussian Process, denoted as $GP(\mu, \kappa)$, defines a distribution over functions.
  - The GP is defined by mean function $\mu$ and covariance/kernel function $\kappa$.
- Can use GP as a prior distribution over functions.
What is Gaussian Process?

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

Mean function $\mu$ models the “average” function from $\mathcal{GP}(\mu, \kappa)$

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

Cov. function $\kappa$ models “shape/smoothness” of these functions

$\kappa(.,.)$ is a function that computes similarity between two inputs (just like a kernel function)

Note: $\kappa(.,.)$ needs to be positive definite (just like kernel functions)

Can even learn $\mu$ and especially $\kappa$ (makes GP very flexible to model, possibly nonlinear, functions)
What is Gaussian Process?

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

- Mean function $\mu$ models the “average” function $f$ from $\mathcal{GP}(\mu, \kappa)$
  $$\mu(x) = \mathbb{E}[f(x)]$$
What is Gaussian Process?

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

- Can use GP as a prior distribution over functions

- Draw from a $\mathcal{GP}(\mu, \kappa)$ will give us a random function $f$ (imagine it as an infinite dim. vector)

- Mean function $\mu$ models the “average” function $f$ from $\mathcal{GP}(\mu, \kappa)$
  $$\mu(x) = \mathbb{E}[f(x)]$$

- Cov. function $\kappa$ models “shape/smoothness” of these functions
What is Gaussian Process?

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

- Can use GP as a prior distribution over functions.
- Draw from a $\mathcal{GP}(\mu, \kappa)$ will give us a random function $f$ (imagine it as an infinite dim. vector).

- Mean function $\mu$ models the “average” function $f$ from $\mathcal{GP}(\mu, \kappa)$.
  \[ \mu(x) = \mathbb{E}[f(x)] \]

- Cov. function $\kappa$ models “shape/smoothness” of these functions.
  - $\kappa(., .)$ is a function that computes similarity between two inputs (just like a kernel function).
What is Gaussian Process?

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

![Gaussian Process example](image)

- Mean function $\mu$ models the “average” function $f$ from $\mathcal{GP}(\mu, \kappa)$
  $$\mu(x) = \mathbb{E}[f(x)]$$
- Cov. function $\kappa$ models “shape/smoothness” of these functions
  - $\kappa(\cdot, \cdot)$ is a function that computes similarity between two inputs (just like a kernel function)
  - Note: $\kappa(\cdot, \cdot)$ needs to be positive definite (just like kernel functions)
What is Gaussian Process?

- A Gaussian Process, denoted as \( GP(\mu, \kappa) \), defines a distribution over functions.
  - The GP is defined by mean function \( \mu \) and covariance/kernel function \( \kappa \).
- Can use GP as a prior distribution over functions.
- Draw from a \( GP(\mu, \kappa) \) will give us a random function \( f \) (imagine it as an infinite dim. vector).

Mean function \( \mu \) models the “average” function \( f \) from \( GP(\mu, \kappa) \):

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

Cov. function \( \kappa \) models “shape/smoothness” of these functions.
- \( \kappa(., .) \) is a function that computes similarity between two inputs (just like a kernel function).
- Note: \( \kappa(., .) \) needs to be positive definite (just like kernel functions).
- Can even learn \( \mu \) and especially \( \kappa \) (makes GP very flexible to model, possibly nonlinear, functions).
Gaussian Process

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

$$
\begin{bmatrix}
  f(x_1) \\
  f(x_2) \\
  \vdots \\
  f(x_N)
\end{bmatrix}
\sim
\mathcal{N}
\left(
\begin{bmatrix}
  \mu(x_1) \\
  \mu(x_2) \\
  \vdots \\
  \mu(x_N)
\end{bmatrix},
\begin{bmatrix}
  \kappa(x_1, x_1) & \cdots & \kappa(x_1, x_N) \\
  \kappa(x_2, x_1) & \cdots & \kappa(x_2, x_N) \\
  \vdots & \cdots & \vdots \\
  \kappa(x_N, x_1) & \cdots & \kappa(x_N, x_N)
\end{bmatrix}
\right)
$$
Gaussian Process

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

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

- The above means that $f$’s values at any finite set of inputs are jointly Gaussian.
Gaussian Process

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

\[
\begin{bmatrix}
  f(x_1) \\
  f(x_2) \\
  \vdots \\
  f(x_N)
\end{bmatrix}
\sim \mathcal{N}
\begin{bmatrix}
  \mu(x_1) \\
  \mu(x_2) \\
  \vdots \\
  \mu(x_N)
\end{bmatrix},
\begin{bmatrix}
  \kappa(x_1, x_1) \ldots \kappa(x_1, x_N) \\
  \kappa(x_2, x_1) \ldots \kappa(x_2, x_N) \\
  \vdots \\
  \kappa(x_N, x_1) \ldots \kappa(x_N, x_N)
\end{bmatrix}
\]

- The above means that \( f \)'s values at any finite set of inputs are jointly Gaussian
- We can also write the above more compactly as \( f \sim \mathcal{N}(\mu, K) \) where

\[
\begin{bmatrix}
  f(x_1) \\
  f(x_2) \\
  \vdots \\
  f(x_N)
\end{bmatrix},
\begin{bmatrix}
  \mu(x_1) \\
  \mu(x_2) \\
  \vdots \\
  \mu(x_N)
\end{bmatrix},
\begin{bmatrix}
  \kappa(x_1, x_1) \ldots \kappa(x_1, x_N) \\
  \kappa(x_2, x_1) \ldots \kappa(x_2, x_N) \\
  \vdots \\
  \kappa(x_N, x_1) \ldots \kappa(x_N, x_N)
\end{bmatrix}
\]

Note that \( p(f) = \mathcal{N}(\mu, K) \) can be seen as the finite-dimensional version of the GP prior over \( f \).

If mean function is zero, we will have \( p(f) = \mathcal{N}(0, K) \).

Important: \( p(f_i | f_{-i}) \) is also Gaussian (where \( i \) denotes any subset of inputs and \( -i \) denotes rest of the inputs) due to Gaussian properties.
Gaussian Process

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

\[
\begin{bmatrix}
f(x_1) \\
f(x_2) \\
\vdots \\
f(x_N)
\end{bmatrix}
\sim \mathcal{N}
\begin{bmatrix}
\mu(x_1) \\
\mu(x_2) \\
\vdots \\
\mu(x_N)
\end{bmatrix},
\begin{bmatrix}
\kappa(x_1, x_1) \ldots \kappa(x_1, x_N) \\
\kappa(x_2, x_1) \ldots \kappa(x_2, x_N) \\
\vdots \\
\kappa(x_N, x_1) \ldots \kappa(x_N, x_N)
\end{bmatrix}
\]

- The above means that \( f \)'s values at any finite set of inputs are jointly Gaussian
- We can also write the above more compactly as \( f \sim \mathcal{N}(\mu, K) \) where

\[
\begin{bmatrix}
f(x_1) \\
f(x_2) \\
\vdots \\
f(x_N)
\end{bmatrix}, \quad
\begin{bmatrix}
\mu(x_1) \\
\mu(x_2) \\
\vdots \\
\mu(x_N)
\end{bmatrix}, \quad
\begin{bmatrix}
\kappa(x_1, x_1) \ldots \kappa(x_1, x_N) \\
\kappa(x_2, x_1) \ldots \kappa(x_2, x_N) \\
\vdots \\
\kappa(x_N, x_1) \ldots \kappa(x_N, x_N)
\end{bmatrix}
\]

- Note that \( \rho(f) = \mathcal{N}(\mu, K) \) can be seen as the finite-dimensional version of the GP prior over \( f \)
Gaussian Process

- $f$ is said to be drawn from a $\mathcal{GP}(\mu, \kappa)$ if its finite dim. version is the following joint Gaussian
  
  $\begin{bmatrix}
  f(x_1) \\
  f(x_2) \\
  \vdots \\
  f(x_N)
  \end{bmatrix}
  \sim \mathcal{N}
  \begin{bmatrix}
  \mu(x_1) \\
  \mu(x_2) \\
  \vdots \\
  \mu(x_N)
  \end{bmatrix},
  \begin{bmatrix}
  \kappa(x_1, x_1) \ldots \kappa(x_1, x_N) \\
  \kappa(x_2, x_1) \ldots \kappa(x_2, x_N) \\
  \vdots \\
  \kappa(x_N, x_1) \ldots \kappa(x_N, x_N)
  \end{bmatrix}$

- The above means that $f$’s values at any finite set of inputs are jointly Gaussian
- We can also write the above more compactly as $f \sim \mathcal{N}(\mu, K)$ where
  
  $f = \begin{bmatrix}
  f(x_1) \\
  f(x_2) \\
  \vdots \\
  f(x_N)
  \end{bmatrix}, \quad \mu = \begin{bmatrix}
  \mu(x_1) \\
  \mu(x_2) \\
  \vdots \\
  \mu(x_N)
  \end{bmatrix}, \quad K = \begin{bmatrix}
  \kappa(x_1, x_1) \ldots \kappa(x_1, x_N) \\
  \kappa(x_2, x_1) \ldots \kappa(x_2, x_N) \\
  \vdots \\
  \kappa(x_N, x_1) \ldots \kappa(x_N, x_N)
  \end{bmatrix}$

- Note that $p(f) = \mathcal{N}(\mu, K)$ can be seen as the finite-dimensional version of the GP prior over $f$
- If mean function is zero, we will have $p(f) = \mathcal{N}(0, K)$
Gaussian Process

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

$$
\begin{bmatrix}
    f(x_1) \\
    f(x_2) \\
    \vdots \\
    f(x_N)
\end{bmatrix}
\sim \mathcal{N}
\begin{bmatrix}
    \mu(x_1) \\
    \mu(x_2) \\
    \vdots \\
    \mu(x_N)
\end{bmatrix},
\begin{bmatrix}
    \kappa(x_1, x_1) & \cdots & \kappa(x_1, x_N) \\
    \kappa(x_2, x_1) & \cdots & \kappa(x_2, x_N) \\
    \vdots & \cdots & \vdots \\
    \kappa(x_N, x_1) & \cdots & \kappa(x_N, x_N)
\end{bmatrix}
$$

- The above means that $f$’s values at any finite set of inputs are jointly Gaussian
- We can also write the above more compactly as $f \sim \mathcal{N}(\mu, K)$ where

$$
\begin{aligned}
f &= \begin{bmatrix}
    f(x_1) \\
    f(x_2) \\
    \vdots \\
    f(x_N)
\end{bmatrix},
\mu &= \begin{bmatrix}
    \mu(x_1) \\
    \mu(x_2) \\
    \vdots \\
    \mu(x_N)
\end{bmatrix},
K &= \begin{bmatrix}
    \kappa(x_1, x_1) & \cdots & \kappa(x_1, x_N) \\
    \kappa(x_2, x_1) & \cdots & \kappa(x_2, x_N) \\
    \vdots & \cdots & \vdots \\
    \kappa(x_N, x_1) & \cdots & \kappa(x_N, x_N)
\end{bmatrix}
\end{aligned}
$$

- Note that $p(f) = \mathcal{N}(\mu, K)$ can be seen as the finite-dimensional version of the GP prior over $f$
- If mean function is zero, we will have $p(f) = \mathcal{N}(0, K)$. Important: $p(f_i | f_{-i})$ is also Gaussian (where $i$ denotes any subset of inputs and $-i$ denotes rest of the inputs) due to Gaussian properties
A Perspective from Bayesian Linear Regression

Let’s first consider the (probabilistic) linear regression model

\[
\begin{align*}
p(w) &= \mathcal{N}(w | \mu_0, \Sigma_0) \quad \text{(Prior)} \\
p(y|X, w) &= \mathcal{N}(Xw, \beta^{-1}I_N) \quad \text{(Likelihood w.r.t. } N \text{ obs.)}
\end{align*}
\]
A Perspective from Bayesian Linear Regression

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

\[
p(w) = \mathcal{N}(w|\mu_0, \Sigma_0) \quad \text{(Prior)}
\]

\[
p(y|X, w) = \mathcal{N}(Xw, \beta^{-1}I_N) \quad \text{(Likelihood w.r.t. \(N\) obs.)}
\]

\[
p(y|X) = \int p(y|X, w)p(w)dw = \mathcal{N}(X\mu_0, \beta^{-1}I_N + XX^\top) \quad \text{(Marginal likelihood)}
\]
A Perspective from Bayesian Linear Regression

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

\[
p(w) = \mathcal{N}(w|\mu_0, \Sigma_0) \quad \text{(Prior)}
\]

\[
p(y|X, w) = \mathcal{N}(Xw, \beta^{-1}I_N) \quad \text{(Likelihood w.r.t. N obs.)}
\]

\[
p(y|X) = \int p(y|X, w)p(w)dw = \mathcal{N}(X\mu_0, \beta^{-1}I_N + X\Sigma_0X^T) \quad \text{(Marginal likelihood)}
\]

\[
p(y|X) = \mathcal{N}(0, \beta^{-1}I_N + XX^T) \quad \text{(if } \mu_0 = 0 \text{ and } \Sigma_0 = I)\]
A Perspective from Bayesian Linear Regression

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

\[
p(w) = \mathcal{N}(w | \mu_0, \Sigma_0) \quad \text{(Prior)}
\]

\[
p(y|X, w) = \mathcal{N}(Xw, \beta^{-1}I_N) \quad \text{(Likelihood w.r.t. } N \text{ obs.)}
\]

\[
p(y|X) = \int p(y|X, w)p(w)dw = \mathcal{N}(X\mu_0, \beta^{-1}I_N + XX^T) \quad \text{(Marginal likelihood)}
\]

\[
p(y|X) = \mathcal{N}(0, \beta^{-1}I_N + XX^T) \quad \text{(if } \mu_0 = 0 \text{ and } \Sigma_0 = I)
\]

\[
p(y|X) = \mathcal{N}(0, XX^T) \quad \text{(if } \beta^{-1} = \infty, \text{ i.e., zero noise)}
\]
A Perspective from Bayesian Linear Regression

Let’s first consider the (probabilistic) linear regression model

\[
p(w) = \mathcal{N}(w|\mu_0, \Sigma_0) \quad \text{(Prior)}
\]

\[
p(y|X, w) = \mathcal{N}(Xw, \beta^{-1}I_N) \quad \text{(Likelihood w.r.t. N obs.)}
\]

\[
p(y|X) = \int p(y|X, w)p(w)dw = \mathcal{N}(X\mu_0, \beta^{-1}I_N + XX^T) \quad \text{(Marginal likelihood)}
\]

\[
p(y|X) = \mathcal{N}(0, \beta^{-1}I_N + XX^T) \quad \text{(if } \mu_0 = 0 \text{ and } \Sigma_0 = I)\]

\[
p(y|X) = \mathcal{N}(0, XX^T) \quad \text{(if } \beta^{-1} = \infty, \text{ i.e., zero noise)}
\]

Thus the joint marginal distr. of y conditioned on X is the following multivariate Gaussian

\[
\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \sim \mathcal{N} \left( \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} x_1^T x_1 \ldots x_1^T x_N \\ x_2^T x_1 \ldots x_2^T x_N \\ \vdots \\ x_N^T x_1 \ldots x_N^T x_N \end{bmatrix} \right)
\]
A Perspective from Bayesian Linear Regression

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

\[ p(w) = \mathcal{N}(w|\mu_0, \Sigma_0) \quad \text{(Prior)} \]
\[ p(y|X, w) = \mathcal{N}(Xw, \beta^{-1}I_N) \quad \text{(Likelihood w.r.t. } N \text{ obs.)} \]
\[ p(y|X) = \int p(y|X, w)p(w)dw = \mathcal{N}(X\mu_0, \beta^{-1}I_N + XX^\top) \quad \text{(Marginal likelihood)} \]

Thus the joint marginal distr. of \( y \) conditioned on \( X \) is the following multivariate Gaussian

\[
\begin{bmatrix}
  y_1 \\
  y_2 \\
  \vdots \\
  y_N
\end{bmatrix}
\sim \mathcal{N}
\left(
\begin{bmatrix}
  0 \\
  0 \\
  \vdots \\
  0
\end{bmatrix},
\begin{bmatrix}
  x_1^\top x_1 & \ldots & x_1^\top x_N \\
  x_2^\top x_1 & \ldots & x_2^\top x_N \\
  \vdots & \ddots & \vdots \\
  x_N^\top x_1 & \ldots & x_N^\top x_N
\end{bmatrix}
\right)
\]

A “function space” view of linear regression as opposed to “weight space” view (both equivalent)
GP for Regression and Classification

(Note that GP only defines the score $f(x)$ but $y = f(x) + \text{“noise”}$)

(“noise” may be Gaussian, sigmoid-Bernoulli, or something else)
GP Regression

Training data: \( \{x_n, y_n\}_{n=1}^{N} \). \( x_n \in \mathbb{R}^D, y_n \in \mathbb{R} \)
GP Regression

Training data: \( \{x_n, y_n\}_{n=1}^{N} \). \( x_n \in \mathbb{R}^D, y_n \in \mathbb{R} \)

Assume the responses to be a noisy function of the inputs

\[ y_n = f(x_n) + \epsilon_n = f_n + \epsilon_n \]
GP Regression

- Training data: \( \{x_n, y_n\}^N_{n=1} \). \( x_n \in \mathbb{R}^D, y_n \in \mathbb{R} \)
- Assume the responses to be a noisy function of the inputs
  \[ y_n = f(x_n) + \epsilon_n = f_n + \epsilon_n \]
- Assume a zero-mean Gaussian noise: \( \epsilon_n \sim \mathcal{N}(\epsilon_n|0, \sigma^2) \)
GP Regression

- Training data: \( \{x_n, y_n\}_{n=1}^N \). \( x_n \in \mathbb{R}^D \), \( y_n \in \mathbb{R} \)
- Assume the responses to be a noisy function of the inputs
  \[
y_n = f(x_n) + \epsilon_n = f_n + \epsilon_n
\]
- Assume a zero-mean Gaussian noise: \( \epsilon_n \sim \mathcal{N}(\epsilon_n|0, \sigma^2) \)
- This implies the following likelihood model: \( p(y_n|f_n) = \mathcal{N}(y_n|f_n, \sigma^2) \)
GP Regression

- Training data: \( \{x_n, y_n\}_{n=1}^{N} \). \( x_n \in \mathbb{R}^D, y_n \in \mathbb{R} \)
- Assume the responses to be a noisy function of the inputs
  \[ y_n = f(x_n) + \epsilon_n = f_n + \epsilon_n \]
- Assume a zero-mean Gaussian noise: \( \epsilon_n \sim \mathcal{N}(\epsilon_n | 0, \sigma^2) \)
- This implies the following likelihood model: \( p(y_n|f_n) = \mathcal{N}(y_n|f_n, \sigma^2) \)
- Denote \( f = [f_1, \ldots, f_N] \) and \( y = [y_1, \ldots, y_N] \).
GP Regression

- Training data: \( \{ \mathbf{x}_n, y_n \}_{n=1}^{N} \). \( \mathbf{x}_n \in \mathbb{R}^D, y_n \in \mathbb{R} \)
- Assume the responses to be a noisy function of the inputs
  \[ y_n = f(x_n) + \epsilon_n = f_n + \epsilon_n \]
- Assume a zero-mean Gaussian noise: \( \epsilon_n \sim \mathcal{N}(0, \sigma^2) \)
- This implies the following likelihood model: \( p(y_n | f_n) = \mathcal{N}(y_n | f_n, \sigma^2) \)
- Denote \( f = [f_1, \ldots, f_N] \) and \( y = [y_1, \ldots, y_N] \). For i.i.d. responses, the joint likelihood will be
  \[ p(y | f) = \mathcal{N}(y | f, \sigma^2 I_N) \]
Training data: \( \{ x_n, y_n \}_{n=1}^{N} \). \( x_n \in \mathbb{R}^D \), \( y_n \in \mathbb{R} \)

Assume the responses to be a noisy function of the inputs
\[
y_n = f(x_n) + \epsilon_n = f_n + \epsilon_n
\]

Assume a zero-mean Gaussian noise: \( \epsilon_n \sim \mathcal{N}(0, \sigma^2) \)

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

Denote \( f = [f_1, \ldots, f_N] \) and \( y = [y_1, \ldots, y_N] \). For i.i.d. responses, the joint likelihood will be
\[
p(y|f) = \mathcal{N}(y|f, \sigma^2 I_N)
\]

We now need a prior on the function \( f \) that enables us to model a nonlinear \( f \)
GP Regression

- Training data: \( \{x_n, y_n\}_{n=1}^{N} \), \( x_n \in \mathbb{R}^D, y_n \in \mathbb{R} \)
- Assume the responses to be a noisy function of the inputs
  \[ y_n = f(x_n) + \epsilon_n = f_n + \epsilon_n \]
- Assume a zero-mean Gaussian noise: \( \epsilon_n \sim \mathcal{N}(\epsilon_n|0, \sigma^2) \)
- This implies the following likelihood model: \( p(y_n|f_n) = \mathcal{N}(y_n|f_n, \sigma^2) \)
- Denote \( f = [f_1, \ldots, f_N] \) and \( y = [y_1, \ldots, y_N] \). For i.i.d. responses, the joint likelihood will be
  \[ p(y|f) = \mathcal{N}(y|f, \sigma^2 I_N) \]
- We now need a prior on the function \( f \) that enables us to model a nonlinear \( f \)
- Let’s choose zero mean Gaussian Process prior \( \mathcal{GP}(0, \kappa) \) on \( f \), which is equivalent to
  \[ p(f) = \mathcal{N}(f|0, K) \]
  where \( K_{nm} = \kappa(x_n, x_m) \).
GP Regression

- Training data: \( \{x_n, y_n\}_{n=1}^N \). \( x_n \in \mathbb{R}^D, y_n \in \mathbb{R} \)
- Assume the responses to be a noisy function of the inputs
  \[ y_n = f(x_n) + \epsilon_n = f_n + \epsilon_n \]
- Assume a zero-mean Gaussian noise: \( \epsilon_n \sim \mathcal{N}(\epsilon_n|0, \sigma^2) \)
- This implies the following likelihood model: \( p(y_n|f_n) = \mathcal{N}(y_n|f_n, \sigma^2) \)
- Denote \( f = [f_1, \ldots, f_N] \) and \( y = [y_1, \ldots, y_N] \). For i.i.d. responses, the joint likelihood will be
  \[ p(y|f) = \mathcal{N}(y|f, \sigma^2 I_N) \]
- We now need a prior on the function \( f \) that enables us to model a nonlinear \( f \)
- Let’s choose zero mean Gaussian Process prior \( \mathcal{GP}(0, \kappa) \) on \( f \), which is equivalent to
  \[ p(f) = \mathcal{N}(f|0, K) \]
where \( K_{nm} = \kappa(x_n, x_m) \). For now, assume \( \kappa \) is a known function with fixed hyperparameters.
GP Regression

- The likelihood model: $p(y|f) = \mathcal{N}(y|f, \sigma^2 I_N)$. The prior distribution: $p(f) = \mathcal{N}(f|0, K)$
GP Regression

- The likelihood model: \( p(y|f) = \mathcal{N}(y|f, \sigma^2 I_N) \). The prior distribution: \( p(f) = \mathcal{N}(f|0, K) \)
- The posterior \( p(f|y) \propto p(f)p(y|f) \), which will be another Gaussian (Exercise: Find its expression)

What's the posterior predictive \( p(y^*|x^*, y, X) \) or \( p(y^*|y) \) (skipping \( X, x^* \) from the notation)?

\[
p(y^*|y) = \int p(y^*|f^*) p(f^*|y) df^*
\]

where \( p(f^*|y) = \int p(f^*|f) p(f|y) df \) and note that \( p(f^*|f) \) must be Gaussian for GP

For this case (GP regression), we actually don't need to compute \( p(y^*|y) \) using the above method

Reason: The marginal distribution of the training data responses \( p(y) = \int p(y|f) p(f) df = \mathcal{N}(y|0, K + \sigma^2 I_N) = \mathcal{N}(y|0, C_N) \)
GP Regression

- The likelihood model: \( p(y|f) = \mathcal{N}(y|f, \sigma^2 I_N) \). The prior distribution: \( p(f) = \mathcal{N}(f|0, K) \)

- The posterior \( p(f|y) \propto p(f)p(y|f) \), which will be another Gaussian (Exercise: Find its expression)

- What’s the posterior predictive \( p(y_*|x_*, y, X) \) or \( p(y_*|y) \) (skipping \( X, x_* \) from the notation)?
GP Regression

- The likelihood model: \( p(y|f) = \mathcal{N}(y|f, \sigma^2 I_N) \). The prior distribution: \( p(f) = \mathcal{N}(f|0, K) \)

- The posterior \( p(f|y) \propto p(f)p(y|f) \), which will be another Gaussian (Exercise: Find its expression)

- What’s the posterior predictive \( p(y_\ast|x_\ast, y, X) \) or \( p(y_\ast|y) \) (skipping \( X, x_\ast \) from the notation)?

\[
p(y_\ast|y) = \int p(y_\ast|f_\ast)p(f_\ast|y)df_\ast
\]
GP Regression

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

\[
p(y^*|y) = \int p(y^*|f^*)p(f^*|y)df^*
\]

where \( p(f^*|y) = \int p(f^*|f)p(f|y)df \) and note that \( p(f^*|f) \) must be Gaussian for GP
GP Regression

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

\[
p(y_*|y) = \int p(y_*|f_*)p(f_*|y)df_*
\]

where \( p(f_*|y) = \int p(f_*|f)p(f|y)df \) and note that \( p(f_*|f) \) must be Gaussian for GP
- For this case (GP regression), we actually don’t need to compute \( p(y_*|y) \) using the above method
GP Regression

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

\[
p(y_*|y) = \int p(y_*|f_*)p(f_*|y)df_*
\]

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

\[
p(y) = \int p(y|f)p(f)df = \mathcal{N}(y|0, K + \sigma^2 I_N) = \mathcal{N}(y|0, C_N)
\]
GP Regression

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

\[
p(y_*|y) = \int p(y_*|f_*)p(f_*|y)df_*
\]

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

\[
p(y) = \int p(y|f)p(f)df = \mathcal{N}(y|0, K + \sigma^2 I_N) = \mathcal{N}(y|0, C_N)
\]
- Using the same result, the marginal distribution \( p(y_*) = \mathcal{N}(y_*|0, \kappa(x_*, x_*) + \sigma^2) \)
GP Regression: Making Predictions

Let’s consider the joint distr. of $N$ training responses $y$ and test response $y_*$

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

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

$$C_{N+1} = \begin{bmatrix} C_N & k_* \\ k_*^T & c \end{bmatrix}$$
Let's consider the joint distr. of $N$ training responses $y$ and test response $y_*$

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

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

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

and $\mathbf{k}_* = [\kappa(x_*, x_1), \ldots, \kappa(x_*, x_N)]^\top$, $c = \kappa(x_*, x_*) + \sigma^2$
GP Regression: Making Predictions

- Let's consider the joint distr. of $N$ training responses $y$ and test response $y^*$

$$p\left(\begin{bmatrix} y \\ y^* \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} y \\ y^* \end{bmatrix} \middle| \begin{bmatrix} 0 \\ 0 \end{bmatrix}, C_{N+1}\right)$$

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

$$C_{N+1} = \begin{bmatrix} C_N & k_* \\ k_*^\top & c \end{bmatrix}$$

and $k_* = [\kappa(x_*, x_1), \ldots, \kappa(x_*, x_N)]^\top$, $c = \kappa(x_*, x_*) + \sigma^2$

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

$$p(y_* \mid y) = \mathcal{N}(y_* \mid \mu_*, \sigma_*^2)$$
GP Regression: Making Predictions

Let’s consider the joint distr. of \(N\) training responses \(y\) and test response \(y_*\)

\[
p \left( \begin{bmatrix} y \\ y_* \end{bmatrix} \right) = \mathcal{N} \left( \begin{bmatrix} y \\ y_* \end{bmatrix} \middle| \begin{bmatrix} 0 \\ 0 \end{bmatrix}, c_{N+1} \right)
\]

where the \((N + 1) \times (N + 1)\) matrix \(c_{N+1}\) is given by

\[
c_{N+1} = \begin{bmatrix} C_N & k_* \\ k_*^T & c \end{bmatrix}
\]

and \(k_* = [\kappa(x_*, x_1), \ldots, \kappa(x_*, x_N)]^T\), \(c = \kappa(x_*, x_*) + \sigma^2\)

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

\[
p(y_* \mid y) = \mathcal{N}(y_* \mid \mu_*, \sigma_*^2)
\]

\[
\mu_* = k_*^T c_N^{-1} y
\]
GP Regression: Making Predictions

Let’s consider the joint distr. of $N$ training responses $y$ and test response $y_*$

$$p \left( \begin{bmatrix} y \\ y_* \end{bmatrix} \right) = \mathcal{N} \left( \begin{bmatrix} y \\ y_* \end{bmatrix} \bigg| \begin{bmatrix} 0 \\ 0 \end{bmatrix}, C_{N+1} \right)$$

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

$$C_{N+1} = \begin{bmatrix} C_N & k_* \\ k_*^\top & c \end{bmatrix}$$

and $k_* = [\kappa(x_*, x_1), \ldots, \kappa(x_*, x_N)]^\top$, $c = \kappa(x_*, x_*) + \sigma^2$

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

$$p(y_* | y) = \mathcal{N}(y_* | \mu_*, \sigma_*^2)$$

$$\mu_* = k_*^\top C_N^{-1} y$$

$$\sigma_*^2 = \kappa(x_*, x_*) + \sigma^2 - k_*^\top C_N^{-1} k_*$$
GP Regression: Interpreting GP Predictions

Let’s look at the predictions made by GP regression

\[ p(y_*|y) = \mathcal{N}(y_*|\mu_*, \sigma_*^2) \]

\[ \mu_* = k_*^\top C_N^{-1} y \]

\[ \sigma_*^2 = k(x_*, x_*) + \sigma^2 - k_*^\top C_N^{-1} k_* \]

Two interpretations for the mean prediction

A kernel SVM like interpretation

\[ \mu_* = k_*^\top C_N^{-1} \alpha \]

where \( \alpha \) is akin to the weights of support vectors

A nearest neighbors interpretation

\[ \mu_* = w^\top y = N \sum_{n=1}^{N} w_n y_n \]

where \( w \) is akin to the weights of the neighbors
GP Regression: Interpreting GP Predictions

Let’s look at the predictions made by GP regression

\[ p(y_*|y) = \mathcal{N}(y_*|\mu_*, \sigma_*^2) \]
\[ \mu_* = k_*^\top C_N^{-1} y \]
\[ \sigma_*^2 = k(x_*, x_*) + \sigma^2 - k_*^\top C_N^{-1} k_* \]

Two interpretations for the mean prediction \( \mu_* \)
GP Regression: Interpreting GP Predictions

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

\[
p(y_*|y) = \mathcal{N}(y_*|\mu_*, \sigma_*^2)
\]

\[
\mu_* = k_*^\top C_N^{-1} y
\]

\[
\sigma_*^2 = k(x_*, x_*) + \sigma^2 - k_*^\top C_N^{-1} k_*
\]

- Two interpretations for the mean prediction \( \mu_* \)
  - A kernel SVM like interpretation
    \[
    \mu_* = k_*^\top C_N^{-1} y = k_*^\top \alpha = \sum_{n=1}^{N} k(x_*, x_n) \alpha_n
    \]
    where \( \alpha \) is akin to the weights of support vectors
GP Regression: Interpreting GP Predictions

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

\[
p(y_*|y) = \mathcal{N}(y_*|\mu_*, \sigma_*^2)
\]

\[
\mu_* = k_*^\top C_N^{-1} y
\]

\[
\sigma_*^2 = k(x_*, x_*) + \sigma^2 - k_*^\top C_N^{-1} k_*
\]

- Two interpretations for the mean prediction \(\mu_*\)
  - A kernel SVM like interpretation
    \[
    \mu_* = k_*^\top C_N^{-1} y = k_*^\top \alpha = \sum_{n=1}^{N} k(x_*, x_n) \alpha_n
    \]
    where \(\alpha\) is akin to the weights of support vectors
  - A nearest neighbors interpretation
    \[
    \mu_* = k_*^\top C_N^{-1} y = w^\top y = \sum_{n=1}^{N} w_n y_n
    \]
    where \(w\) is akin to the weights of the neighbors
Next Class

- Properties of GP based models, choice of kernels, etc
- Learning hyperparameters in GP based models
- GP for classification and GLMs
- Making GP models scalable
- Some recent advances in GP