Gaussian Processes: The Basics

CS698X: Topics in Probabilistic Modeling and Inference

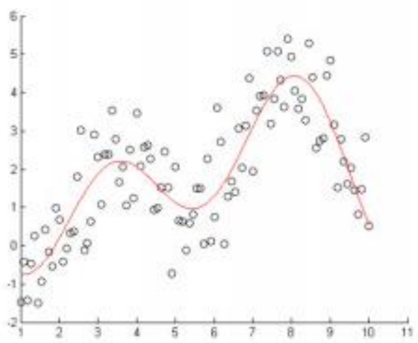
Piyush Rai

Linear Models and Their Limitations

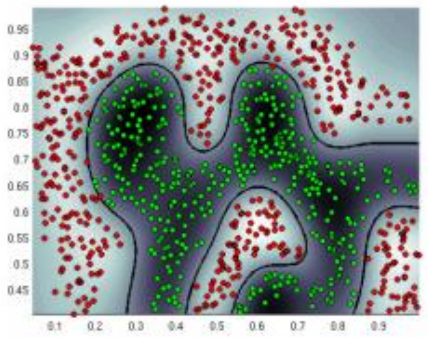
- Consider learning to map an input $\boldsymbol{x}$ to the output $\boldsymbol{y}$
- We've seen various discriminative models (linear and generalized linear models)

$$
\begin{aligned}
p(y|\boldsymbol{w},\boldsymbol{x}) &= \mathcal{N}(y|\mathbf{w}^\top\mathbf{x},\beta^{-1}) && \text{(Linear Regression)}\\
p(y|\boldsymbol{w},\boldsymbol{x}) &= [\sigma(\boldsymbol{w}^\top\boldsymbol{x})]^y[1-\sigma(\boldsymbol{w}^\top\boldsymbol{x})]^{1-y} && \text{(Logistic Regression)}\\
p(y|\boldsymbol{w},\boldsymbol{x}) &= \text{ExpFam}(\underbrace{\boldsymbol{w}^\top\boldsymbol{x}}_{\text{Natural param of canonical GLM}}) && \text{(Generalized Linear Model)}
\end{aligned}
$$

- These have limited expressive power – can't learn nonlinear patterns



Nonlinear Regression



Nonlinear Classification

Learning Nonlinear Functions

- Assume the input to output relationship to be modeled by a nonlinear function $\boldsymbol{f}$

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

In all of these, the linear score $\boldsymbol{w}^\top \boldsymbol{x}$ has been replaced by a nonlinear function $\boldsymbol{f}(\boldsymbol{x})$

- Would like to model this function in a probabilistic/Bayesian manner
  - Nonlinearity + all the benefits of probabilistic/Bayesian modeling

Example: Assuming $\boldsymbol{x}$ is scalar, $\phi(x) = [1, x, x^2, \ldots, x^k]$, for some $\boldsymbol{k}$

- Some ways to achieve this
  - Ad-hoc: Manually define nonlinear features $\boldsymbol{\phi}(\boldsymbol{x})$ + train Bayesian linear model
  - Ad-hoc: Use a pre-train deep neural net to extract features $\boldsymbol{\phi}(\boldsymbol{x})$ + train Bayesian linear model
  - Bayesian Neural Networks (later)
  - Gaussian Processes (a Bayesian approach to kernel based nonlinear learning; today)

Gaussian Process

Any choice of the GP covariance function has an associated feature map $\phi(x)$ for the inputs x

Hmmm.. So GPs look like kernel methods with all the benefits of probabilistic/Bayesian modeling

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

Akin to how we define a Gaussian distribution over scalars/vectors, defined by a mean and variance/covariance matrix

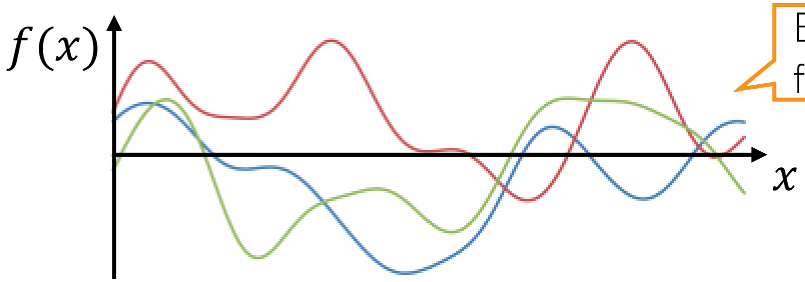
Mean Function

Covariance Function

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

Can also think of a function as an infinite dimensional vector of function's values at different inputs (x), i.e.,
$\boldsymbol{f} = [f(x_1), f(x_2), f(x_3), \ldots]$

- Every draw/sample from $\mathcal{GP}(\mu, \kappa)$ will give a random function f



Each of these curves is a random function drawn from the GP

μ and κ can be pre-defined or can even be learned

Mean Function $\mu(.)$ defines the "average" function looks like:
$\mu(x) = \mathbb{E}[f(x)]$

Covariance Function $\kappa(.,.)$ defines similarity between pairs of inputs and controls the shape of these curves (also needed to be pos-sem-def)

- IMP: If $f \sim \mathcal{GP}(\mu, \kappa)$ then f's value at any finite set of inputs is jointly Gaussian

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

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

$N \times 1$ vector of f's values: $\mathbf{f}$

$N \times 1$ mean vector: $\boldsymbol{\mu}$

$N \times N$ cov/kernel matrix (PSD): $\mathbf{K}$

Very useful property for making predictions: Knowing f's value at some N "training" inputs, say, $x_1, x_2, \ldots, x_N$, we can easily compute its value at a new test input x_*, using the Gaussian joint-to-conditional formula

Weight Space View vs Function Space View

- GPs are defined w.r.t. a function space that models input-output relationship
- In contrast, we have seen models that are defined w.r.t. a weight space, e.g.,

$$p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{w}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{X}\boldsymbol{w},\beta^{-1}\boldsymbol{I}_N)$$

Likelihood

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}_0,\boldsymbol{\Sigma}_0)$$

Prior over weight vector

$$p(\boldsymbol{y}|\boldsymbol{X}) = \int p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{w})p(\boldsymbol{w})d\boldsymbol{w} = \mathcal{N}(\boldsymbol{y}|\boldsymbol{X}\boldsymbol{\mu}_0,\beta^{-1}\boldsymbol{I}_N + \boldsymbol{X}\boldsymbol{\Sigma}_0\boldsymbol{X}^\top)$$

Marginal likelihood after integrating out the weights

$$p(\boldsymbol{y}|\boldsymbol{X}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{0},\beta^{-1}\boldsymbol{I}_N + \boldsymbol{X}\boldsymbol{X}^\top)$$

Marginal likelihood assuming $\boldsymbol{\mu}_0 = \boldsymbol{0}$ and $\boldsymbol{\Sigma}_0 = \boldsymbol{I}$

$$p(\boldsymbol{y}|\boldsymbol{X}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{0},\boldsymbol{X}\boldsymbol{X}^\top)$$

Assuming noise-free likelihood

- Thus the joint distribution of the N responses $y_1, y_2, \ldots, y_N$ is a multivariate Gaussian

This equivalence also shows that Bayesian linear regression is a special case of GP with linear kernel

$$p\left(\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} x_1^\top x_1 & \cdots & x_1^\top x_N \\ x_2^\top x_1 & & x_2^\top x_N \\ \vdots & \ddots & \vdots \\ x_N^\top x_1 & \cdots & x_N^\top x_N \end{bmatrix}\right)$$

Same as a GP $f(x_i) = y_i$, $\mu(x) = 0$ and linear covariance/kernel function $\kappa(x_i, x_j) = x_i^\top x_j$

- Thus GPs can be seen as bypassing the weight space and directly defining the model using a marginal likelihood via a function space defined by the GP

Predicting using GP

The results we saw here relating the score f_* to $\mathbf{f}$ will still hold ☺

We just need to use a likelihood model for y_n to handle such "noisy settings (will see soon)

- We have already seen that

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

The setting considered on this slide is the "noiseless" setting where the response y_n is simply given by $y_n = f_n = f(x_n)$. More realistic settings with have each output y_n as a transformation of a "score" given by GP: $f_n = f(x_n)$

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

- Let's assume the mean function $\mu(x) = 0$, thus $\boldsymbol{\mu} = \mathbf{0}$ and $p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \mathbf{K})$
- Assume we know $\mathbf{f} = [f(x_1), f(x_2), \ldots, f(x_N)]$ and want to compute $f(x_*)$
- Due to the GP property, joint distribution of f's values will always be Gaussian

$$p\left(\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix}\right) = \mathcal{N}\left(\underbrace{\begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}}_{(N+1)\times 1 \text{ vector}}, \underbrace{\begin{bmatrix} \mathbf{K} & \mathbf{k}_* \\ \mathbf{k}_*^\top & \kappa(x_*, x_*) \end{bmatrix}}_{(N+1)\times(N+1) \text{ matrix}}\right)$$

where $\mathbf{k}_* = [\kappa(x_1, x_*), \kappa(x_2, x_*), \ldots, \kappa(x_N, x_*)]^\top$

$N \times 1$ vector of similarities of x_* with each of the N training inputs

Important result

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

Form of prediction similar to kernel methods but also get variances σ_*^2

- Exercise: Show that predictive mean $\mu_* = \sum_{i=1}^N \beta_i f_i = \sum_{i=1}^N \alpha_i \kappa(x_i, x_*)$

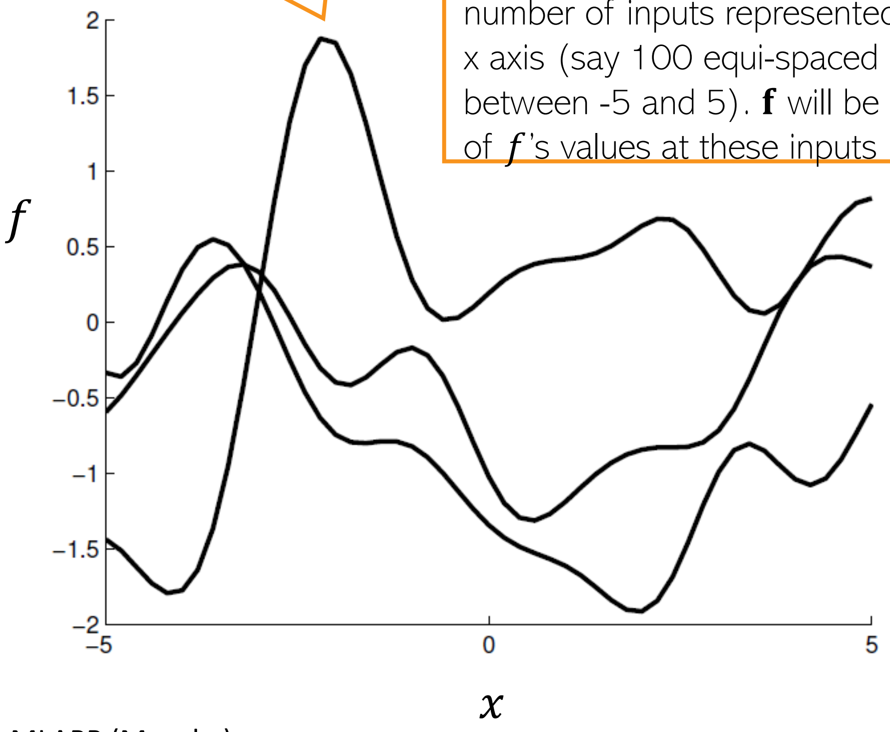
GP: A Visualization

$$k_{\mathrm{SE}}(x, x') = \sigma^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right)$$

- Assumed zero mean function and a squared exponential kernel

Each curve below is obtained by drawing a random $\mathbf{f}$ from the GP "prior" $p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \mathbf{K})$ and plotting it.

$\mathbf{K}$ is the kernel matrix of a finite number of inputs represented on the x axis (say 100 equi-spaced points between -5 and 5). $\mathbf{f}$ will be a vector of f's values at these inputs



Shaded area shows the predictive uncertainty for each of the test inputs (+/- 2 std)

Each curve below is obtained by drawing a random $\mathbf{f}$'s drawn from the GP posterior $p(\mathbf{f}|\mathbf{f}_{train})$ which is also a Gaussian

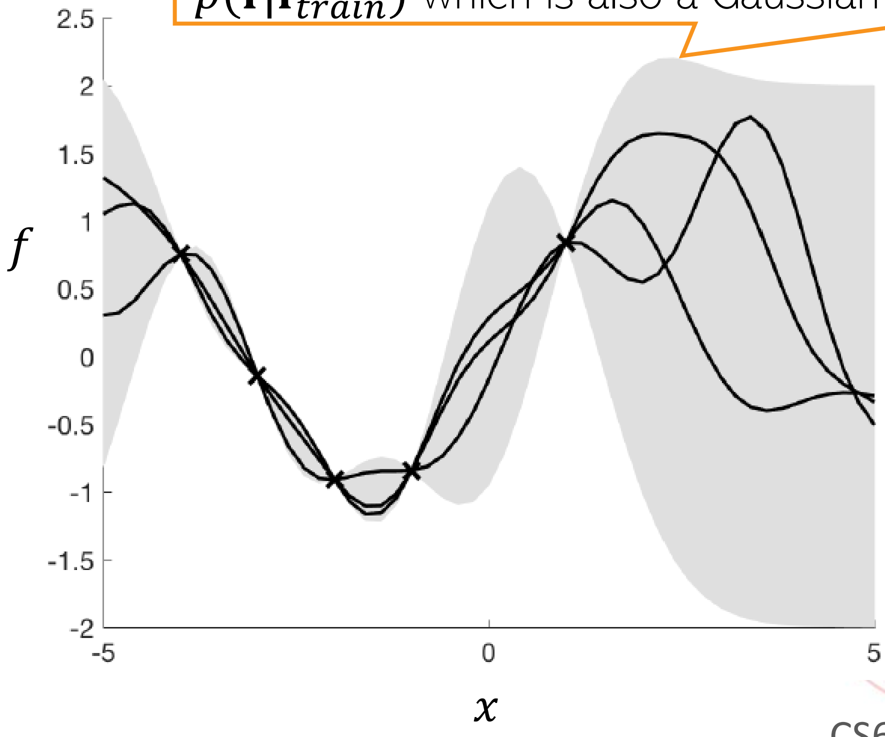


Figure courtesy: MLAPP (Murphy)

Coming Up

- GP for the "noisy" setting
  - Regression with Gaussian likelihood
  - Classification with Bernoulli/multinoulli likelihood
- Estimating the covariance/kernel function
- Connections with deep neural networks

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