**10-708: Probabilistic Graphical Models 10-708, Spring 2017**

# 24 : Gaussian Process and Deep Kernel Learning

*Lecturer: Zhiting Hu* *Scribes: Chao-Ming Yen, Biwei Huang*

## 1 Gaussian Process

### 1.1 Parametric v.s. Nonparametric Model

To better understand Gaussian Process, we first review some basic ideas of both parametric and nonparametric models. For parametric models, we assume Assumes all data can be represented using a fixed, finite number of parameters. For example, Mixture of K Gaussians and polynomial regression are both parametric models. On the other hand, for non-parametric models, the number of parameter can grow with sample size, which means that the sample number is undetermined when model is created.

In addition to conventional nonparametric model, there are also Bayesian nonparametrics, which are Bayesian models where the underlying finite-dimensional random variable is replaced by a stochastic process. A finite data set will only use a finite number of parameters, and the other parameters are integrated out.

### 1.2 Gaussian Process

A Gaussian process is a stochastic process where any finite number of random variables have a joint Gaussian distribution. Given the stochastic process $f$ and index $\mathbf{x}$ of sequence of random variables, the Gaussian Process is specified by a *mean function*

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{1}$$

and a *covariance function* (positive definite, also called *kernel function*)

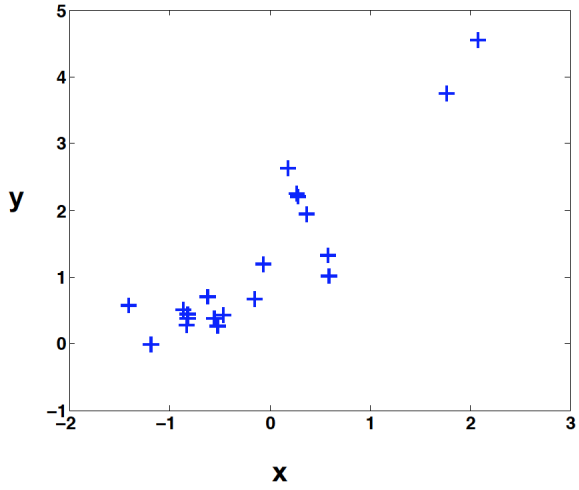
$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \tag{2}$$

We can write the Gaussian process as

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{3}$$

Intuitively, we can think Gaussian process as an infinite-dimensional Gaussian distribution.

## 1.3 Regression with Gaussian Process




To better understand Gaussian Process, we start from the classic regression problem. Same as conventional regression, we assume data is generated according to some latent function, and our goal is to infer this function to predict future data.

## 1.4 Probabilistic Approach

From the view of probabilistics, the uncertainty accounts for noise in the model, and we can assume a noise function $\epsilon(x)$. Let true underlying function $f$ and parameters $\mathbf{w}$, our observation $y$ can be written as

$$y(x) = f(x, \mathbf{w}) + \epsilon(x) \tag{4}$$

For the noise $\epsilon(x)$, one commonly takes $\epsilon(x) = N(0, \sigma^2)$ for i.i.d. additive Gaussian noise. In this case, we have

- Observation model
$$p(y(x)|x, \mathbf{w}, \sigma^2) = N(y(x); f(x, \mathbf{w}), \sigma^2) \tag{5}$$
- Likelihood
$$p(\mathbf{y}|x, \mathbf{w}, \sigma^2) = \sum_{i=1}^{N} N(y(x_i); f(x_u, \mathbf{w}), \sigma^2) \tag{6}$$

## 1.5 Bayesian Approach

From the view of Bayesian, we assume a prior over the parameters. In this case, we put a zero-mean Gaussian prior with covariance matrix $\Sigma_p$ on the weights $w = N(0, \Sigma_p)$. According to the Bayes' rule

$$posterior = \frac{likelihood * prior}{marginal \;\; likelihood} \tag{7}$$

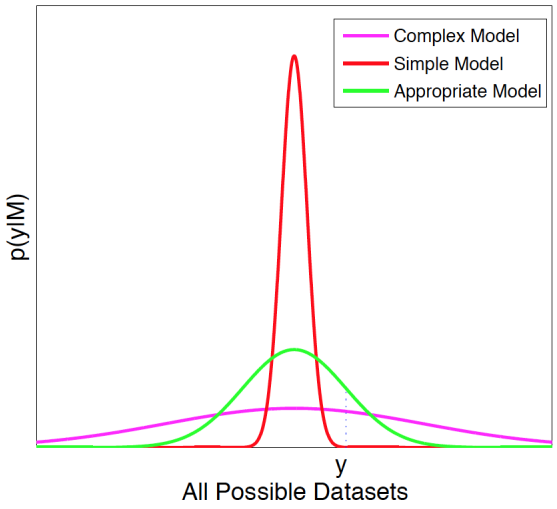
We have

$$p(\mathbf{w}|\mathbf{y}, X, \sigma^2) = \frac{p(\mathbf{y}|X, \mathbf{w}, \sigma^2)p(\mathbf{w})}{p(\mathbf{y}|X, \sigma^2)} \tag{8}$$

and the marginal likelihood

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})d\mathbf{w} \tag{9}$$

It's the average of infinitely many models weighted by their posterior probabilities. Typically we are more interested in distribution over functions than in parameters.



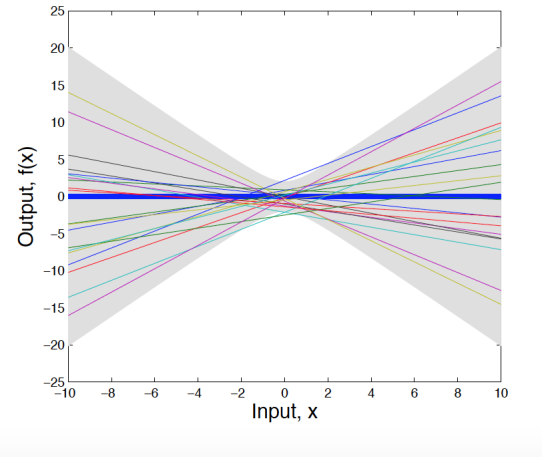

According to the rule of Occam's razor, the most appropriate model should cover all possible datasets while using the appropriate size of parameters. Simplest model might not be able to capture the whole datasets while complicated model is often over-fitting. For our Bayesian model, one way to resolve the model complexity problem is to average over infinitely many models by the posterior probabilities of those models. It allows us to calibrate the complexity while avoid over-fitting.

## 1.6 Weight-Space View

Consider a single linear regression model:

$$f(x) = a_0 + a_1 x, \text{where } a_0, a_1 \sim N(0, 1) \tag{10}$$

then the weight space construct many possible lines just like the following figure:




## 1.7 Function-Space View

Since we are more interested in the induced distribution over functions than the parameters, we can characterize the properties of the functions directly:

$$\mathbb{E}[f(x)] = \mathbb{E}[a_0] + \mathbb{E}[a_1]x = 0 \tag{11}$$

$$cov[f(x_b), f(x_c)] = \mathbb{E}[f(x_b)f(x_c)] - \mathbb{E}[f(x_b)]\mathbb{E}[f(x_c)] \tag{12}$$

$$= \mathbb{E}[a_o^2 + a_0 a_1(x_b + x_c) + a_1^2 x_b x_c] - 0 \tag{13}$$

$$= \mathbb{E}[a_0^2] + \mathbb{E}[a_1^2 x_b x_c] + \mathbb{E}[a_0 a_1(x_b + x_c)] \tag{14}$$

$$= 1 + x_b x_c + 0 = 1 + x_b x_c \tag{15}$$

Therefore any collection of values has a joint Gaussian distribution

$$[f(x_1), ..., f(x_N)] \sim N(0, K), \text{where} K_{ij} = cov(f(x_i), f(x_j)) = k(x_i, x_j) = 1 + x_b x_c \tag{16}$$

We can see that $f(x)$ is actually a collection of random variables of which have a joint Gaussian distribution. By definition, it is a Gaussian process. More specifically, for any collection of input values $x_1, ..., x_N$

$$[f(x_1), ..., f(X_N)] \sim N(\mu, \mathbf{K}), \text{where} \tag{17}$$

$$\mu_i = m(x_i) \text{and} \mathbf{K}_{ij} = k(x_i, x_j) \tag{18}$$

## 1.8 Linear Basis Function Models

Given the model specification:

$$f(x, \mathbf{w}) = \mathbf{w}^T \phi(x) \tag{19}$$

$$p(\mathbf{w}) = N(0, \Sigma_w) \tag{20}$$

By modeling the regression function as a linear regression in the basis function space, we can show that it is a Gaussian process by check the moments of induced distribution over functions:

$$\mathbb{E}[f(x, \mathbf{w})] = m(x) = \mathbb{E}[\mathbf{w}^T]\phi(x) = 0 \tag{21}$$

$$cov(f(x_i), f(x_j)) = k(x_i, x_j) = \mathbb{E}[f(x_i)f(x_j)] - \mathbb{E}[f(x_i)]\mathbb{E}[f(x_j)] \tag{22}$$

$$= \phi(x_i)^T \mathbb{E}[\mathbf{w}\mathbf{w}^T]\phi(x_j) - 0 \tag{23}$$

$$= \phi(x_i)^T \sum_w \phi(x_j) \tag{24}$$

Here, $f(x, \mathbf{w})$ is a Gaussian process and $f(x) \sim GP(m, k)$ with mean function $m(x) = 0$ and covariance kernel $k(x_i, x_j) = \phi(x_i)^T \sum_w \phi(x_j)$, and the entire basis function model of the model specification is encapsulated as a distribution over functions with kernel $k(x, x')$. The kernel controls the support and inductive biases of our model, and thus its ability of generalization.

## 2 Covariance Kernel

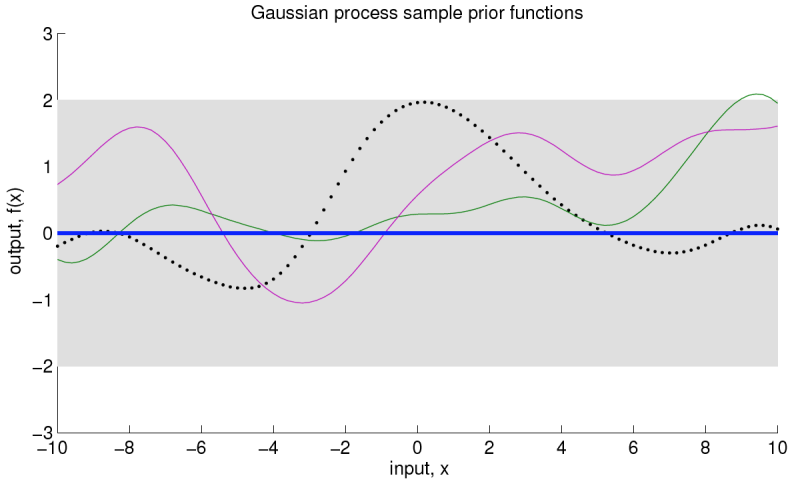
From the previous section, we know the properties of the distribution over functions are controlled by the covariance function. If we want to capture the idea that similar inputs have similar outputs, we typically want a covariance function that decays smoothly with distance. Take the most popular RBF kernel as an example:

$$k_{RBF}(x, x') = cov(f(x), f(x')) = a^2 exp(-\frac{||x - x'||^2}{2l^2}) \tag{25}$$

Where $a$ and $l$ are kernel hyper-parameters, controlling amplitudes and wiggliness of the functions. The RBF kernel expresses the intuition that function values at nearby inputs are more correlated than function values at far away inputs. A Gaussian process with RBF kernel is a powerful modeling tool since it has large support and universal approximators.

The following figure shows some samples from a GP with an RBF Kernel:

$$f_* \sim N(0, K(X_*, X_*)) \tag{26}$$



# 3 Working with Gaussian Process

A Gaussian process defines a prior over functions, but usually we only have data at a finite number of locations. To walk around, we can marginalize over all the locations at which we do not have data. That means at any time, we are only working with a finite multivariate Gaussian. As a consequence, we can learn the form of the covariance function based on these finite locations, and which gives us the covariance between the existing points and the target point. This allows us to predict values at any location via the conditional distribution.

# 4 Gaussian Process Inference

## 4.1 Inference

Suppose that we have observed noisy data $\mathbf{y} = (y(x_1), \cdots, y(x_N))^T$ at input locations $\mathbf{X}$. We assume the standard regression assumption: $y(x) \sim \mathcal{N}(f(x), \sigma^2)$, and we put a Gaussian process prior over noise free functions $f(x) \sim \mathcal{GP}(0, k_\theta)$. Let $f_*$ represent the noise free function evaluated at $X_*$, and then the joint distribution of $\mathbf{y}$ and $f_*$ can be represented as

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K_\theta(\mathbf{X}, \mathbf{X}) + \sigma^2 & K_\theta(\mathbf{X}, X_*) \\ K_\theta(X_*, \mathbf{X}) & K_\theta(X_*, X_*) \end{bmatrix}\right).$$

Then the predictive distribution of $f_*$ can be derived easily:

$$f_* | X_*, \mathbf{X}, \mathbf{y}, \theta \sim \mathcal{N}(\bar{f}_*, \mathtt{cov}(f_*)),$$

where

$$\bar{f}_* = K_\theta(X_*, \mathbf{X})[K_\theta(\mathbf{X}, \mathbf{X}) + \sigma^2 I]^{-1}\mathbf{y},$$

and

$$\mathtt{cov}(f_*) = K_\theta(X_*, X_*) - K_\theta(X_*, \mathbf{X})[K_\theta(\mathbf{X}, \mathbf{X}) + \sigma^2 I]^{-1}K_\theta(\mathbf{X}, X_*).$$

## 4.2 Learning and Model Selection

The marginal likelihood can be derived by integrating out $f(x)$, as a function of kernel hyperparameters alone.

$$p(\mathbf{y}|\mathbf{X}, \theta) = \int p(\mathbf{y}|\mathbf{f}, \mathbf{X})p(\mathbf{f}|\mathbf{X}, \theta)d\mathbf{f}.$$

With the Gaussian process prior $\mathbf{f}|\mathbf{X}, \theta \sim \mathcal{N}(0, K_\theta)$, and with the likelihood as a factorized Gaussian $\mathbf{y}|\mathbf{f} \sim \mathcal{N}(\mathbf{f}, \sigma^2 I)$, we can represent the log marginal likelihood as

$$p(\mathbf{y}|\mathbf{X}, \theta) = -1/2\mathbf{y}^T(K_\theta + \sigma^2 I)^{-1}\mathbf{y} - 1/2 \log |K_\theta + \sigma^2 I| - n/2 \log 2\pi, \tag{27}$$

where the first term is used for model fitting, and the second term is for model complexity regularization.

The estimation of marginal likelihood involves an inverse and a determinant of $n \times n$ matrix, which require $\mathcal{O}(n^3)$ operations and $\mathcal{O}(n^2)$ storage. In practice, one may use Cholesky decomposition to speed up the matrix inversion and matrix determinant.

The hyperparameters $\theta$ can be learned by maximizing the log marginal likelihood. For example, the RBF kernel in one dimension has the following form

$$k_y(x_i, x_j) = \sigma_f^2 \exp\big(-\frac{1}{2l^2}(x_i - x_j)^2\big) + \sigma_n^2 \delta_{ij},$$

The covariance is denoted by $k_y$ as it is for the noisy targets $y$ rather than for the underlying function $f$. Observe that the length-scale $l$, the signal variance $\sigma_f^2$ and the noise variance $\sigma_n^2$ can be varied. In general we call the free parameters hyperparameters. The hyperparameters $\sigma_f^2$, $l$, and $\sigma_n^2$ can be learned by maximizing the marginal likelihood in equation (27). Figure 1 shows the effects from hyperparameters.

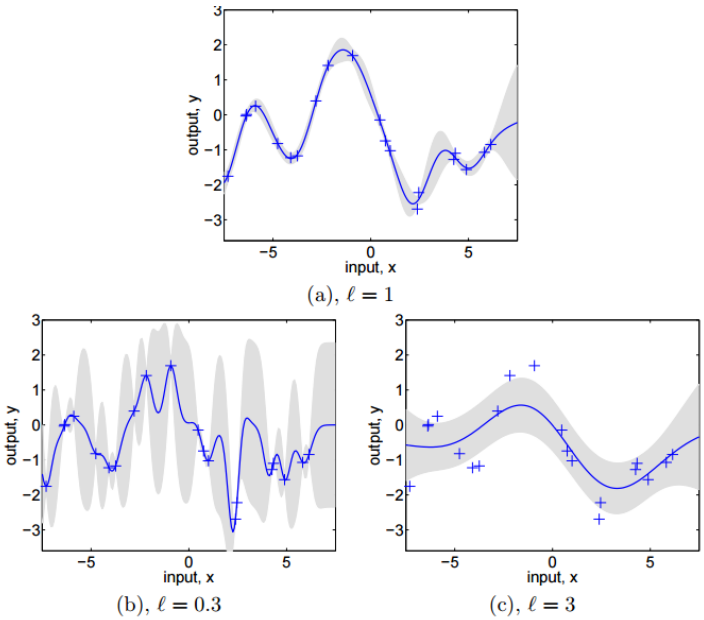



Figure 1: (a) Data is generated from a GP with hyperparameters $(l, \sigma_f, \sigma_n) = (1, 1, 0.1)$, as shown by + symbols. Using Gaussian process prediction with these hyperparameters we obtain a 95 % confidence region for the underlying function $f$ (shown in grey). Panels (b) and (c) again show the 95 % confidence region, but this time for hyperparameter values $(0.3, 1.08, 0.00005)$ and $(3.0, 1.16, 0.89)$ respectively.

## 4.3 Scaling Up Gaussian Process

A decision function can be written as

$$f(x) = \langle \mathbf{w}, \phi(x) \rangle = \langle \sum_{i=1}^{N} \alpha_i \phi(x_i), \phi(x) \rangle = \sum_{i=1}^{N} \alpha_i k(x_i, x).$$

Representer theorem says that this function exists with finitely many coefficients even when $\phi$ is infinitely dimensional (infinite number of basis functions). It is initially viewed as a strength of kernel methods, for datasets not extremely large. Unfortunately, the number of nonzero $\alpha_i$ often grows linearly in the size of the training set $n$. Therefore, efficient approaches need to be developed to scale up GP.

To scale up Gaussian process, roughly there are three types of approaches.

1. Approximate non-parametric kernels in a finite basis 'dual space'. It requires $\mathcal{O}(m^2 n)$ computations and $\mathcal{O}(m)$ storage for $m$ basis functions. Examples: SSGP, Random Kitchen Sinks, Fastfood, À la Carte.

2. Inducing point based sparse approximations. Examples: SoR, FITC, KISS-GP.

3. Exploit existing structure in K to quickly (and exactly) solve linear systems and log determinants. Examples: Toeplitz and Kronecker methods.

### 4.3.1 Inducing Points

Suppose that Gaussian processes $\mathbf{f}$ and $f_*$ are evaluated at $n$ training points and $J$ testing points. We further assume that there are $m$ inducing points $\mathbf{u}$ ($m \ll n$), where $p(\mathbf{u}) = \mathcal{N}(\mathbf{0}, K_{\mathbf{u},\mathbf{u}})$. The joint distribution of $\mathbf{f}$ and $f_*$ can be written as

$$p(\mathbf{f}, f_*) = \int p(\mathbf{f}, f_*|\mathbf{u})p(\mathbf{u})d\mathbf{u}.$$

We assume that $\mathbf{f}$ and $f_*$ are conditionally independent given $\mathbf{u}$, and then we have

$$p(\mathbf{f}, f_*) \approx q(\mathbf{f}, f_*) = \int q(\mathbf{f}|\mathbf{u})q(f_*|\mathbf{u})p(\mathbf{u})d\mathbf{u},$$

where

$$q(\mathbf{f}|\mathbf{u}) = \mathcal{N}(K_{f,u}K_{u,u}^{-1}\mathbf{u}, K_{f,f} - K_{f,u}K_{u,u}^{-1}K_{u,f}),$$

and

$$q(f_*|\mathbf{u}) = \mathcal{N}(K_{f_*,u}K_{u,u}^{-1}\mathbf{u}, K_{f_*,f_*} - K_{f_*,u}K_{u,u}^{-1}K_{u,f_*}).$$

With $m$ inducing points, the cost for predictions is reduced from $\mathcal{O}(n^3)$ to $\mathcal{O}(m^2 n)$.

### 4.3.2 Kronecker Methods

Suppose that if $x \in \mathbb{R}^P$, $k$ decomposes as a product of kernels across each input dimension: $k(x_i, x_j) = \prod_{p=1}^{P} k^p(x_i^p, x_j^q)$. Then $K$ can be decomposed into a Kronecker product of matrices over each input dimension

$$K = K^1 \otimes \cdots \otimes K^P.$$

The eigendecomposition of $K$ into $QVQ^T$ also decomposes:

$$Q = Q^1 \otimes \cdots \otimes Q^P,$$

and

$$V = V^1 \otimes \cdots \otimes V^P.$$

Assuming equal cardinality for each input dimension, we can thus eigendecompose an $n \times n$ matrix $K$ in $\mathcal{O}(PN^3/P)$ operations instead of $\mathcal{O}(n^3)$ operations.

Then the inference and learning are highly efficient:

$$(K + \sigma^2 I)^{-1} = (QVQ^T + \sigma^2 I)^{-1} = Q(V + \sigma^2 I)^{-1}Q^T$$
$$\log|K + \sigma^2 I| = \log|QVQ^T + \sigma^2 I| = \sum_{i=1}^{n} \log(\lambda_i + \sigma^2).$$

# 5 Kernel

Informally, kernel $k$ describes the similarities between pairs of data points. For example, far away points may be considered less similar than nearby points. We list several widely-used kernels:

$$k_{\mathrm{SE}}(\tau) = \exp(-\tfrac{1}{2}\tau^2/l^2)$$
$$k_{\mathrm{MA}}(\tau) = a(1 + \tfrac{\sqrt{3}\tau}{l})\exp(-\tfrac{\sqrt{3}\tau}{l})$$
$$k_{\mathrm{RQ}}(\tau) = (1 + \tfrac{\tau^2}{2al^2})^{-\alpha}$$
$$k_{\mathrm{PE}}(\tau) = \exp(-1\sin^2(\pi\tau w)/l^2),$$

where $\tau = x_i - x_j$.

## 6 Deep Kernel Learning

Scalable deep kernels combine the structural properties of deep learning architectures with the non-parametric flexibility of kernel methods. Specifically, it transforms the inputs of a spectral mixture base kernel with a deep architecture,

$$k(x_i, x_j|\theta) \rightarrow k(g(x_i, w), g(x_j, w)|\theta, w),$$

where $g(x, w)$ is a non-linear mapping given by a deep architecture, such as a deep convolutional network, parametrized by weights $w$. These closed-form kernels can be used as drop-in replacements for standard kernels, with benefits in expressive power and scalability. Then the properties of these kernels are jointly learned through the marginal likelihood of a Gaussian process. Inference and learning cost $\mathcal{O}(n)$ for $n$ training points, and predictions cost $\mathcal{O}(1)$ per test point.