Gaussian Process

Lei Tang

Arizona State University

Jul. 31th, 2007

- Gaussian Process, (kriging in geostatistics)
- Autoregressive moving average model, Kalman filters, and radial basis function networks can be viewed as forms of Gaussian process models.

Linear regression revisited

$$\begin{aligned} y(x) &= w^T \phi(x) \\ p(w) &= \mathcal{N}(w|0, \alpha^{-1}\mathbf{I}) \end{aligned}$$

$$\begin{aligned} \mathbf{y} &= \Phi \mathbf{w} \\ E[\mathbf{y}] &= \Phi E[\mathbf{w}] \\ cov[\mathbf{y}] &= E[\mathbf{y}\mathbf{y}^T] = \Phi E[\mathbf{w}\mathbf{w}^T]\Phi^T = \frac{1}{\alpha}\Phi\Phi^T = \mathbf{K} \end{aligned}$$

where **K** is Gram matrix with elements

$$K_{nm} = k(x_n, x_m) = \frac{1}{\alpha}\phi(x_n)^T \phi(x_m)$$

Gaussian Process

- A Gaussian process is defined as a probability distribution over functions $y(x)$ suth that the set of values of $y(x)$ evaluated at an arbitrary set of points $x_1, \cdots, x_N$ jointly have a Gaussian distribution.
- *Gaussian random field*: when the input vector x is two-dimentional.
- *Stochastic process*: $y(x)$ is specified by giving the joint probability distirubtion for any finite set of values $y(x_1), \cdots, y(x_N)$ in a consistent manner.

GP Connection to Kernel

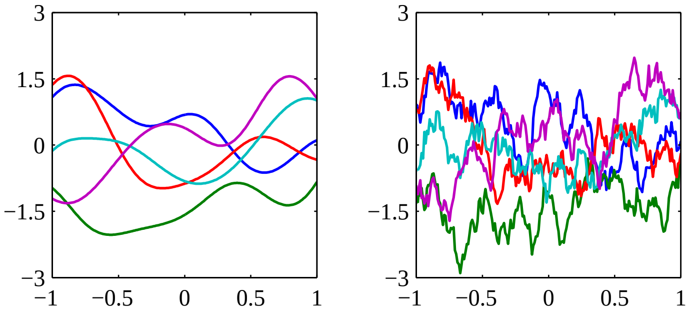
- For Gaussian stochstic process, the joint distribution over N variables $y_1, \cdots, y_N$ is specified completely by the second-order statistics.
- For most applications, we have no prior knowledge, so by symmetry(also for sparsity) we take the mean of $y(x)$ to be zero.
- Then the Gaussian process is deteremined by the covariance of $y(x)$ which is specified by the kernel function:

$$E[y(x_n), y(x_m)] = k(x_n, x_m)$$

Two Examples of GP

Specificy the covaraince (kernel) directly.

1. Gaussian Kernel: $k(x, x') = exp(-||x - x'||^2/2\sigma^2)$
2. Exponential Kernel: $k(x, x') = exp(-\theta|x - x'|)$ (correpsonds to the *Ornstein-Uhlenbeck* process original introduced for Brownian motion)



GP for Regression with Random Noise

If the noise on the observed target values are considered:

$$\begin{aligned}
p(t_n|y_n) &= \mathcal{N}(t_n|y_n, \beta^{-1}) \\
p(\mathbf{t}|\mathbf{y}) &= \mathcal{N}(\mathbf{t}|\mathbf{y}, \beta^{-1}\mathbf{I}_N) \\
p(\mathbf{y}) &= \mathcal{N}(\mathbf{y}|0, \mathbf{K}) \\
p(\mathbf{t}) &= \int p(\mathbf{t}|y)p(\mathbf{y})d\mathbf{y} = \mathcal{N}(t|0, \mathbf{C})
\end{aligned}$$

where $C(x_n, x_m) = k(x_n, x_m) + \beta^{-1}\delta_{nm}$. Covraince simply add.

Hint: matrix inverse lemma

$$[B^{-1} + CD^{-1}C^T]^{-1} = B - BC(D + C^T BC)^{-1}C^T B$$

GP for Regression with Random Noise

If the noise on the observed target values are considered:

$$\begin{aligned}
p(t_n|y_n) &= \mathcal{N}(t_n|y_n, \beta^{-1}) \\
p(\mathbf{t}|\mathbf{y}) &= \mathcal{N}(\mathbf{t}|\mathbf{y}, \beta^{-1}\mathbf{I}_N) \\
p(\mathbf{y}) &= \mathcal{N}(\mathbf{y}|0, \mathbf{K}) \\
p(\mathbf{t}) &= \int p(\mathbf{t}|y)p(\mathbf{y})d\mathbf{y} = \mathcal{N}(t|0, \mathbf{C})
\end{aligned}$$

where $C(x_n, x_m) = k(x_n, x_m) + \beta^{-1}\delta_{nm}$. Covraince simply add.

Hint: matrix inverse lemma

$$[B^{-1} + CD^{-1}C^T]^{-1} = B - BC(D + C^T BC)^{-1}C^T B$$

GP for Regression with Random Noise

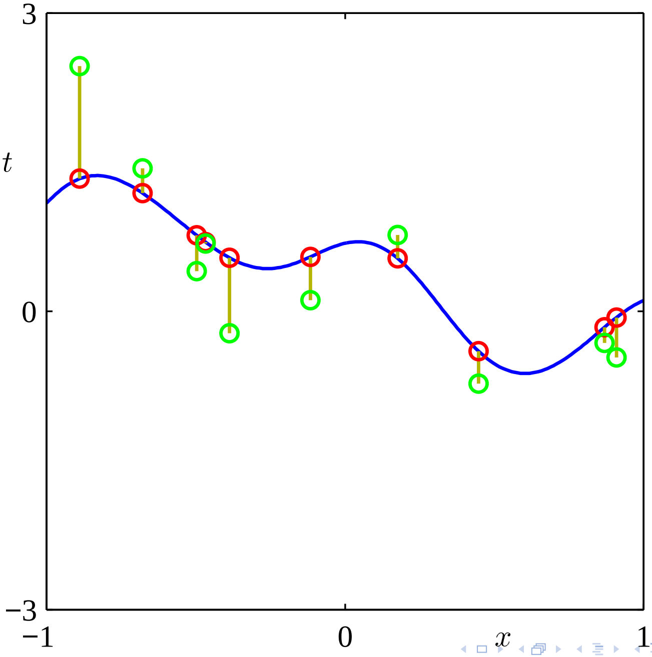
If the noise on the observed target values are considered:

$$\begin{aligned}
p(t_n|y_n) &= \mathcal{N}(t_n|y_n, \beta^{-1}) \\
p(\mathbf{t}|\mathbf{y}) &= \mathcal{N}(\mathbf{t}|\mathbf{y}, \beta^{-1}\mathbf{I}_N) \\
p(\mathbf{y}) &= \mathcal{N}(\mathbf{y}|0, \mathbf{K}) \\
p(\mathbf{t}) &= \int p(\mathbf{t}|y)p(\mathbf{y})d\mathbf{y} = \mathcal{N}(t|0, \mathbf{C})
\end{aligned}$$

where $C(x_n, x_m) = k(x_n, x_m) + \beta^{-1}\delta_{nm}$. Covraince simply add.

Hint: matrix inverse lemma

$$[B^{-1} + CD^{-1}C^T]^{-1} = B - BC(D + C^T BC)^{-1}C^T B$$


3
0
−3
t
−1
0
1
x

Commonly Used Kernel for Regression

$$k(x_n, x_m) = \theta_0 exp\left\{-\frac{\theta_1}{2}||x_n - x_m||^2\right\} + \theta_2 + \theta_3 x_n^T x_m$$

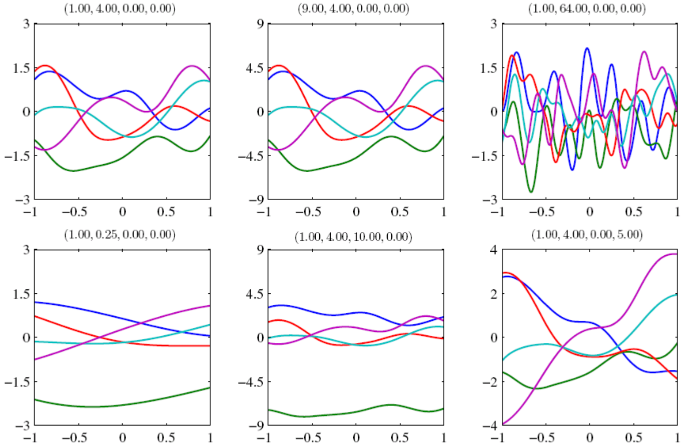


Figure 6.5 Samples from a Gaussian process prior defined by the covariance function (6.63). The title above each plot denotes $(\theta_0, \theta_1, \theta_2, \theta_3)$.

GP for Prediction

$$\begin{aligned} p(\mathbf{t}_{N+1}) &= \mathcal{N}(\mathbf{t}_{N+1}|0, \mathbf{C}_{N+1}) \\ C_{N+1} &= \begin{pmatrix} \mathbf{C}_N & \mathbf{k} \\ \mathbf{k}^T & c \end{pmatrix} \end{aligned}$$

$$\begin{aligned} m(x_{N+1}) &= \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{t} \\ \sigma^2(x_{N+1}) &= c - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k} \end{aligned}$$

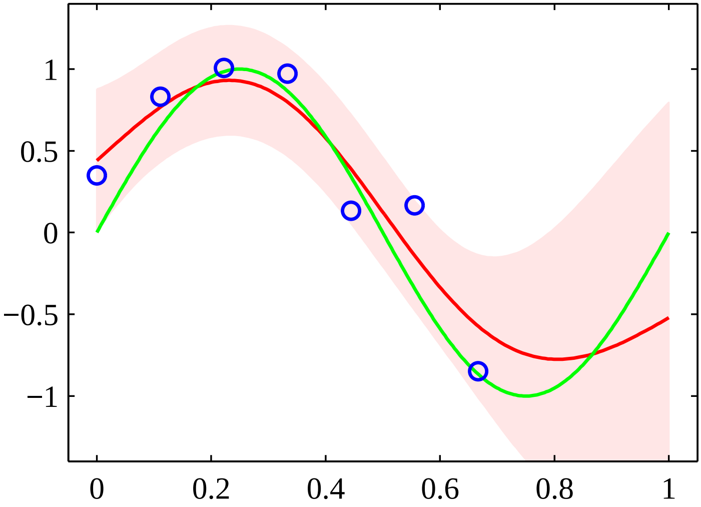
If we rewrite $m(x_{N+1}) = \sum_{n=1}^{N} a_n k(x_n, x_{N+1})$, and define a kernel function depending only on the distance $||x_n - x_m||$, we obtain an expansion in radial basis function.

GP for Prediction

$$\begin{aligned} p(\mathbf{t}_{N+1}) &= \mathcal{N}(\mathbf{t}_{N+1}|0, \mathbf{C}_{N+1}) \\ C_{N+1} &= \begin{pmatrix} \mathbf{C}_N & \mathbf{k} \\ \mathbf{k}^T & c \end{pmatrix} \end{aligned}$$

$$\begin{aligned} m(x_{N+1}) &= \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{t} \\ \sigma^2(x_{N+1}) &= c - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k} \end{aligned}$$

If we rewrite $m(x_{N+1}) = \sum_{n=1}^{N} a_n k(x_n, x_{N+1})$, and define a kernel function depending only on the distance $||x_n - x_m||$, we obtain an expansion in radial basis function.


1
0.5
0
−0.5
−1
0
0.2
0.4
0.6
0.8
1

Computation time for GP regression

1. Training:
   - GP: inversion of a $N \times N$ matrix $O(N^3) + O(N^2)$.
   - Linear basis function model: inversion of a $M \times M$ matrix $O(M^3) + O(M^2)$.
2. Prediction:
   - GP: $O(N)$.
   - Linear basis function: $O(M)$.

Advantages of GP

- If the number of basis functions is larger than the number of data points, GP is computionally more efficient.
- Donot need to construct the basis function.
- Can learn the hyperparameters (maximum likelihood estimation)

Computation time for GP regression

1. Training:
   - GP: inversion of a $N \times N$ matrix $O(N^3) + O(N^2)$.
   - Linear basis function model: inversion of a $M \times M$ matrix $O(M^3) + O(M^2)$.
2. Prediction:
   - GP: $O(N)$.
   - Linear basis function: $O(M)$.

Advantages of GP

- If the number of basis functions is larger than the number of data points, GP is computionally more efficient.
- Donot need to construct the basis function.
- Can learn the hyperparameters (maximum likelihood estimation)

Automatic relevance determination

- Previous example doesn't consider the relevave importance of each dimension.
- Define a kernel as

$$k(x, x') = \theta_0 exp \left\{ -\frac{1}{2} \sum_{i=1}^{2} \gamma^i (x_i - x_i')^2 \right\}$$

- Atuomatically learn the hyperparameters resulting ARD which automatically determine the relative importance of each basis.

GP for Classification

- Similar to logistic/probit regression, using a nonlinear activation function to transform $(-\infty, +\infty)$ into probability interval $(0, 1)$.
- Assume latent variable a and the target output given latent variables are determined:

$$p(t|a) = \sigma(a)^t (1 - \sigma(a))^{1-t}$$

- Latent variables a follows the Gaussian Process
- For prediction,

$$p(t_{N+1} = 1|t_N) = \int p(t_{N+1} = 1|a_{N+1}) p(a_{N+1}|\mathbf{t}_N) da_{N+1}$$

Unfortunately, this is analytically intractable and may be approximated using sampling methods or analytical approximation.

GP for Classification

- Similar to logistic/probit regression, using a nonlinear activation function to transform $(-\infty, +\infty)$ into probability interval $(0, 1)$.
- Assume latent variable a and the target output given latent variables are determined:

$$p(t|a) = \sigma(a)^t (1 - \sigma(a))^{1-t}$$

- Latent variables a follows the Gaussian Process
- For prediction,

$$p(t_{N+1} = 1|t_N) = \int p(t_{N+1} = 1|a_{N+1}) p(a_{N+1}|\mathbf{t}_N) da_{N+1}$$

Unfortunately, this is analytically intractable and may be approximated using sampling methods or analytical approximation.

GP for Classification

- Similar to logistic/probit regression, using a nonlinear activation function to transform $(-\infty, +\infty)$ into probability interval $(0, 1)$.
- Assume latent variable a and the target output given latent variables are determined:

$$p(t|a) = \sigma(a)^t (1 - \sigma(a))^{1-t}$$

- Latent variables a follows the Gaussian Process
- For prediction,

$$p(t_{N+1} = 1|t_N) = \int p(t_{N+1} = 1|a_{N+1}) p(a_{N+1}|\mathbf{t}_N) da_{N+1}$$

Unfortunately, this is analytically intractable and may be approximated using sampling methods or analytical approximation.

GP for classification prediction

Gaussin approximation to the posterior distribution over a_{N+1}.

$$\begin{aligned} p(a_{N+1}|\mathbf{t}_N) &= \int p(a_{N+1}|\mathbf{a}_N)p(\mathbf{a}_N|\mathbf{t}_N)d\mathbf{a}_N \\ p(a_{N+1}|\mathbf{a}_N) &= \mathcal{N}(a_{N+1}|\mathbf{k}^T\mathbf{C}_N^{-1}\mathbf{a}_N, c - \mathbf{k}^T\mathbf{C}_N^{-1}\mathbf{k}) \end{aligned}$$

Need to estimate $p(\mathbf{a}_N|\mathbf{t}_N)$: use Gaussian Approximation

- The shape of single-mode distribution is close to Gaussian distribution.
- Increasing the number of data points falling in a fixed region of x space, then the corresponding uncertainty in the function $a(x)$ will decrease, asymptotically leading to a Gaussian.

GP for classification prediction

Gaussin approximation to the posterior distribution over a_{N+1}.

$$\begin{aligned} p(a_{N+1}|\mathbf{t}_N) &= \int p(a_{N+1}|\mathbf{a}_N)p(\mathbf{a}_N|\mathbf{t}_N)d\mathbf{a}_N \\ p(a_{N+1}|\mathbf{a}_N) &= \mathcal{N}(a_{N+1}|\mathbf{k}^T\mathbf{C}_N^{-1}\mathbf{a}_N, c - \mathbf{k}^T\mathbf{C}_N^{-1}\mathbf{k}) \end{aligned}$$

Need to estimate $p(\mathbf{a}_N|\mathbf{t}_N)$: use Gaussian Approximation

- The shape of single-mode distribution is close to Gaussian distribution.
- Increasing the number of data points falling in a fixed region of x space, then the corresponding uncertainty in the function $a(x)$ will decrease, asymptotically leading to a Gaussian.

Different approach to obtain a Gaussian approximation

1. variational inference
2. expectation propagation
3. Laplace approximation

Laplace Approximation

- $p(\mathbf{a}_N)$ is given by a zero-mean Gaussian process with covariance matrix C_N:

$$\begin{aligned} p(\mathbf{a}_N) &= \mathcal{N}(0, C_N) \\ p(\mathbf{t}_N|a_N) &= \prod_{n=1}^{N} \sigma(a_n)^{t_n}(1-\sigma(a_n))^{1-t_n} \sum_{n=1}^{N} e^{a_n t_n}\sigma(-a_n) \end{aligned}$$

$$\begin{aligned} \Psi(\mathbf{a}_N) &= \ln p(\mathbf{a}_N) + \ln p(\mathbf{t}_N|\mathbf{a}_N) \\ &= -\frac{1}{2}\mathbf{a}_N^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{a}_N - \frac{N}{2}\ln(2\pi) - \frac{1}{2}\ln|\mathbf{C}_N| + \mathbf{t}_N^{\mathrm{T}}\mathbf{a}_N \\ &\quad - \sum_{n=1}^{N}\ln(1+e^{a_n}) + \text{const.} \end{aligned}$$

$$\nabla\Psi(\mathbf{a}_N) = \mathbf{t}_N - \boldsymbol{\sigma}_N - \mathbf{C}_N^{-1}\mathbf{a}_N$$

$$\nabla\nabla\Psi(\mathbf{a}_N) = -\mathbf{W}_N - \mathbf{C}_N^{-1}$$

where w_N is a diagonal matrix with elements $\sigma(a_n)(1-\sigma(a_n))$.

- The hessian matrix $A = -\nabla\nabla\Psi(\mathbf{a}_N)$ is positive definite. So the posterior is log convex and has a single model that is the global maximum.

Laplace Approximation (2)

How to find the mode

Use Newton method,

$$
\begin{aligned}
a_N^{new} &= a_N^{old} - \nabla\nabla\Psi(a_N)^{-1}\nabla\Psi(a_N) \\
&= a_N^{old} + (W_N + C_N^{-1})^{-1}(t_N - \sigma_N - C_N^{-1}a_N) \\
&= C_N(I + W_N C_N)^{-1}\{t_N - \sigma_N + W_N a_N\}
\end{aligned}
$$

At the mode,

$$a_N^* = C_N(t_N - \sigma_N)$$

How to get the Hessian

$$H = -\nabla\nabla\Psi(a_N) = W_N + C_N^{-1}$$

$$q(a_N) = \mathcal{N}(a_N | a_N^*, H^{-1}) \tag{1}$$

Laplace Approximation (2)

How to find the mode

Use Newton method,

$$\begin{aligned} a_N^{new} &= a_N^{old} - \nabla\nabla\Psi(a_N)^{-1}\nabla\Psi(a_N) \\ &= a_N^{old} + (W_N + C_N^{-1})^{-1}(t_N - \sigma_N - C_N^{-1}a_N) \\ &= C_N(I + W_N C_N)^{-1}\{t_N - \sigma_N + W_N a_N\} \end{aligned}$$

At the mode,

$$a_N^* = C_N(t_N - \sigma_N)$$

How to get the Hessian

$$H = -\nabla\nabla\Psi(a_N) = W_N + C_N^{-1}$$

$$q(a_N) = \mathcal{N}(a_N | a_N^*, H^{-1}) \tag{1}$$

Laplace Approximation for Prediction(1)

$$
\begin{aligned}
p(a_{N+1}|\mathbf{t}_N) &\approx \int p(a_{N+1}|\mathbf{a}_N)q(\mathbf{a}_N|\mathbf{t}_N)d\mathbf{a}_N \\
E[a_{N+1}|t_N] &= k^T(t_N - \sigma_N) \\
var[a_{N+1}|t_N] &= c - k^T C_N^{-1} k + k^T C_N^{-1}(W_N + C_N^{-1})^{-1} C_N^{-1} k \\
&= c - k^T C_N^{-1} k + k^T (C_N W_N C_N + C_N)^{-1}) k \\
&= c - k^T C_N^{-1} k + k^T (C_N^{-1} - C_N^{-1} C_N (W_N^{-1} + C_N)^{-1} C_N C_N^{-1}) \\
&= c - k^T C_N^{-1} k + k^T (C_N^{-1} - (W_N^{-1} + C_N)^{-1}) k \\
&= c - k^T (W_N^{-1} + C_N)^{-1} k
\end{aligned}
$$

Laplace Approximation for Prediction(1)

$$
\begin{aligned}
p(a_{N+1}|\mathbf{t}_N) &\approx \int p(a_{N+1}|\mathbf{a}_N) q(\mathbf{a}_N|\mathbf{t}_N) d\mathbf{a}_N \\
E[a_{N+1}|t_N] &= k^T(t_N - \sigma_N) \\
var[a_{N+1}|t_N] &= c - k^T C_N^{-1} k + k^T C_N^{-1} (W_N + C_N^{-1})^{-1} C_N^{-1} k \\
&= c - k^T C_N^{-1} k + k^T (C_N W_N C_N + C_N)^{-1}) k \\
&= c - k^T C_N^{-1} k + k^T (C_N^{-1} - C_N^{-1} C_N (W_N^{-1} + C_N)^{-1} C_N C_N^{-1}) \\
&= c - k^T C_N^{-1} k + k^T (C_N^{-1} - (W_N^{-1} + C_N)^{-1}) k \\
&= c - k^T (W_N^{-1} + C_N)^{-1} k
\end{aligned}
$$

Laplace Approximation for Prediction(2)

- Recall that

$$\begin{aligned} p(a_{N+1}|\mathbf{t}_N) &= \int p(a_{N+1}|\mathbf{a}_N)p(\mathbf{a}_N|\mathbf{t}_N)d\mathbf{a}_N \\ &= \int \sigma(a_{N+1})p(a_{N+1}|\mathbf{t}_N)da_{N+1} \end{aligned}$$

- Use a probit function to approximate the sigmoid function:

$$\sigma(a) \approx \Phi(\lambda a) \qquad \text{where} \quad \lambda^2 = \frac{\pi}{8}$$

$$\int \Phi(\lambda a)\mathcal{N}(a|\mu, \sigma^2)da = \Phi\left(\frac{\mu}{(\lambda^{-2} + \sigma^2)^{\frac{1}{2}}}\right)$$

Connection to Neural Network

- The functions represented by a neural network is governed by the number of hidden units (M). Hence, the number of hidden units is limited based on the size of training data to avoid over-fitting. In a Bayesian perspective, it makes no sense to limit the number of parameters according to the size of training data.
- For a broad class of prior distributions over w, the distribution of functions generated by a neural network will tend to a Gaussian process in the limit $M \rightarrow \infty$.
- In the limit, the output variables of the neural network are independent. But in neural network, they can still borrow strength from each other.

END
TS