Bayesian Regression using Gaussian Process priors

Carl Edward Rasmussen
carl@cs.toronto.edu
http://www.cs.toronto.edu/~carl

Dept. of Computer Science
University of Toronto

Institute for Mathematical Modelling
Technical University of Denmark

Rambøll Informatics

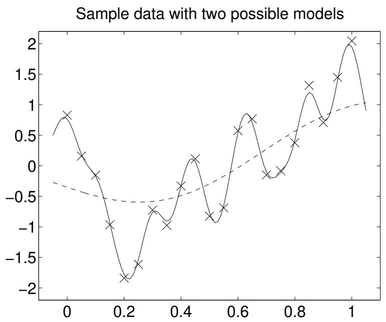
Take-home message

Gaussian Processes are:

- neat; both theoretical and applicational interest
- simple; you can use them at home
- powerful; in DELVE GP's perform better than the rest

It's not a new idea [O'Hagan, 1978]. GP's are closely related to spline models, regularisation networks, neural networks, variable metric kernel methods, support vector machines and nearest neighbor methods.

A language for priors

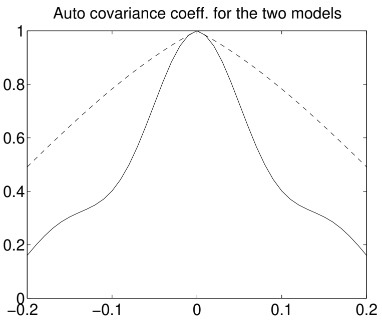


We 'best' function is the one that has a suitable trade-off between accuracy and 'simplicity'.

How do we measure simplicity? There are many suggestions:

- use a regularizer, eg: $\int (f'(x))^2 dx$
- penalize weights in a neural network

Let's use the covariance structure



The plot shows $E[(y^{(i)} - \mu)(y^{(j)} - \mu)]$ normalised as a function of $x^{(i)} - x^{(j)}$. The auto covariance drops much faster for the more wobbly function.

Is it possible to specify a prior in terms of a covariance structure?

Yes. This is the central idea in the following.

Gaussian Process

Reverse engineering: How to construct a function with a specified covariance structure

Begin by specifying a covariance function, eg:

$$\text{cov}\left(y^{(i)}, y^{(j)}\right) = \nu \exp\left(-\frac{(x^{(i)} - x^{(j)})^2}{2\sigma^2}\right). \tag{2}$$

Here the parameter ν gives the overall scale of variance, and σ defines a length scale on the inputs.

The concept of a Gaussian Process will help us get from this covariance specification to an actual function.

A Gaussian Process is: a collection of random variables, any finite set of which have a joint Gaussian distribution.

A multivariate Gaussian distribution is fully specified by its mean and covariance matrix.

Generating functions from a Gaussian Process

Select a (finite) collection of inputs, $\mathbf{x} = (x^{(1)}, \ldots x^{(n)})^\top$, where you want to evaluate your function.

Invent a set of random variables, $\mathbf{y} = (y^{(1)}, \ldots, y(n))^\top$, to represent the corresponding function values.

Use a Gaussian Process to define the joint distribution between the y's:

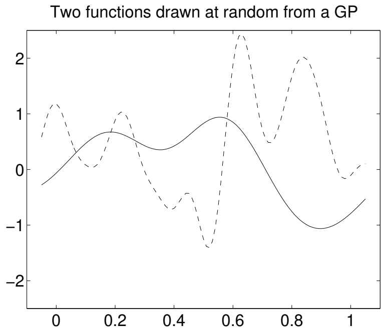
$$p(\mathbf{y}|\mathbf{x}) \propto \exp\left(-\frac{1}{2}\mathbf{y}^\top \Sigma^{-1} \mathbf{y}\right), \tag{4}$$

where the covariance matrix Σ is given by the covarince function:

$$\Sigma_{ij} = \text{cov}\left(y^{(i)}, y^{(j)}\right). \tag{6}$$

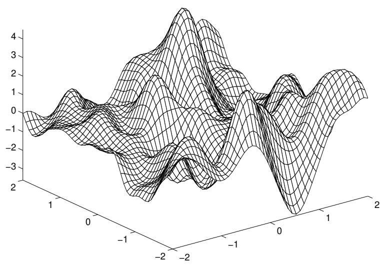
Now that we have a probabilistic model for $\mathbf{y}$, we draw can random functions, by drawing samples from the joint Gaussian distribution.

Random functions drawn from a GP



Parameters: $\nu = 1$ and $\sigma = 0.05$ or $\sigma = 0.15$.

We can easily generalise this to more input dimensions:



How do we use this for predictions?

Gaussian Process Prior over functions

$$p\left(y^{(1)}, \ldots, y^{(n+1)} \middle| x^{(1)}, \ldots, x^{(n+1)}\right) \propto \exp\left(-\frac{1}{2}\mathbf{y}^\top \Sigma^{-1} \mathbf{y}\right). \tag{8}$$

Example covariance function

$$\begin{aligned}\Sigma_{pq} &= \text{cov}\left(y^{(p)}, y^{(q)}\right) = C\left(x^{(p)}, x^{(q)}\right) \\ &\propto \exp\left(-\frac{1}{2}\sum_{j=1}^{m}\left(x_j^{(p)} - x_j^{(q)}\right)^2\right).\end{aligned} \tag{10}$$

Likelihood: assume Gaussian noise:

$$p\left(t^{(1)}, \ldots, t^{(n)} \middle| y^{(1)}, \ldots, y^{(n)}\right) \propto \exp\left(-\frac{1}{2}(\mathbf{y} - \mathbf{t})^\top \Omega^{-1} (\mathbf{y} - \mathbf{t})\right), \tag{12}$$

where $\Omega = r^2 I$.

Now, turn the crank ...

Posterior, by Bayes' rule

$$p\Big(y^{(1)},\dots,y^{(n+1)}\Big|x^{(n+1)},\mathcal{D}\Big) \propto \qquad (14)$$
$$p\Big(t^{(1)},\dots,t^{(n)},y^{(1)},\dots,y^{(n+1)}\Big|x^{(1)},\dots,x^{(n+1)}\Big).$$

Perform two steps on the posterior distribution:

1) Marginalise $y^{(1)},\dots,y^{(n)}$ to obtain

$$p\Big(t^{(1)},\dots,t^{(n)},y^{(n+1)}\Big|x^{(1)},\dots,x^{(n+1)}\Big) \propto \exp\left(-\frac{1}{2}\tilde{\mathrm{t}}^\top\big(\Sigma^{-1}+\Omega^{-1}\big)\tilde{\mathrm{t}}\right), \qquad (16)$$

where $\tilde{\mathrm{t}} = \Big(t^{(1)},\dots,t^{(n)},y^{(n+1)}\Big)$.

2) Condition on the observed targets to obtain

$$p\Big(y^{(n+1)}\Big|x^{(n+1)},\mathcal{D}\Big) \sim \mathcal{N}(\mu,\sigma^2), \qquad (18)$$

where

$$\begin{aligned} &\mu = \mathbf{a}^\top Q^{-1}\mathbf{t} \qquad \sigma^2 = b - \mathbf{a}^\top Q^{-1}\mathbf{a} \\ &Q_{pq} = C\Big(x^{(p)},x^{(q)}\Big) + r^2\delta_{pq}, \quad p,q=1,\dots,n \\ &\mathbf{a}_p = C\Big(x^{(n+1)},x^{(p)}\Big), \quad p=1,\dots,n \\ &b = C\Big(x^{(n+1)},x^{(n+1)}\Big). \end{aligned} \qquad (20)$$

Comments

Because the distributions are Gaussian, we can do all the computations analytically

We get a predictive *distribution*, not just a point prediction

The mean prediction is *linear* in the training targets

The computational work scales with the number of training points like n^3 allowing upto $n = 1000$, but we can have many input dimensions

We can also use this for noise-free data

Applying the model to data

Using a parameterised *Automatic Relevance Determination* (ARD) covariance function

$$C\Big(x^{(p)},x^{(q)}\Big) = \nu \exp\left(-\frac{1}{2}\sum_{j=1}^{m} w_j\Big(x_j^{(p)}-x_j^{(q)}\Big)^2\right), \qquad (22)$$

with *hyperparameters* $\theta = (\nu, w_1,\dots,w_m, r^2)$.

The likelihood for θ is

$$\begin{aligned} &\log p(\mathcal{D}|\theta) \\ &\quad = \log p\Big(t^{(1)},\dots,t^{(n)}\Big|x^{(1)},\dots,x^{(n)},\theta\Big) \\ &\quad = -\frac{1}{2}\log\det Q - \frac{1}{2}\mathrm{t}^\top Q^{-1}\mathrm{t} - \frac{n}{2}\log(2\pi), \end{aligned} \qquad (24)$$

with partial derivatives

$$\begin{aligned} &\frac{\partial}{\partial\theta_i}\log p(\mathcal{D}|\theta) \\ &\quad = -\frac{1}{2}\mathrm{trace}\Big(Q^{-1}\frac{\partial Q}{\partial\theta_i}\Big) + \frac{1}{2}\mathrm{t}^\top Q^{-1}\frac{\partial Q}{\partial\theta_i}Q^{-1}\mathrm{t} \end{aligned} \qquad (26)$$

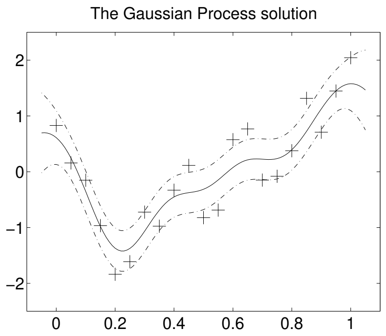
Matlab example

```
function C = covariance(theta, x1, x2, noise)
for i=1:length(x1);
  for j=1:length(x2);
    C(i,j) = exp(theta(1)) ..
        * exp(-0.5*(x1(i)-x2(j))^2/exp(theta(2)));
  end
  if noise == 1, C(i,i) = C(i,i) + exp(theta(3)); end
end

function minusloglik = f(theta, x, t)
Q = covariance(theta, x, t, 1);
minusloglik = 0.5*log(det(Q))+t'*inv(Q)*t ..
        + 0.5*length(x)*log(2*pi);

theta=fmins('f', [0 0 0], 0, [], x_train,t_train);
A = covariance(theta, x_test, x_train, 0);
Q = covariance(theta, x_train, x_train, 1);
plot(x_test, A*inv(Q)*t_train)
```

Gaussian Process model



Dash-dotted lines indicate one standard deviation. The parameters found where signal variance $\nu = 1.7$, input length scale, $\sigma = 0.12$ and noise variance $r^2 = 0.4$.

Priors for infinite networks

Consider networks with independent Gaussian weight priors and input $x^{(p)}$. Under the prior the hidden unit activities are independent, identically distributed and bounded with variance $V_h(x^{(p)})$. Thus

$$E[f(x^{(p)})] = E\Big[\sum_{i=1}^{H} v_i h_i(x^{(p)})\Big] = 0,$$

$$E[(f(x^{(p)})^2)] = H\sigma_v^2 V_h(x^{(p)}) + \sigma_b^2.$$

Central limit theorem: for an infinite network the output is Gaussian with variance $H\sigma_v^2 V_h(x^{(1)}) + \sigma_b^2$; therefore, scale the prior σ_v^2 as H^{-1}.

As $H \to \infty$ the joint distribution of outputs corresponding to different inputs become Gaussian with zero mean and covariance

$$\begin{aligned} E[f(x^{(p)})f(x^{(q)})] &= \sigma_b^2 + \sum_{i=1}^{H} \sigma_v^2 E[h_i(x^{(p)})h_i(x^{(q)})] \\ &= \sigma_b^2 + C(x^{(p)}, x^{(q)}) \end{aligned}$$

Gaussian Process: the joint distribution of any finite number of variables is multivariate Gaussian.

Conclusions

GP's are a simple, practical, powerful, Bayesian tool for data analysis for upto about 1000 cases.

One can use more complicated parameterised covariance functions for ARD, additive functions, input dependent noise, etc.

GP's can be extended for classification and non-Gaussian noise models

Software and references:
`http://www.cs.toronto.edu/~carl/gp.html`