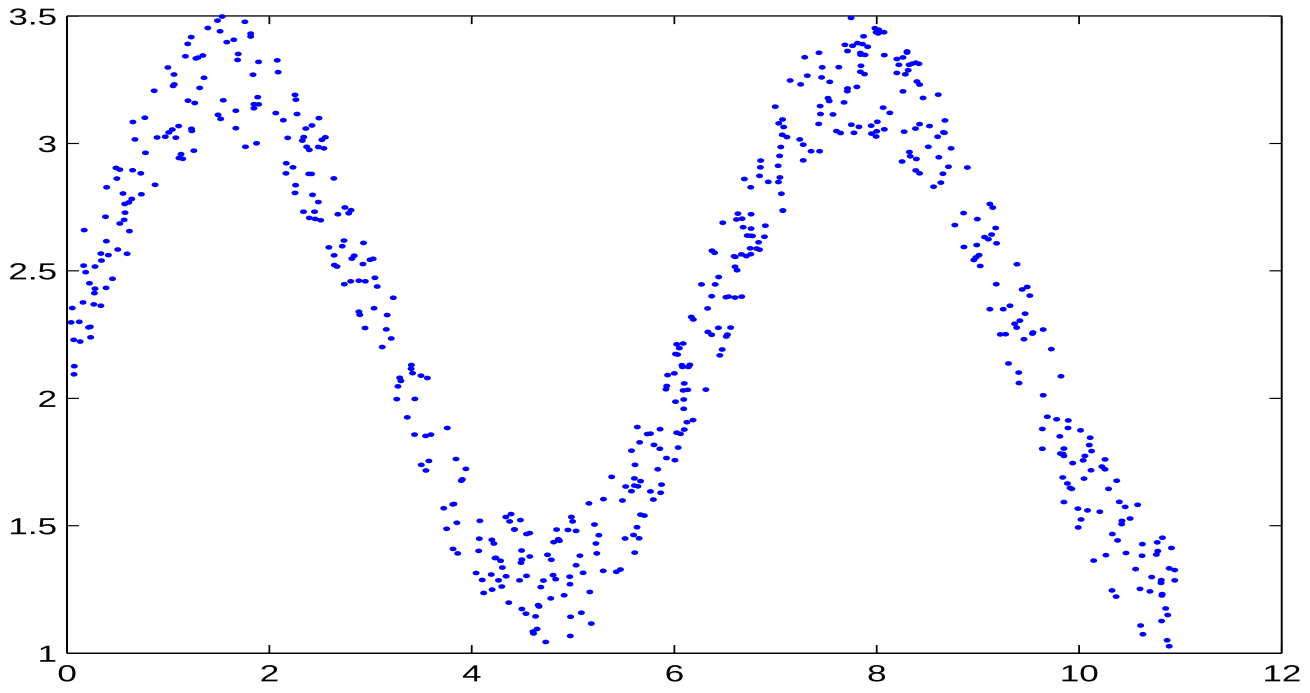
Nonparametric Regression With Gaussian Processes

From Chap. 45, Information Theory, Inference and Learning Algorithms, D. J. C. McKay

Presented by Micha Elsner

A Non-linear Dataset

We have a group of N data points in d-dimensional space, X, and associated values t.



Linear Least-Squares Regression

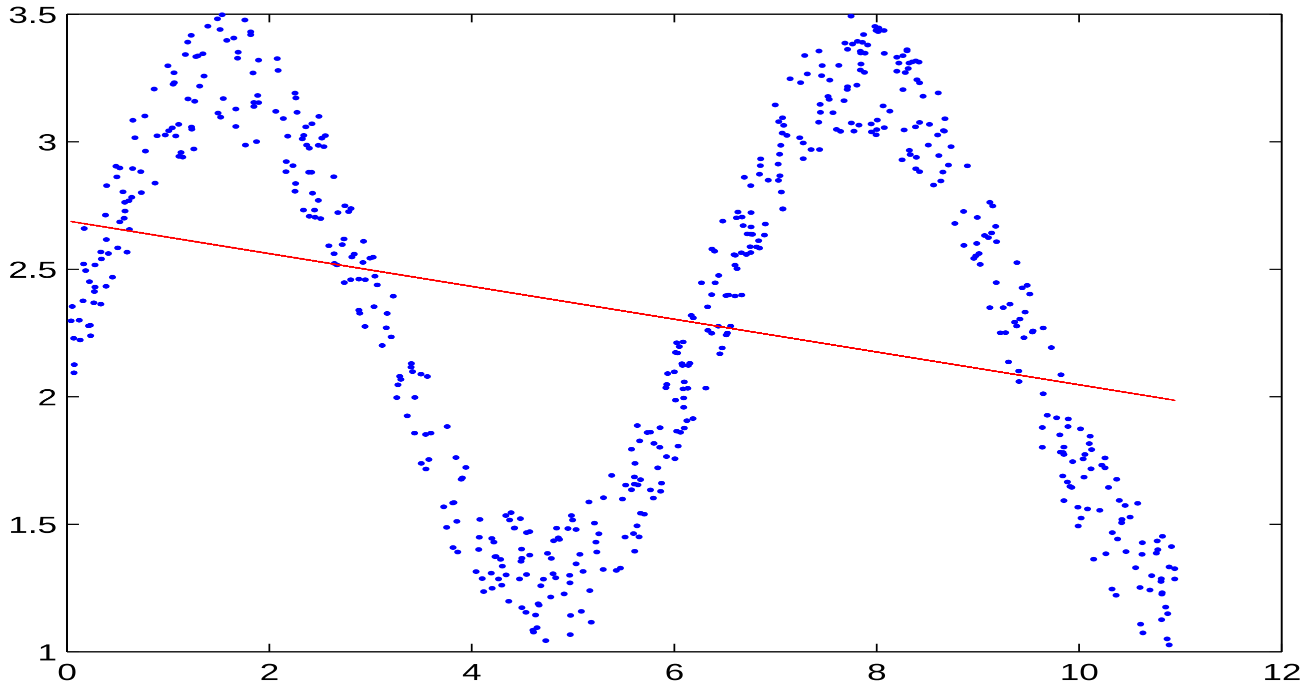
We wish to find a function $y(x)$ for which the mean squared error is small:

$$\hat{y}(x) = argmin_y \frac{1}{N} \sum_i (y(x_i) - t_i)^2$$

We hypothesize that y is a linear function of the training data parameterized by the weight vector w,
$y(x; w) = w_0 + w^T x$.
Our optimal w is now:

$$w^* = (X^T X)^{-1} X^T t$$

Which looks like this:



A Statistical Model

Linear least-squares regression is a procedure for finding a w; let's turn it into a model for our data.
We suppose that there exists a function $y(x; w^*)$, and our data t is generated from this function, then corrupted by Gaussian white noise (mean 0, variance σ_ν^2).

$$t_i = y(x_i; w^*) + \nu_i$$
$$\nu_i \sim N(\cdot|0, \sigma_\nu^2)$$

In this case, our data t consists of samples from a Gaussian:

$$t(x) \sim N(\cdot|y(x; w), \sigma_\nu^2)$$

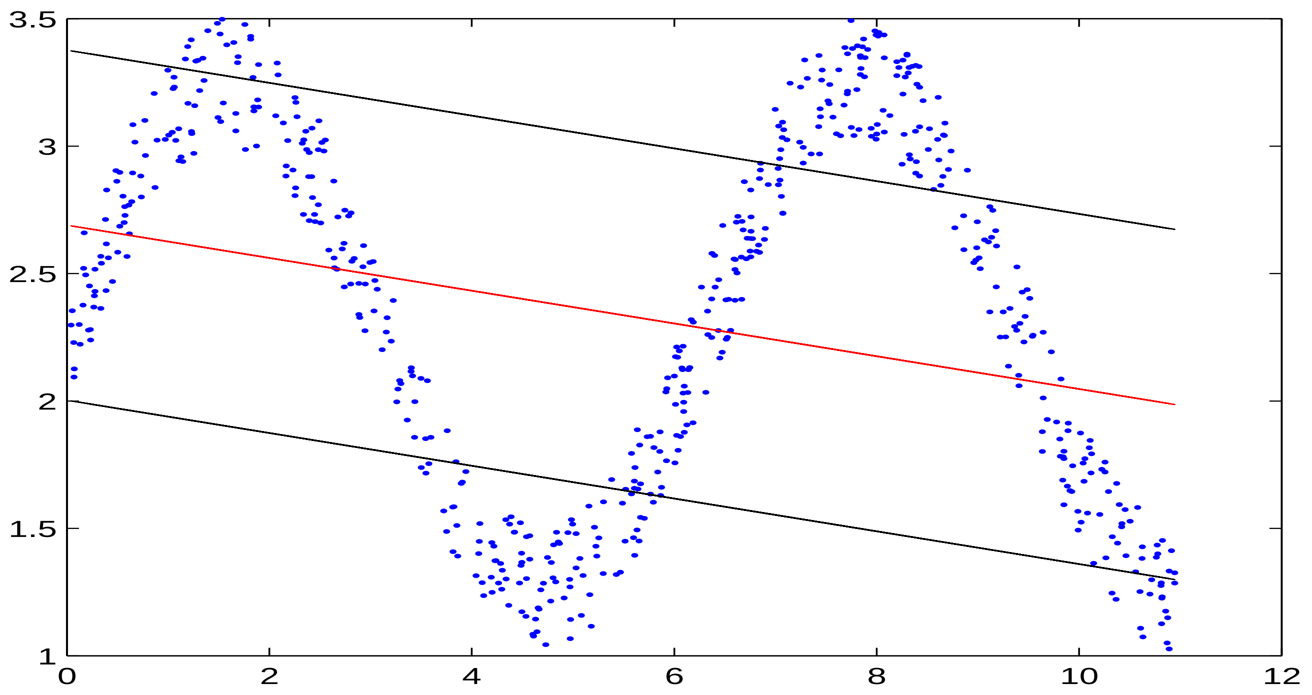
Bayesian inference

Now the probability of our parameter w is:

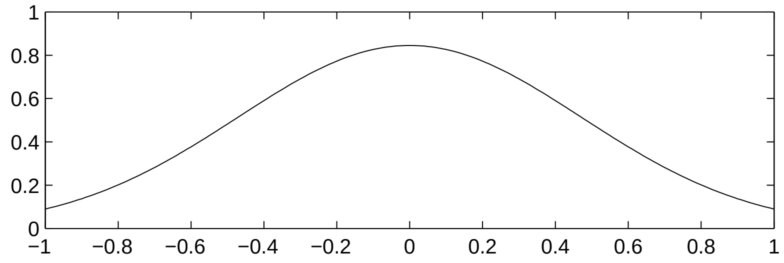
$$p(w, \sigma_\nu^2 | t, X) = \frac{p(t | y(x; w, \sigma_\nu^2), X) p(w, \sigma_\nu^2)}{p(t|X)}$$

The solution (in case of an appropriate form for y and a prior over w and σ_ν^2) will be the same least-squares solution as before.

Only now we have error bars



Gaussian white noise:



Non-parametric Bayesian inference

We could dispense with the parameter w and compute over the space of functions directly:

$$p(y|t, X) = \frac{p(t|y(x), X)p(y(x))}{p(t|X)}$$

This is a generalization of the previous equation; to work with it we need to be able to define priors over the infinite space of functions y.
However, we'd like to retain some useful properties:

- Simple descriptions of our favorite parametric models.
- Tractable (finite-space) representation of the posterior.
- Distribution of the data given the model is Gaussian.

Gaussian Processes

A prior which is a Gaussian process has all the properties we desire.
A Gaussian process is a distribution over the space of functions, with mean function $\mu(x)$ (a *function* because it need not be stationary– as above it may be something like $f(x;w)$) and an operator A analogous to the covariance:

$$P(y(x)|\mu(x), A) = \frac{1}{Z} exp(-\frac{1}{2}(y(x) - \mu(x))^T A(y(x) - \mu(x)))$$

The inner product $(y(x) - \mu(x))^T A(y(x) - \mu(x))$ is defined as:

$$\int (y(x) - \mu(x))^T A(y(x) - \mu(x)) dx$$

Linear Regression Revisited

Back in our simple linear regression model, let's impose a prior on w; unsurprisingly, let's make it Gaussian, with mean 0 and spherical covariance $\sigma_w^2 I$.

$$w \sim N(\cdot|0, \sigma_w^2 I)$$

Let Y be the value of $y(x)$ at each point x.
What are our prior beliefs about Y like? Y must still be Gaussian, and has expected mean 0.
The covariance $Q = YY^T$ is:

$$= (Xw)(Xw)^T$$
$$= X(ww^T)X^T$$
$$= X\sigma_w^2 X^T$$

Covariance Terms

t is Y, plus the additive Gaussian noise terms ν. It's still Gaussian with mean 0.
The covariance C is Q, plus the noise covariance, $\sigma_\nu^2 I$.
Any particular element of Q is the data covariance at that point, times the variance in w:

$$Q_{i,j} = (\sigma_w^2 X X^T)_{i,j}$$

A particular element of C is:

$$C_{i,j} = Q_{i,j} + \mathbb{I}_{i=j}\sigma_\nu^2$$

The Kernel Trick

So far, we've been working explicitly in the input space (the dimensions of X– plus an extra feature always equal to 1, to allow a bias term).

We can project the data into feature space using a set of basis functions ϕ. One way to do this would be to form

$R = \left[\phi_1(x) ... \phi_H(x)\right]$.

But our set of basis functions could be huge, even infinite.

However, our covariance Q only depends on the dot products between vectors.

So all we really need is a function $C(x, x')$ that produces the entries of the covariance matrix Q.

Doing Inference

So suppose we have fixed C as some covariance function. Actually using our regression model (predicting t for some new x) requires inference over the posterior $p(t_{n+1}(x_{n+1}|t, X) = \frac{p(t_{n+1}(x_{n+1}),t(X))}{p(t)}$.
This distribution is a Gaussian (because our prior distribution over t is Gaussian). So the posterior is:

$$p(t_{n+1}(x_{n+1}|t(X)) \propto exp(-\frac{1}{2}[t_N t_{n+1}]C_{N+1}^{-1}[t_N t_{n+1}]^T)$$

This is just an ordinary Gaussian distribution. The tough part is the covariance matrix C_{N+1}^{-1}, which is the inverse of the covariance C for the data X *augmented by the new point* t_{n+1}.

Inverting the Covariance Matrix

Inverting matrices is hard. Ideally we only want to invert the data once, especially if we have many training points.
The inverse covariance of the augmented data matrix can be computed only in terms of the inverse covariance of the original data, C^{-1}, and the covariance of the new point against the original data, k.
The prediction of the new point is:

$$\hat{t}_{n+1} = k^T C^{-1} t$$

The (Gaussian) posterior distribution for the point is:

$$p(t_{n+1}(x_{n+1})|t(X)) \propto exp\left(-\frac{(t_{n+1} - \hat{t}_{n+1})^2}{2\sigma^2_{n+1}}\right)$$

Incremental learning

Suppose we get a new point x_{n+1} and its label t_{n+1}. We could recompute C_{n+1} and invert it all over again. But that would be expensive.
Notice that:

$$C_{n+1} = \begin{bmatrix} \begin{bmatrix} \quad C_n \quad \end{bmatrix} & \begin{bmatrix} k \end{bmatrix} \\ \begin{bmatrix} \quad k^T \quad \end{bmatrix} & \begin{bmatrix} \kappa \end{bmatrix} \end{bmatrix}$$

Here again k is the covariance of x_{n+1} with the training set:

$$k_i = C(x_i, x_{n+1})$$

κ is $C(x, x) + \sigma_\nu$ and $C(x, x)$ is usually 1.
We can compute the inverse C_{n+1}^{-1} without inverting the matrix from scratch.

Incremental inversion

$$C_{n+1} = \begin{bmatrix} \begin{bmatrix} \quad M \quad \end{bmatrix} & \begin{bmatrix} m \end{bmatrix} \\ \begin{bmatrix} \quad m^T \quad \end{bmatrix} & \begin{bmatrix} sm \end{bmatrix} \end{bmatrix}$$

(Partitioned inverse equations; Barnett 1979).

$$sm = (\kappa - k^T C_n^{-1} k)^{-1}$$

$$m = -sm C_n^{-1} k$$

$$M = C_n^{-1} + \frac{1}{sm} m m^T$$

A few useful priors

The *radial basis* kernel assumes that points near one another have similar values; the similarity between points is expressed as a Gaussian:

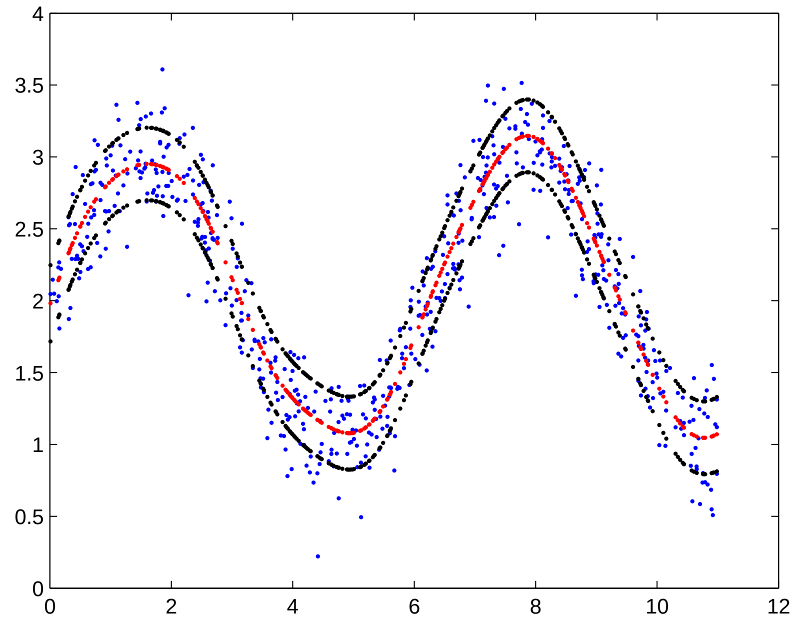
$$d(x, y) = exp - \frac{|x - y|^2}{2r^2}$$

If we assume our basis contains an infinite number of these functions and set our covariance function to:

$$C(x, y) = \theta_1 exp - \frac{|x - y|^2}{4r^2}$$

We get the same thing.

Result looks like:



Exploiting periodicity

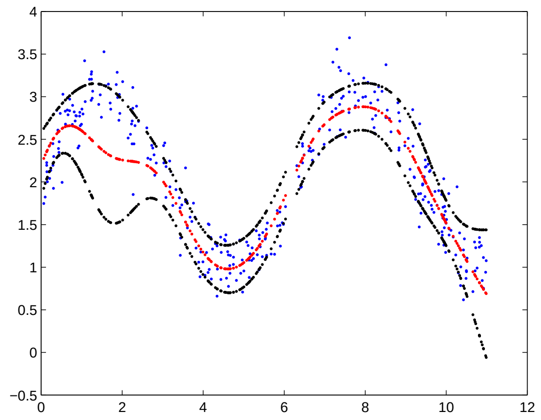
We can use a prior that assumes points nearby in the period of the function have similar values:

$$C(x, y) = \theta_1 exp - \frac{sin(.5 * (x - y))^2}{r}$$

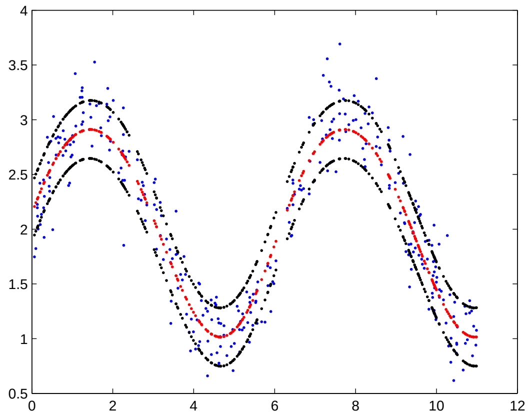
So $C(0, 2\pi)$ is 0.

Result looks like:

With only 25 training points we get some sparsity issues:



A sinusoidal prior helps us reconstruct the data:



Why Bayesian analysis?

Why not just use kernel methods?

- Error bars: how unlikely is a given prediction?
- Generative model: given an x, efficiently sample a t.
- Hierarchical inference: sample hyperparameters (the θ parameters that specify C, and the parameters for the noise model).
- Posterior inference: can compute data likelihood given the posterior distribution over the parameters, not a single set of parameters.

Review: How do you do it?

- Get some data.
- Decide on a covariance function C (with some hyperparameters θ).
- Guess the noise variance σ_ν.
- Construct the covariance matrix C, where $C_{i,j} = C(x_i, x_j)$ and $C_{i,i}$ has an additional term σ_ν.
- Invert the covariance matrix. (This is the hard part!)
- Predict for new points x_{n+1}: take the distances $k_i = C(i, n+1)$ and compute $k^T C^{-1} t$.

Remember, if you can't directly invert your training covariance matrix (~ 1000 points) you have to be cleverer!