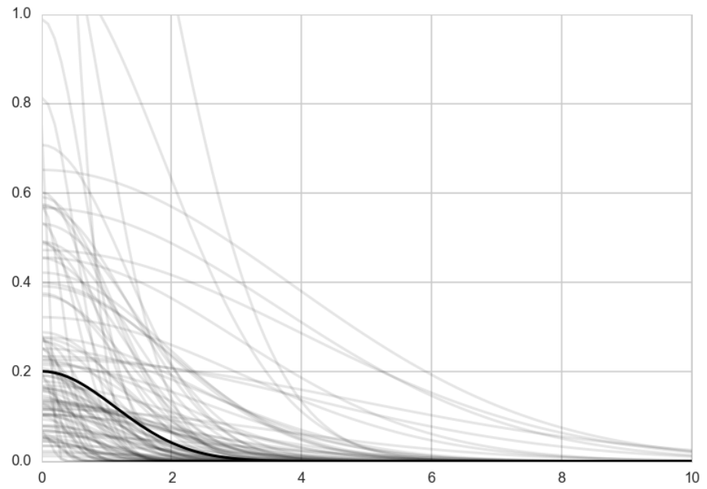
Lecture 25

GAUSSIAN PROCESSES

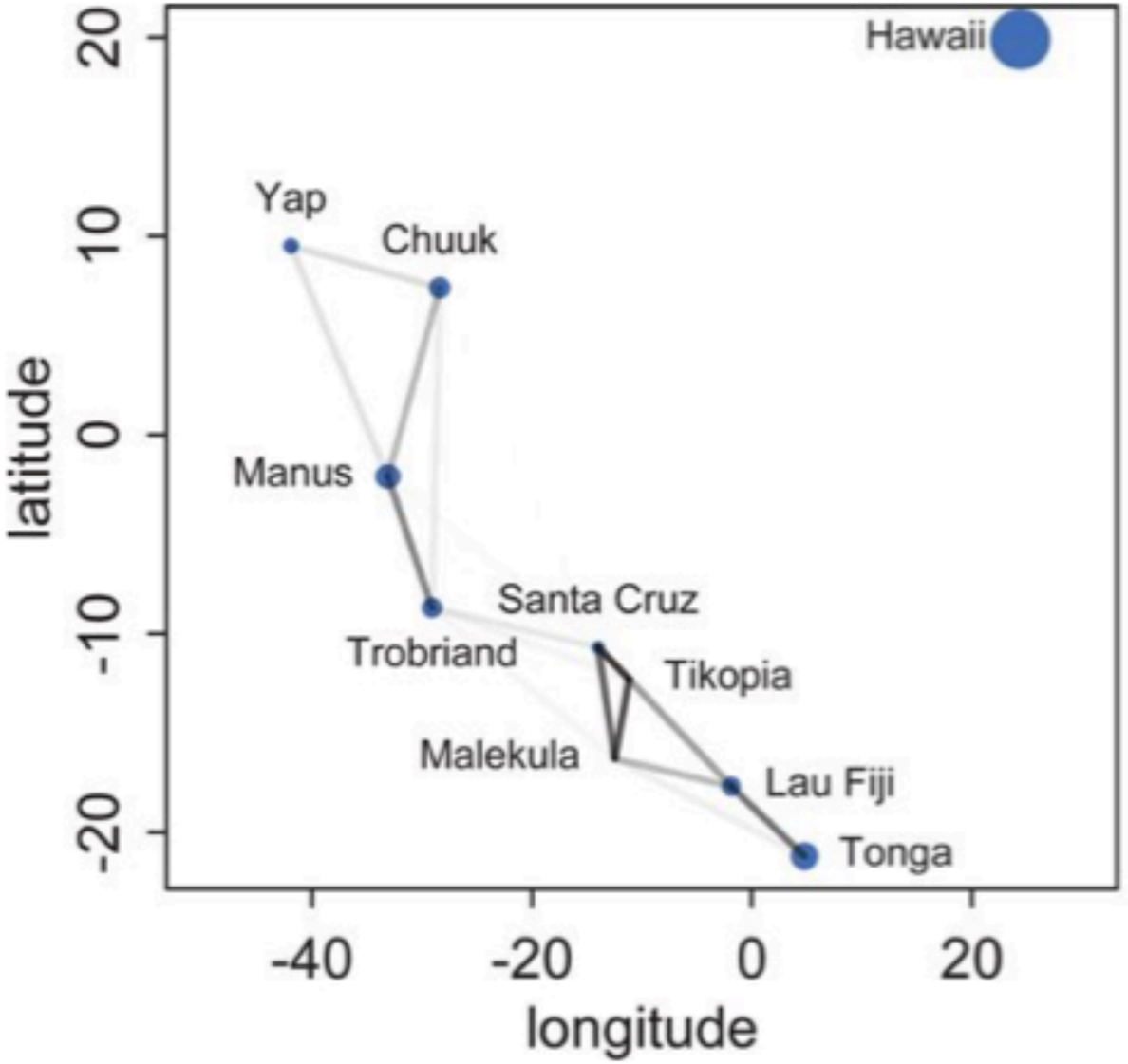
Last lab

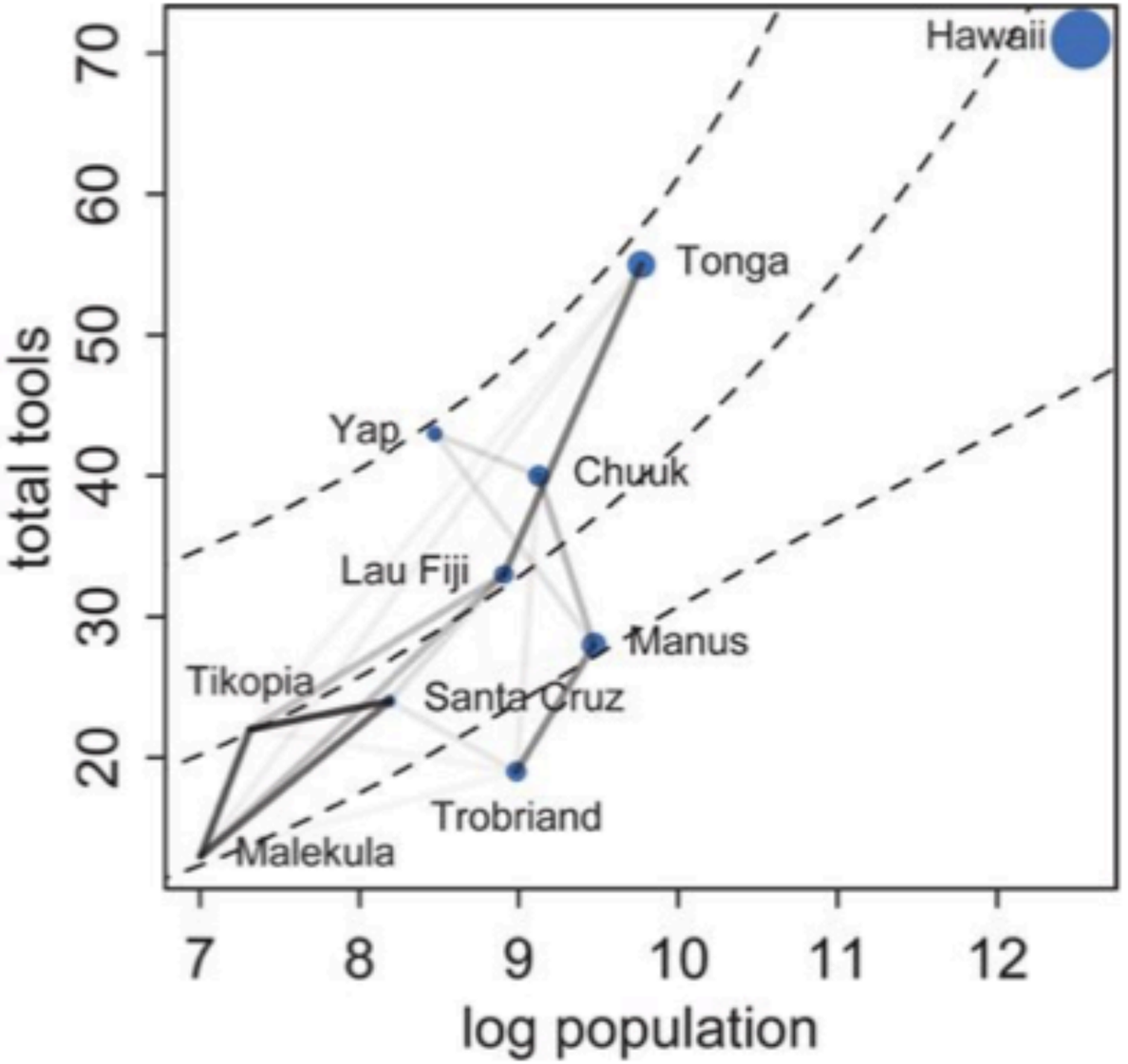
We modeled society specific intercepts for oceanic tools as draws from a 0 mean multivariate gaussian.

Covariance posteriors:



$$
\begin{aligned}
T_i &\sim \text{Poisson}(\lambda_i) \\
\log \lambda_i &= \alpha + \gamma_{\text{SOCIETY}[i]} + \beta_P \log P_i \\
\gamma &\sim \text{MVNormal}\big((0, \ldots, 0), \mathbf{K}\big) \\
\mathbf{K}_{ij} &= \eta^2 \exp(-\rho^2 D_{ij}^2) + \delta_{ij}(0.01) \\
\alpha &\sim \text{Normal}(0, 10) \\
\beta_P &\sim \text{Normal}(0, 1) \\
\eta^2 &\sim \text{HalfCauchy}(0, 1) \\
\rho^2 &\sim \text{HalfCauchy}(0, 1)
\end{aligned}
$$


Hawaii
Yap
Chuuk
Manus
Santa Cruz
Trobriand
Tikopia
Malekula
Lau Fiji
Tonga
latitude
longitude
20
10
0
-10
-20
-40
-20
0
20


Hawaii
Tonga
Yap
Chuuk
Lau Fiji
Manus
Tikopia
Santa Cruz
Trobriand
Malekula
total tools
log population
70
60
50
40
30
20
7
8
9
10
11
12

This is an example of a **gaussian process regression** for $log(\lambda)$.

The intercepts are modeled as coming from a MVN with a covariance matrix coming from a correlation function that depends on distance.

The key idea is that **nearer "distances" have similar intercepts, or effects**.

Lets see these concepts in more detail

MVN Primer

$$p(x \mid \mu, \Sigma) = (2\pi)^{-n/2}|\Sigma|^{-1/2}\exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}$$

JOINT: $p(x, y) = \mathcal{N}\left(\begin{bmatrix}\mu_x \\ \mu_y\end{bmatrix}, \begin{bmatrix}\Sigma_x & \Sigma_{xy} \\ \Sigma_{xy}^T & \Sigma_y\end{bmatrix}\right)$

MARGINAL: $p(x) = \int p(x, y)dy = \mathcal{N}(\mu_x, \Sigma_x)$

CONDITIONAL: $p(x \mid y) = \mathcal{N}(\mu_x + \Sigma_{xy}\Sigma_y^{-1}(y - \mu_y), \Sigma_x - \Sigma_{xy}\Sigma_y^{-1}\Sigma_{xy}^T)$

Modeling correlation

General expectation of continuity as you move from one adjacent point to another. In the absence of significant noise, two adjacent points ought to have fairly similar f values.

$$k(x_i, x_j) = \sigma_f^2 exp(\frac{-(x_i - x_j)^2}{2l^2})$$

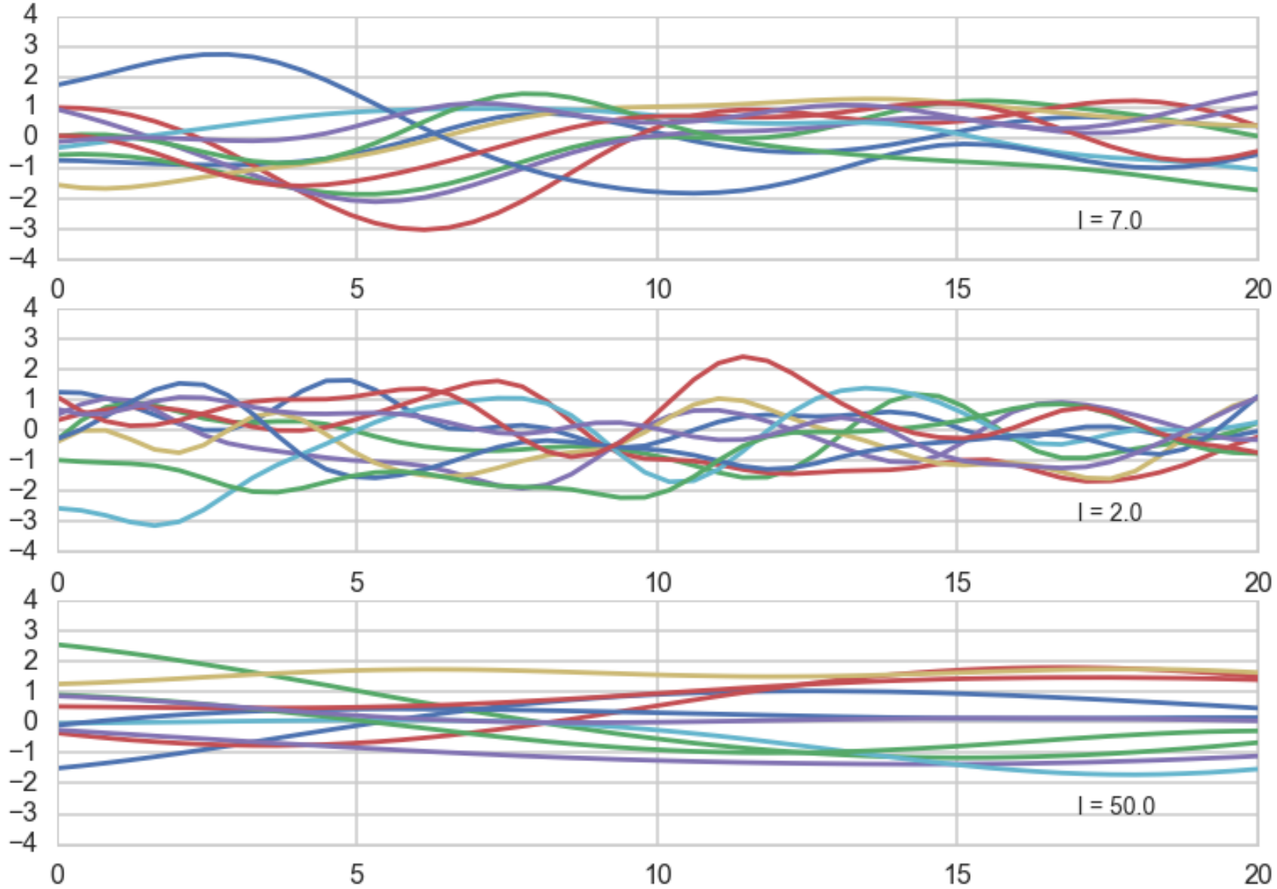
l is correlation length, σ_f^2 amplitude.

```
#Correlation Kernel
def exp_kernel(x1,x2, params):
    amplitude=params[0]
    scale=params[1]
    return amplitude * amplitude*np.exp(-((x1-x2)**2) / (2.0*scale))

#Covariance Matrix
covariance = lambda kernel, x1, x2, params: \
    np.array([[kernel(xi1, xi2, params) for xi1 in x1] for xi2 in x2])
```

Each curve in plots is generated as:

```
a = 1.0
nsamps = 50
xx = np.linspace(0,20,nsamps)

#Create Covariance Matrix
sigma = covariance(exp_kernel,xx,xx, [a,ell]) + np.eye(nsamps)*1e-06

#Draw samples from a 0-mean gaussian with cov=sigma
samples = np.random.multivariate_normal(np.zeros(nsamps), sigma)
```

The greater the correlation length, the smoother the curve.



What did we just do?

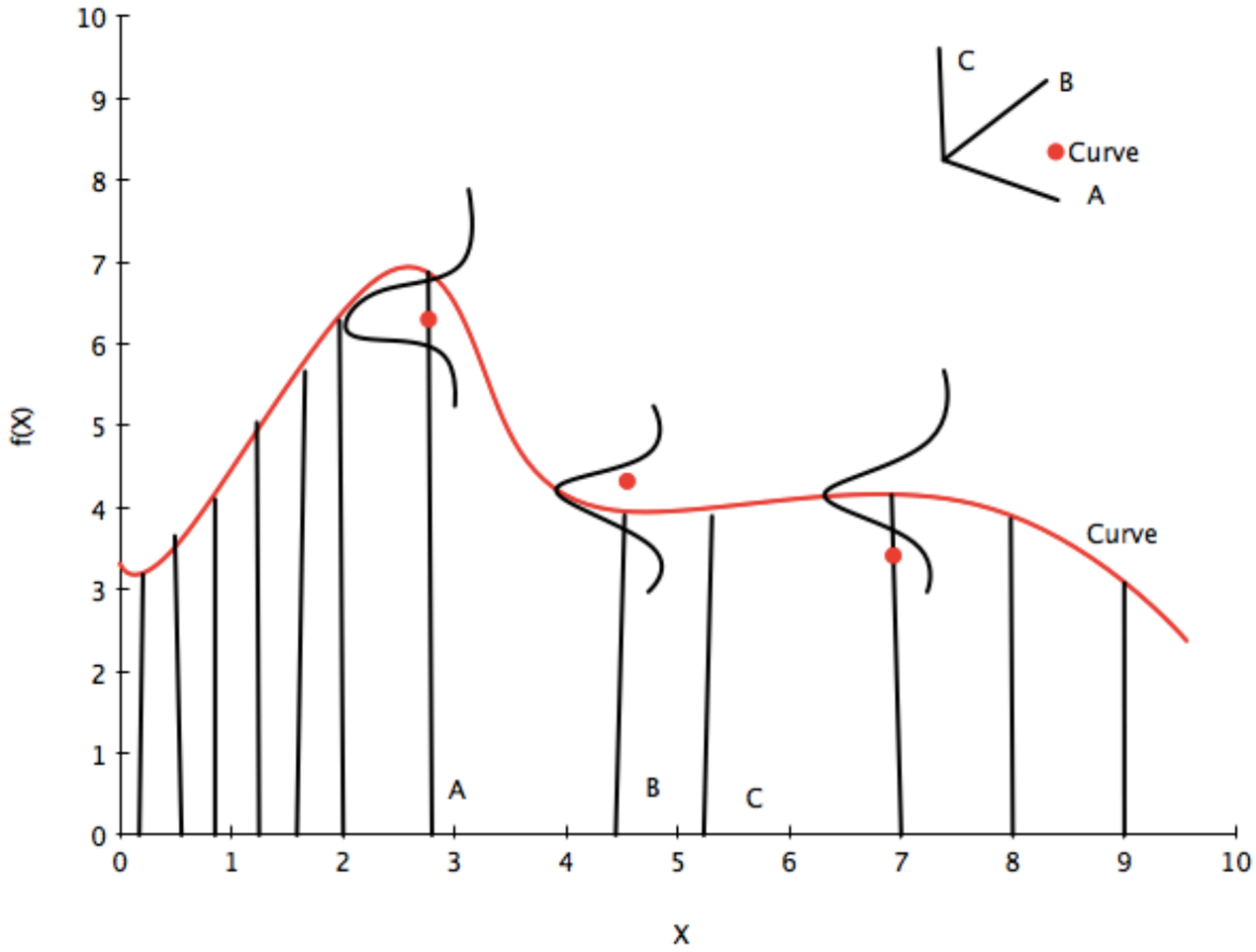
- we have not seen any data yet
- but we expect the function representing our data to have some level of continuity
- thus we considered different **PRIOR** functions that might represent our data
- as having come from MVNs with a covariance matrix based on the correlation length we think we have

The red curve represents one of these functions from the calculation above.

We have 3 red data points, so it would be seem to be one of the curves consistent with the data.

We can consider the curve as a point in a multi-dimensional space, a draw from a multivariate gaussian with as many points as points on the curve.

Consider the 3 red data points to have been generated IID from some regression function $f(x)$ (like $w \cdot x$) with some univariate gaussian noise σ^2 at each point.



Add some data

Augments the dimension of our multivariate normal. Then conditional directly gives us a predictive distribution!!

$$JOINT: \; p(y, f^*) = \mathcal{N}\left(\begin{bmatrix}\mu_y \\ \mu_{f^*}\end{bmatrix}, \begin{bmatrix}\Sigma_{yy} & \Sigma_{yf^*} \\ \Sigma_{yf^*}^T & \Sigma_{f^*f^*}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mu \\ \mu_*\end{bmatrix}, \begin{bmatrix}K + \sigma^2 I & K_* \\ K_*^T & K_{**}\end{bmatrix}\right)$$

$$MARGINAL: \; p(f^*) = \int p(f^*, y)dy = \mathcal{N}(\mu_*, K_{**})$$

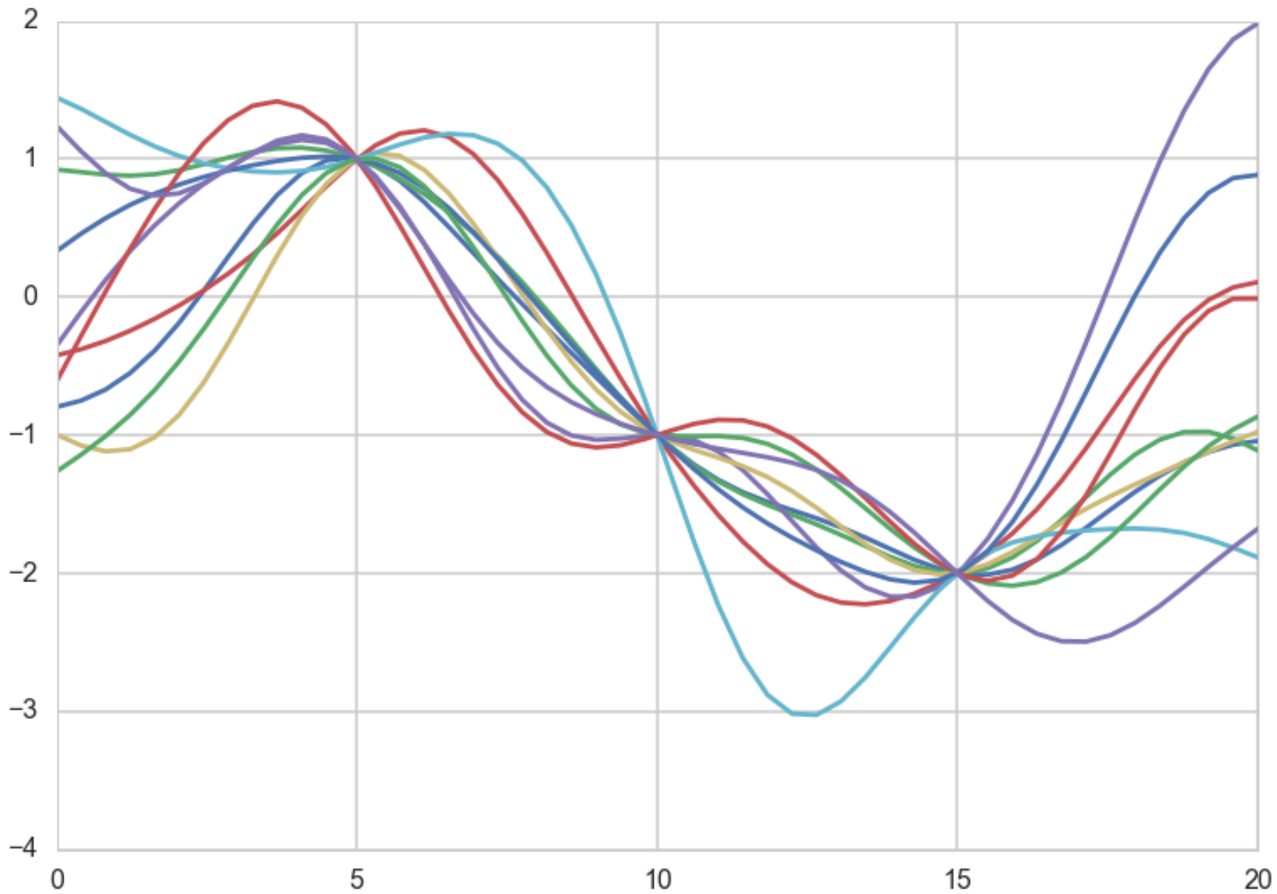
$$CONDITIONAL: \; p(f^* \mid y) = \mathcal{N}\left(\mu_* + K_*(K + \sigma^2 I)^{-1}(y - \mu), \; K_{**} - K_*(K + \sigma^2 I)^{-1}K_*^T\right)$$

Note here that:

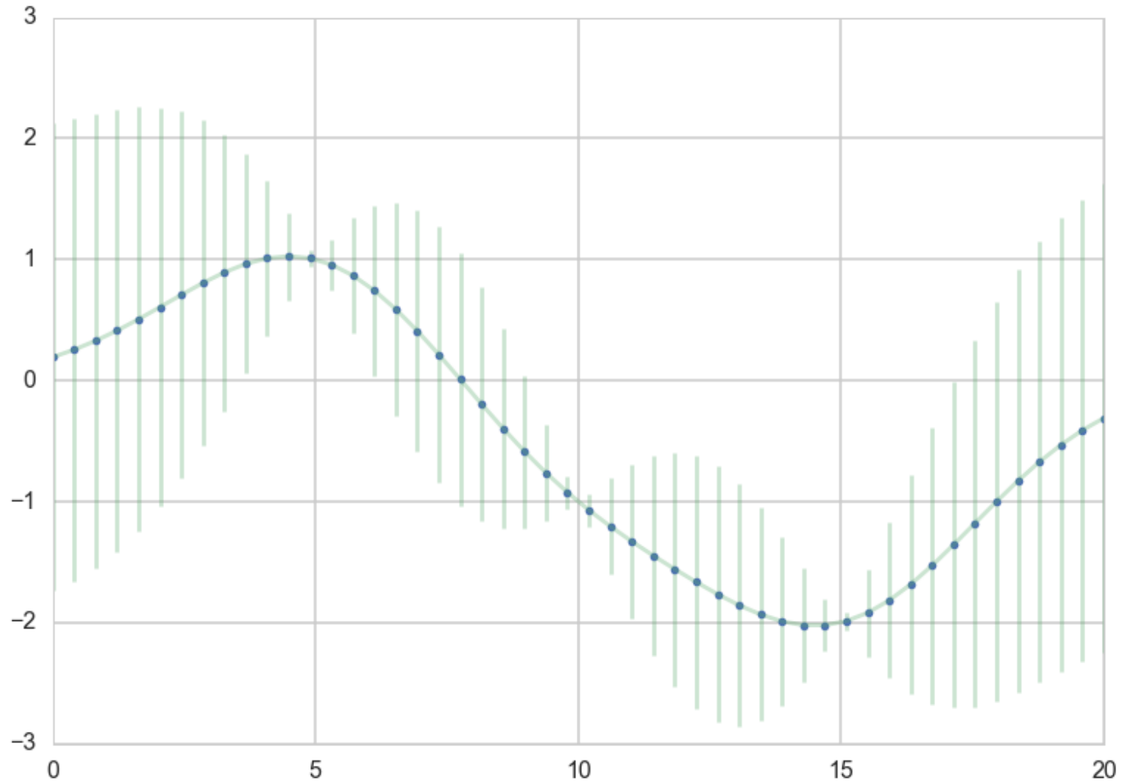
$$K = K(x, x); K_* = K(x, x^*); K_{**} = K(x^*, x^*)$$

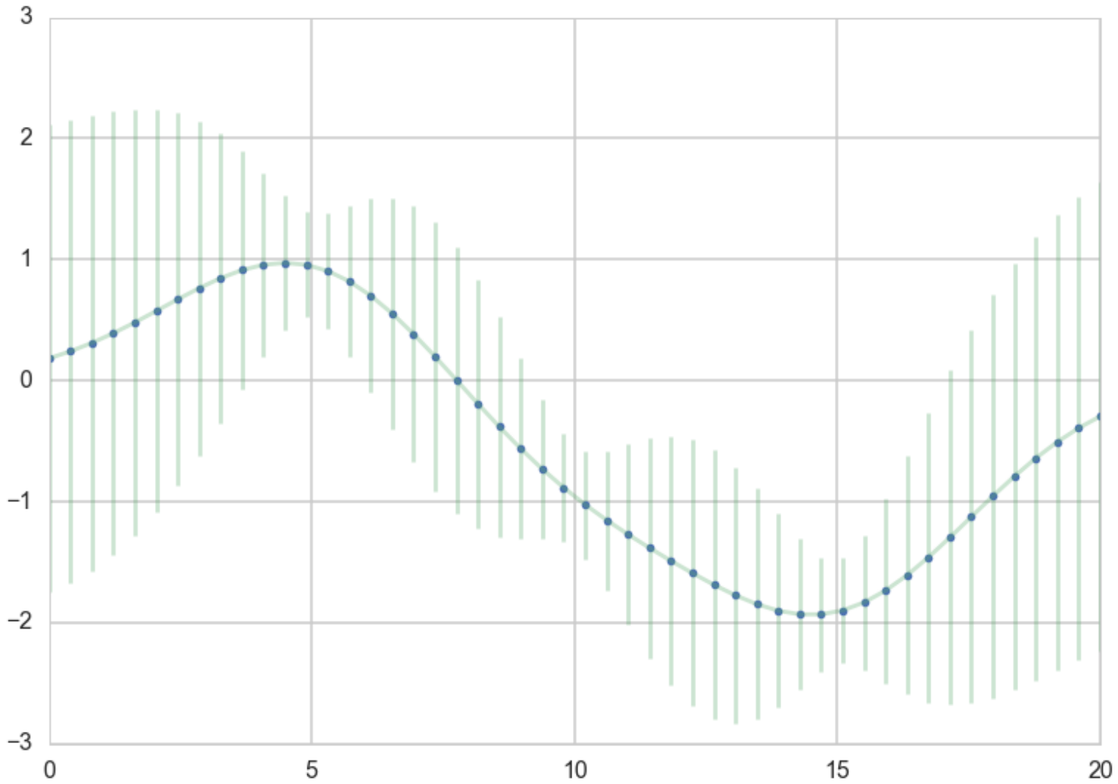
$l = 7$, added a small noise term.

```
#"test" data
x_star = np.linspace(0,20,nsamps)

# defining the training data
x = np.array([5.0, 10.0, 15.0]) # shape 3
f = np.array([1.0, -1.0, -2.0]).reshape(-1,1)


K = covariance(exp_kernel, x,x,[a,ell])
#shape 3,3

K_star = covariance(exp_kernel,x,x_star,[a,ell])
#shape 50, 3

K_star_star = covariance(exp_kernel, x_star, x_star, [a,ell])
#shape 50,50

K_inv = np.linalg.inv(K)
#shape 3,3

mu_star = np.dot(np.dot(K_star, K_inv),f)
#shape 50

sigma_star = K_star_star  - np.dot(np.dot(K_star, K_inv),K_star.T)
#shape 50, 50
```



Posterior and predictive





So far

1. We built a covariance matrix from a kernel function
2. Use the covariance matrix to generate a "curve" as a point in a multi-dimensional space from a MVN
3. multiple such curves serve as prior fits for our data
4. now we bring in the data and condition on it (with noise added if needed) using normal distribution formulae
5. the conditional has the form of a predictive and we are done
6. Also notice that the marginal only has quantities from the predictive block. This means that we dont care about the size of the original block in calculating the marginal.

These observations are the building blocks of the GP.

Formulae, again

$$JOINT: \; p(y, f^*) = \mathcal{N}\left(\begin{bmatrix} \mu_y \\ \mu_{f^*} \end{bmatrix}, \begin{bmatrix} \Sigma_{yy} & \Sigma_{yf^*} \\ \Sigma_{yf^*}^T & \Sigma_{f^*f^*} \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \mu \\ \mu_* \end{bmatrix}, \begin{bmatrix} K + \sigma^2 I & K_* \\ K_*^T & K_{**} \end{bmatrix}\right)$$

$$MARGINAL: \; p(f^*) = \int p(f^*, y) dy = \mathcal{N}(\mu_*, K_{**})$$

$$CONDITIONAL: \; p(f^* \mid y) = \mathcal{N}\left(\mu_* + K_*(K + \sigma^2 I)^{-1}(y - \mu), \; K_{**} - K_*(K + \sigma^2 I)^{-1} K_*^T\right)$$

Note here that:

$$K = K(x, x); K_* = K(x, x^*); K_{**} = K(x^*, x^*)$$

Parametric models

- In general, parametrization restricts the class of functions we use. If our data is not well modeled by our choices, we might underfit.
- Increasing flexibility might lead to overfitting.

INSTEAD: consider every possible function and associate a prior probability with this function. e.g. assign smoother functions higher prior probability. But how are we possibly going to calculate over an uncountably infinite set of functions in finite time?

Linear Regression

$$y = f(X) + \epsilon, \epsilon \sim N(0, \sigma^2), f(X) = X^T w, w \sim N(0, \Sigma)$$

Posterior: $p(w|X, y) \sim N(\frac{1}{\sigma^2} A^{-1} Xy, A^{-1})$ where $A = \frac{1}{\sigma^2} XX^T + \Sigma^{-1}$

Posterior predictive distribution:

$p(f(x_*)|x_*, X, y) = N(\frac{1}{\sigma^2} x_*^T A^{-1} Xy, x_*^T A^{-1} x_*).$

For posterior predictive of y_*, just add σ^2 to the variance above.

We can show that we can rewrite the above posterior predictive as:

$$p(f(x_*)|x_*, X, y) = N(x_*^T \Sigma X^T (K + \sigma^2 I)^{-1} y, x_*^T \Sigma x_* - x_*^T \Sigma X^T (K + \sigma^2 I)^{-1} X \Sigma x_*)$$

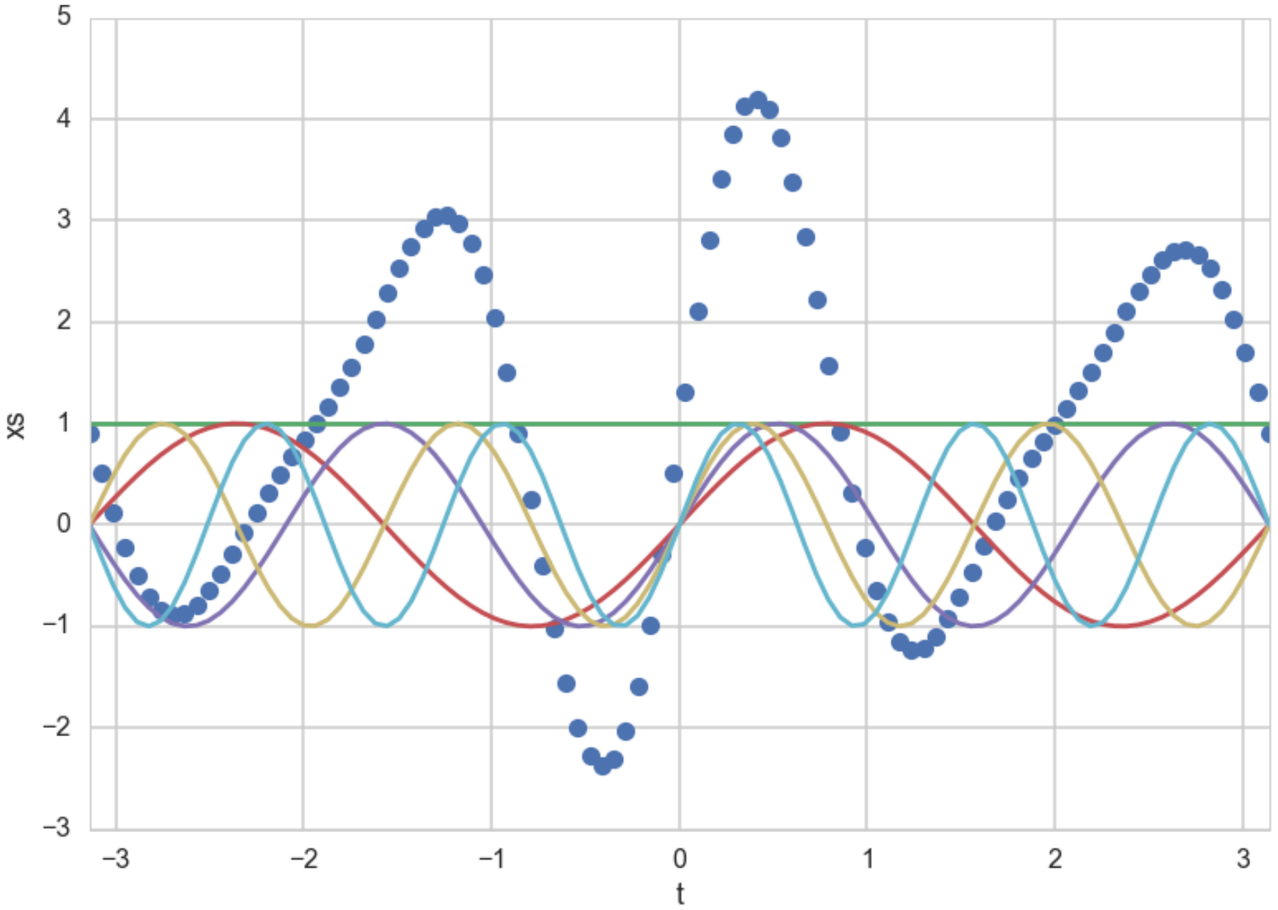
where $K = X \Sigma X^T$ which is of size N x N.

Notice now that the features only appear in the combination $\kappa(x, x') = x^T \Sigma x'$, thus, the dual representation:

$$p(f(x_*)|x_*, X, y) =$$

$$N\left(\kappa(x_*, X)\left(\kappa(X^T, X) + \sigma^2 I\right)^{-1} y,\ \ \kappa(x_*, x_*) - \kappa(x_*, X^T)\left(\kappa(X^T, X) + \sigma^2 I\right)^{-1} \kappa(X^T, x_*)\right)$$

Basis functions



A scalar x could be projected into a polynomial space: $\phi(x) = (1, x, x^2, x^3, \dots)$. So let us now have

$$f(x) = \phi(x)^T w$$

Let $\Phi(X)$ be the aggregation of columns $\phi(x)$ for all training set cases $x \in X$.

Then the posterior predictive is $p(f(x_*)|x_*, X, y) = N(\frac{1}{\sigma^2}\phi_*^T A^{-1} \Phi y, \phi_*^T A^{-1} \phi_*)$.

where $\phi_* = \phi(x_*)$ and $A = \frac{1}{\sigma^2}\Phi\Phi^T + \Sigma^{-1}$

which can as before be written as

$$N(\kappa(x_*, X)\left(\kappa(X^T, X) + \sigma^2 I\right)^{-1} y, \;\; \kappa(x_*, x_*) - \kappa(x_*, X^T)\left(\kappa(X^T, X) + \sigma^2 I\right)^{-1} \kappa(X^T, x_*)$$

where the kernel is now $\kappa(x, x') = \phi(x)^T \Sigma \phi(x')$

Then defining $\psi(x) = \Sigma^{(1/2)}\phi(x)$, we have $\kappa(x, x') = \psi(x)^T \psi(x')$

Kernel Trick

If an algorithm is defined just in terms of inner products in input space then we can make the algorithm work in higher-dimensional feature space by replacing occurrences of those inner products by $\kappa(x, x')$.

So we learn that covariance can be kernelized, and dimensions can be lifted. This might remind you of SVM.

Infinite basis sets

Now consider an infinite set of $\phi(x)$. Like a fourier series or a Bessel series.

We can construct an infinitely parametric model.

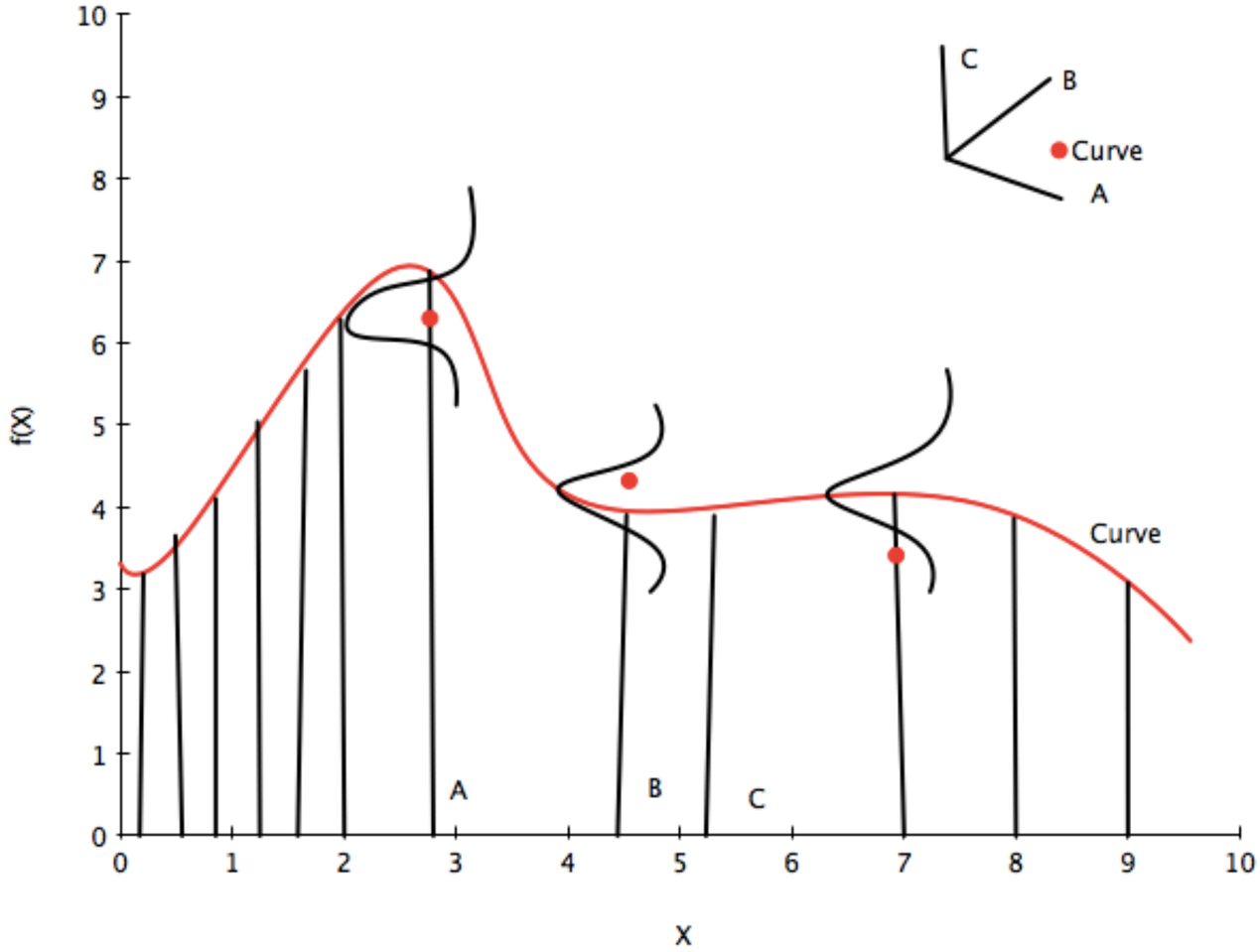
This is called a non-parametric model.

We just need to be able to define a finite kernel $\kappa(x, x') = \psi(x)^T \psi(x')$!!

Even simpler: use infinite gaussians

- think of the function as an infinite vector.
- Draw $\overline{f}$ from some 'infinite' gaussian distribution with some mean and some kernel.

  This then is the Gaussian Process, which we use to set a prior on the space of functions.



Key Insight

for the marginal of a gaussian, only the covariance of the block of the matrix involving the unmarginalized dimensions matters! Thus "if you ask only for the properties of the function (you are fitting to the data) at a finite number of points, then inference in the Gaussian process will give you the same answer if you ignore the infinitely many other points, as if you would have taken them all into account!"

-Rasmunnsen

Definition of Gaussian Process

Assume we have this function vector $f = (f(x_1), \dots f(x_n))$. If, for ANY choice of input points, $(x_1, \dots, x_n)$, the marginal distribution over f:

$$P(F) = \int_{f \notin F} P(f) df$$

is multi-variate Gaussian, then the distribution $P(f)$ over the function f is said to be a Gaussian Process.

a Gaussian Process defines a prior distribution over functions!

Once we have seen some data, this prior can be converted to a posterior over functions, thus restricting the set of functions that we can use based on the data.

Since the size of the "other" block of the matrix does not matter, we can do inference from a finite set of points.

Any m observations in an arbitrary data set, $y = y_1, \ldots, y_n = m$ can always be represented as a single point sampled from some m-variate Gaussian distribution. Thus, we can work backwards to 'partner' a GP with a data set, by marginalizing over the infinitely-many variables that we are not interested in, or have not observed.

GP regression

Using a Gaussian process as a prior for our model, and a Gaussian as our data likelihood, then we can construct a Gaussian process posterior.

Likelihood: $y|f(x), x \sim \mathcal{N}(f(x), \sigma^2 I)$

where the infinite $f(x)$ takes the place of the parameters.

Prior: $f(x) \sim \mathcal{GP}(m(x) = 0, k(x, x\prime))$

Infinite normal posterior process: $f(x)|y \sim \mathcal{GP}(m_{post}, \kappa_{post}(x, x\prime))$.

The posterior distribution for f is:

$$m_{post} = k(x\prime, x)[k(x, x) + \sigma^2 I]^{-1} y$$
$$k_{post}(x, x\prime) = k(x\prime, x\prime) - k(x\prime, x)[k(x, x) + \sigma^2 I]^{-1} k(x, x\prime)$$

Posterior predictive distribution for $f(x_*)$ for a test vector input x_*, given a training set X with values y for the GP is:

$$m_* = k(x_*, X)[k(X^T, X) + \sigma^2 I]^{-1} y$$
$$k_* = k(x_*, x_*) - k(x_*, X^T)[k(X^T, X) + \sigma^2 I]^{-1} k(X^T, x_*)$$

The predictive distribution of test targets y_* : add $\sigma^2 I$ to k_*.

Tidbits

- No free lunch: calculation involves inverting a NxN matrix as in the kernel space representation of regression.
- Exact correspondence between the gaussian process (direct usage of gaussians in space) to the basis function regression (in feature space with gaussians for prior parameters) in the kernelized representation, as long as we identify the GP covariance function k with the kernel function $\kappa(x, x') = \phi(x)^T \Sigma \phi(x')$. (Mercer's theorem)

INFERENCE

Use the marginal likelihood:

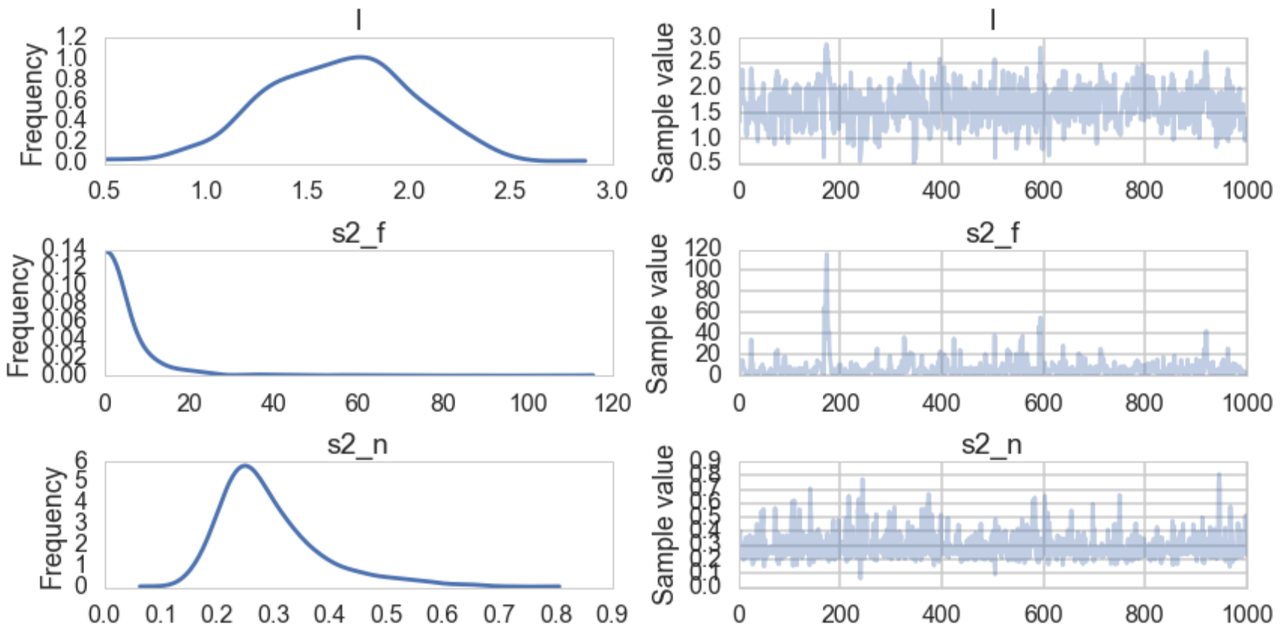
$$p(y|X) = \int_f p(y|f,X)p(f|X)df$$

The Marginal likelihood given a GP prior and a gaussian likelihood is:

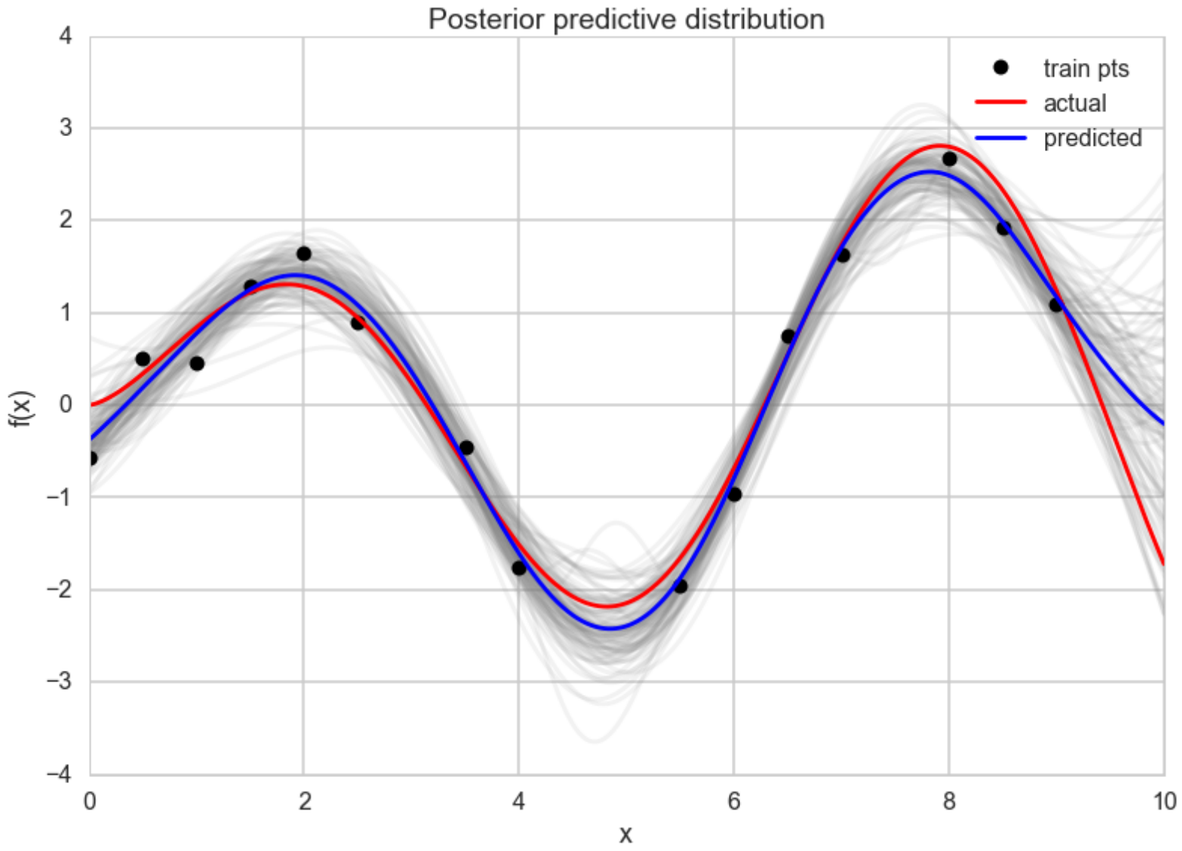
$$\log p(y|X) = -\frac{n}{2}\log 2\pi - \frac{1}{2}\log|K+\sigma^2 I| - \frac{1}{2}y^T(K+\sigma^2 I)^{-1}y$$

Fitting in pymc3

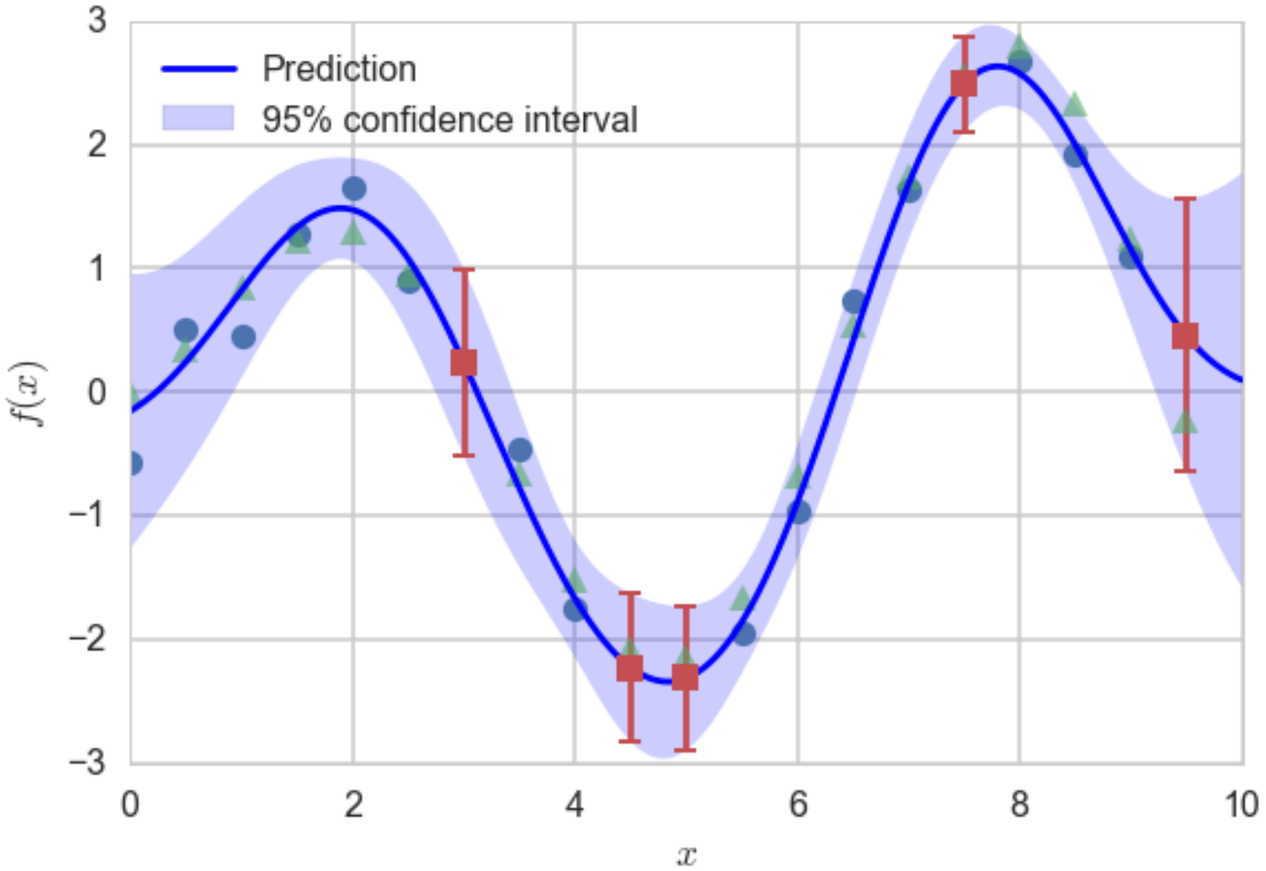
```
import theano.tensor as tt
with pm.Model() as model:
    # priors on the covariance function hyperparameters
    l = pm.Uniform('l', 0, 10)

    # uninformative prior on the function variance
    log_s2_f = pm.Uniform('log_s2_f', lower=-10, upper=5)
    s2_f = pm.Deterministic('s2_f', tt.exp(log_s2_f))

    # uninformative prior on the noise variance
    log_s2_n = pm.Uniform('log_s2_n', lower=-10, upper=3)
    s2_n = pm.Deterministic('s2_n', tt.exp(log_s2_n))

    # covariance functions for the function f and the noise
    f_cov = s2_f * pm.gp.cov.ExpQuad(1, l)

    y_obs = pm.gp.GP('y_obs', cov_func=f_cov, sigma=s2_n,
        observed={'X':xtrain.reshape(-1,1), 'Y':ytrain})
with model:
    trace = pm.sample(2000)
with model:
    gp_samples = pm.gp.sample_gp(trace,
        y_obs, x_pred.reshape(-1,1), samples=100)
```



Posterior (predictive) curves



MLE using sklearn




```
gp = GaussianProcess(corr='squared_exponential', theta0=1e-1,
                     thetaL=1e-3, thetaU=10,
                     nugget=(sigma_noise/ytrain)**2,
                     random_start=100)

# Fit to data using Maximum Likelihood Estimation of the parameters
gp.fit(xtrain.reshape(-1,1), ytrain)
# Make the prediction on the meshed x-axis (ask for MSE as well)
Xpred = x_pred.reshape(-1,1)
y_pred, MSE = gp.predict(Xpred, eval_MSE=True)
sigma = np.sqrt(MSE)
```

Where are GPs used?

- geostatistics with kriging, oil exploration
- spatial statistics
- as an interpolator (0 noise case) in weather simulations
- they are equivalent to many machine learning models such as kernelized regression, SVM and neural networks (some)
- ecology since model uncertainty is high
- they are the start of non-parametric regression
- time series analysis (see cover of BDA)
- because of the composability of kernels, in automates statistical analysis (see the automatic statistician)