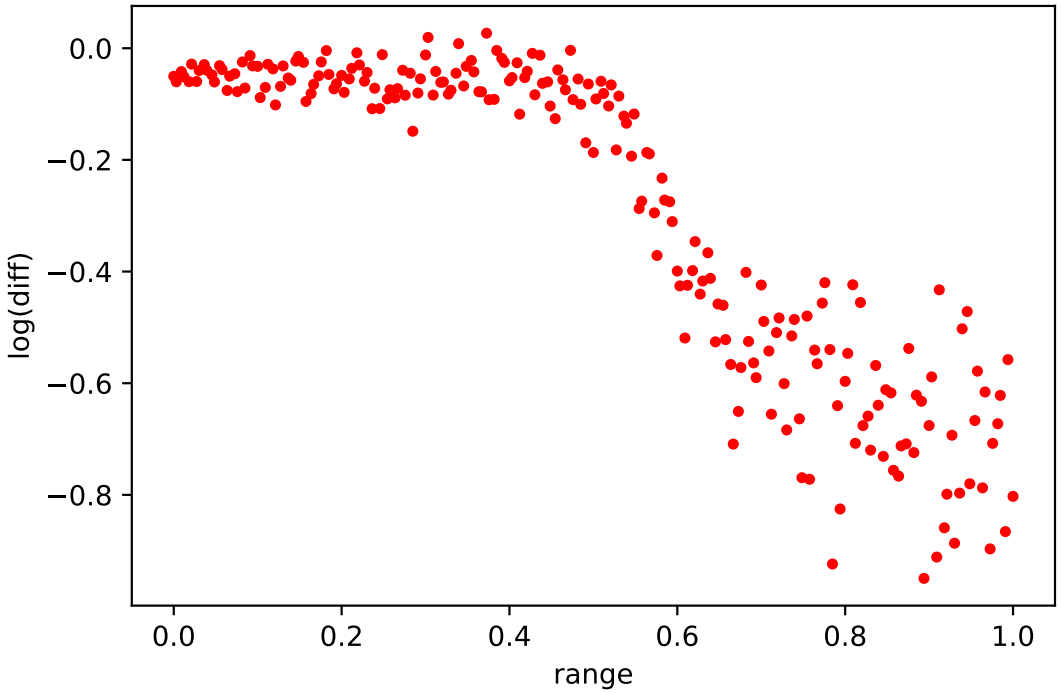
Latent Gaussian processes for non-parameteric regression

Jonas Wallin
(Deptt. of Statistics, Lund Univ.)

September 10, 2019

Goal

- Find a function that describes the data below.



Nonlinear regression

- Linear regression

$$y = f(x) + \epsilon$$
$$f(x) = x^T \beta = \sum_i x_i \beta_i.$$

Nonlinear regression

- Linear regression

$$y = f(x) + \epsilon$$
$$f(x) = x^T \beta = \sum_i x_i \beta_i.$$

- Polynomial regression, $\phi(x) = [1, x, x^2, x^3, \ldots, x^k]$

$$y = f(x) + \epsilon$$
$$f(x) = \phi(x)\beta = \sum_i x^i \beta_i.$$

Nonlinear regression

- Linear regression

$$y = f(x) + \epsilon$$
$$f(x) = x^T \beta = \sum_i x_i \beta_i.$$

- Polynomial regression, $\phi(x) = [1, x, x^2, x^3, \ldots, x^k]$

$$y = f(x) + \epsilon$$
$$f(x) = \phi(x)\beta = \sum_i x^i \beta_i.$$

- More generally: splines.

Nonlinear regression

- Linear regression

$$y = f(x) + \epsilon$$
$$f(x) = x^T \beta = \sum_i x_i \beta_i.$$

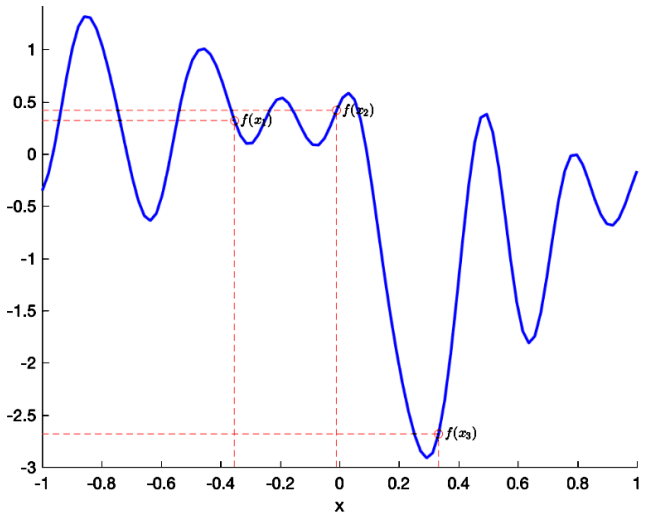
- Polynomial regression, $\phi(x) = [1, x, x^2, x^3, \ldots, x^k]$

$$y = f(x) + \epsilon$$
$$f(x) = \phi(x)\beta = \sum_i x^i \beta_i.$$

- More generally: splines.
- Fitting to data by finding β.

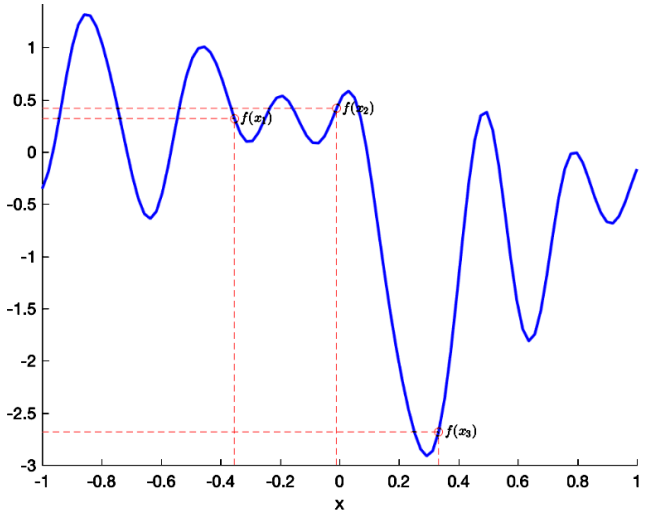
Non parameteric regression

- Non-parameteric regression: does not specific form for $f(\cdot)$.



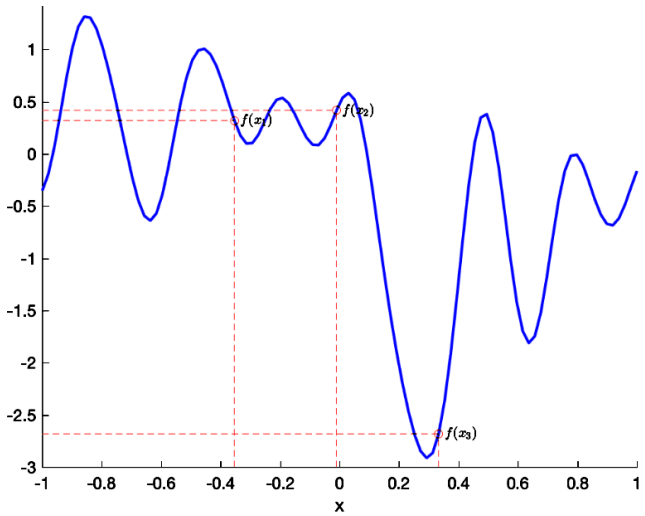
Non parameteric regression

- Non-parameteric regression: does not specific form for $f(\cdot)$.
- Treat $f(x)$ as an unkown parameter for every x



Non parameteric regression

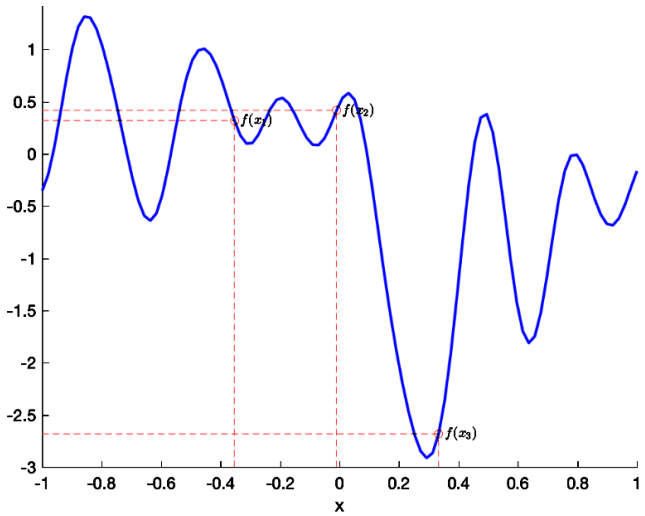
- Non-parameteric regression: does not specific form for $f(\cdot)$.
- Treat $f(x)$ as an unkown parameter for every x



- A new parameter for every x, this means infinitly many of parameters.

Non parameteric regression

- Non-parameteric regression: does not specific form for $f(\cdot)$.
- Treat $f(x)$ as an unkown parameter for every x



- A new parameter for every x, this means infinitly many of parameters.
- Need to put some assumptions of what functions we are looking for.

Bayesian modeling

- We will use a Bayesian modeling approach. Where we put a prior on each $f(x)$ and update the distribution of $f(x)$ as we get data (y_i, x_i).

Bayesian modeling

- We will use a Bayesian modeling approach. Where we put a prior on each $f(x)$ and update the distribution of $f(x)$ as we get data (y_i, x_i).
- We will spesificy a prior on the entire function $f(\cdot)$ so that points close ($x - x' = d$ small) implies that $f(x)$ and $f(x')$ are close.

Bayesian modeling

- We will use a Bayesian modeling approach. Where we put a prior on each $f(x)$ and update the distribution of $f(x)$ as we get data (y_i, x_i).
- We will spesificy a prior on the entire function $f(\cdot)$ so that points close ($x - x' = d$ small) implies that $f(x)$ and $f(x')$ are close.
- If we assume a priori that f is a Gaussian processes we get exactly what we want.

Gaussian Processes

Definition

A Gaussian processes (GP), f, is a random function, such that at each value,x, $f(x)$ is a Normal random variable.

- A GP has two "parameters" the mean and the covaraince function (kernel).

Gaussian Processes

Definition

A Gaussian processes (GP), f, is a random function, such that at each value,x, $f(x)$ is a Normal random variable.

- A GP has two "parameters" the mean and the covaraince function (kernel).
- The mean determines the base structure.

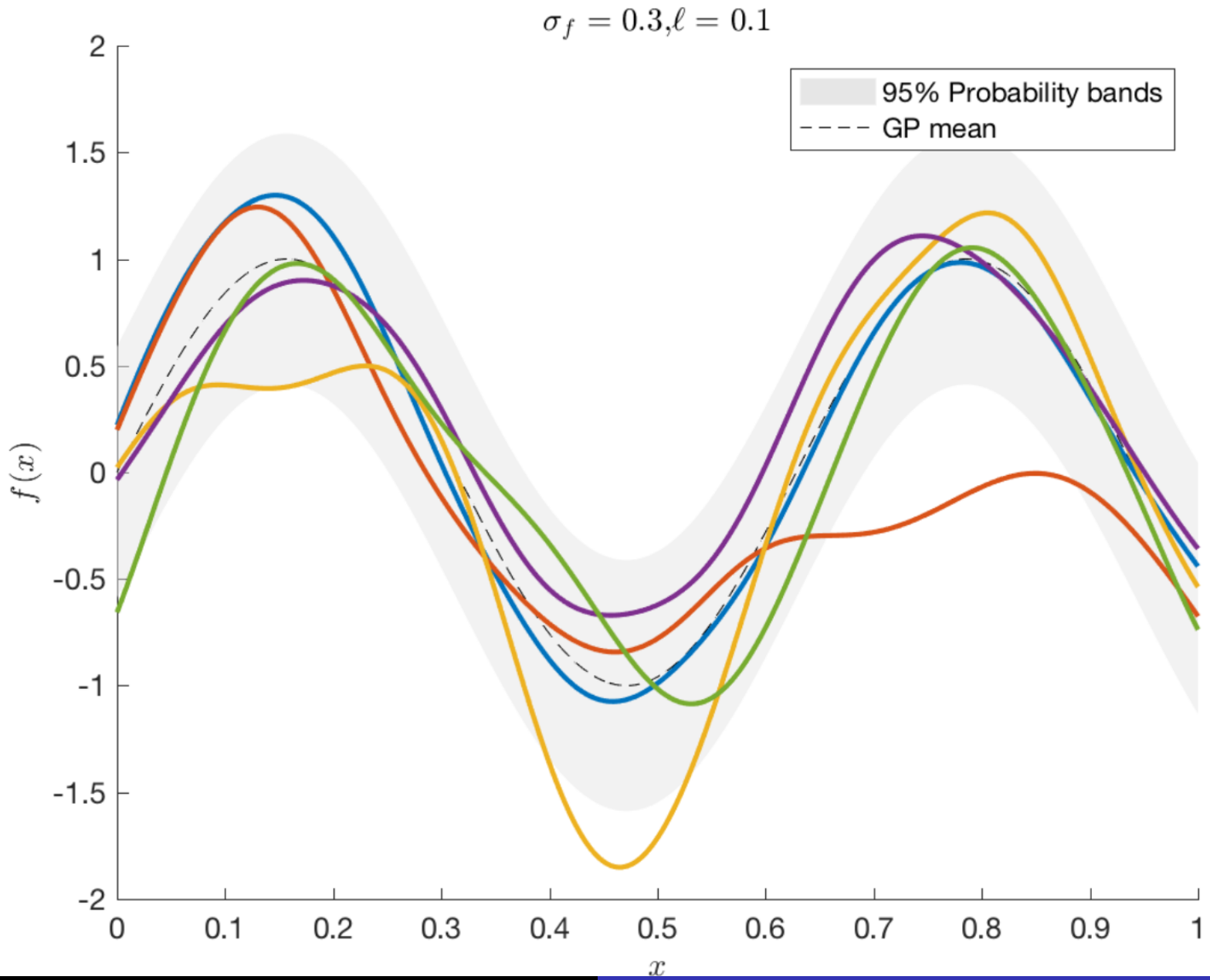
Gaussian Processes

Definition

A Gaussian processes (GP), f, is a random function, such that at each value,x, $f(x)$ is a Normal random variable.

- A GP has two "parameters" the mean and the covaraince function (kernel).
- The mean determines the base structure.
- The kernel $k(||x - x'||)$ determines how dependent (close) the two points, $f(x)$ and $f(x')$, are.

Gaussian Processes simulations



Matérn kernel

Matérn kernel ($l > 0, \sigma_f > 0, \nu > 0$)

$$K_{Matern}(d) = \sigma_f^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} d}{l} \right) K_\nu \left(\frac{\sqrt{2\nu} d}{l} \right)$$

The actual form is equation is not important but the parameters have clear meaning on what functions we want

Matérn kernel

Matérn kernel ($l > 0, \sigma_f > 0, \nu > 0$)

$$K_{Matern}(d) = \sigma_f^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} d}{l} \right) K_\nu \left(\frac{\sqrt{2\nu} d}{l} \right)$$

The actual form is equation is not important but the parameters have clear meaning on what functions we want

- $l-$ length scale determines length of the depence

Matérn kernel

Matérn kernel ($l > 0, \sigma_f > 0, \nu > 0$)

$$K_{Matern}(d) = \sigma_f^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} d}{l} \right) K_\nu \left(\frac{\sqrt{2\nu} d}{l} \right)$$

The actual form is equation is not important but the parameters have clear meaning on what functions we want

- $l-$ length scale determines length of the depence
- σ_f- determines the variability, variance

Matérn kernel

Matérn kernel ($l > 0, \sigma_f > 0, \nu > 0$)

$$K_{Matern}(d) = \sigma_f^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} d}{l} \right) K_\nu \left(\frac{\sqrt{2\nu} d}{l} \right)$$

The actual form is equation is not important but the parameters have clear meaning on what functions we want

- $l-$ length scale determines length of the depence
- σ_f- determines the variability, variance
- $\nu-$ determines the smoothnes, roughness or diffrentiability.

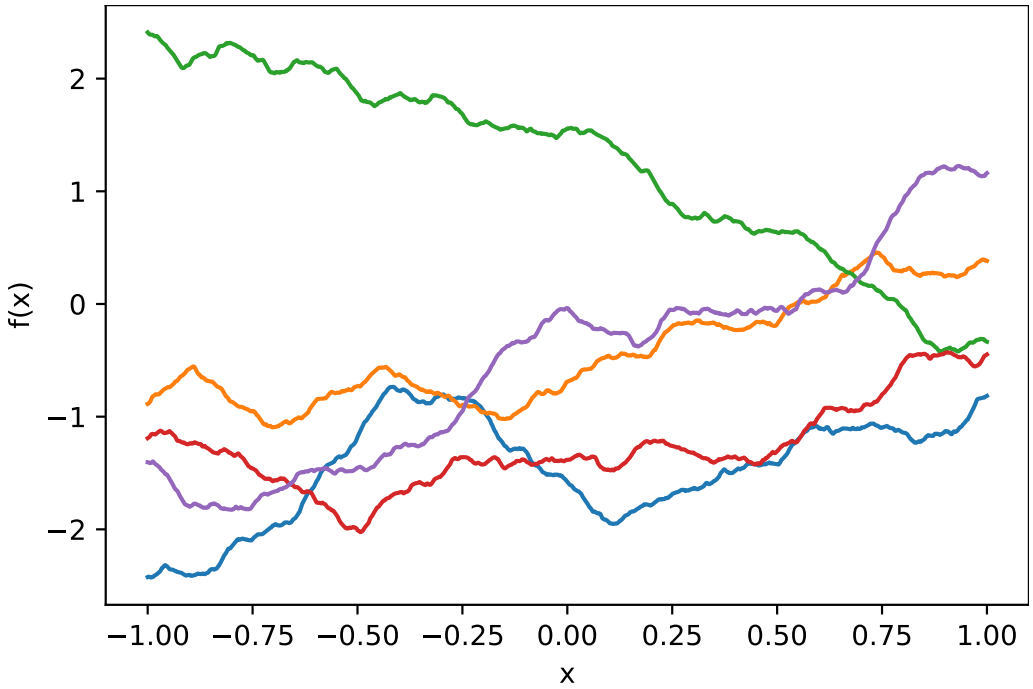
Scikit usage

- Setting up Gaussian Process in sklearn.

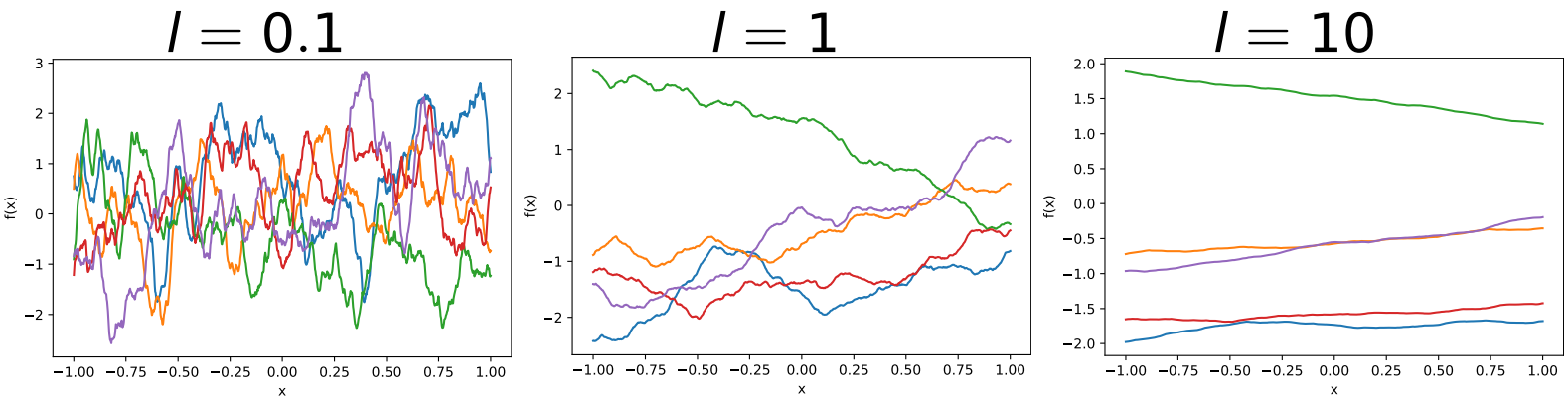
```
from sklearn.gaussian_process import GaussianProcessRegressor
import sklearn.gaussian_process.kernels as kernels
l     = 1
nu    = 1
sigma = 1
kernel = sigma**2*kernels.Matern(length_scale=l, nu=nu)
gp     = GaussianProcessRegressor(kernel=kernel)
```

Simulation without data

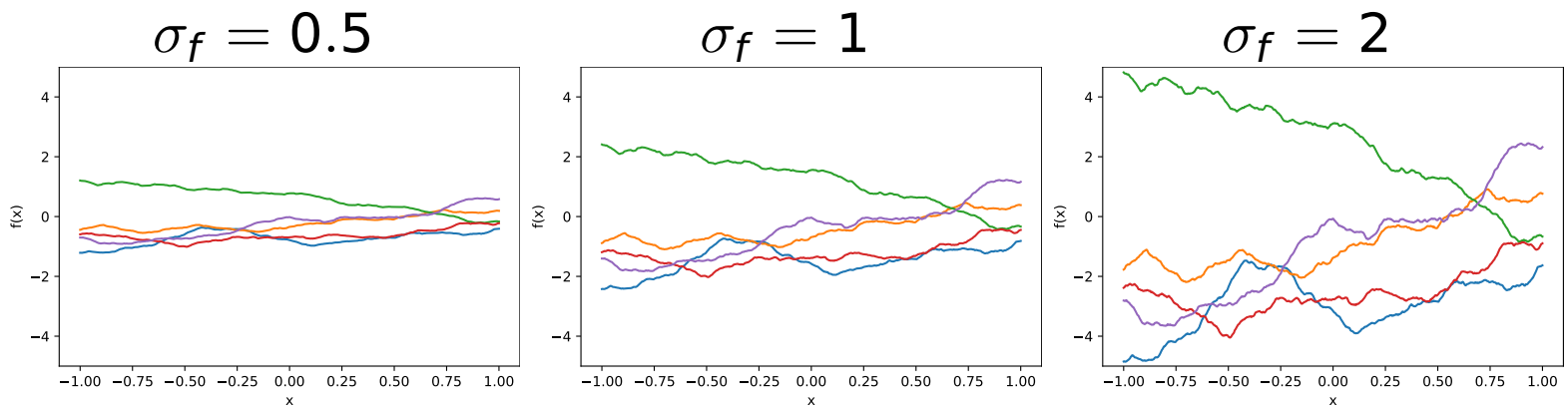
```
x       = np.linspace(-1,1,1000)
fx      = gp.sample_y(x[:,np.newaxis], n_samples=5)
plt.plot(x,fx)
plt.xlabel('x')
plt.ylabel('f(x)')
```



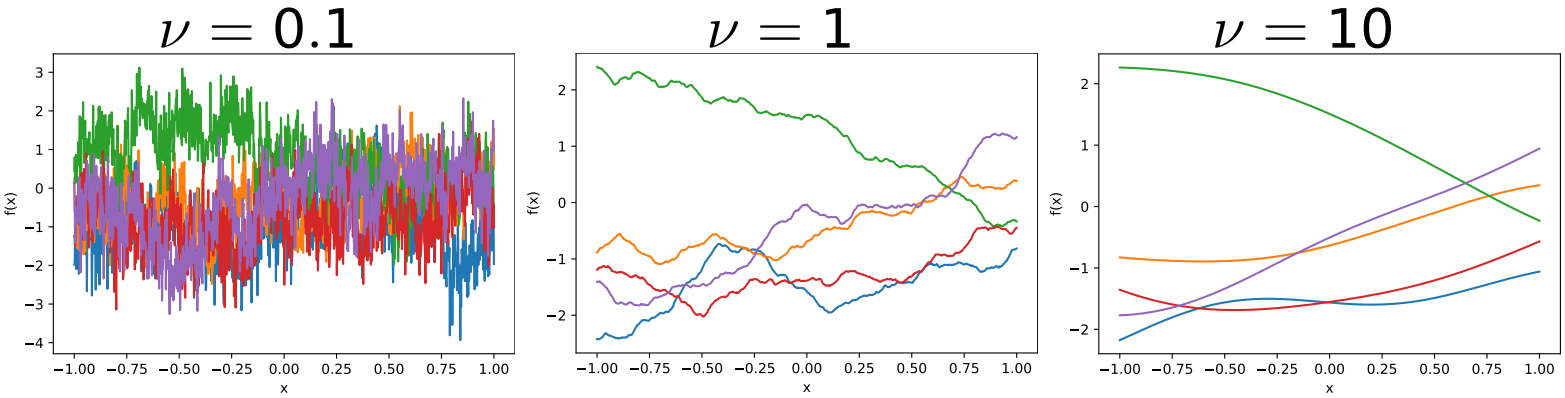
l— controls the length of dependence



$\sigma-$ controls variability



$\nu-$ controls smoothness

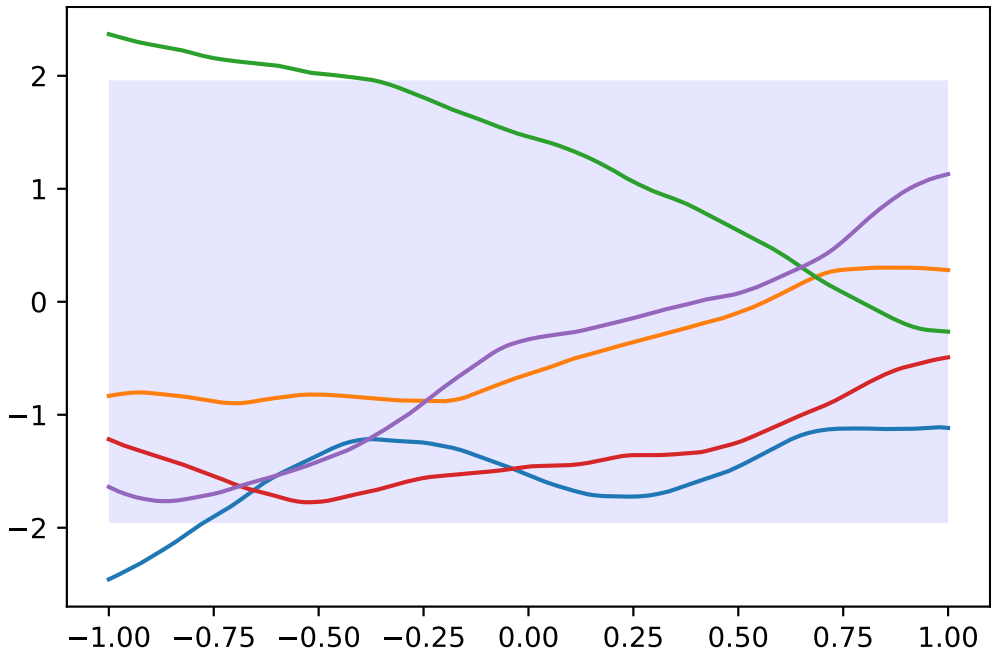


The model prior to data

- From a Bayesian presepective we know have a functional prior due to that we assume

$$f \sim GP$$

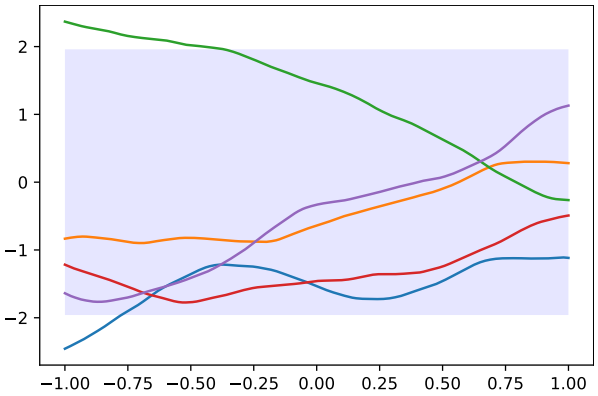
before observing any data.



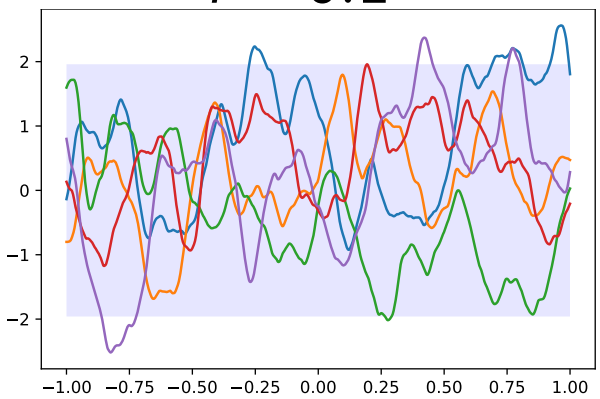
Two models

```
kernel = kernels.Matern(length_scale=1, nu=2)
gp     = GaussianProcessRegressor(kernel=kernel)
kernel2 = kernels.Matern(length_scale=0.1, nu=2)
gp2     = GaussianProcessRegressor(kernel=kernel2)
#plotting the blue area
f_m, f_sd = gp.predict(x[:,np.newaxis], return_std=True)
plt.fill(np.concatenate([x, x[::-1]]),
np.concatenate([f_m - 1.9600 * f_sd,
(f_m + 1.9600 * f_sd)[::-1]]),
alpha=.1, fc='b')
```

$l = 1$



$l = 0.1$



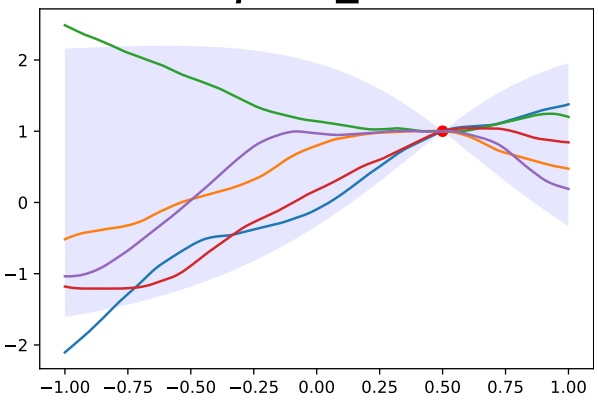
An observation

- Suppose we know observe that $f(0.5) = 1$, how does this update our distribution of functions.

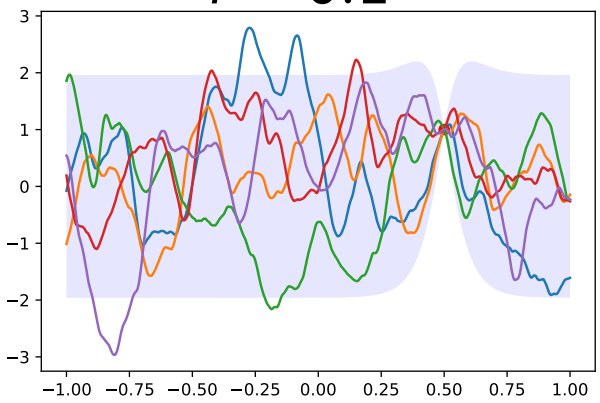
An observation

- Suppose we know observe that $f(0.5) = 1$, how does this update our distribution of functions.

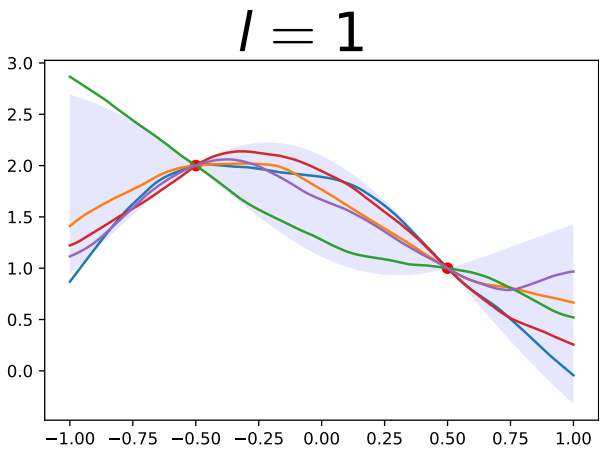
$l = 1$

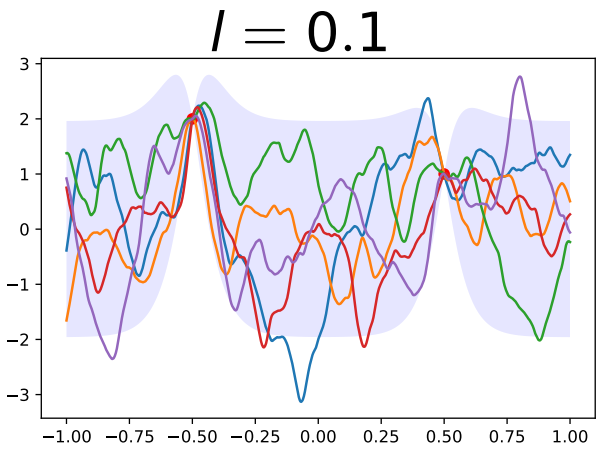


$l = 0.1$



two observation





The latent model

- We know have **prior** on our function

$$f(x) \sim GP(0, k(||x - x'||))$$

The latent model

- We know have **prior** on our function

$$f(x) \sim GP(0, k(||x - x'||))$$

- Model:

$$y_i = f(x_i) + \epsilon_i, \quad \epsilon \sim N(0, \sigma_e^2)$$

The latent model

- We know have **prior** on our function

$$f(x) \sim GP(0, k(||x - x'||))$$

- Model:

$$y_i = f(x_i) + \epsilon_i, \quad \epsilon \sim N(0, \sigma_e^2)$$

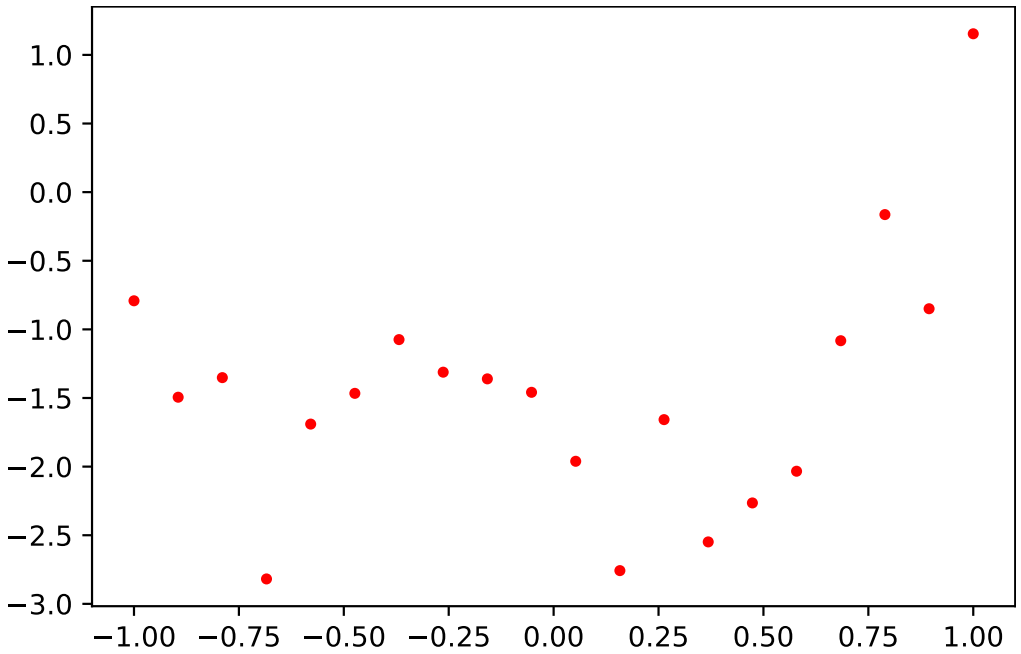
- This can also be written as

$$f(x) \sim GP(0, k(||x - x'||) + \sigma_e^2)$$

```
sigmae = 1.
kernel =  kernels.Matern(length_scale=l, nu=nu) + kernels.WhiteKernel(sigmae**2)
```

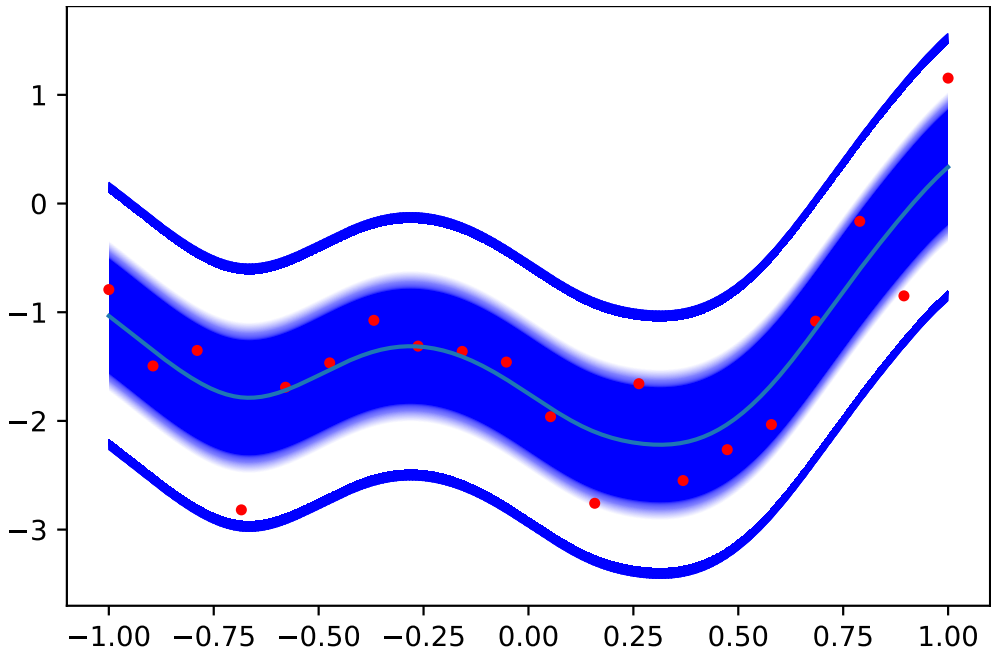
example data

```
xobs  = np.linspace(-1,1,20)
fxobs = gp.sample_y(xobs[:,np.newaxis])
```



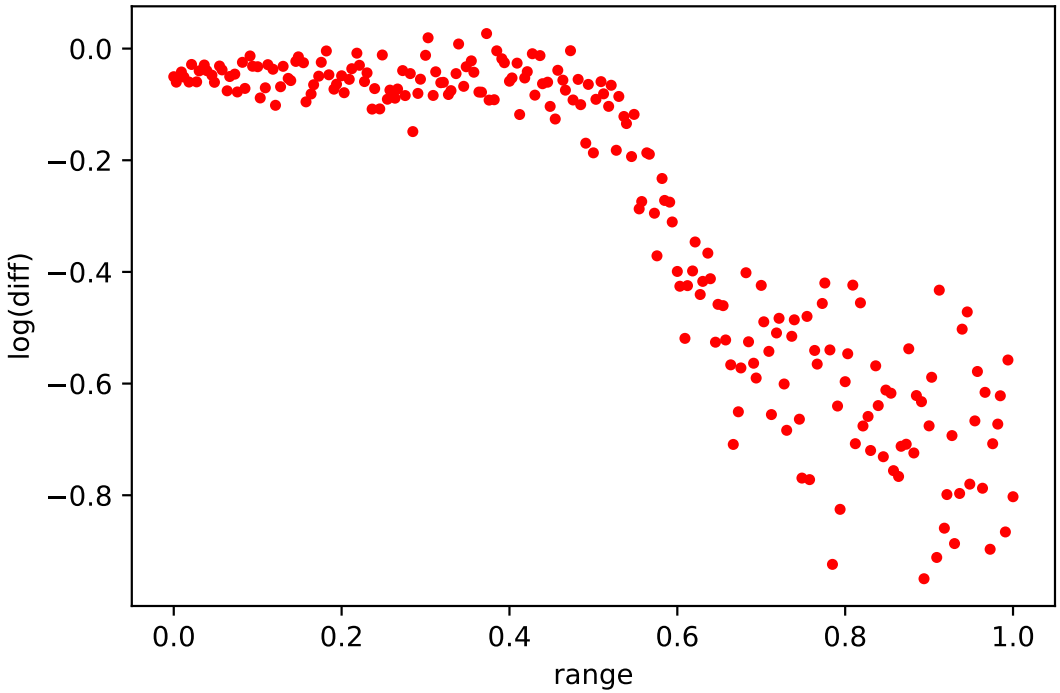
example data

```
x      = np.linspace(-1,1,1000)
gp2.fit(xobs[:,np.newaxis], fxobs)
f_m2, f_sd2 = gp2.predict(x[:,np.newaxis], return_std=True)
plt.fill(np.concatenate([x, x[::-1]]),
np.concatenate([f_m2 - 1.9600 * f_sd2,
(f_m2 + 1.9600 * f_sd2)[::-1]]),
alpha=.01, fc='b')
plt.plot(x, f_m2)
plt.plot(x,f_m2 - 1.9600 * y_sd2,'b--',linewidth=0.1)
plt.plot(x,f_m2 + 1.9600 * y_sd2,'b--',linewidth=0.1)
```



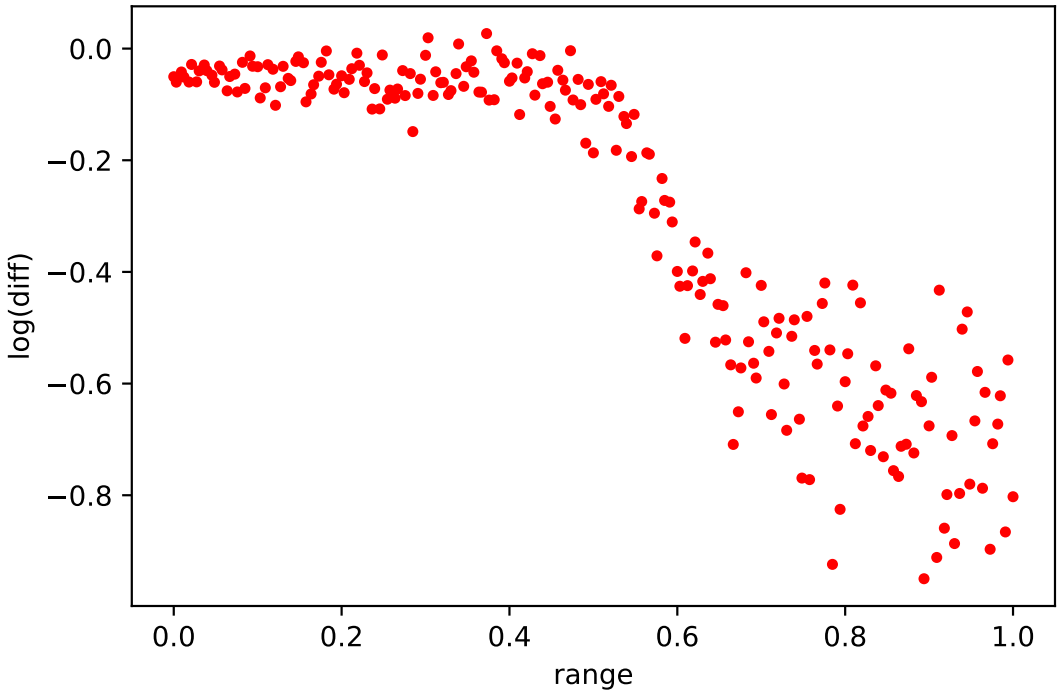
Classical Example

- A classical example in non parameteric regression is the "Lidar data".



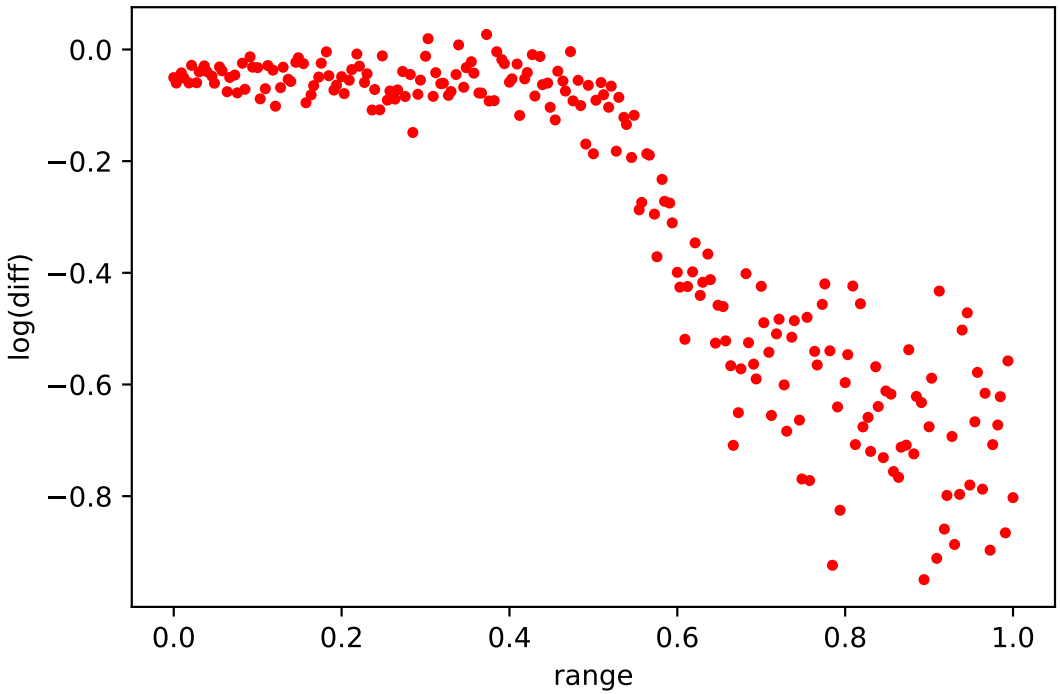
Classical Example

- A classical example in non parameteric regression is the "Lidar data".
- Measurement are the logarthmic of the ratio of two laser wave length as a function of a mercury source.



setting up the Gaussian processes

```
data = np.loadtxt('lidar.txt')
kernel =  kernels.Matern(length_scale=0.1, nu=1) + kernels.WhiteKernel()
gp = GaussianProcessRegressor(kernel=kernel,
optimizer='fmin_l_bfgs_b',
n_restarts_optimizer = 2,
normalize_y=True)
distance  = data[:,0]
logData   = data[:,1]
```

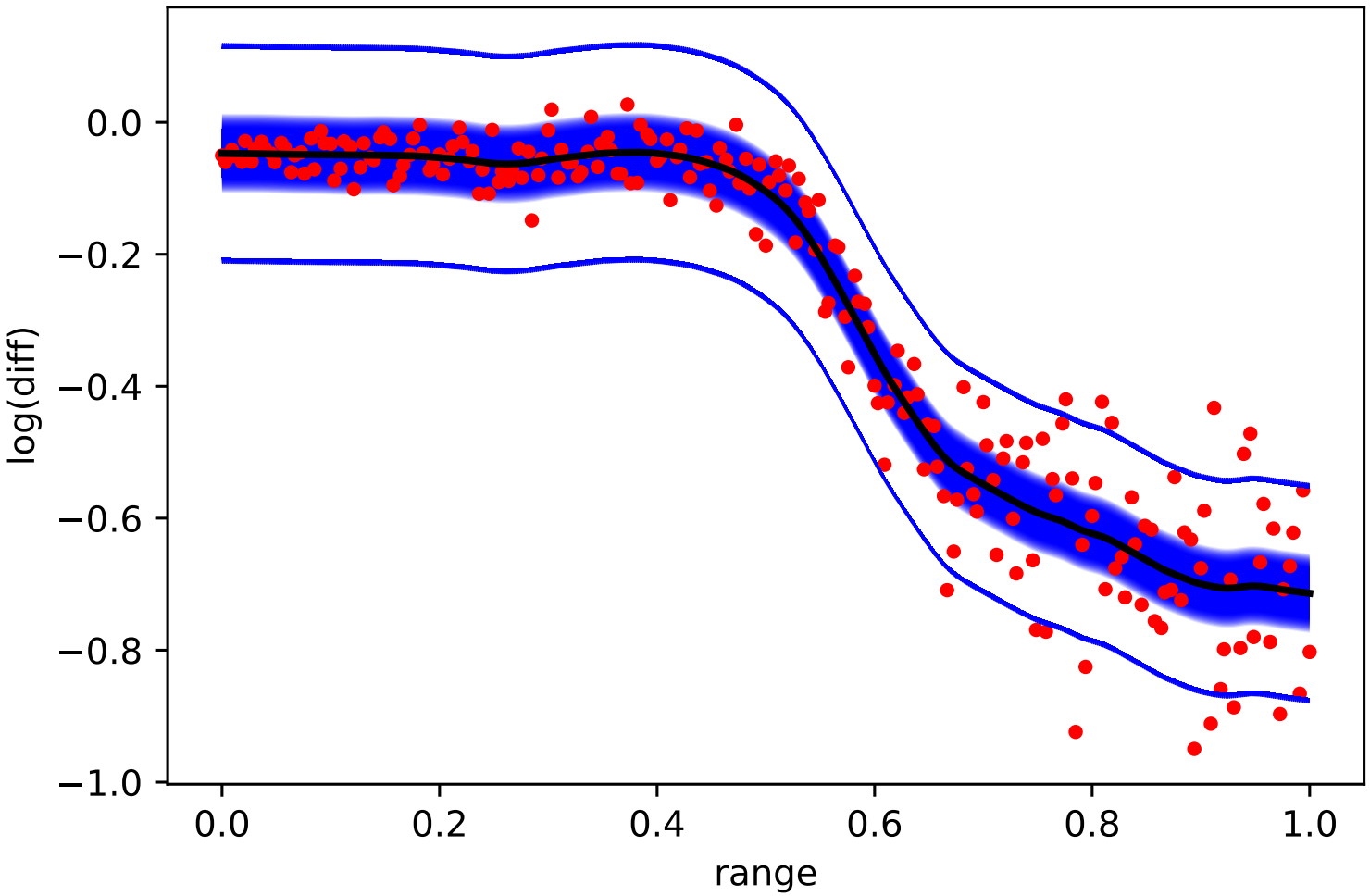


tuning the hyperparameters

- The hyperparameters are not well suited for the data.
- However, they can "easily" be tunned to fit the data. Using either Maximum likelihood, crossvalidation, or MCMC.

```
# Fit to data using Maximum Likelihood Estimation of the parameters
gp.fit(distance[:, np.newaxis], logData[:, np.newaxis])
```

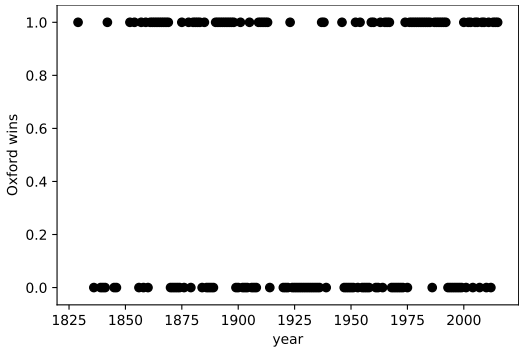
result



Further latent models

- We have looked a the model:

$$y_i = f(x_i) + \epsilon_i, \quad \epsilon \sim N(0, \sigma_e^2)$$



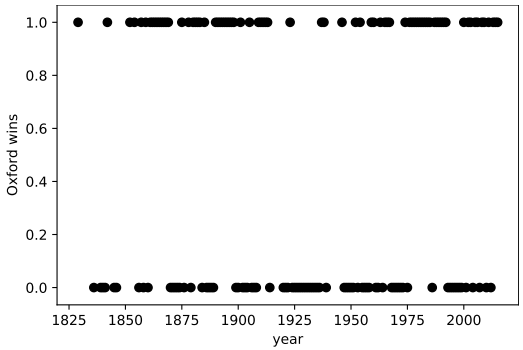
Further latent models

- We have looked a the model:

$$y_i = f(x_i) + \epsilon_i, \quad \epsilon \sim N(0, \sigma_e^2)$$

- An other popular options is

$$y_i \sim Bin(n_i, p(x_i)),$$
$$p(x_i) = \frac{e^{f(x_i)}}{1 + e^{f(x_i)}}$$



Binomial model in Sklearn

```
from sklearn.gaussian_process import GaussianProcessClassifier
gp = GaussianProcessClassifier(kernel= kernel)
prob = gp.predict_proba(X)
```