GP_emu - model analysis with GPs

Sam Coveney - s.coveney@sheffield.ac.uk

April 19, 2017

Contents

What is a model?

Definition

A scientific model is a representation of a phenomenon

- conceptual
- mathematical
- physical

Models generally do the following:

- take **input** information
- **process** this information
- **output** information (i.e. make **predictions**)

We often use computer simulations to run models.

What is a model?

A simple computational model

Simulations can be very simple.

A 'simulation' of the function $f(x) = x(\cos(x) + 3)$ might be:

```
# simple function
def func(x):
    return x * ( np.cos(x) + 3 )

x = 1           # input
y = func(x)     # output
```

It's so simple, it feels wrong to call this a 'simulation'...

What is a model?

A complicated simulation

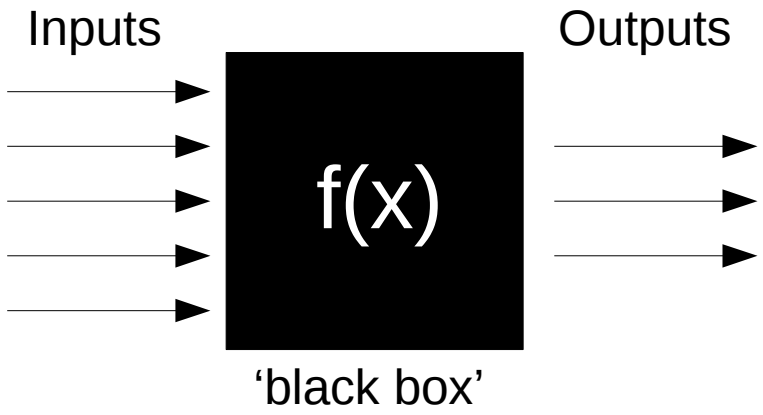
Simulations can be quite complicated, such that we don't know how outputs depend on inputs (else we wouldn't need to run the simulations).

```
# not so simple model
def model(x1, x2, x3, x4, x5):
    # 10,000 lines of complicated code
    return y1, y2, y3

x1,x2,x3,x4,x5 = 1,2,3,4,5          # input
y1,y2,y3 = model(x1,x2,x3,x4,x5)    # outputs
```

Learning a function

Simulations as 'black box' functions

We suppose we can **learn how the outputs depend on the inputs** i.e. 'fit the data' by relating outputs to inputs: $y = f(x)$ where f is a function.



If we could learn the properties of $f(x)$ we could obtain a **surrogate model**, or **emulator**, to run in place of the original simulation.

Learning a function

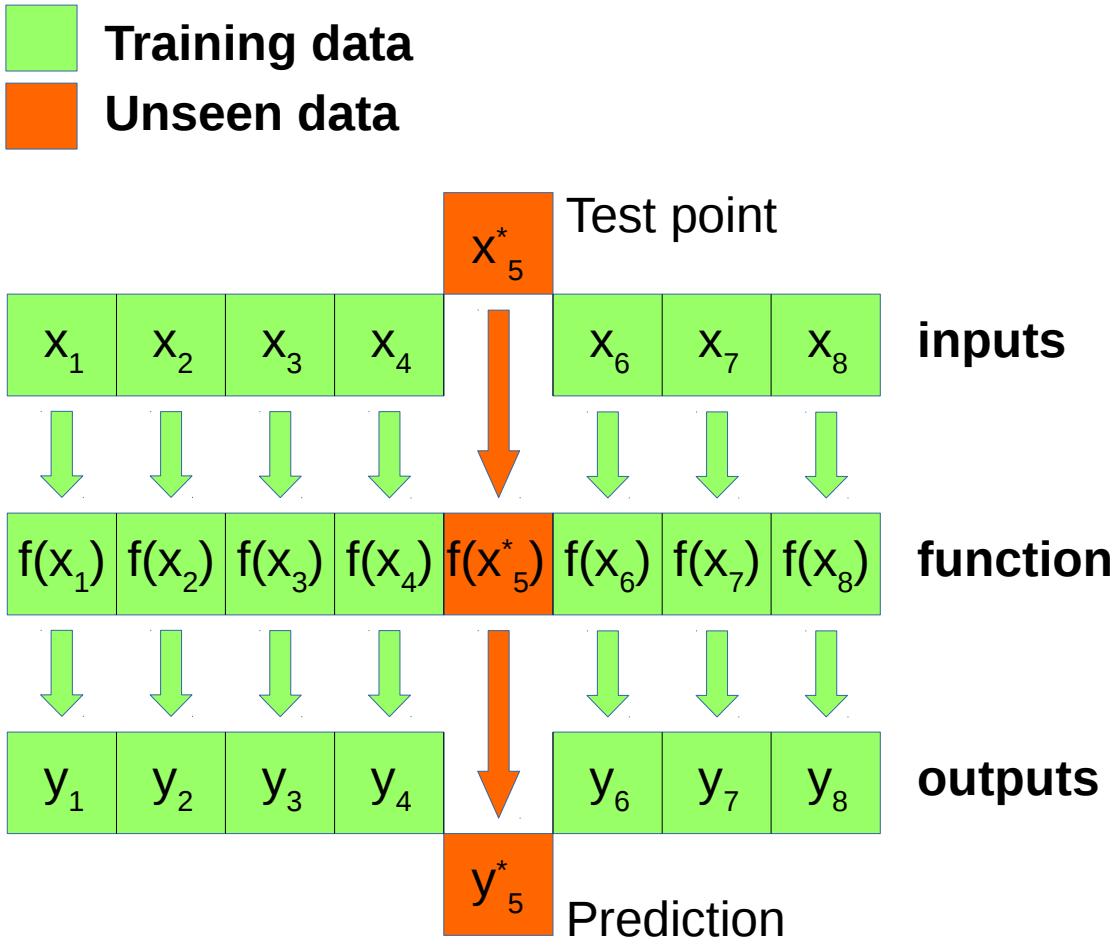
From training data to prediction

From Rasmussen & Williams **Gaussian Processes for Machine Learning**:

> *... think of a function as a very long vector, each entry in the vector specifying the function value $f(x)$ at a particular input x.*
>
> *... we denote the input as x, and the output (or target) as y ... We have a dataset D of n observations, $D = \{(x_i, y_i)|i = 1, \ldots, n\}$*
>
> *... we need to move from the finite training data D to a function f that makes predictions for all possible input values.*

Learning a function

From training data to prediction



What is a Gaussian Process?

Definition of a Gaussian Process

From Rasmussen & Williams **Gaussian Processes for Machine Learning**:

> *A Gaussian process is a collection of random variables ... which have a joint Gaussian distribution ... the random variables represent the value of the function $f(x)$ at location x.*

Represent function with GP

- $f(x) \sim \mathbf{GP}$
- $\mathbf{E}[f(x^*)]$: mean of f for input x^*
- $\mathbf{V}[f(x^*)]$: variance of f for input x^*

What is a Gaussian Process?

Definition of a Gaussian Process

A GP is completely specified by its mean and covariance function.

$$
\begin{aligned}
f(x) &\sim \mathbf{GP}(m(x), k(x, x')) \\
m(x) &= \mathbf{E}[f(x)] \\
k(x, x') &= \mathbf{E}[(f(x) - m(x))(f(x') - m(x'))]
\end{aligned}
$$

The **mean function** $m(x)$ can be anything - common choices are zero, or a linear combination of the inputs.

The **covariance function** $k(x, x')$ is a measure of how dissimilar $f(x)$ and $f(x')$ should be given x and x'.

What is a Gaussian Process?

Training on data

For a linear mean and a Gaussian covariance function:

$$m(x) = \beta_0 + \beta_1 x$$

$$k(x, x') = \sigma^2 \exp\left\{ -\frac{|x - x'|^2}{\delta^2} \right\}$$

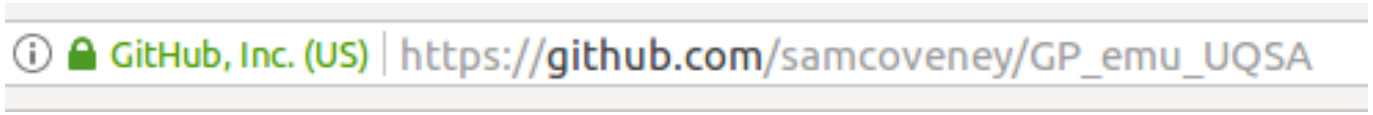
The **hyperparameters** σ, δ, β_0 and β_1 need to be fit to our data.

We usually seek to maximise the loglikelihood function, which balances fitting the data with making the model specific (no fitting elephants).

GP_emu

Free Software for Model Analysis

Open-source Python implementation for doing things with GPs.



www.github.com/samcoveney/GP_emu_UQSA

Why GP_emu ?

- easy to use - needs minimal knowledge of GPs
- incorporates training and validation methodology naturally
- includes an expanding range of powerful model analysis tools

GP_emu

Using GP_emu

How to use GP_emu

- specify parameters in a 'beliefs file'
- specify fitting configuration in a 'config file'
- use GP_emu in a simple python script

```
import gp_emu_uqsa as g

#### set up the emulator
emul = g.setup("config")

#### plot before fitting
g.plot(emul, [0], "mean", points=True)

#### training - fits GP to data
g.train(emul)

#### plot after fitting
g.plot(emul, [0], "mean", points=True)
```

GP_emu

Model Analysis

We can use an emulator to do many useful things, some of which cannot be done easily otherwise.

Uses of emulators in GP_emu :

- **uncertainty quantification** - estimate outputs from uncertain inputs
- **sensitivity analysis** - calculate sensitivity indices for the inputs
- **history matching** - given model outputs, determine which inputs could have plausibly produced these outputs (model calibration)
- **noise fitting** - learn the functional form (input dependence) of output variance for stochastic simulations

GP_emu

UQSA example

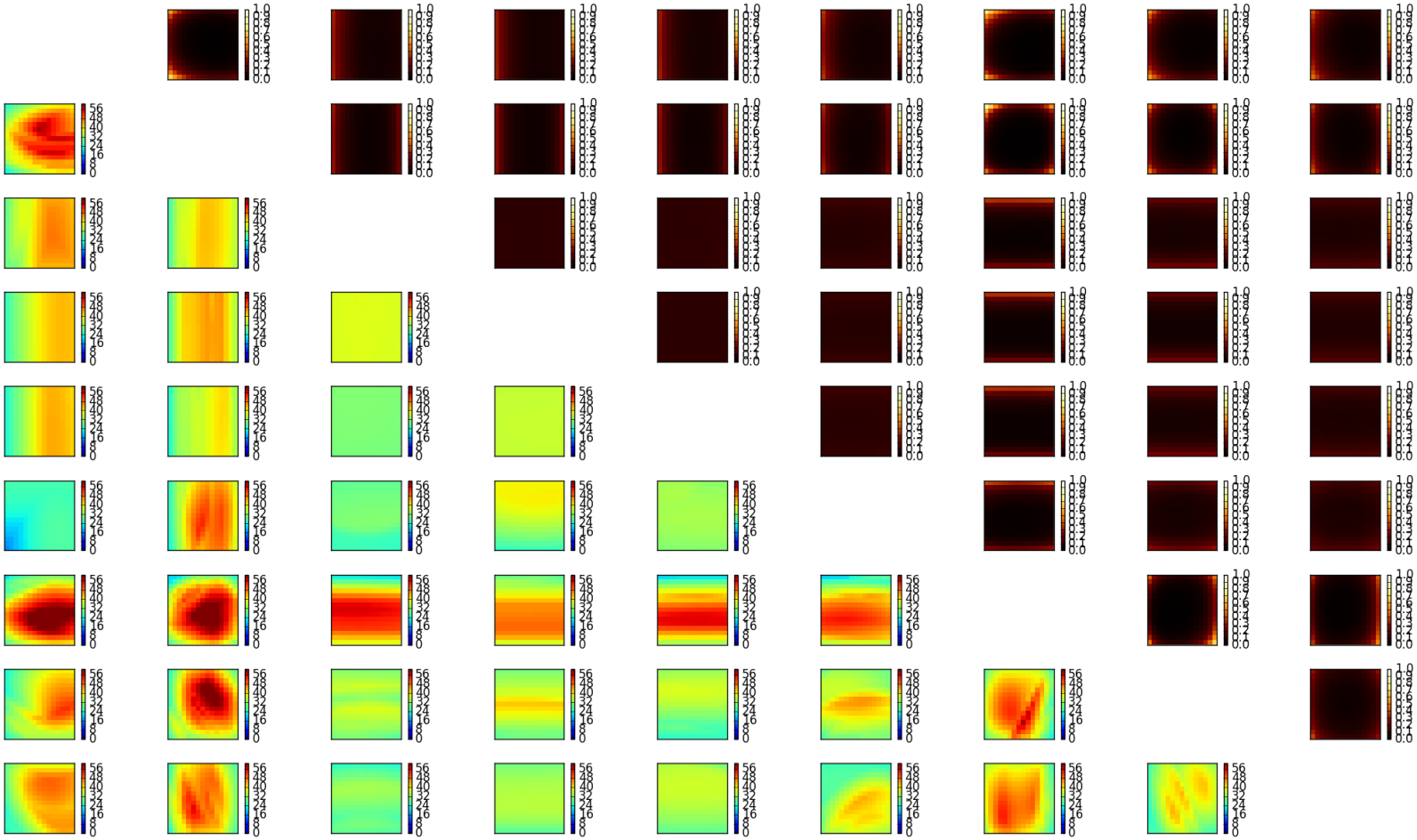
For Uncertainty Quantification and Sensitivity Analysis, inputs can be treated as uncertain using Gaussian distributions with specified mean and variance.

```
#### set up sensitivity analysis
import gp_emu_uqsa.sensitivity as s

## setup sensitivity class
m = [0.50, 0.50]  # mean
v = [0.02, 0.02]  # variance
sens = s.setup(emul, m, v)

## call UQSA functions
sens.uncertainty()
sens.sensitivity()
sens.main_effect(plot=True)
```

GP_emu

History Matching

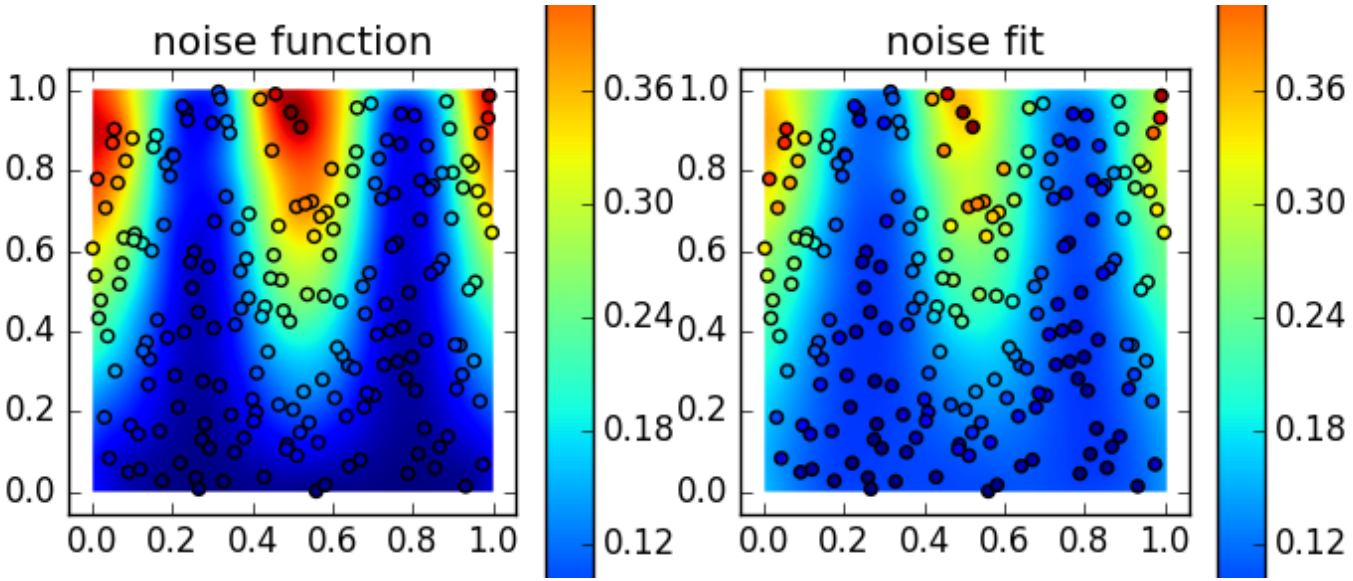
Given some outputs, which inputs could have plausibly been responsible?



GP_emu

Noise Fitting

Sometimes simulations are stochastic, and sometimes the variance of an output depends on the input: $y = f(x) + \sigma(x) * \mathbf{N}$



Note that very few simulation runs have been used, and the simulation has not been repeated for the same input.

GP_emu

Download Page

Get GP_emu today at an Internet near you!

GitHub, Inc. (US) | https://github.com/samcoveney/GP_emu_UQSA

Thanks for listening. Here are some **emu** chicks.