

Gaussian Process Summer School Summary

(by a bewildered layperson)

Dr. Simon Hadfield
S.Hadfield@surrey.ac.uk



Talk Structure

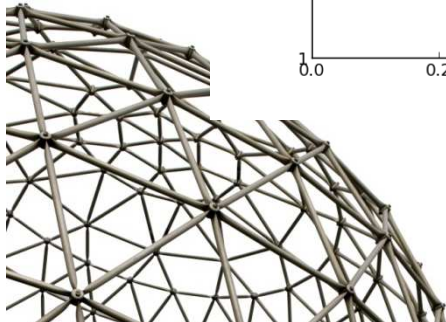
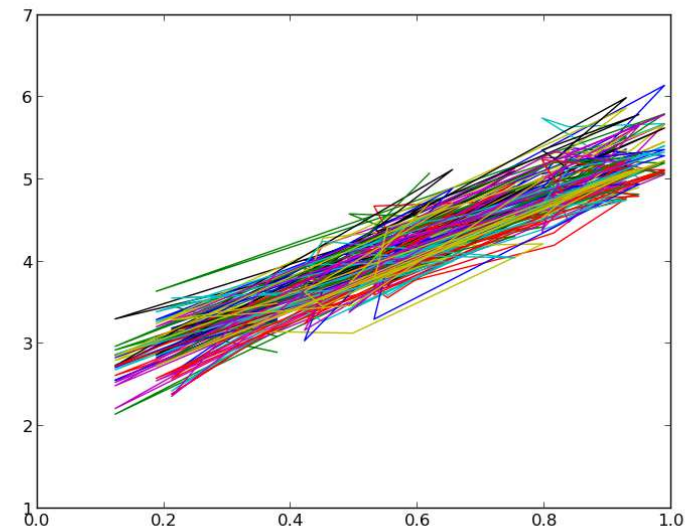
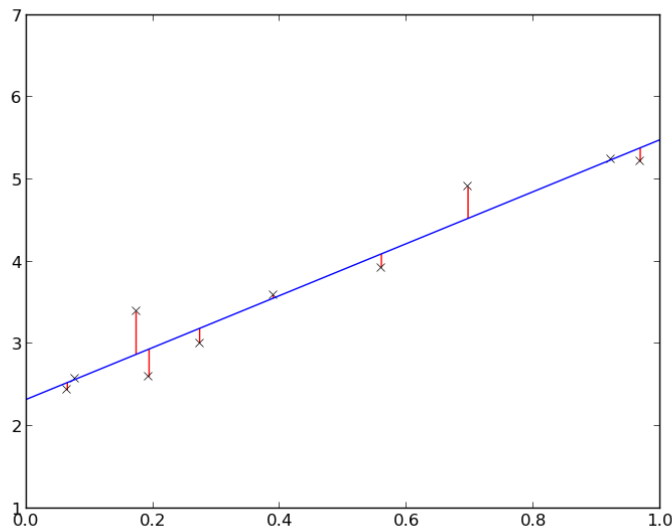
- What are GP's, what are they good for?
- Application to various machine learning tasks:
 - Regression
 - Classification
 - Optimisation
 - Dimensionality reduction + state space representation



1 - Intro	2 - Regression	3 - Classification	4 - Optimisation	5 - Misc
1.1 - What	1.2 - Why	1.3 - How	1.4 - GPy	1.5 - Kernels

What are GPs?

- We want to model something with a function
- Don't just choose the single "best" function (e.g. least squares)
- GP's instead have a distribution over a family functions

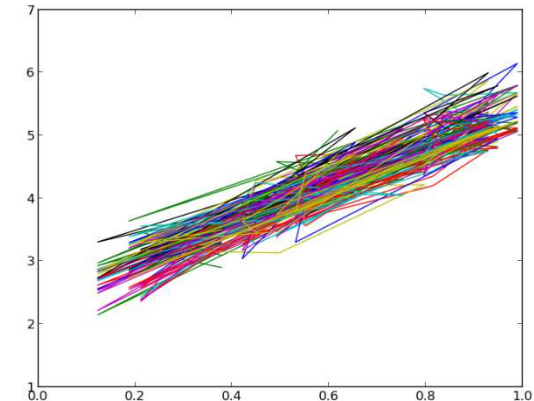
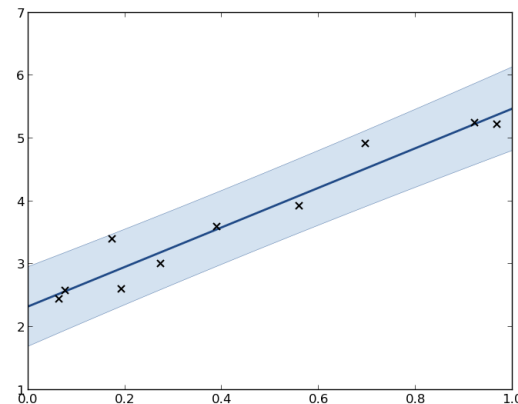
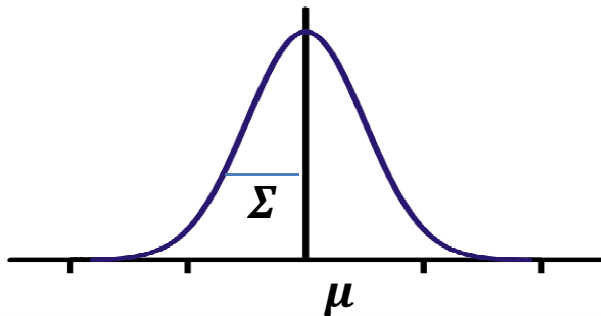


What are GPs?

- A Gaussian distribution is defined by
- A GP is defined by

$$\mathbf{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$\mathbf{X} \sim GP(m(\mathbf{x}), K(\mathbf{x}, \mathbf{x}'))$$



Probability distribution across functions



1 - Intro	2 - Regression	3 - Classification	4 - Optimisation	5 - Misc
1.1 - What	1.2 - Why	1.3 - How	1.4 - GPy	1.5 - Kernels

Why should I use GPs?

Pros

- Model includes uncertainty
- Not limited by model parameterisation (infinitely parametric)
- Can exploit prior knowledge

Cons

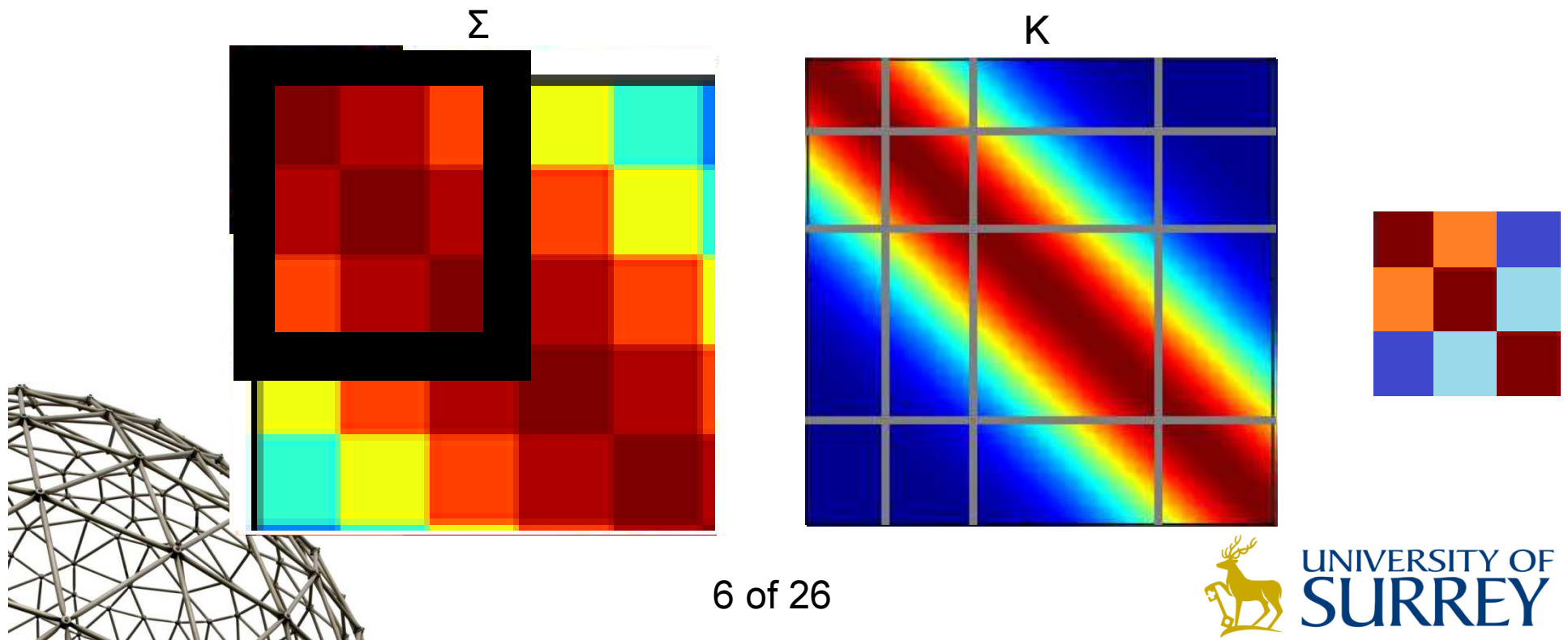
- Need prior knowledge
- Speed = $O(n^3)$ (in data, not dimensions)
- Outliers?



1 - Intro	2 - Regression	3 - Classification	4 - Optimisation	5 - Misc
1.1 - What	1.2 - Why	1.3 - How	1.4 - GPy	1.5 - Kernels

How do GPs work?

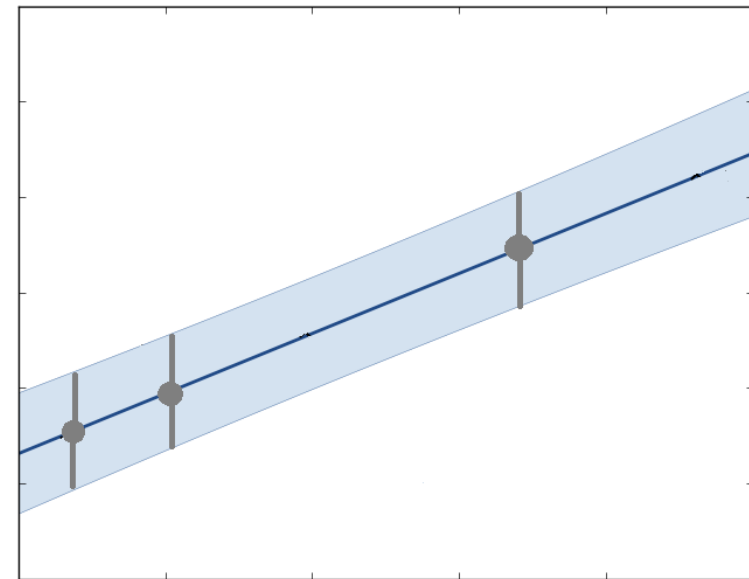
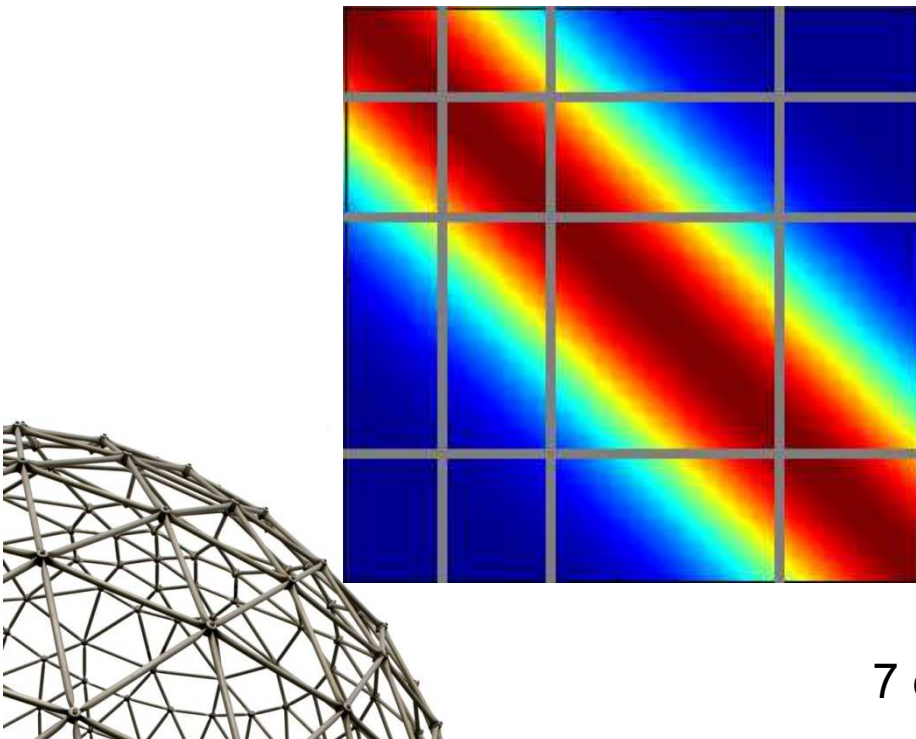
- Define covariance K as a function relating any 2 input points
- $\approx$ Covariance matrix with infinite number of entries
- Marginalising over some dimensions = removing from Σ
- We can “Marginalise” over infinite number of dimensions in K



1 - Intro	2 - Regression	3 - Classification	4 - Optimisation	5 - Misc
1.1 - What	1.2 - Why	1.3 - How	1.4 - GPy	1.5 - Kernels

How do GPs work?

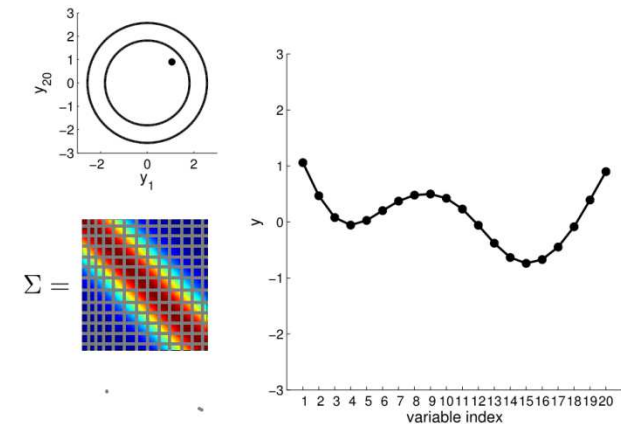
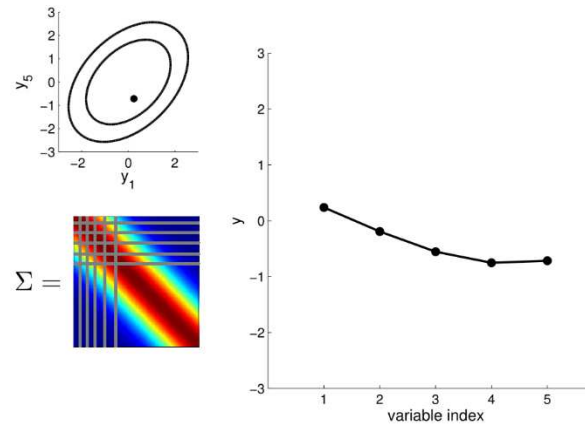
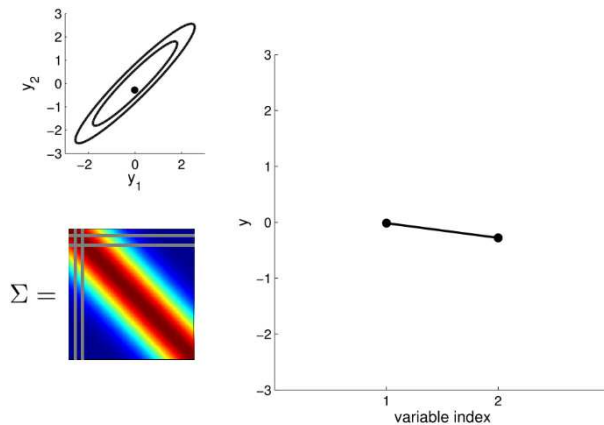
- In these plots, x axis isn't a single dimension of the state X
- It's the indices of these dimensions
- Relates to the continuous covariance function
- Each dimension is independently Gaussian distributed



1 - Intro	2 - Regression	3 - Classification	4 - Optimisation	5 - Misc
1.1 - What	1.2 - Why	1.3 - How	1.4 - GPy	1.5 - Kernels

How do GPs work?

- Plotting dimensions of samples from high dimensional gaussians
- Gaussian assumption on per dimension basis
- Between dimensions we can have crazier relationships



1 - Intro	2 - Regression	3 - Classification	4 - Optimisation	5 - Misc
1.1 - What	1.2 - Why	1.3 - How	1.4 - GPy	1.5 - Kernels

GPs in Python

- GPy provided by Sheffield
- Snippets throughout talk
- Need python 2.7
- Perform the setup step each time you start ipython
- Documentation at <https://github.com/SheffieldML/GPy>

Install GPy

```
pip install GPy
```

Setup GPy

```
ipython  
> import numpy as np  
> import pylab as pb  
> import GPy  
> pylab  
> pb.ion()  
>
```

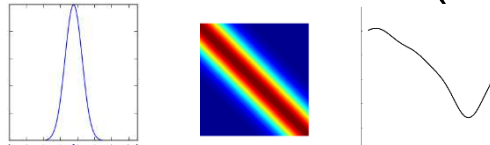


1 - Intro	2 - Regression	3 - Classification	4 - Optimisation	5 - Misc
1.1 - What	1.2 - Why	1.3 - How	1.4 - GPy	1.5 - Kernels

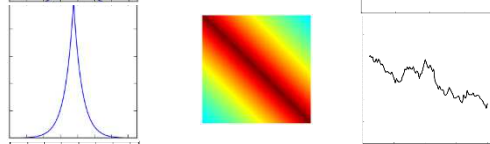
Covariance Functions

- Many possible forms for covariance K (also called the kernel)

- RBF



- Exponential



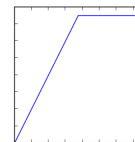
- Constant / Bias



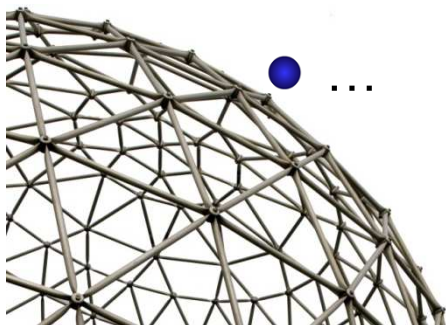
- RBF COS



- Brownian

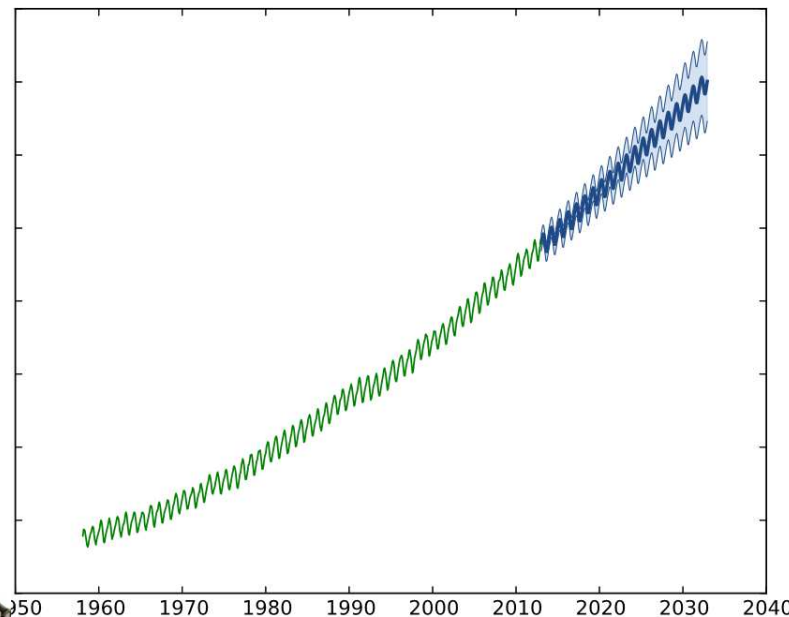
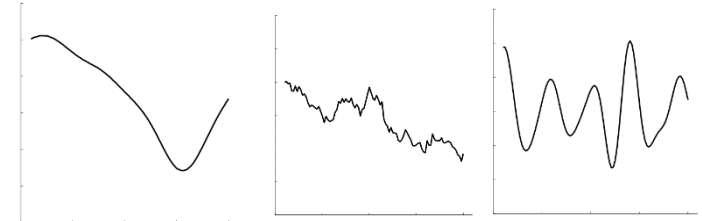


- ...



Covariance Functions

- Some important properties
 - Smoothness
 - Periodicity
 - Stationarity (depends on difference $|x - x'|$)
- Sum and/or Product of kernels is a kernel
- Hyperparameters α, λ



Create kernels

```
> dims=2  
> var=3  
> len=1  
> freq=4  
> k1=GPy.kern.exponential(dims,var,len)  
> k2=GPy.kern.rbfcos(dims,var,len,freq)  
> k3=k1+k2  
> k4=k1*k2
```

1 - Intro	2 - Regression	3 - Classification	4 - Optimisation	5 - Misc
2.1 - Hyperparameters	2.2 - Inference	2.3 - Sparse GPs	2.4 - Multi-output GPs	

Maximum Likelihood parameter selection

- Choose hyperparameters which best fit the observations

$$\log p(\mathbf{y}(\mathbf{x})|\theta, \mathbf{x}) = \underbrace{-\frac{1}{2}\mathbf{y}(\mathbf{x})^T K(\mathbf{x}, \mathbf{x}')^{-1} \mathbf{y}(\mathbf{x})}_{\text{Data term}} \underbrace{-\frac{1}{2}\log \det(K(\mathbf{x}, \mathbf{x}')) - \frac{|\mathbf{x}|}{2}\log 2\pi}_{\text{Complexity term}}$$

Data term

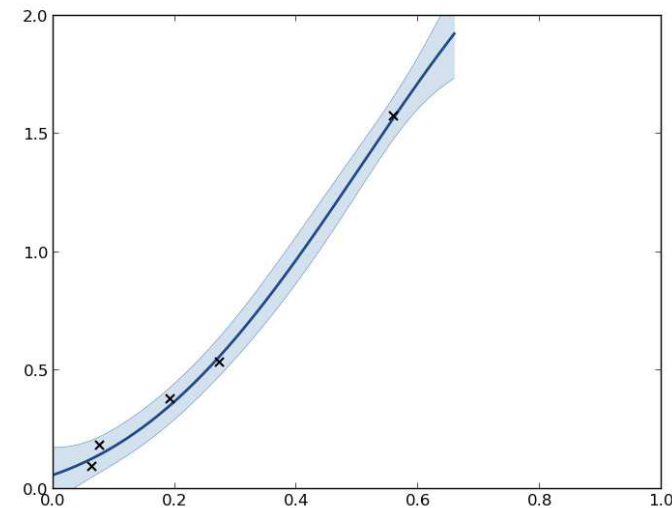
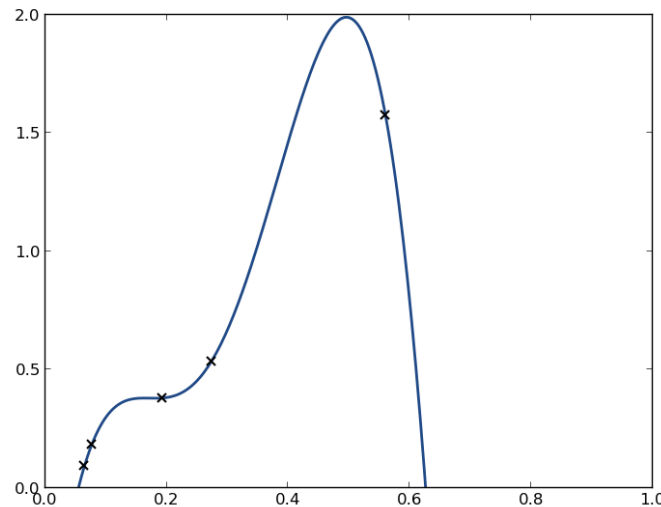
Complexity term

- Warning! Not convex.
- Poor initialisation will lead to poor results



Observation noise

- Typically assume some observation noise
- Functions within distribution do not have to intersect every point
- Maximum likelihood with Gaussian noise = least squares



Setting noise level

```
> m['noise_variance']=0  
> m['noise_variance']=0.1
```



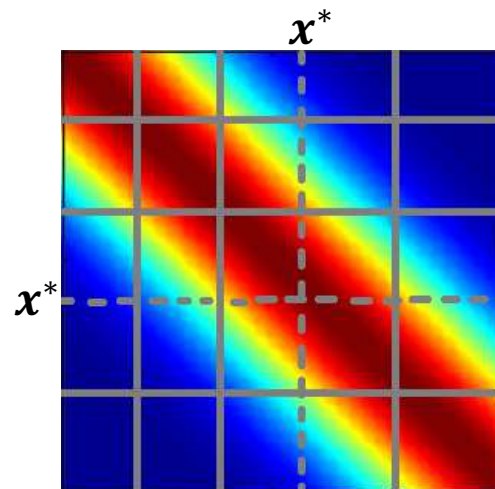
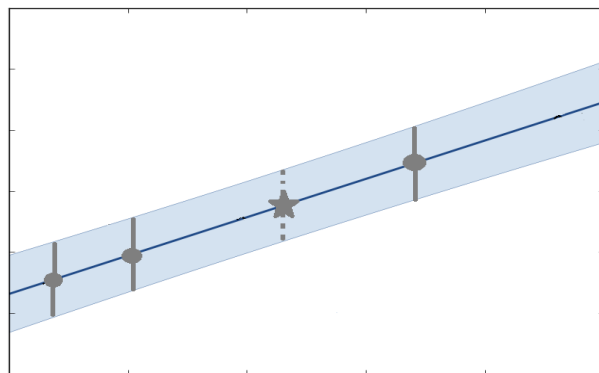
1 - Intro	2 - Regression	3 - Classification	4 - Optimisation	5 - Misc
2.1 - Hyperparameters	2.1 - Inference	2.3 - Sparse GPs	2.4 - Multi-output GPs	

Regression / prediction / inference with GPs

- Want to estimate value of y at unobserved position x^*

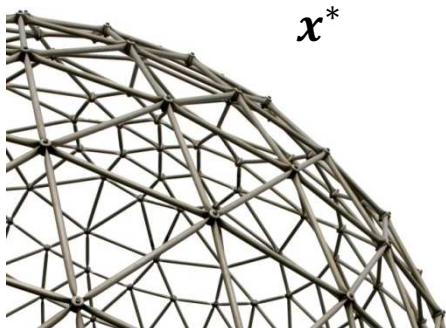
$$y_{\mu}(x^*) = K(x^*, x)K(x, x')^{-1}y(x)$$

$$y_{\Sigma}(x^*) = K(x^*, x) - K(x^*, x)K(x, x')^{-1}K(x^*, x)^T$$



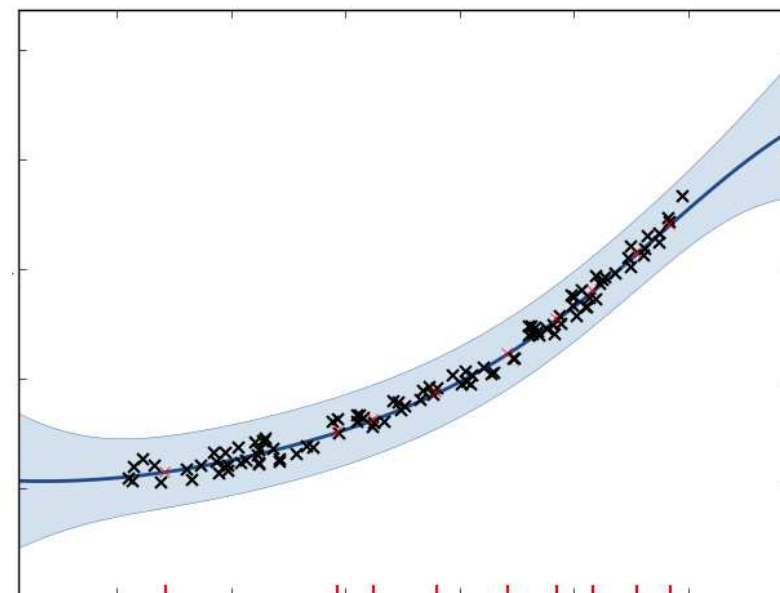
Regression + Parameter Optimisation

```
> m=GPy.models.GPRegression(x,y,k)
> m.ensure_default_constraints()
> m.optimize()
> m.plot()
```



Sparse GP approximation

- A full GP is $O(n^3)$, n is the number of data points
- Instead we can approximate it with $O(nm^2)$, where we choose m
- As $m \rightarrow n$
 - the approximation becomes perfect
 - complexity returns to cubic
- Optimise θ on small subset
- Penalise deviation from original data
- Position of “inducing” variables optimised



Sparse GP

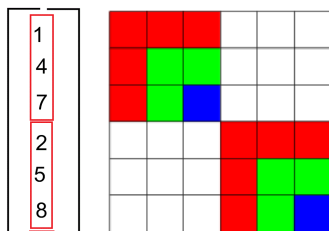
```
> m=Gpy.models.SparseGPRegression(X,Y,k,Z=z)
> m.ensure_default_constraints()
> m.optimize()
> m.plot()
```



Multivariate GPs

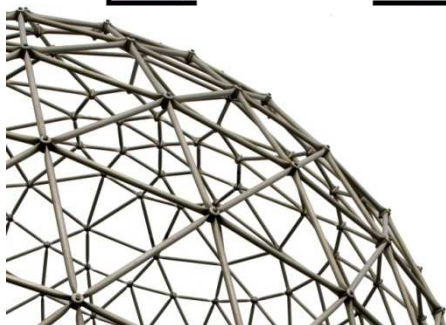
- GPs with a number of independent outputs
- Stack input vectors
- Block diagonal covariance structure
- Produce covariance function via kronecker product with identity
- Useful for markov systems

$$X = \begin{bmatrix} \boxed{1} & \boxed{2} \\ \boxed{4} & \boxed{5} \\ \boxed{7} & \boxed{8} \end{bmatrix}$$



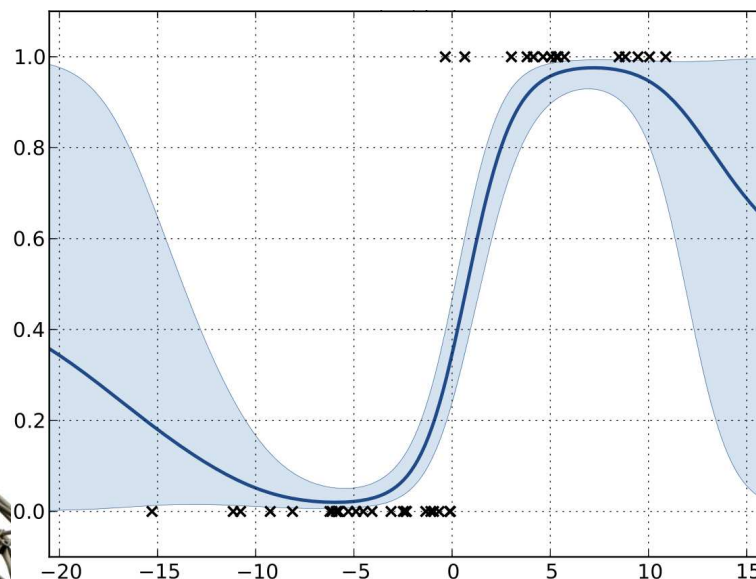
$$= \begin{matrix} & I \\ \begin{bmatrix} \blacksquare & \square \\ \square & \blacksquare \end{bmatrix} & \otimes \end{matrix} \begin{matrix} K_s \\ \begin{bmatrix} \text{red} & \text{red} & \text{red} \\ \text{red} & \text{green} & \text{green} \\ \text{red} & \text{green} & \text{blue} \end{bmatrix} \end{matrix}$$

$$K = I \otimes K_s$$



Non-Gaussian Likelihoods

- Classification implies non-Gaussian likelihoods
- No analytic solution
- Approximate likelihood by product of Gaussians
- Minimise KL divergence of approximate likelihood
- Also encode the fact that values must lie between 0 and 1



GP Classifier

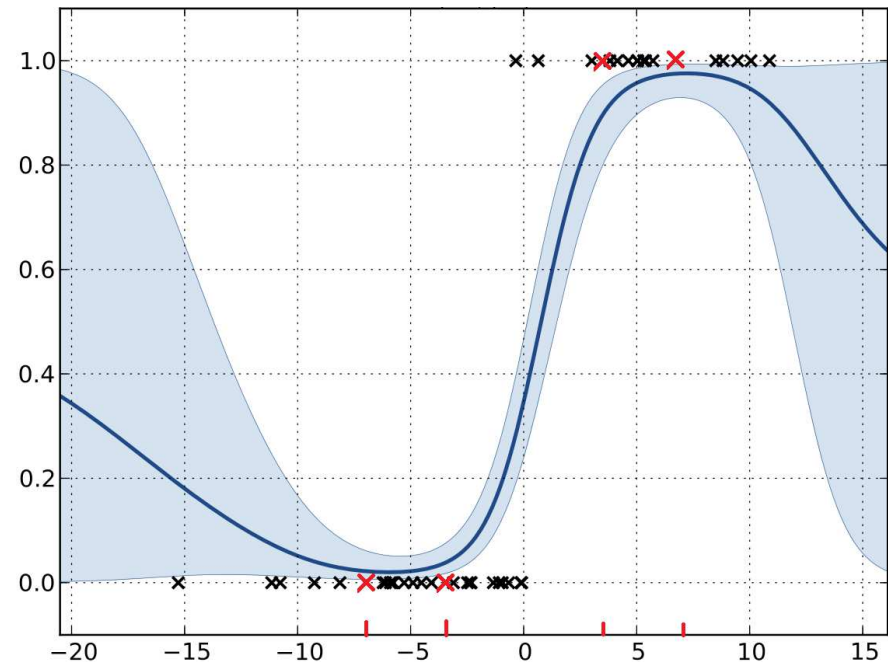
```
> m=Gpy.models.GPClassification(X,Y,k)
> m.ensure_default_constraints()
> for i in range(10):
>   m.update_likelihood_approximation()
>   m.optimise()
> m.plot()
```

Sparse Classification

- GP classification can be slow
- Can use sparse approximation from earlier
- Same iterative optimization

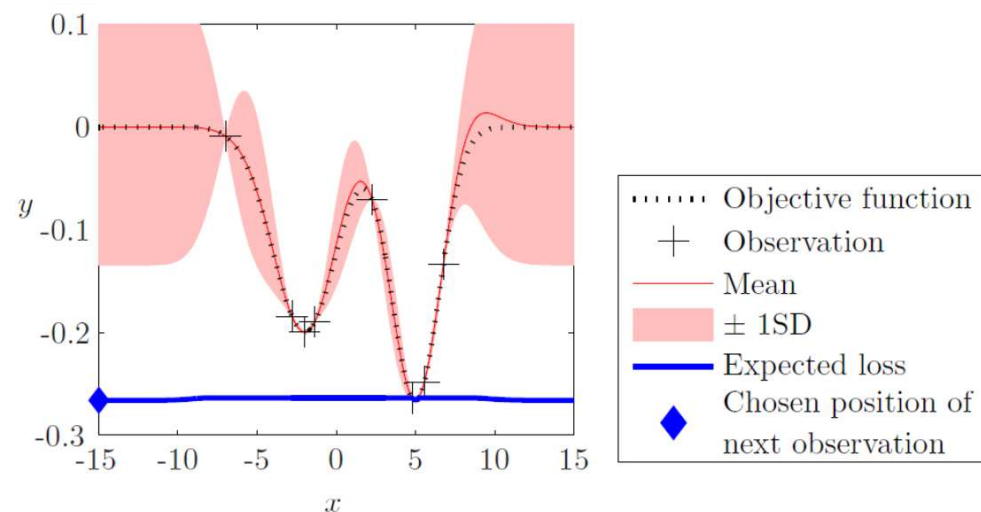
Sparse GP Classifier

```
> m=Gpy.models.SparseGPClassification(X,Y,k,Z)
> m.ensure_default_constraints()
> for i in range(10):
>   m.update_likelihood_approximation()
>   m.optimise()
> m.plot()
```



Gaussian Process Global Optimisation

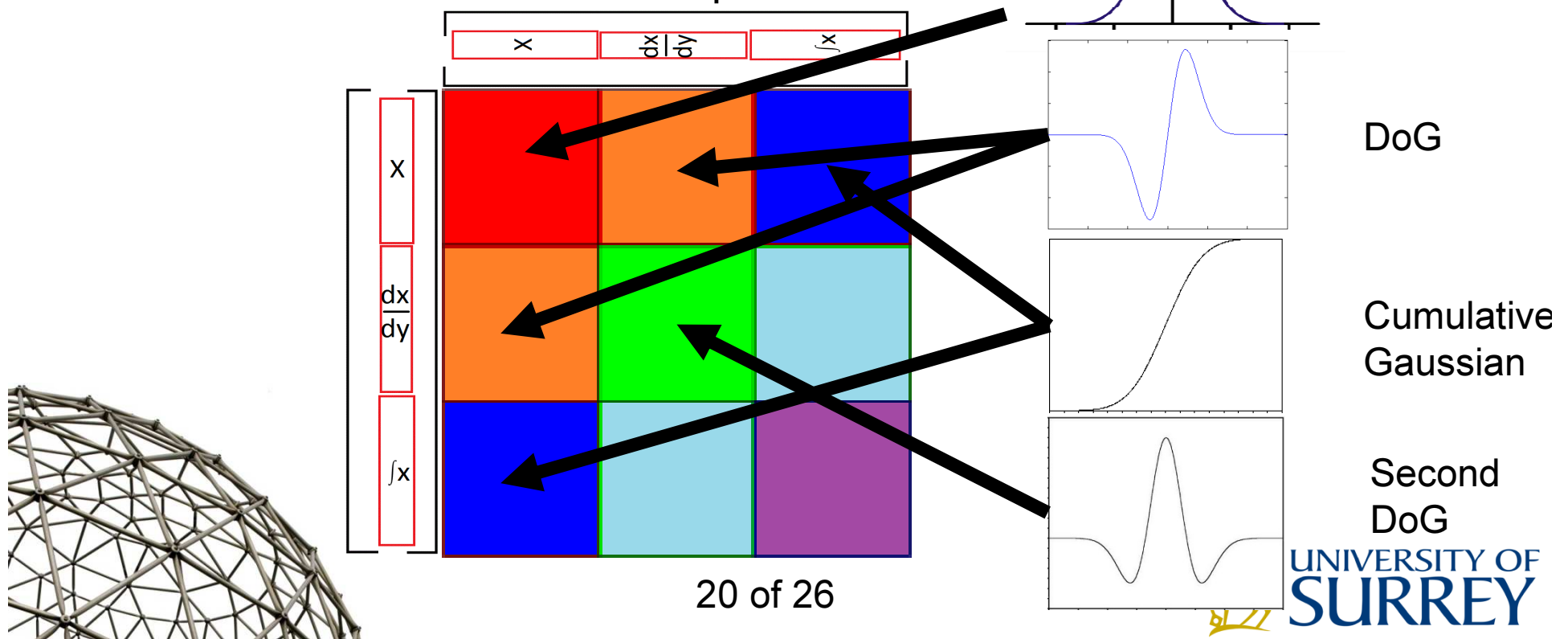
- Optimisation scheme by Michael Osborne (Oxford)
- Model objective function with GP
- Sample where we have best probability of beating current estimate
- Doesn't quit at local minima
- Combine refinement + exploration
- As number of samples $\rightarrow \infty$ obtain global optima?
- Doesn't extend well to high dimensionality



1 - Intro	2 - Regression	3 - Classification	4 - Optimisation	5 - Misc
4.1 - GPGO	4.2 - Calculus in GPs	4.3 - GPs for QN		

Derivatives or integrals within covariance function

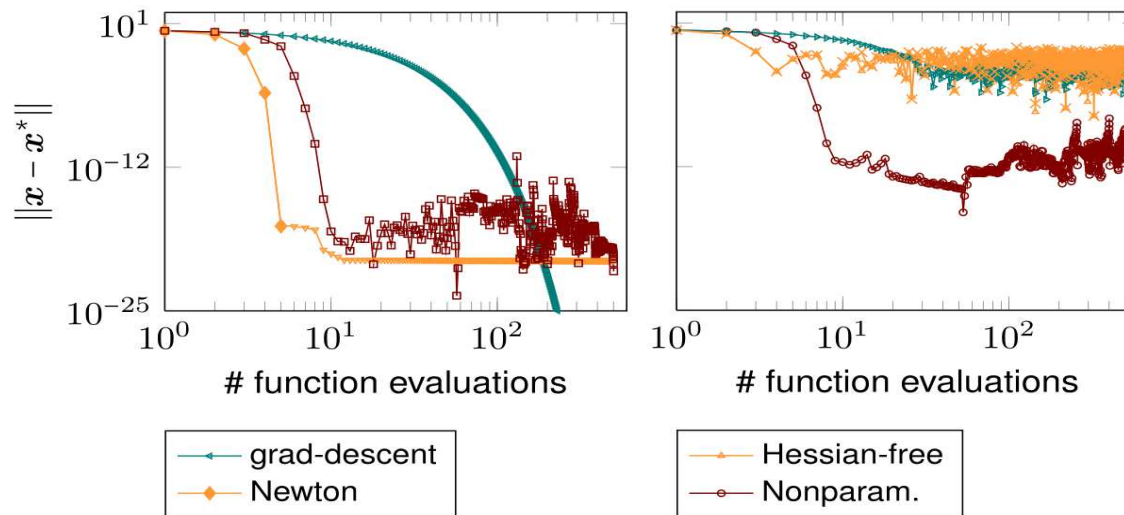
- Returning to stacked states + block diagonal idea
- Can make some blocks of covariance function into:
 - Integral relationships
 - Derivative relationships



1 - Intro	2 - Regression	3 - Classification	4 - Optimisation	5 - Misc
4.1 - GPGO	4.2 - Calculus in GPs		4.3 - GPs for QN	

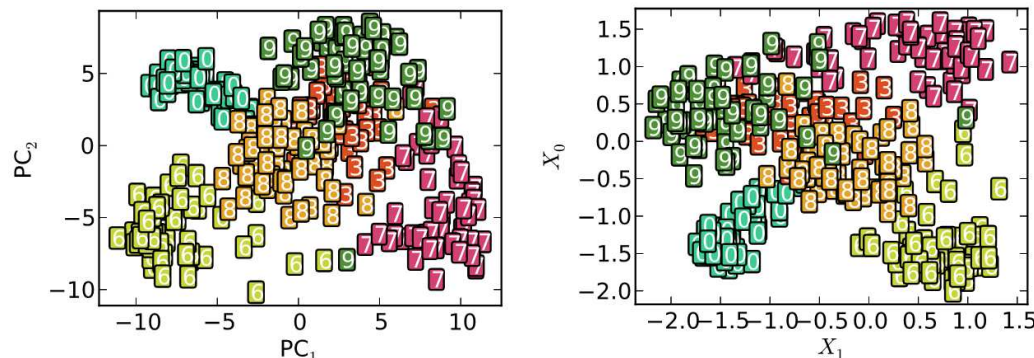
Quasi-Newton optimisation (e.g. BFGS) with GPs

- Good in high dimensions
- QN = max posterior Gaussian inference of Hessian
- Don't pick "best" Hessian, have a distribution over all!
- Same complexity as BFGS $O(n)$ with a constant multiple
- Allows us to handle noisy observations
- No simple interface for GP optimisation in GPy



Dimensionality Reduction

- Mapping from high to low dimensional space
- Minimising lost information
- We can create a distribution of mapping functions (a GP)
- N.B. GP actually low to high
- Adding new data is non-convex



GP dimensionality reduction

```
> m=Gpy.models.GPLVM(Y,input_dims,kernel=k)
> m.ensure_default_constraints()
> m['noise'] = m.likelihood.Y.var()/20. #obs noise < input noise
> m.optimise()
> m.plot()
```

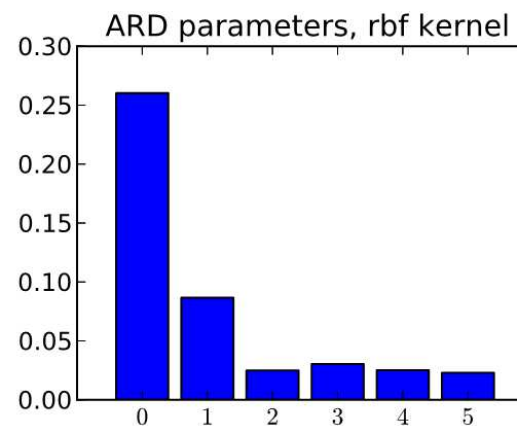
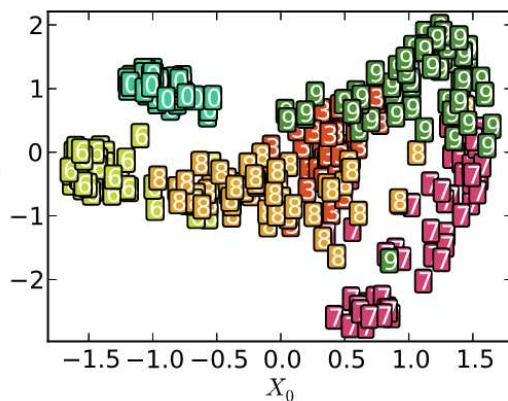
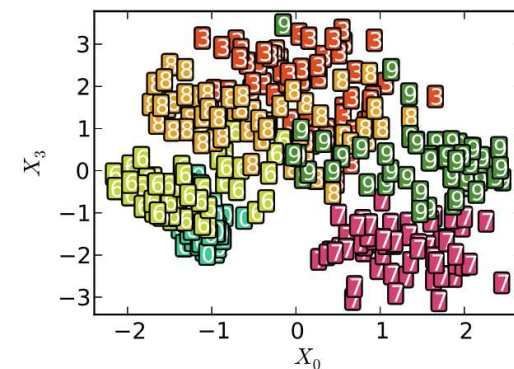
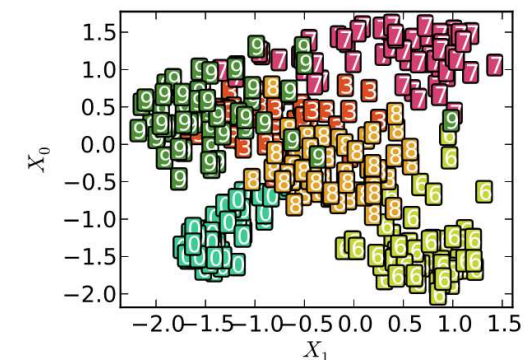
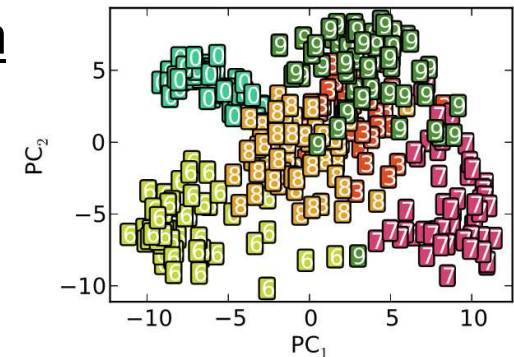


Nonlinear Dimensionality Reduction

- Can use nonlinear bases
- Automatic Relevance Determination (ARD)
- Also full Bayesian schemes possible

Bayesian GP dimensionality reduction with ARD

```
> k=GPy.kern.rbf(input_dims,ARD=True)
> m=Gpy.models.BayesianGPLVM(Y,input_dims,kernel=k)
> ...
> m.kern.plot_ARD()
```



1 - Intro	2 - Regression	3 - Classification	4 - Optimisation	5 - Misc
5.1 - Dimensionality Reduction	5.2 - State Space Models	5.3 - Deep GPs		

State Space

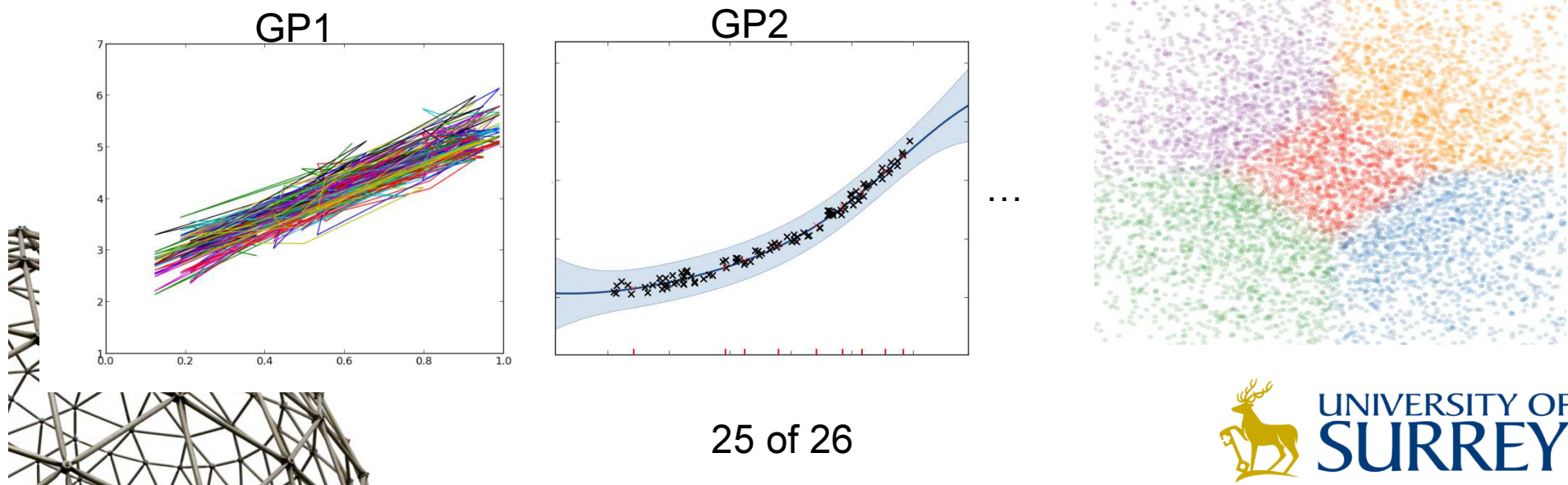
- Covariance function of a GP $\leftrightarrow$ state space
- Kalman Filter can be converted to a form of GP
 - GP form has cubic complexity
 - Flexible time intervals
- Sometimes going the other way brings efficiency
 - If kernels have markov property (stationary?)
 - If GP is 1D
- No need for full covariance
- $O(n)$ with Kalman Filter + Rauch-Tung-Striebel smoother
- Valuable for temporal signals e.g. audio



1 - Intro	2 - Regression	3 - Classification	4 - Optimisation	5 - Misc
5.1 - Dimensionality Reduction	5.2 - State Space Models	5.3 - Deep GPs		

Deep GPs

- A normal GP is a distribution of an infinite number of functions
- Input can be another function distribution (rather than a vector)
- Each of the infinite number of functions contained in GP1 are remapped by the infinite number of functions contained in GP2
- Build a hierarchy of consecutive GP mappings
- Fully Bayesian deep learning
- Can be applied even with small amounts of data



1 - Intro	2 - Regression	3 - Classification	4 - Optimisation	5 - Misc
1.1 - What	1.2 - Why	1.3 - How	1.4 - GPy	1.5 - Kernels

Further Information

- Book - Gaussian Processes for Machine Learning, Carl Edward Rasmusser and Chris Williams, the MIT Press, 2006
 - Free PDF version at www.gaussianprocess.org
- Summer school slides + some recorded talks
 - <http://ml.dcs.ac.uk/gpss>

Questions?

