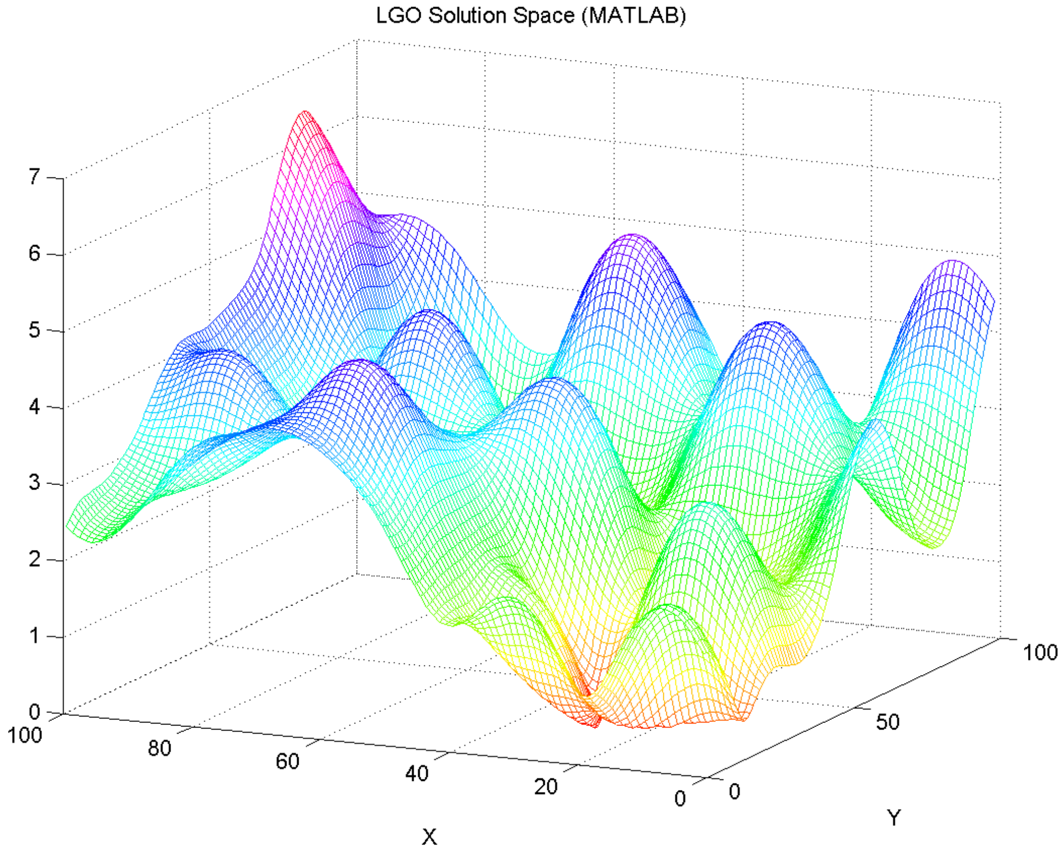
Global Optimisation with Gaussian Processes

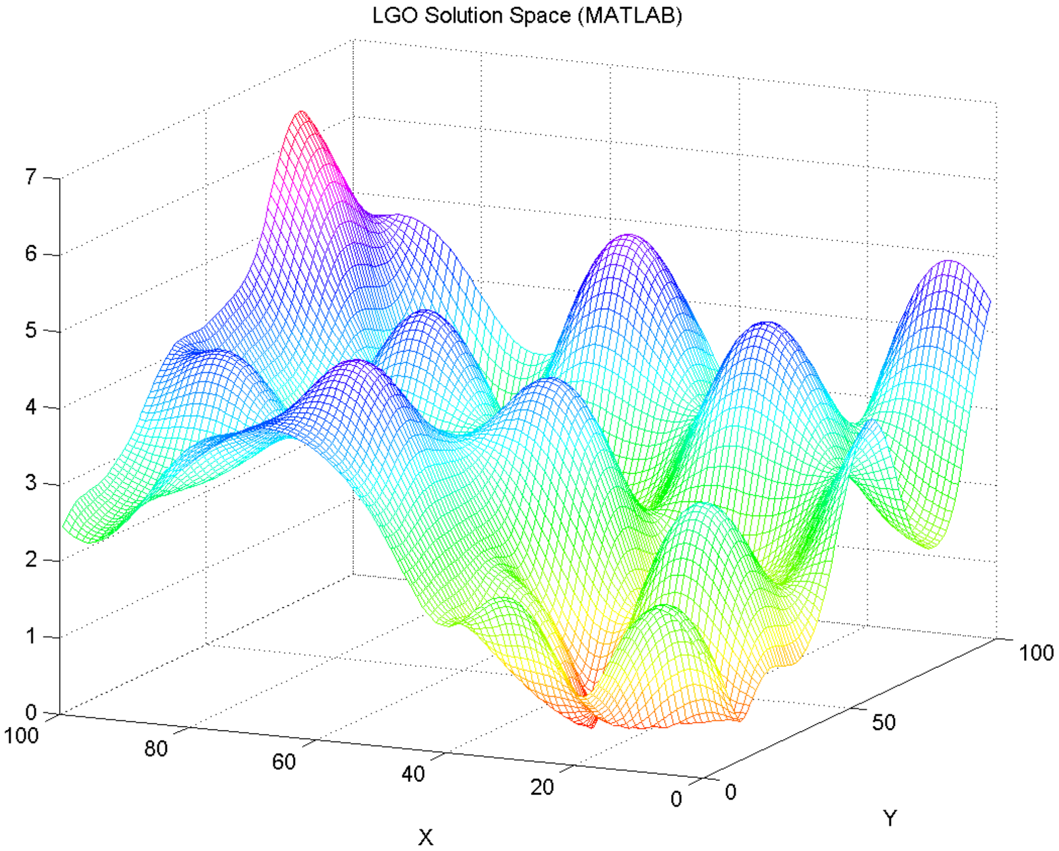
Michael A. Osborne
Machine Learning Research Group
Department of Engineering Science
University of Oxford

UNIVERSITY OF OXFORD

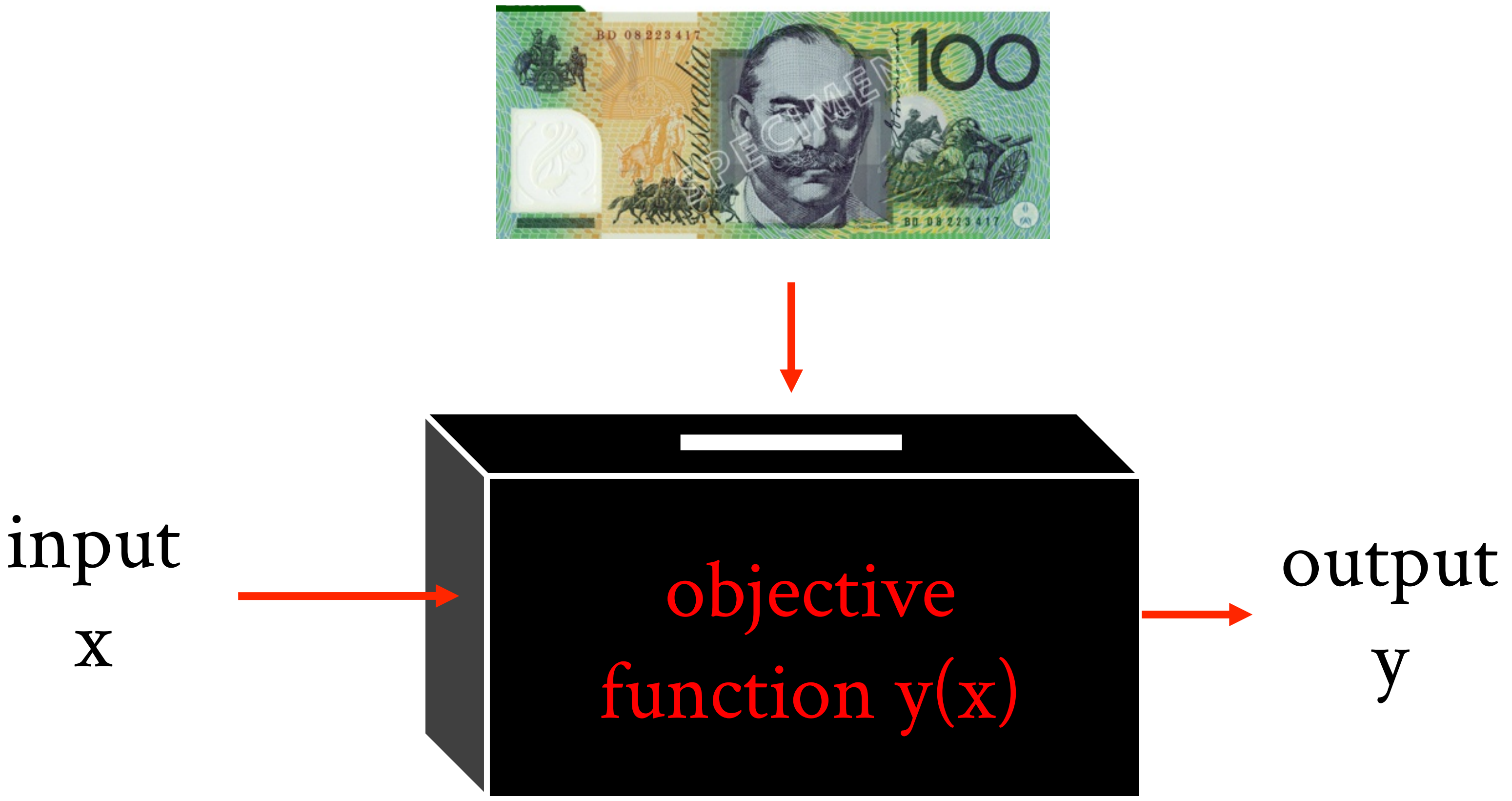
Global optimisation considers objective functions that are multi-modal and expensive to evaluate.



Optimisation is a decision problem: we must select evaluations to determine the minimum. Hence we need to specify a loss function and a probability distribution.



By defining a loss function (including the cost of observation), we can select evaluations optimally by minimising the expected loss.



input x

objective function y(x)

output y

We define a loss function that is equal to the lowest function value found: the lower this value, the lower our loss.

Assuming we have only one evaluation remaining, the loss of it returning value *y* given that the current lowest value obtained is η is

$$\lambda(y) \triangleq \begin{cases} y; & y < \eta \\ \eta; & y \geq \eta \end{cases}.$$

This loss function makes computing the expected loss simple: as such, we'll take a myopic approximation and only ever consider the next evaluation (rather than all possible future evaluations).

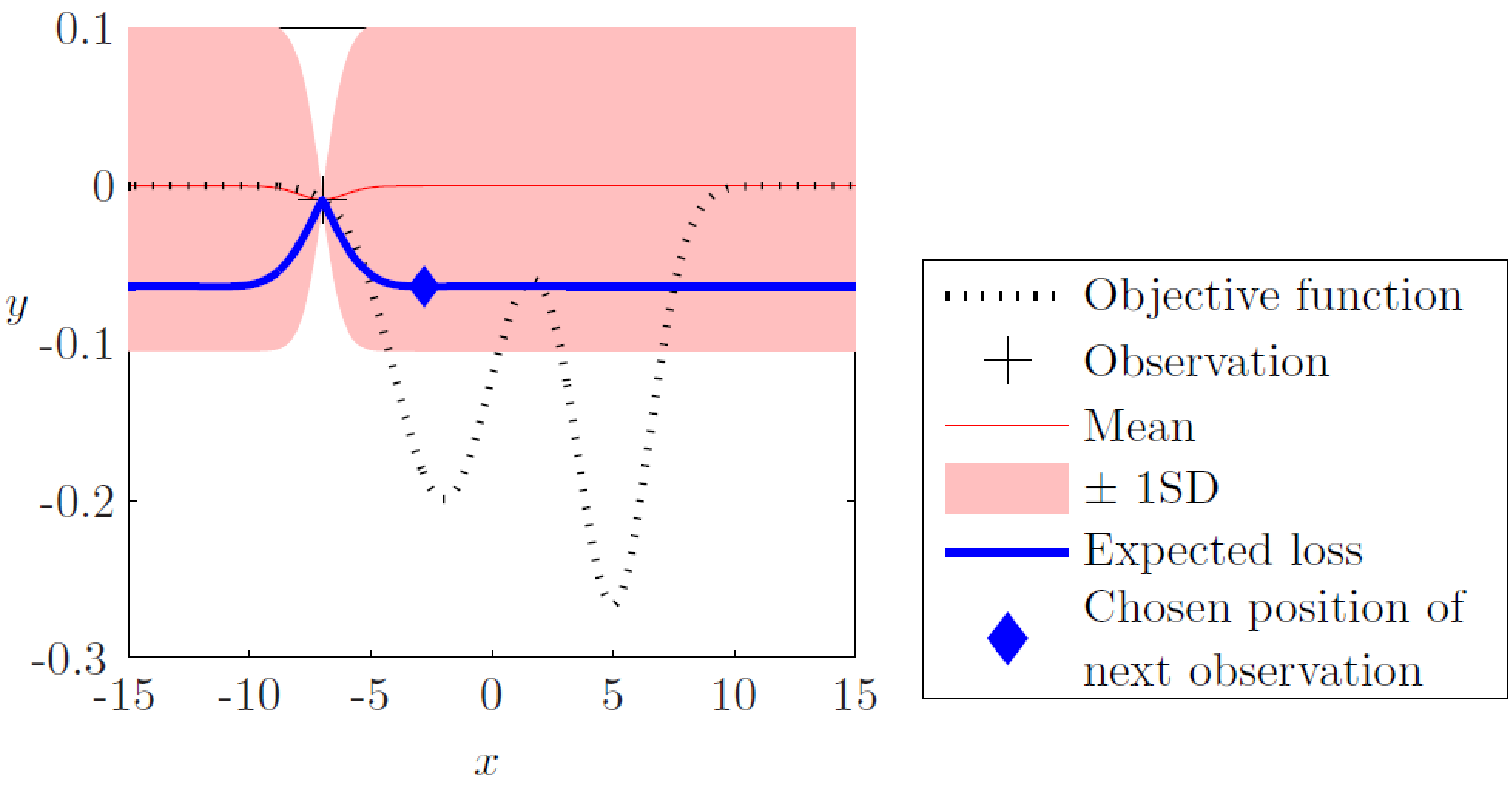
$$\boxed{\int \lambda(y)\, p(\, y \mid x,\, I_0\,)\ \mathrm{d}y}$$

I_0 : All available information.

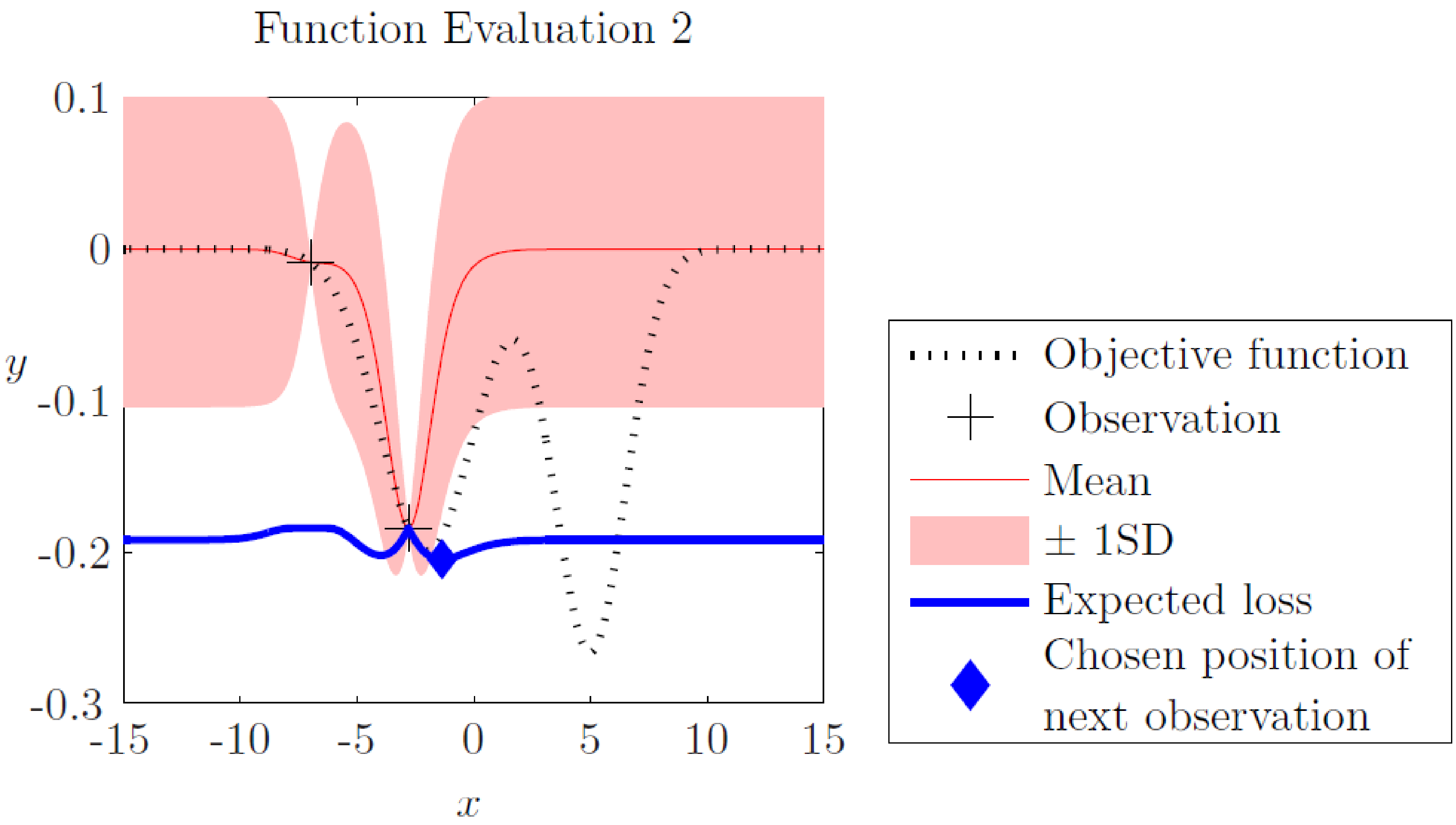
x : next eval. location.

The expected loss is the expected lowest value of the function we've evaluated after the next evaluation.

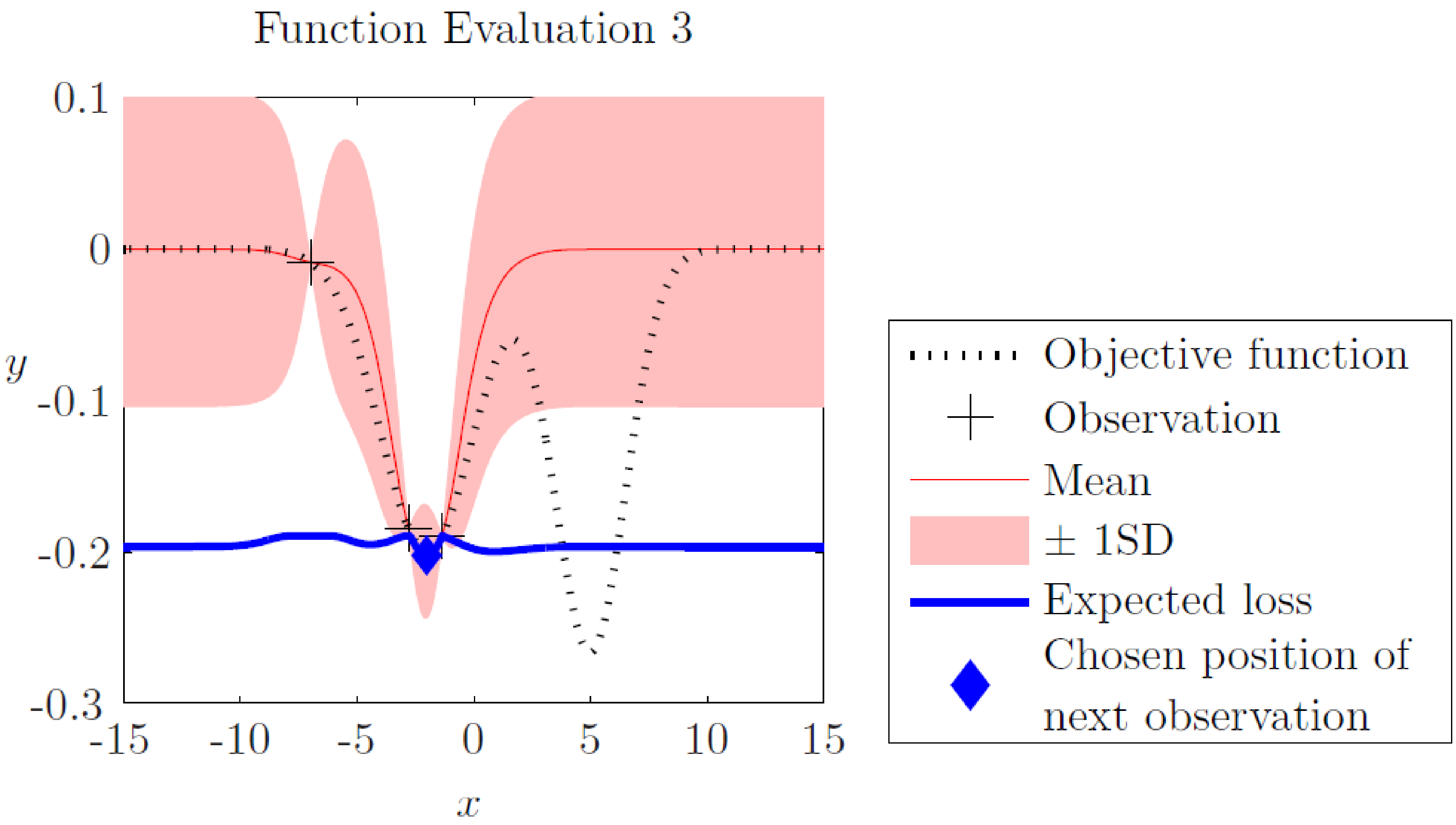
We choose a Gaussian process as the probability distribution for the objective function, giving a tractable expected loss.



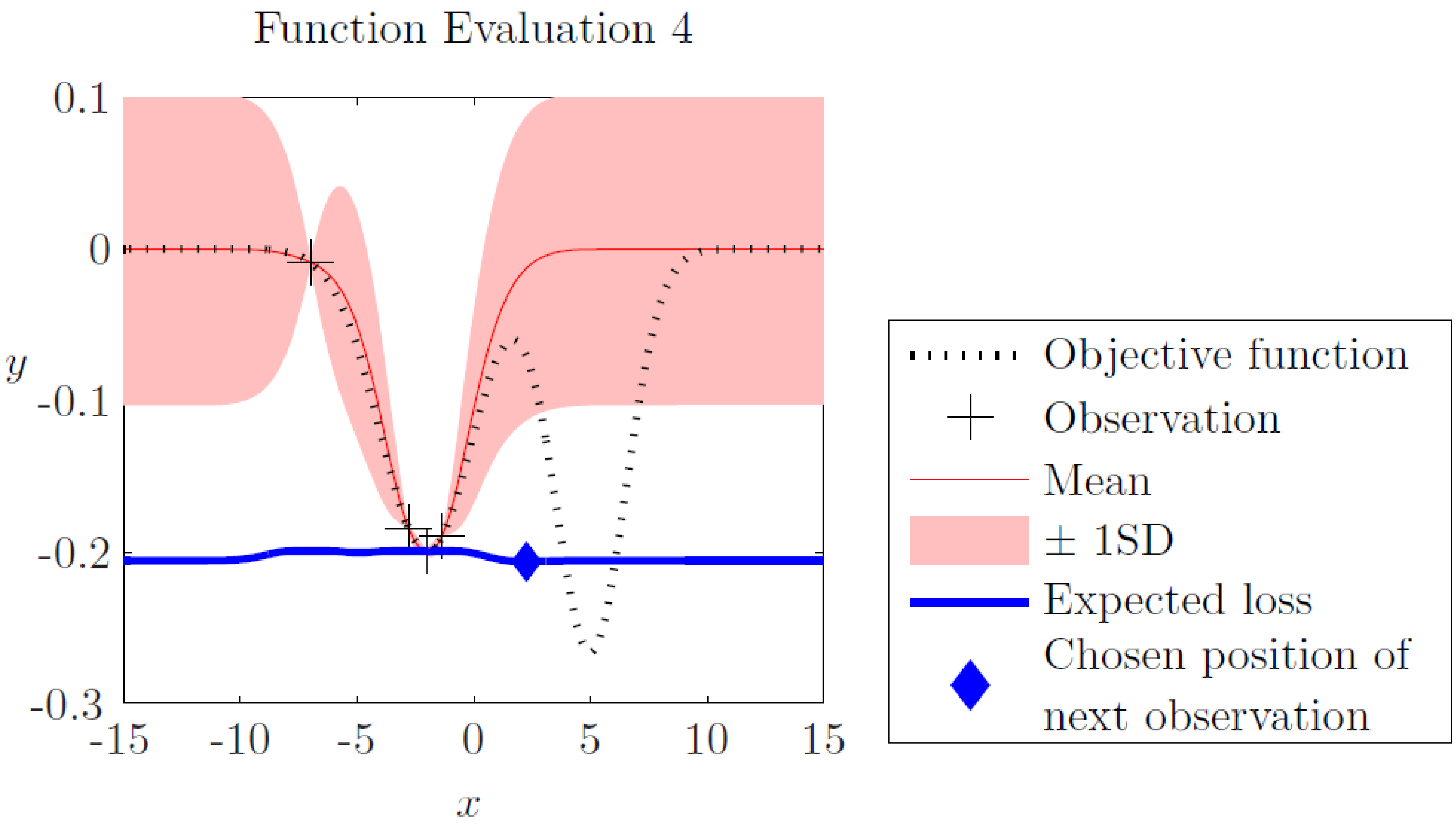
We can use Gaussian processes for optimisation.



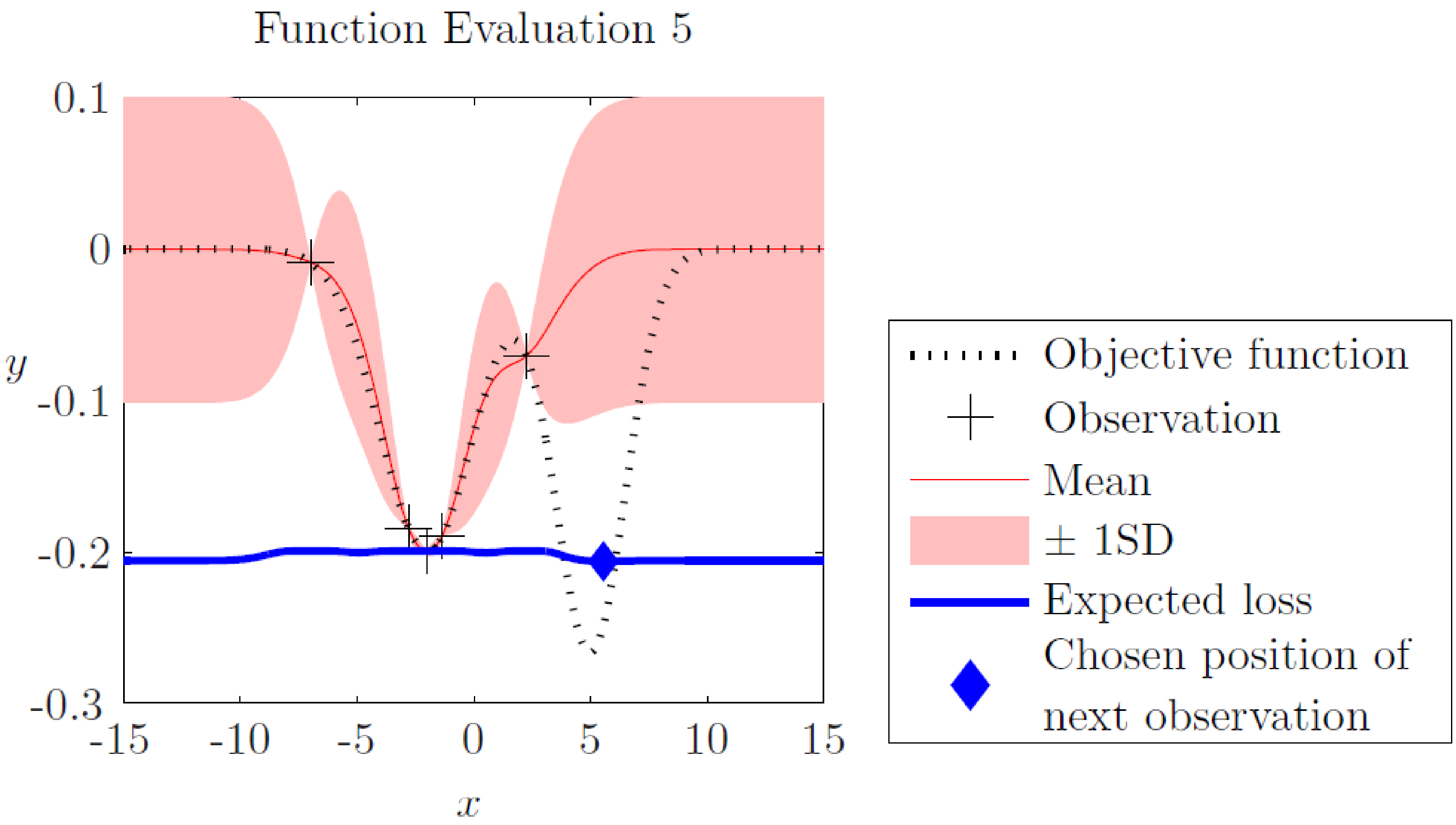
We can use Gaussian processes for optimisation.



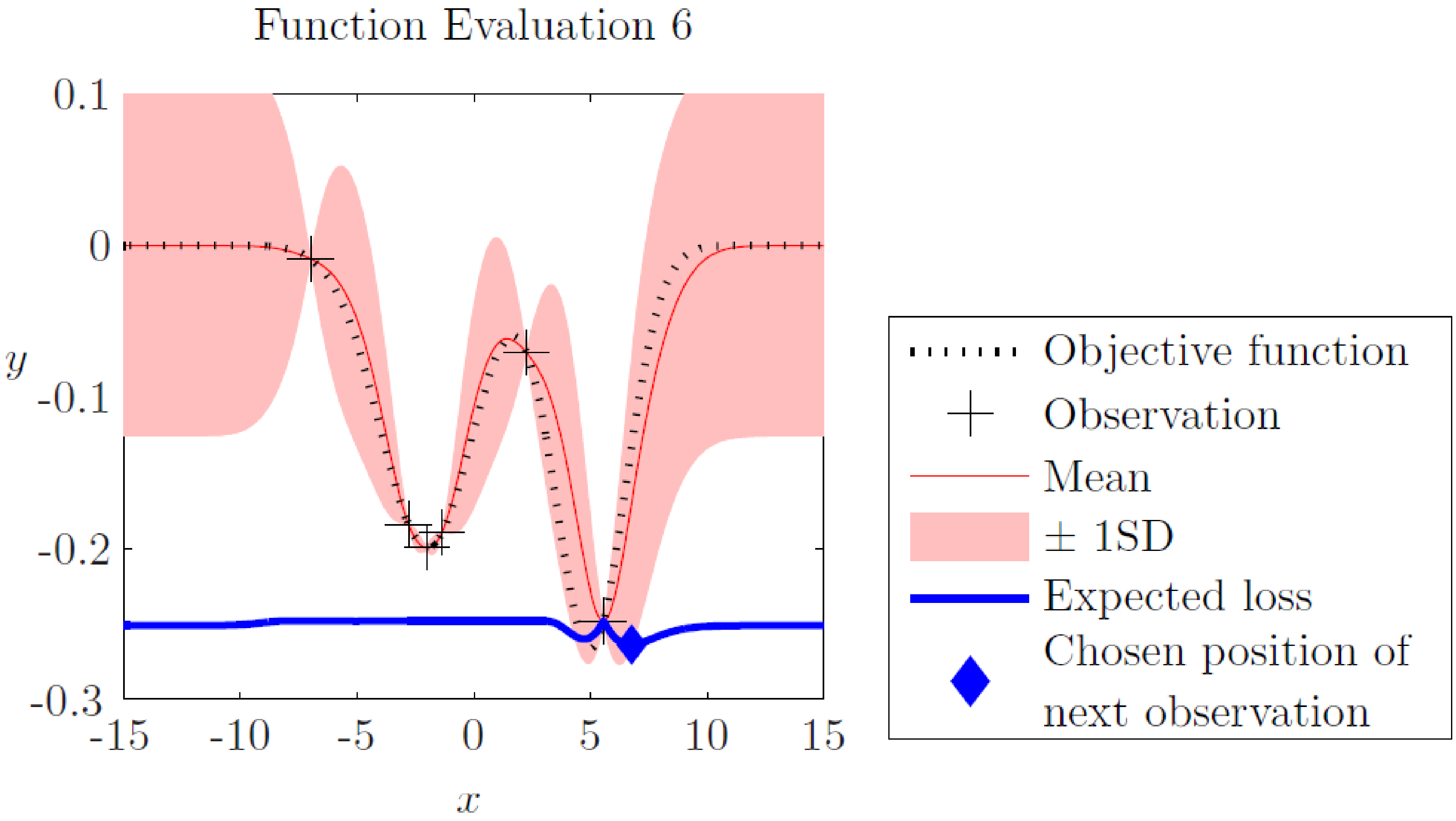
We can use Gaussian processes for optimisation.



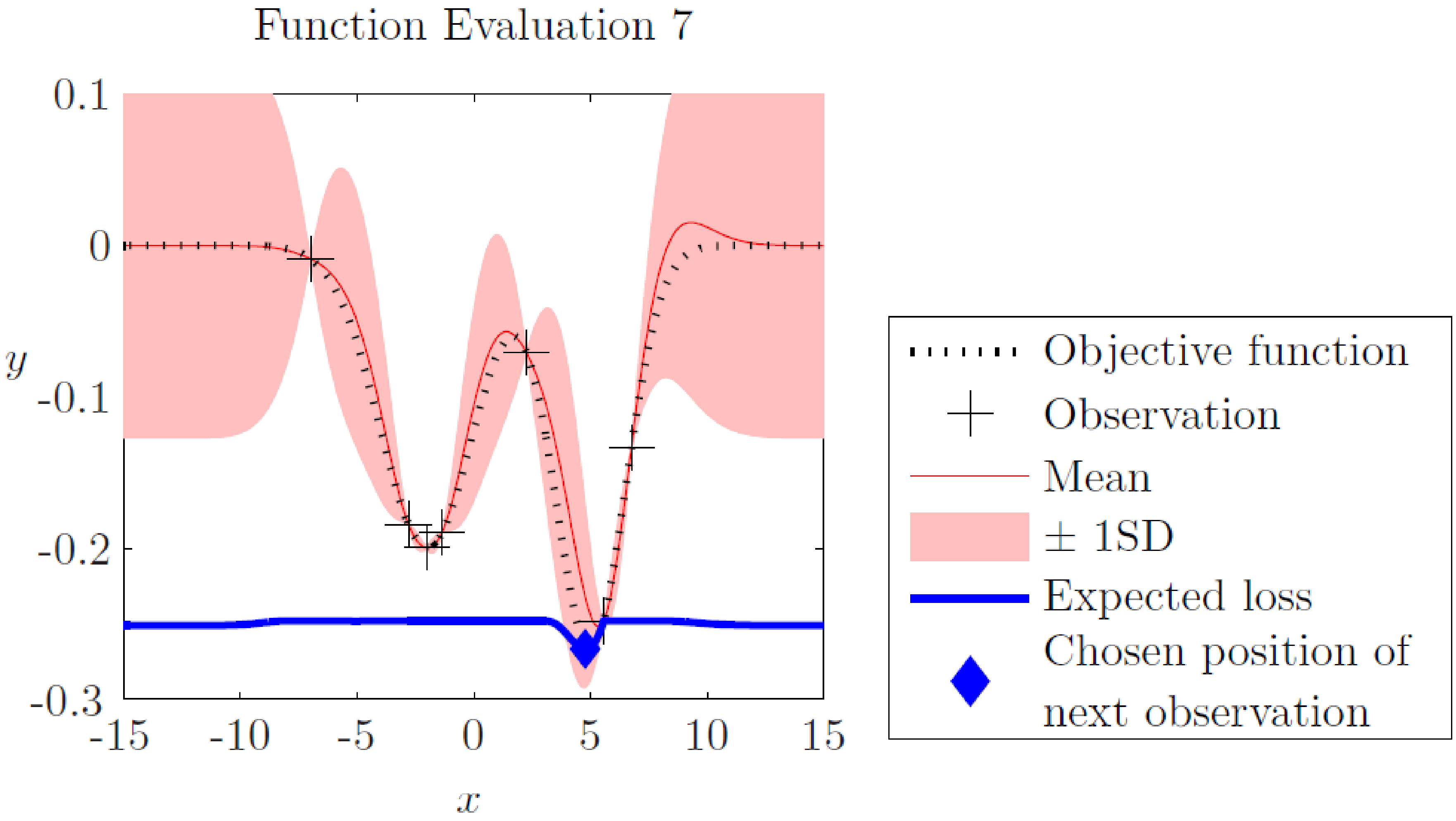
We can use Gaussian processes for optimisation.



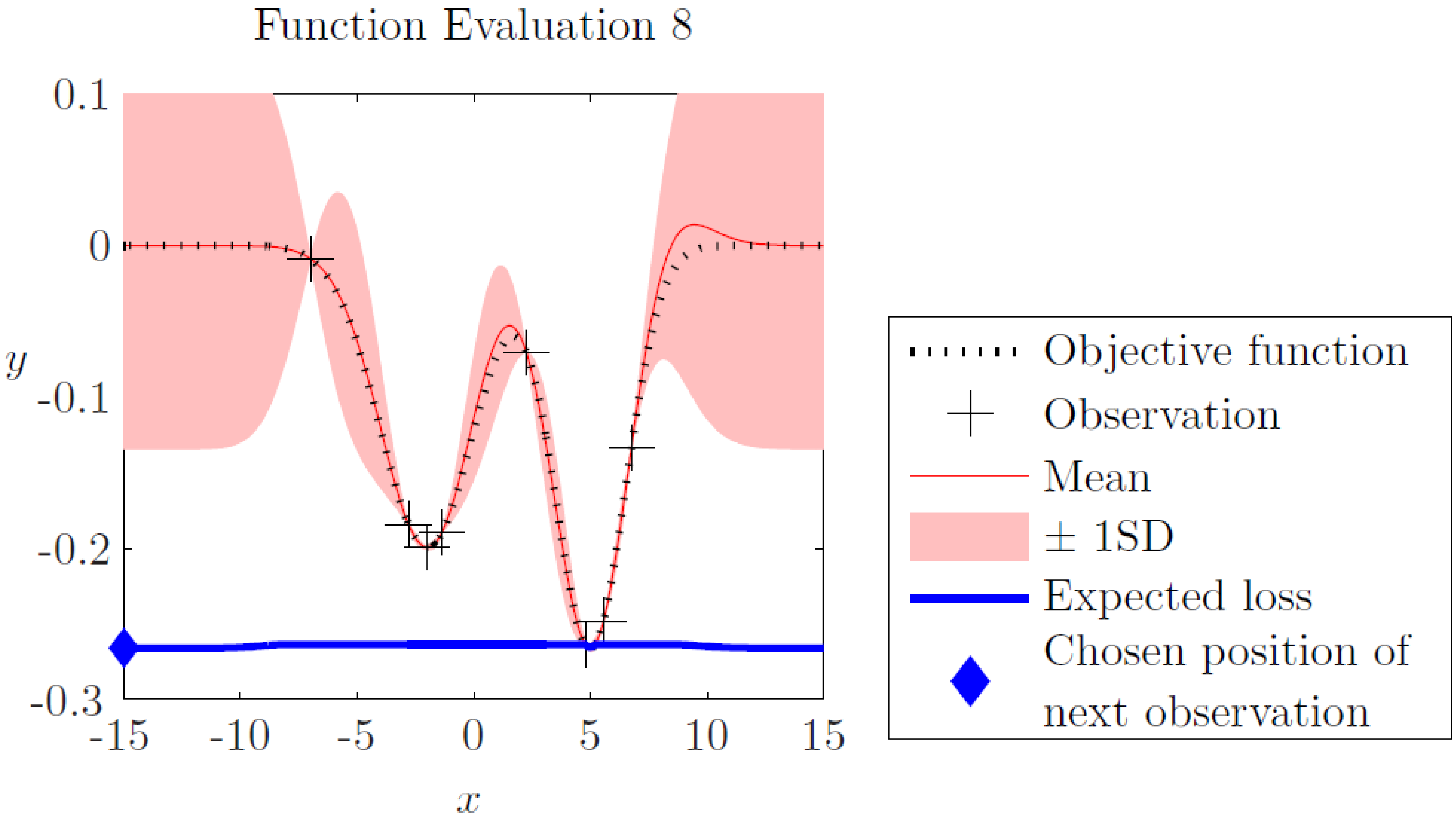
We can use Gaussian processes for optimisation.



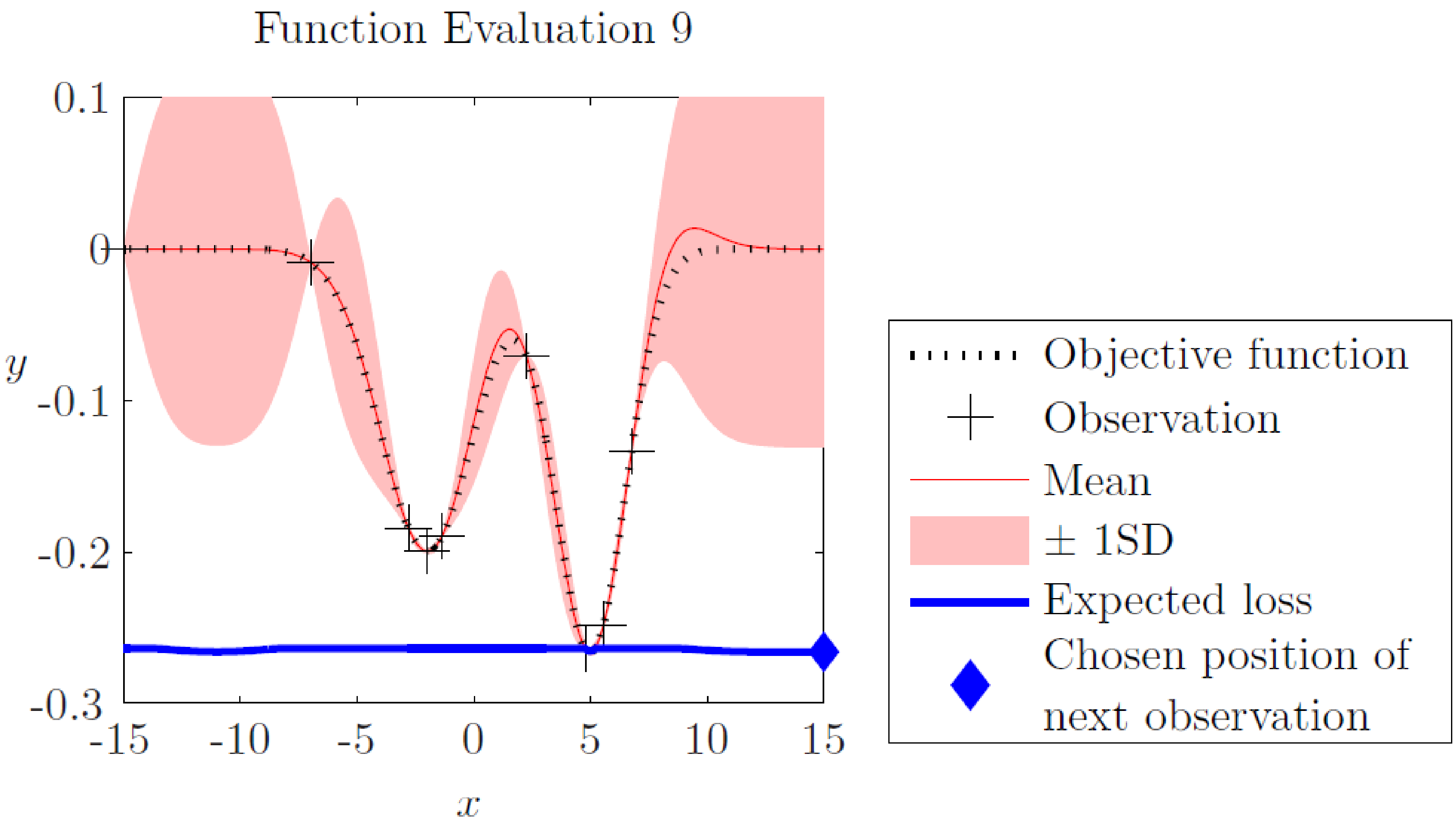
We can use Gaussian processes for optimisation.



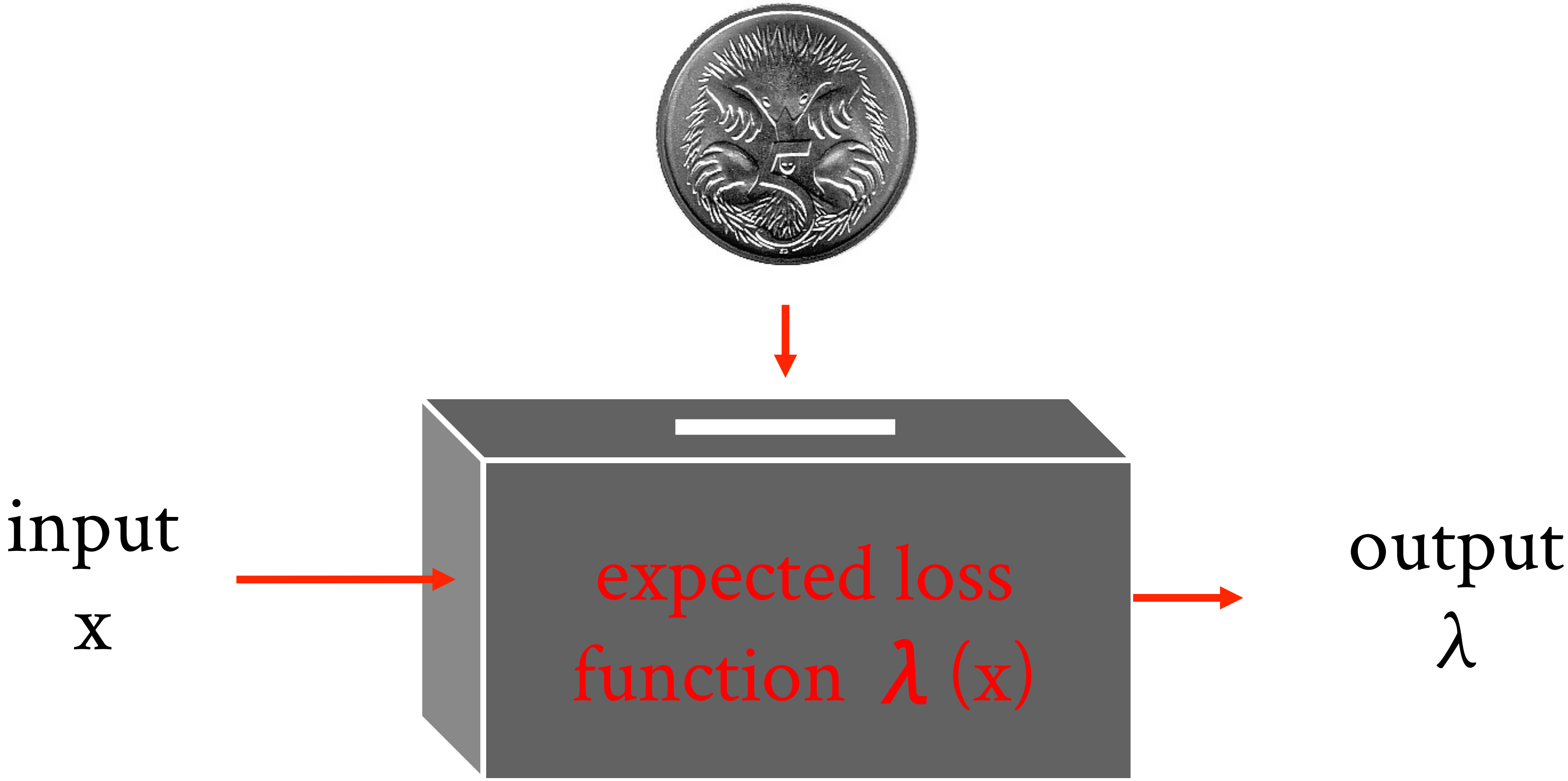
We can use Gaussian processes for optimisation.



We can use Gaussian processes for optimisation.



We have shifted an optimisation problem over an objective to an optimisation problem over an expected loss: the Gaussian process has served as a surrogate (or emulator) for the objective.



input

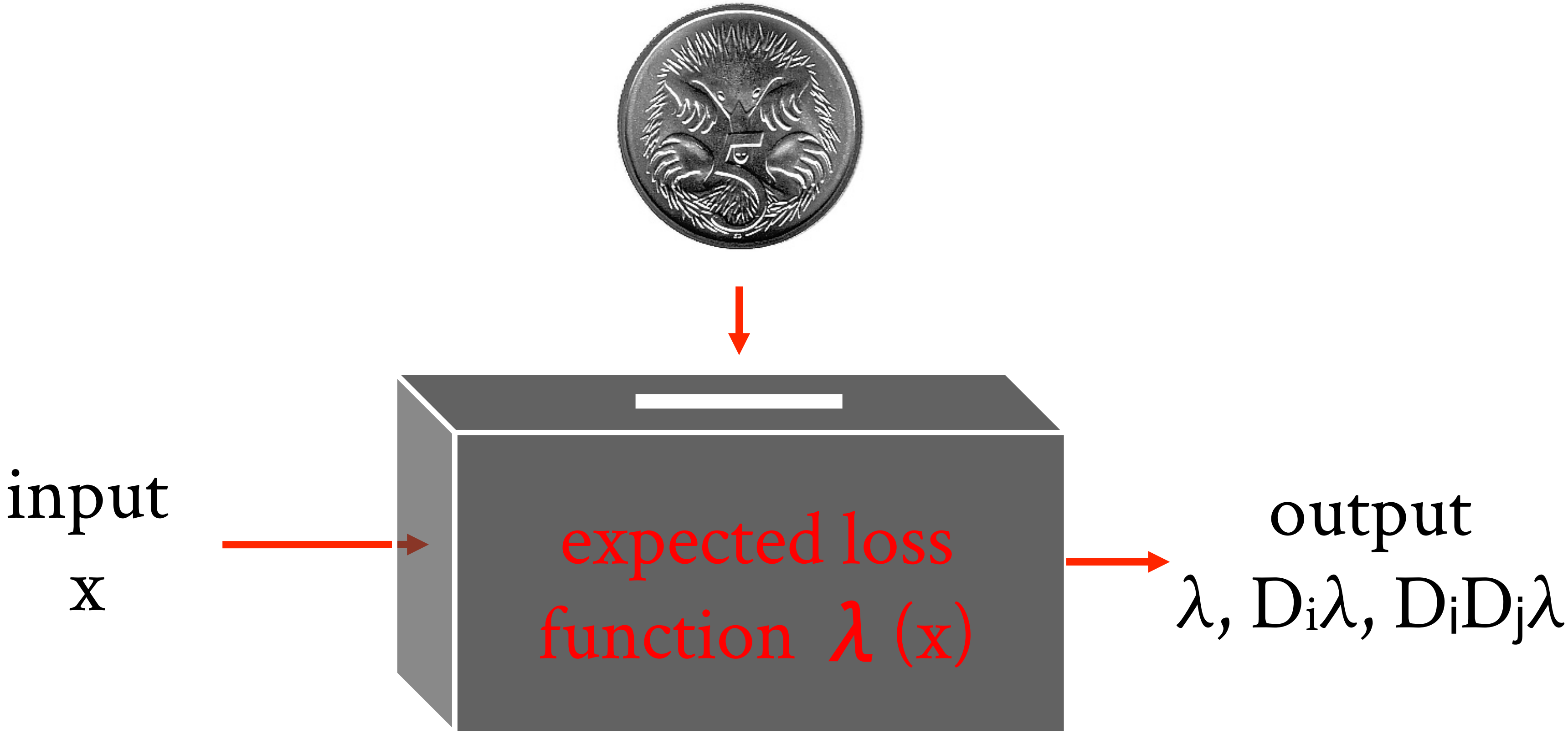
x

expected loss function $\lambda(x)$

output

λ

This is useful because the expected loss is cheaper to evaluate than the objective, and admits gradient and Hessian observations to aid the optimisation.



input

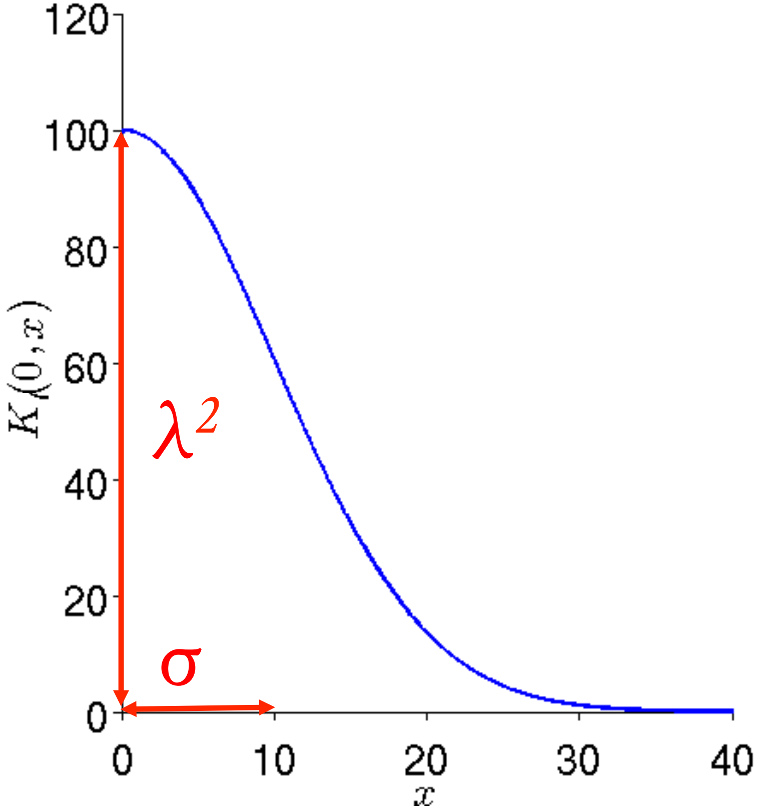
x

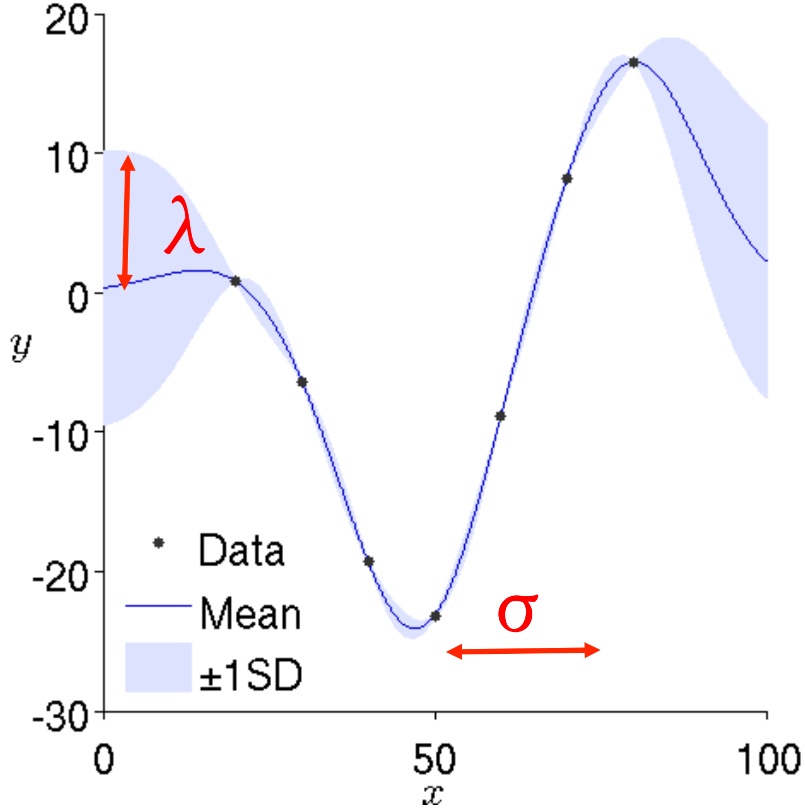
expected loss function $\lambda(x)$

output

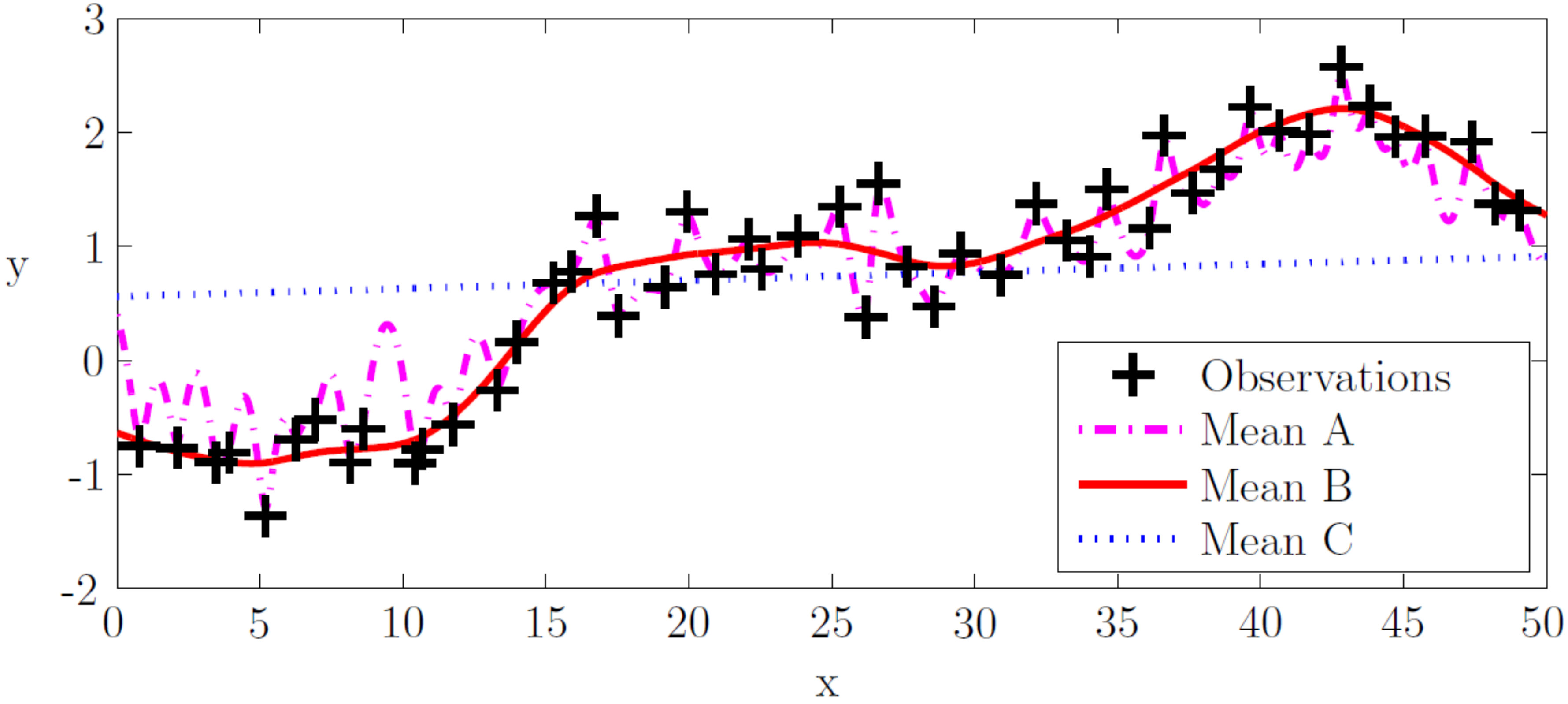
$\lambda, D_i\lambda, D_iD_j\lambda$

Our Gaussian process is specified by hyper-parameters λ and σ, giving expected length scales of the function in output and input spaces respectively.

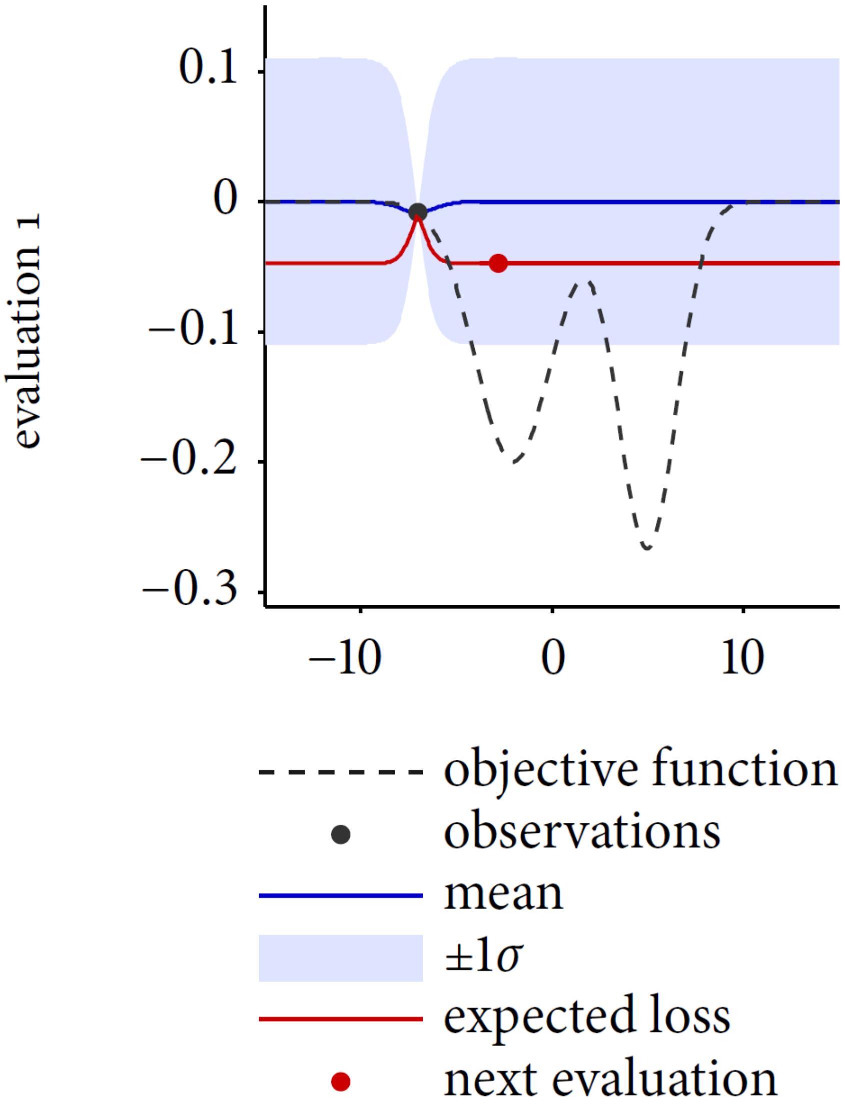


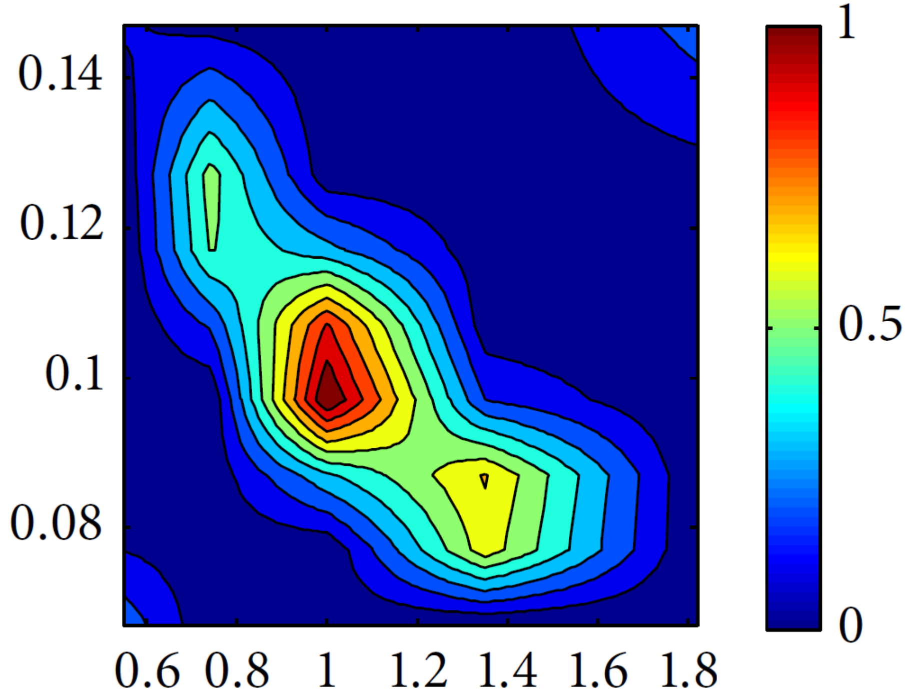


Hyperparameter values can have a significant influence: below, A has small input scale and prior C has large input scale. In optimisation, amongst other things, hyperparameters specify a ‘step size’.



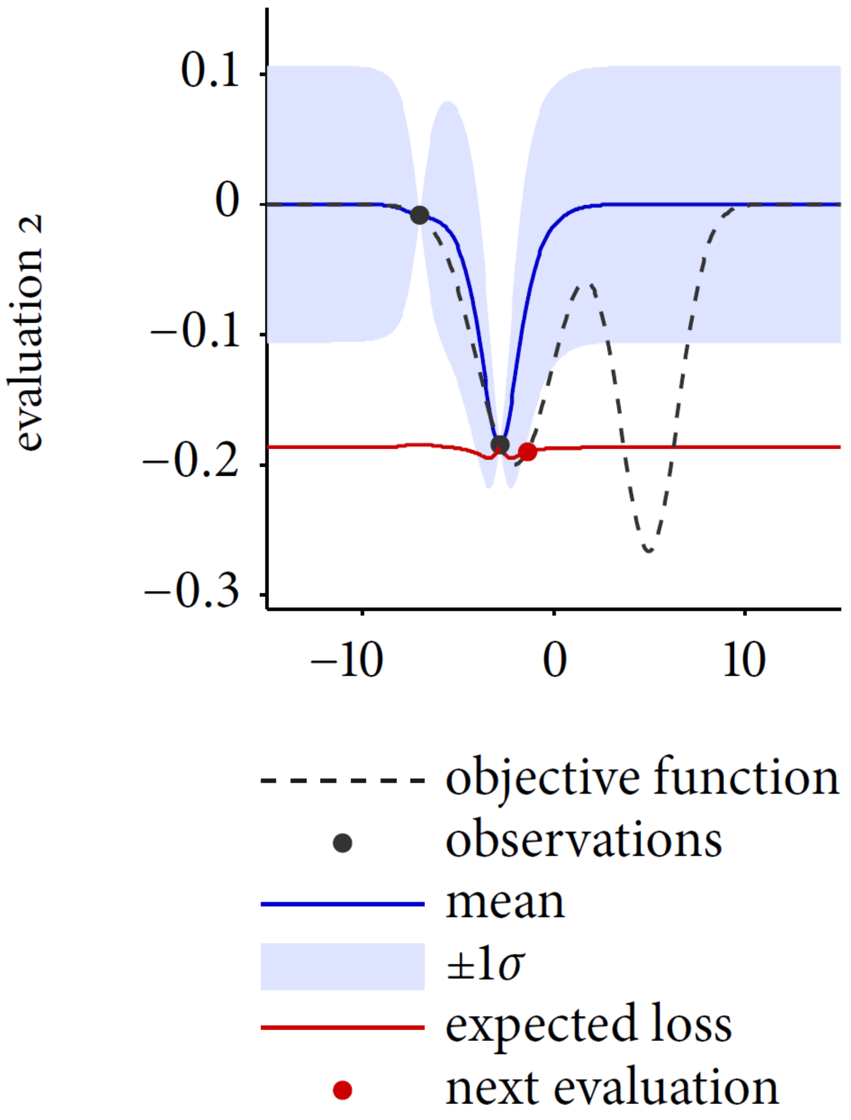
Management of hyperparameters is important for optimisation: we start with no data!

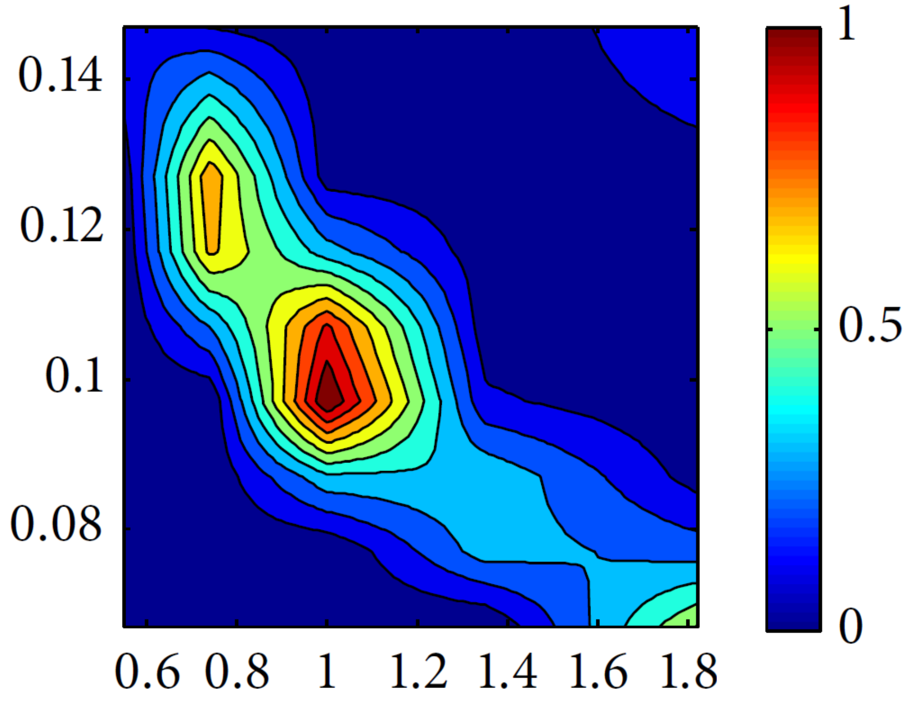




horizontal axis: σ
vertical axis: λ

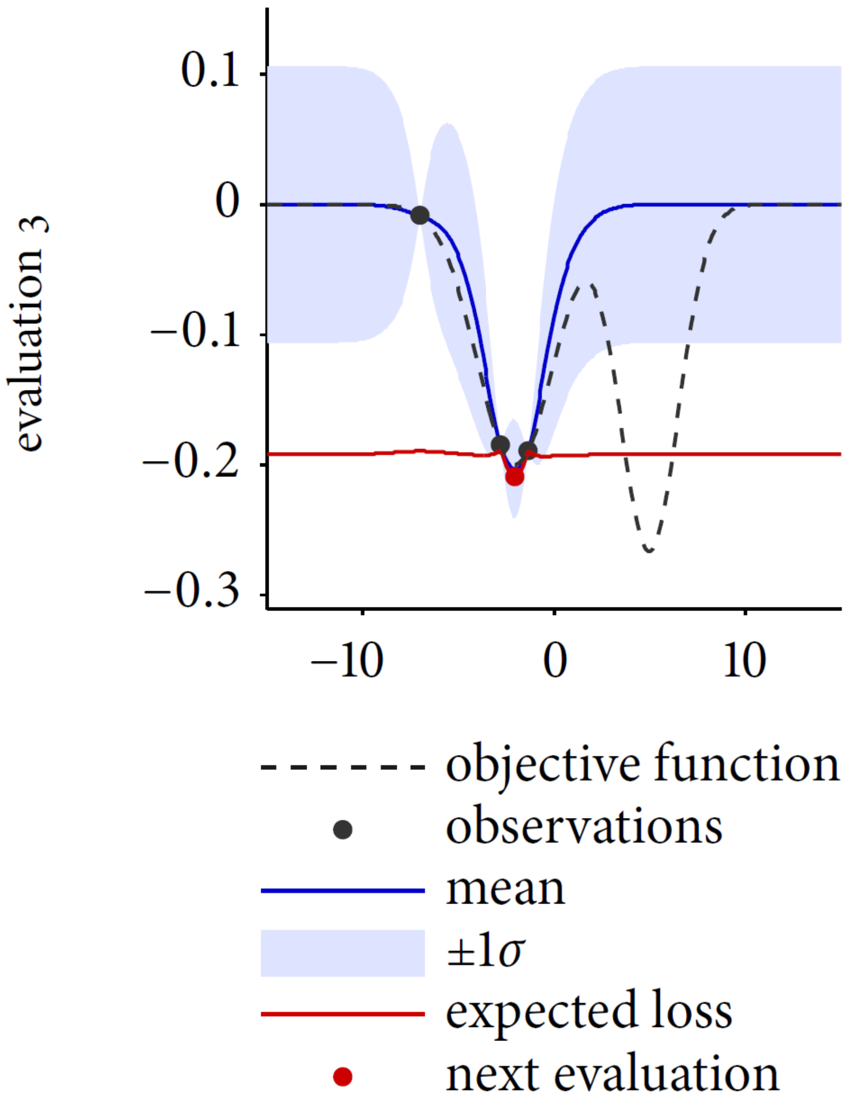
Management of hyperparameters is important.

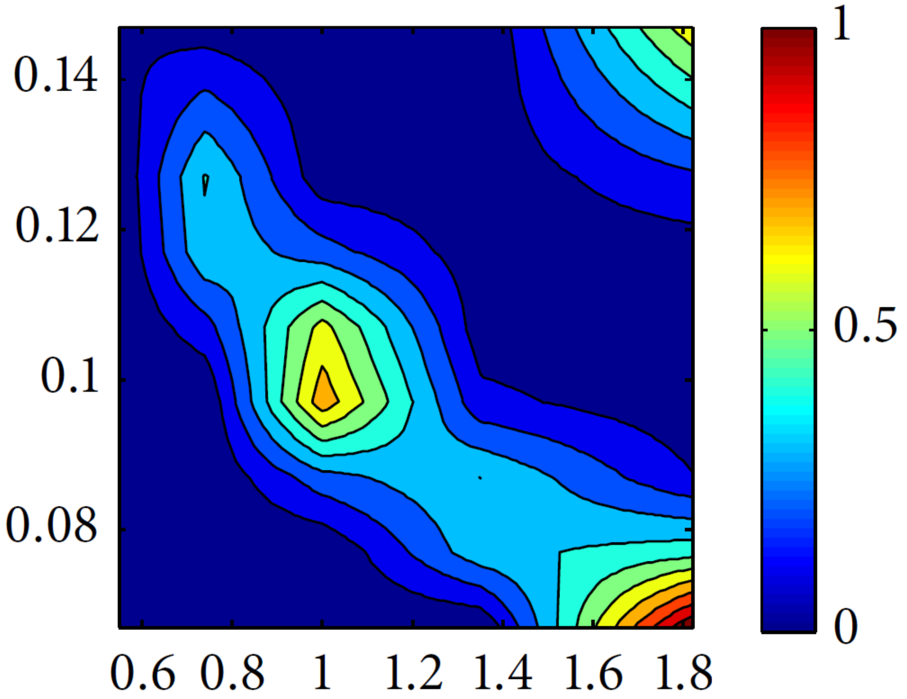




horizontal axis: σ
vertical axis: λ

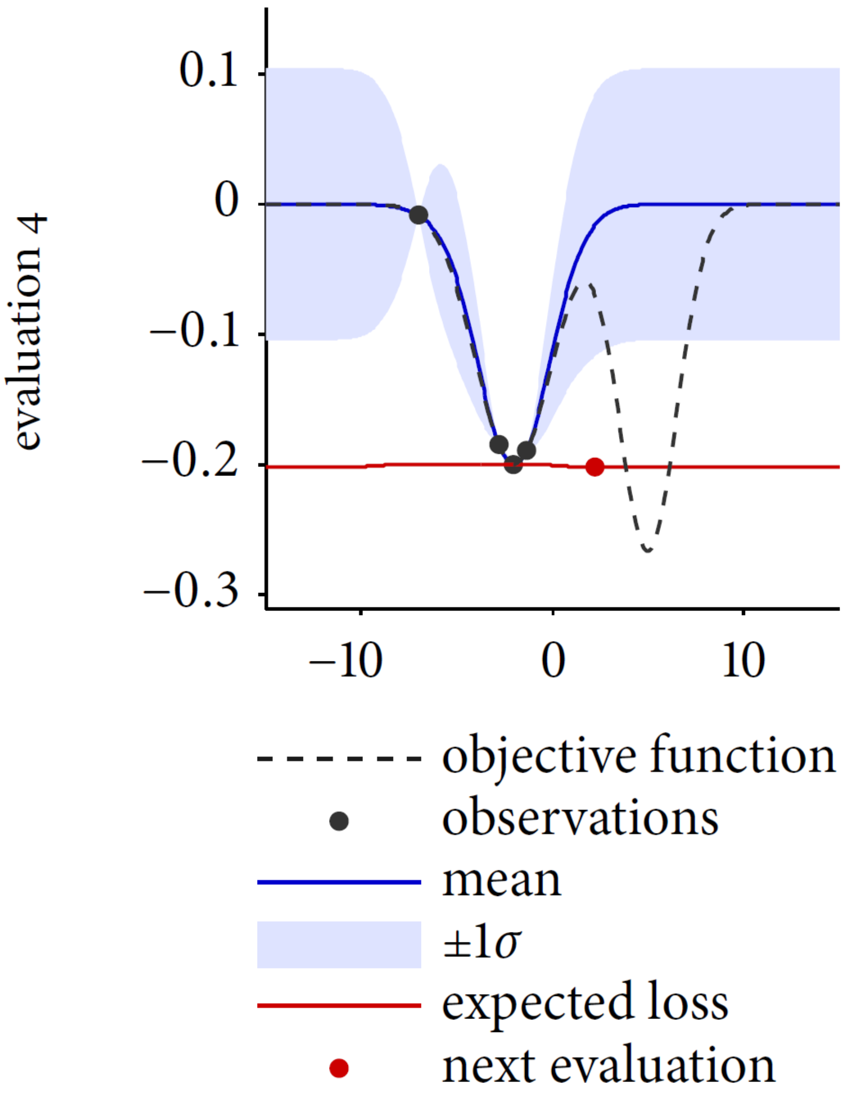
Management of hyperparameters is important.

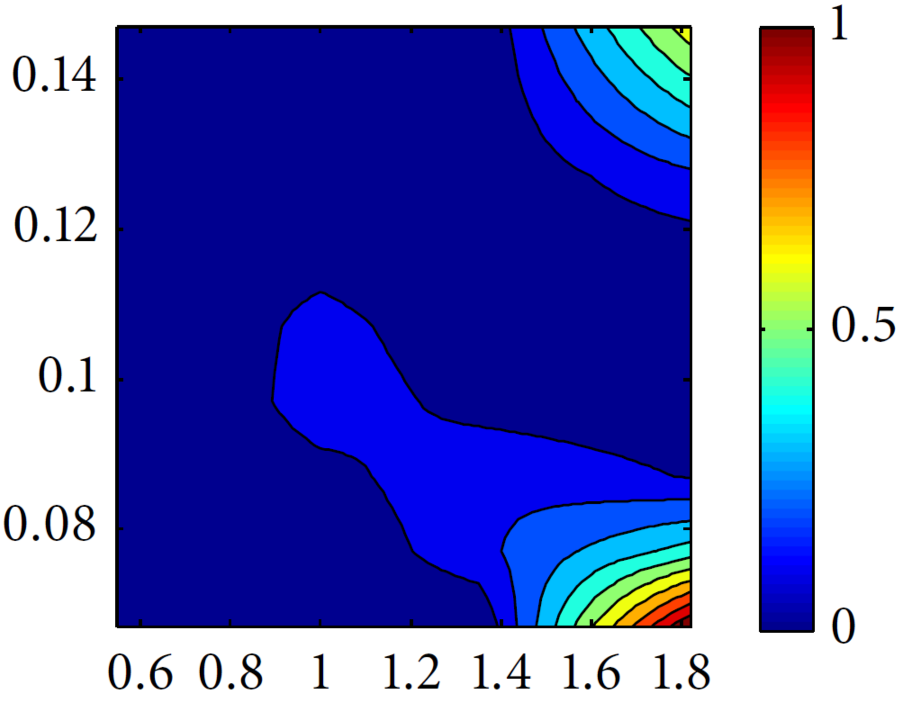




horizontal axis: σ
vertical axis: λ

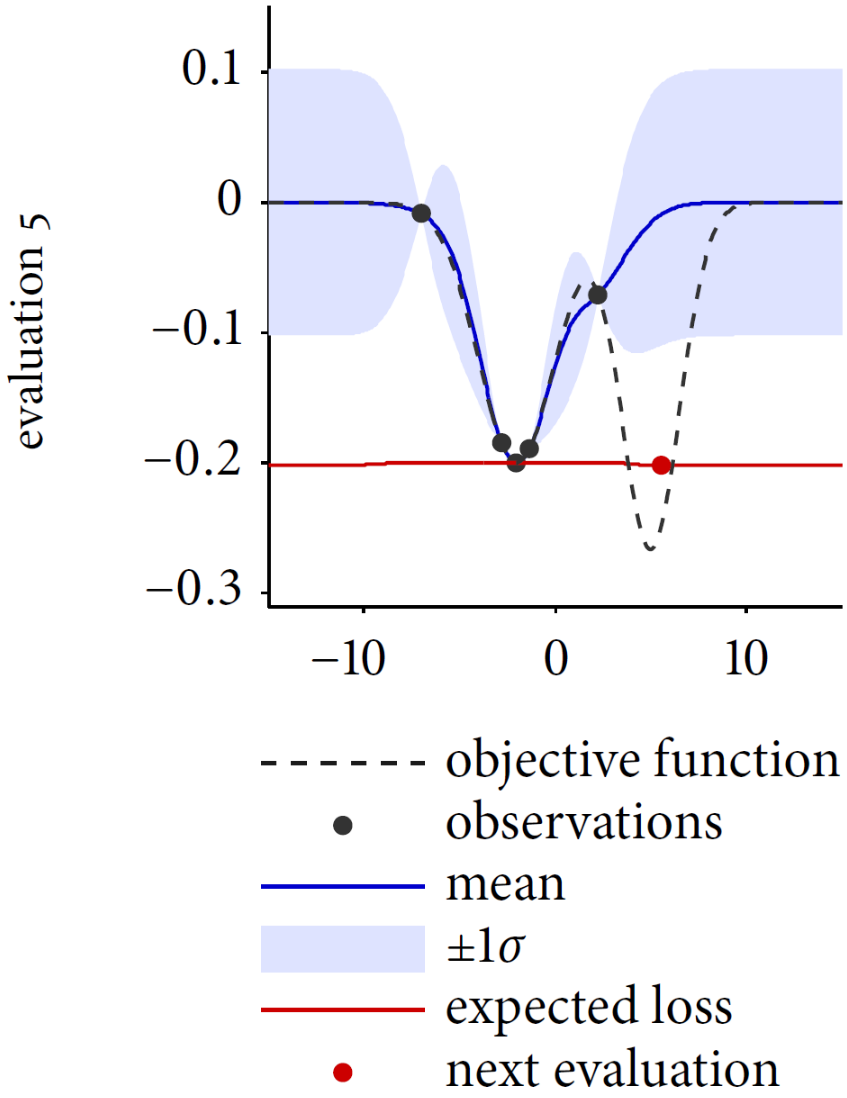
Management of hyperparameters is important.

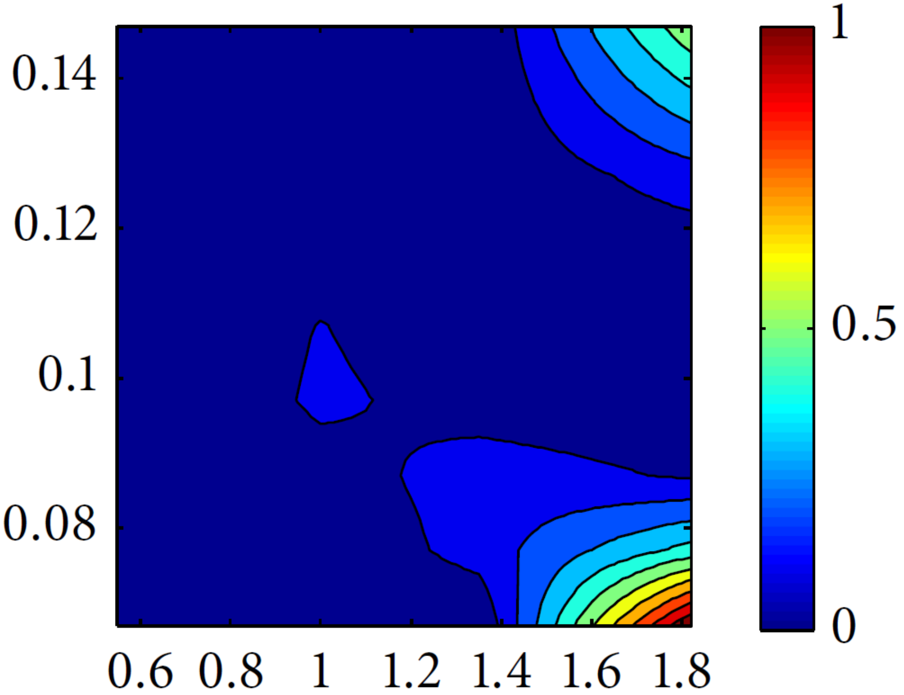




horizontal axis: σ
vertical axis: λ

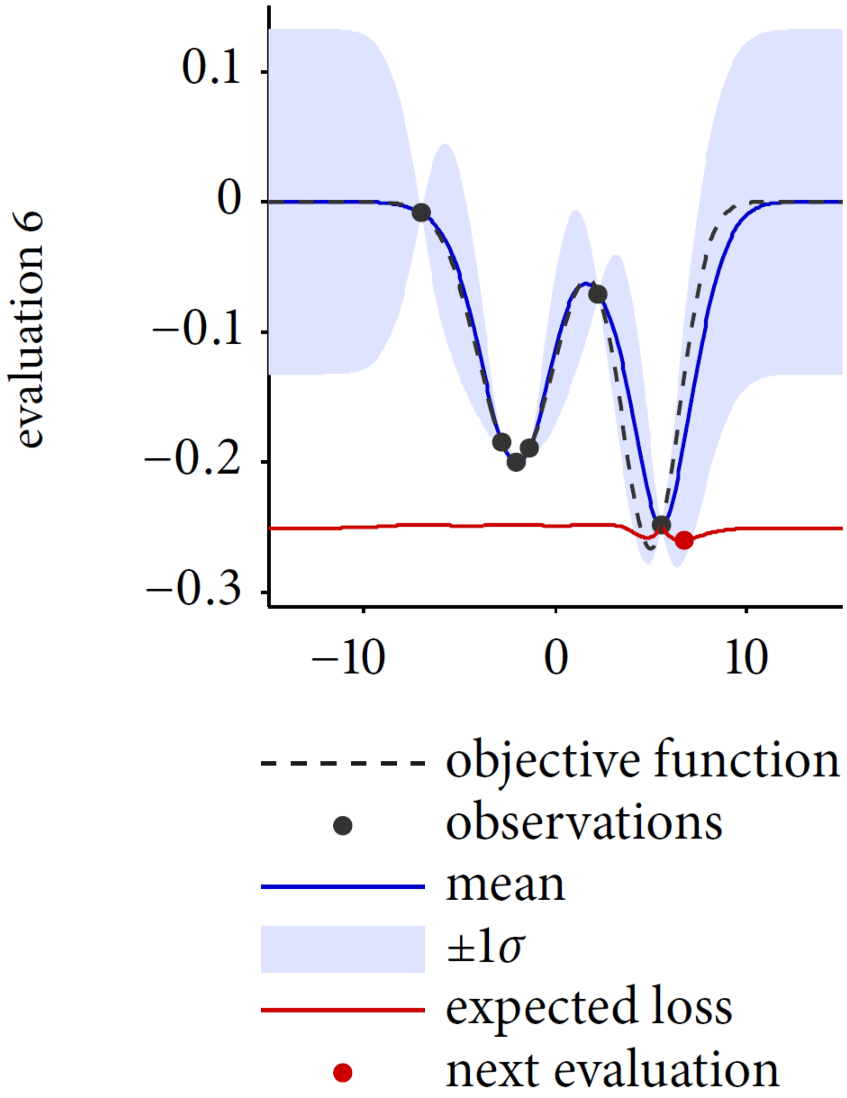
Management of hyperparameters is important.

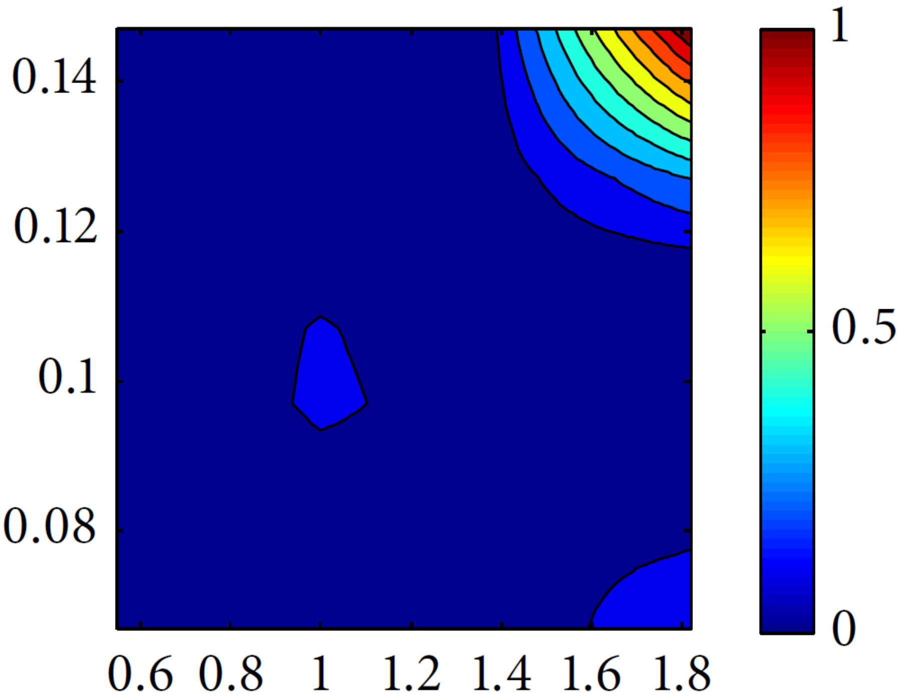




horizontal axis: σ
vertical axis: λ

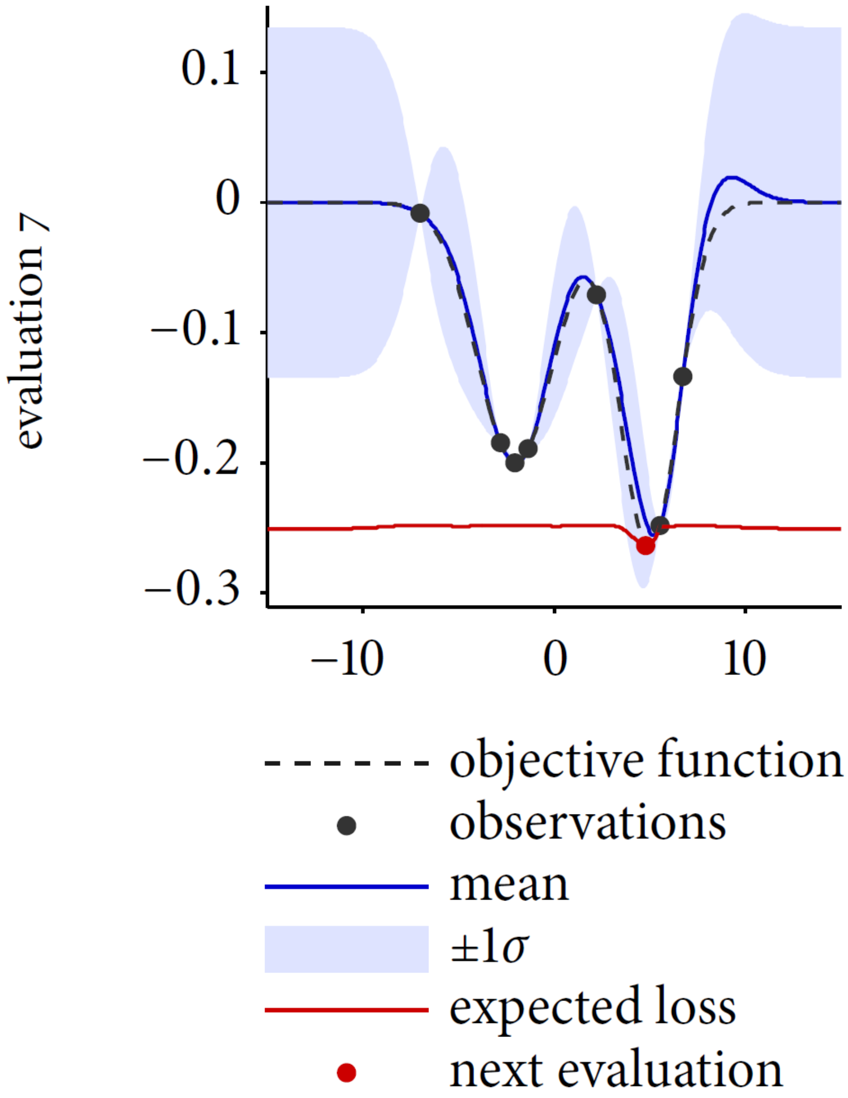
Management of hyperparameters is important.

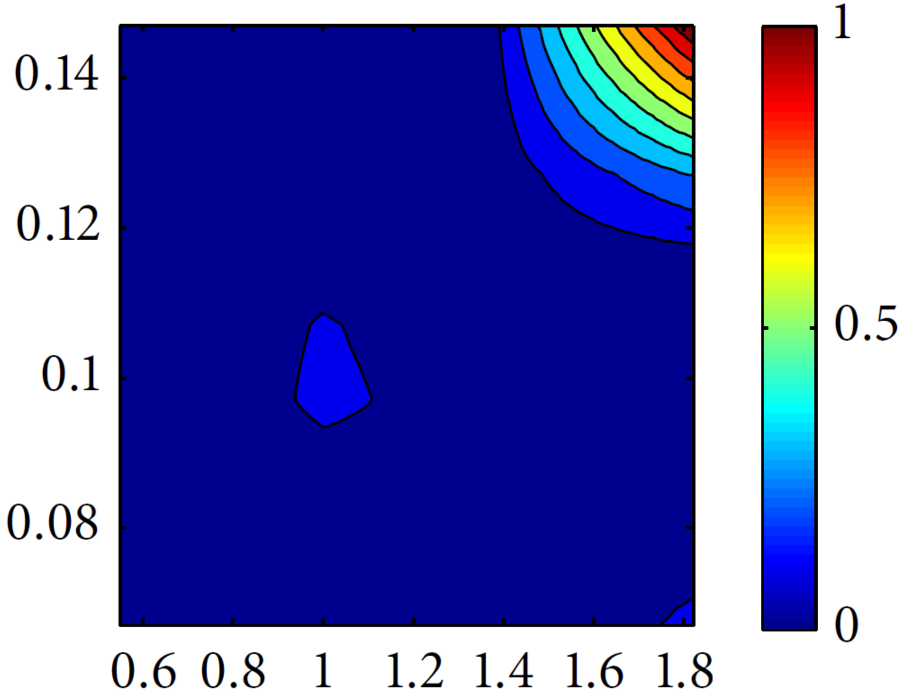




horizontal axis: σ
vertical axis: λ

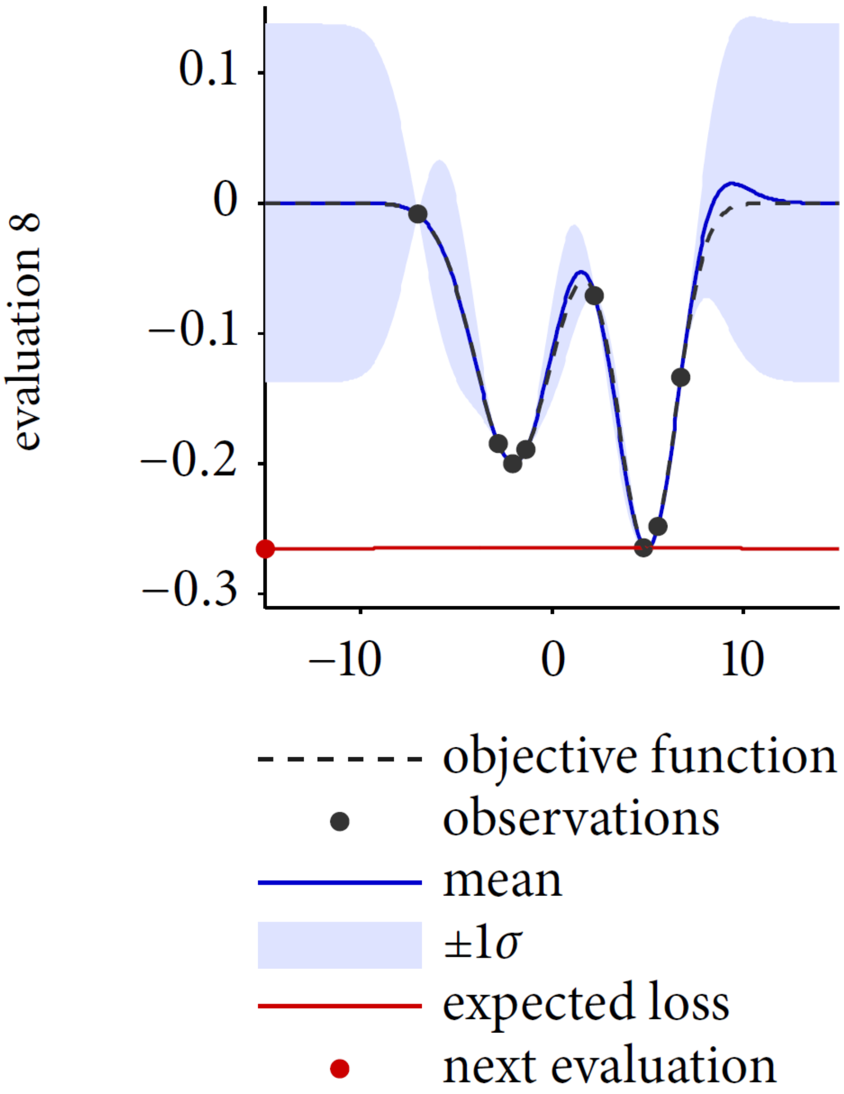
Management of hyperparameters is important.

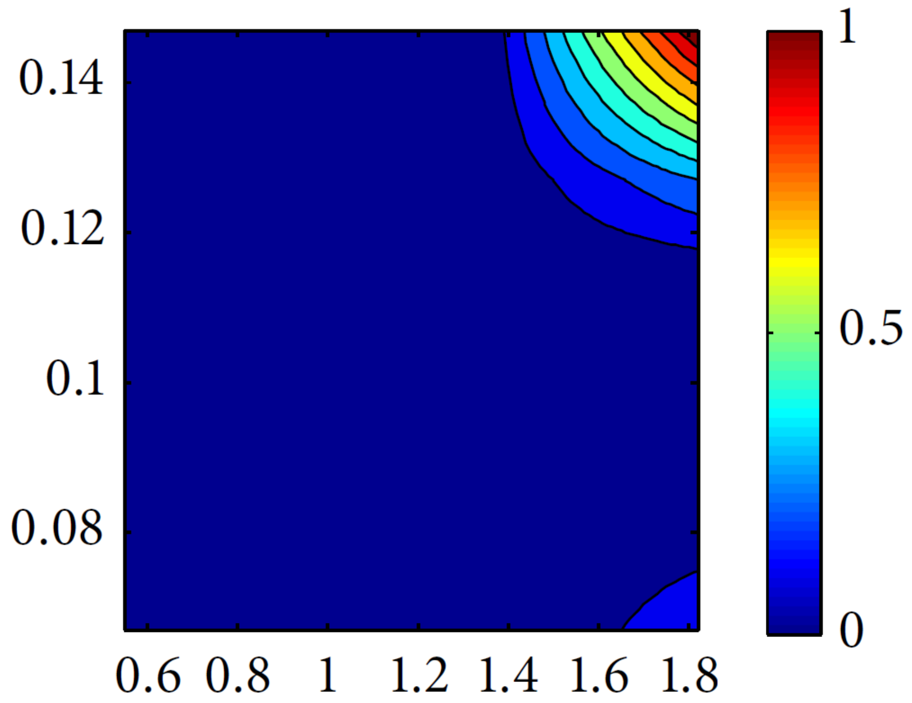




horizontal axis: σ
vertical axis: λ

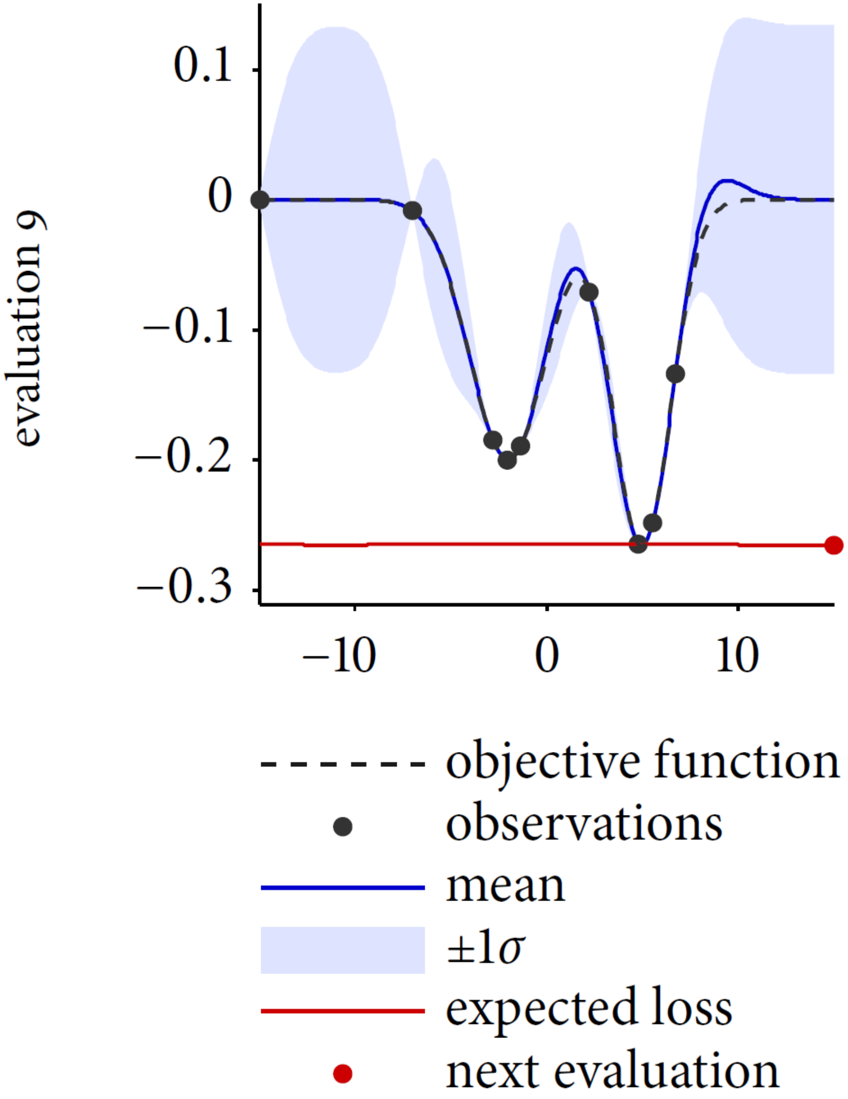
Management of hyperparameters is important.

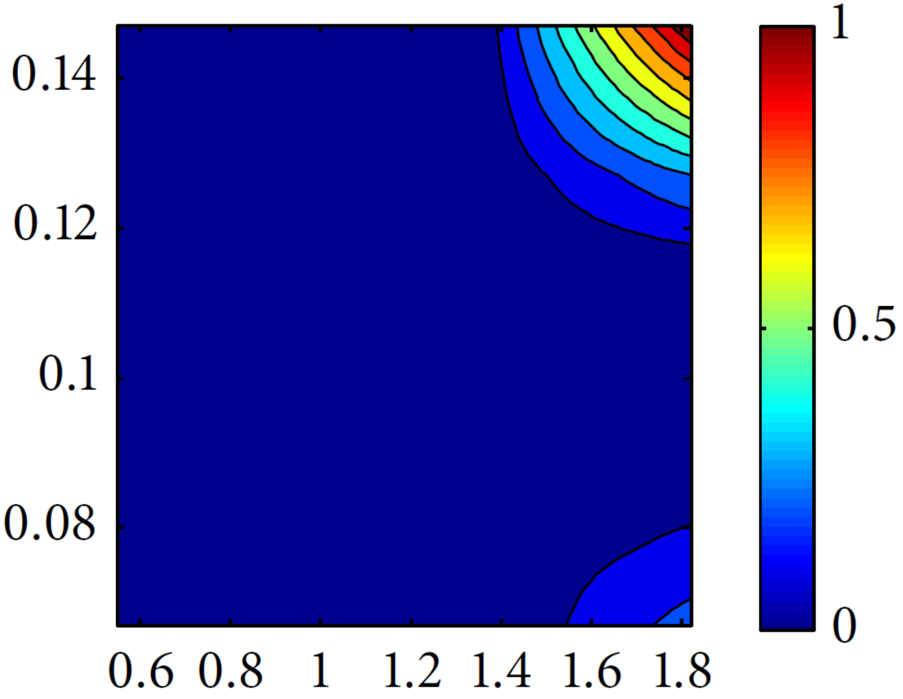




horizontal axis: σ
vertical axis: λ

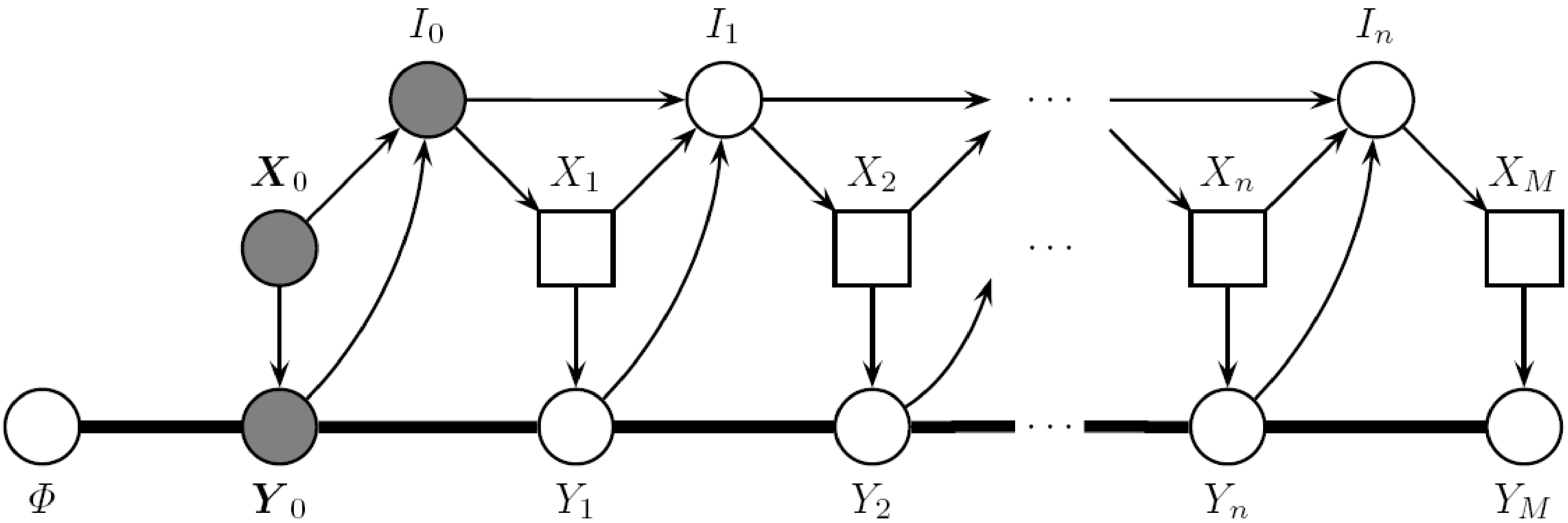
Management of hyperparameters is important (we'll come back to this!).





horizontal axis: σ
vertical axis: λ

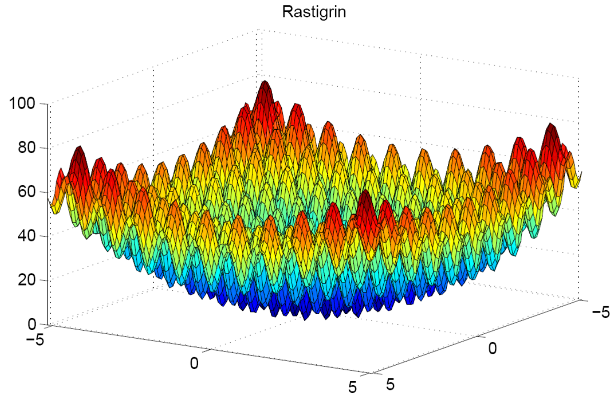
We can improve upon our myopic approximation by considering multiple function evaluations into the future using (expensive) sampling.

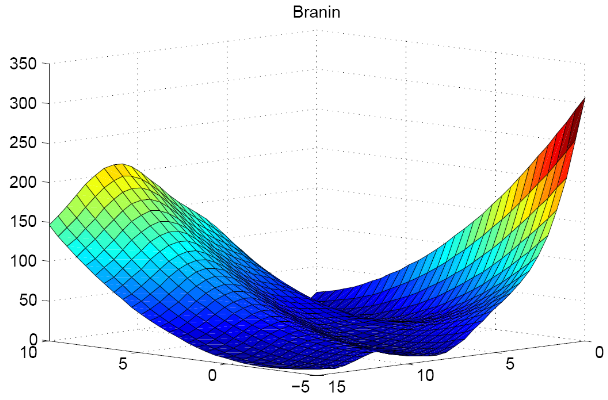


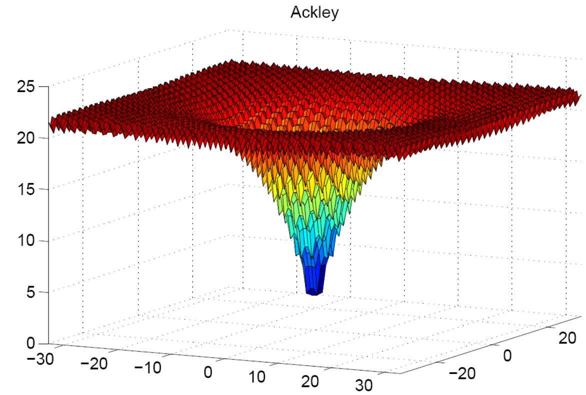
- I_n is our total information up to index n
- M represents the index of our final decision
- All nodes along the dark line are correlated

We tested on a range of problems.

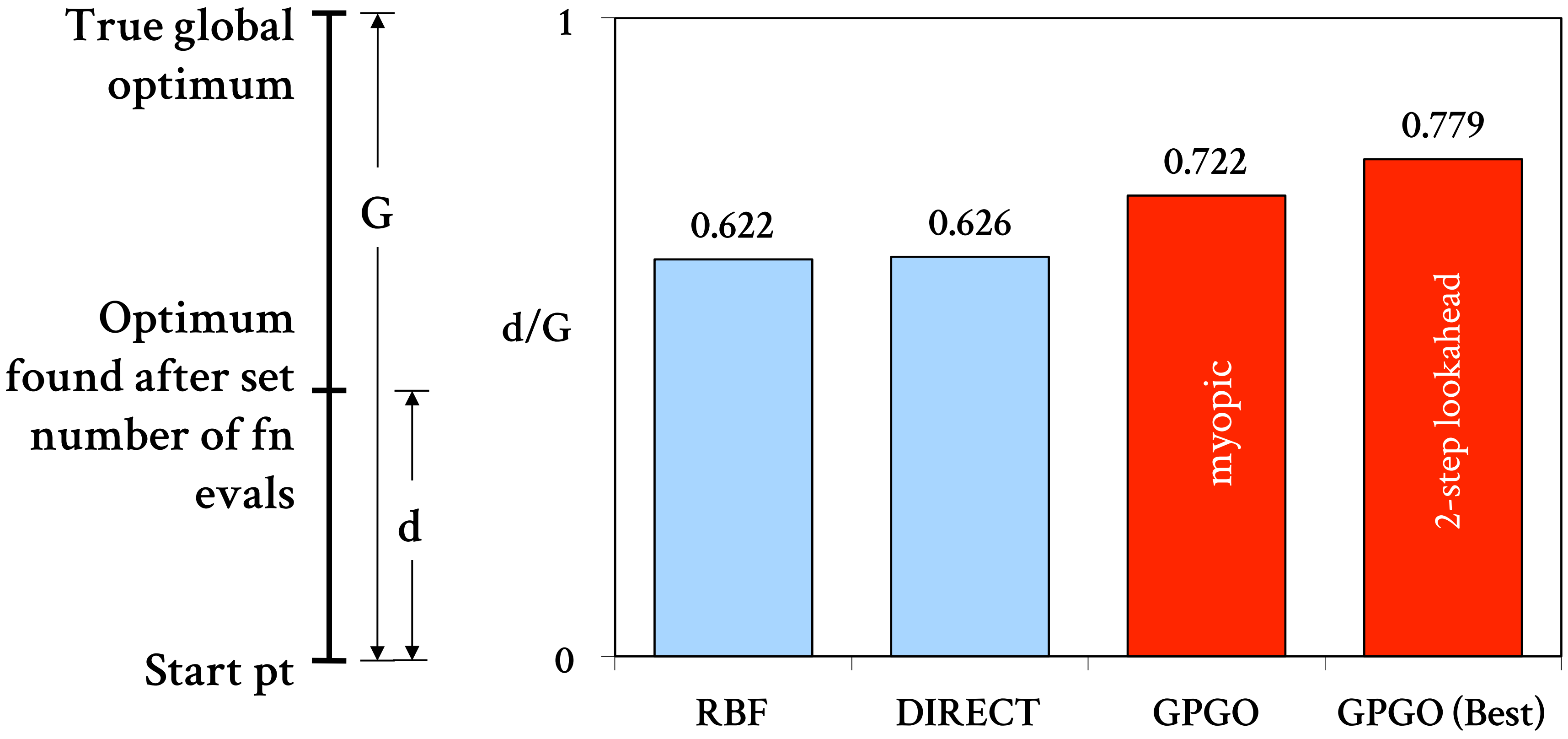
- Ackley 2/5
- Branin
- 6-hump Camelback
- Goldstein-Price
- Griewank 2/5
- Hartman 3/6
- Rastigrin
- Shekel 5/7/10
- Shubert



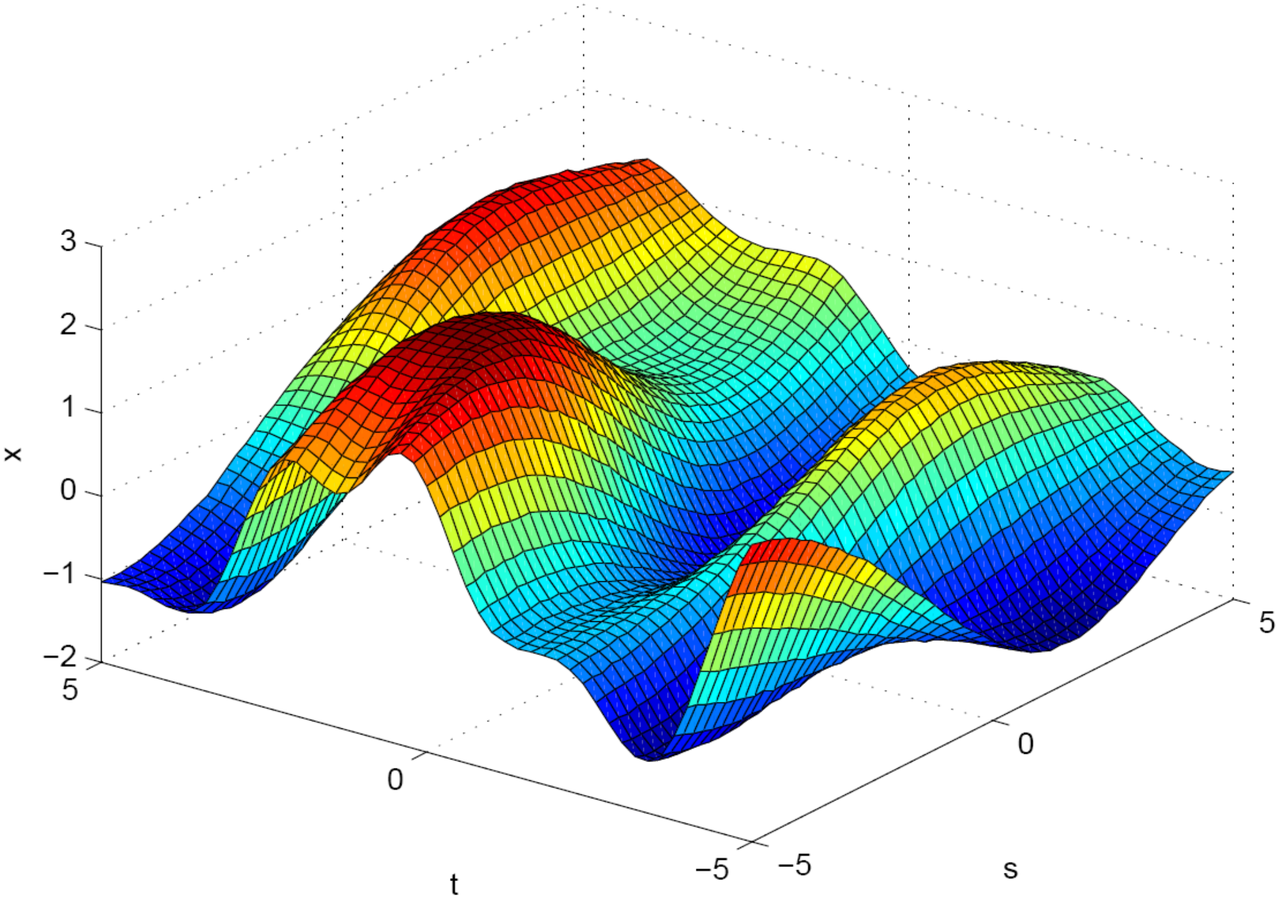




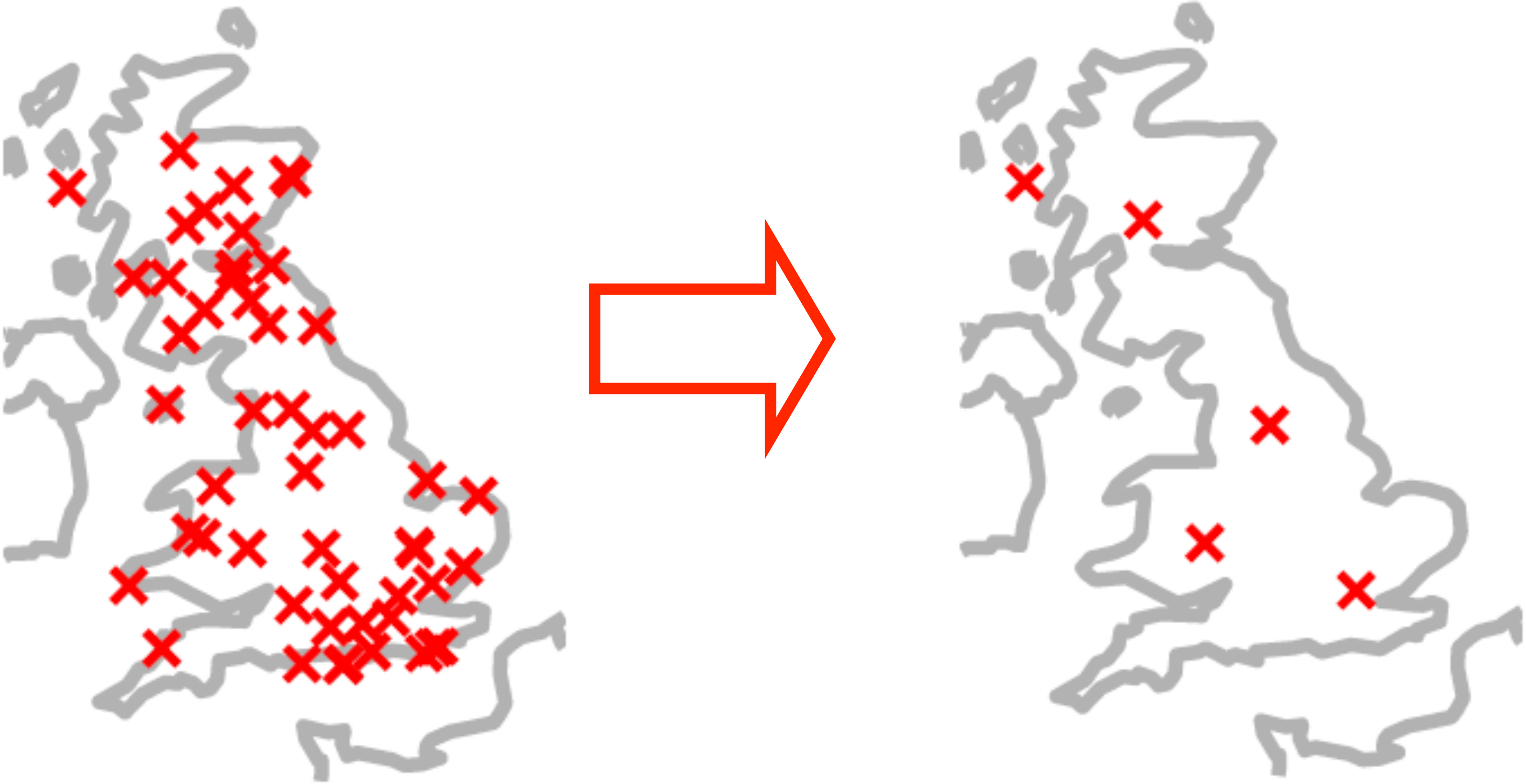
Gaussian process global optimisation (GPGO) outperforms tested alternatives for global optimisation over 140 test problems.



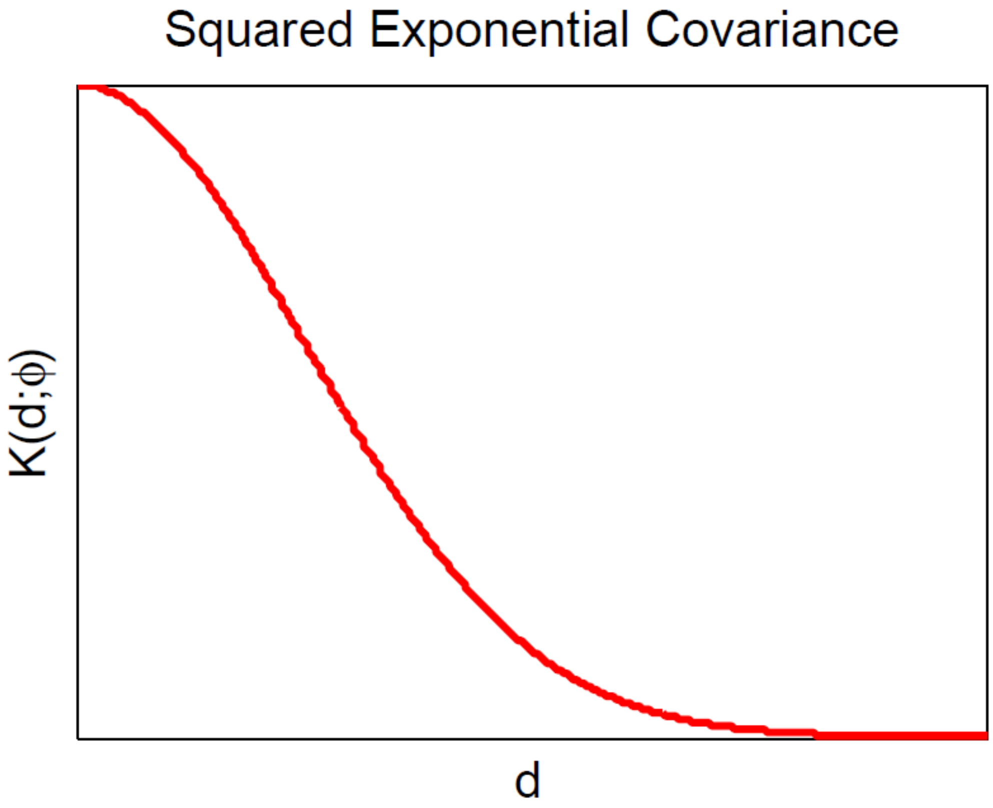
We can extend GPGO to optimise dynamic functions, by simply taking time as an additional input (that we have no control over).



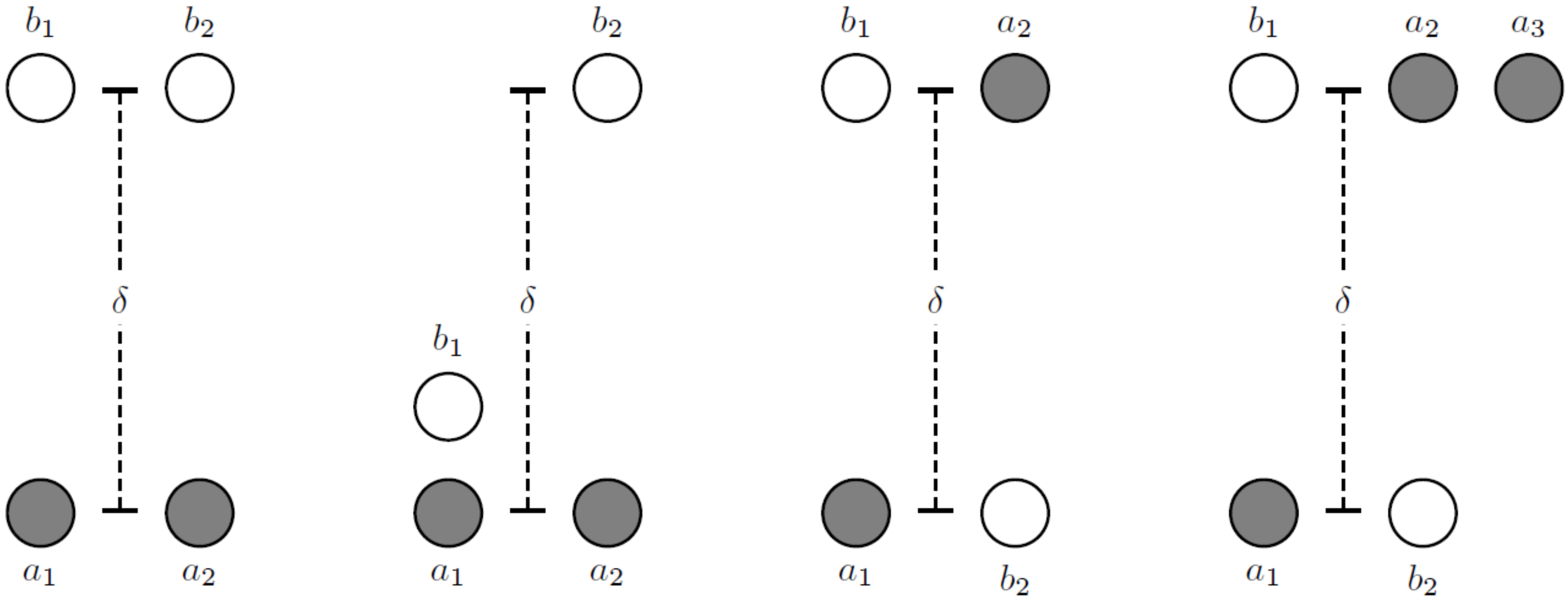
We can use GPGO to select the optimal subset of sensor locations for weather prediction.



Consider functions of sets, such as the predictive quality of a set of sensors. In order to perform inference about such functions, we build a covariance function over sets.



To build such a covariance, we need the distance d (A,B) between sets A and B. For δ sufficiently large, d(A,B) is



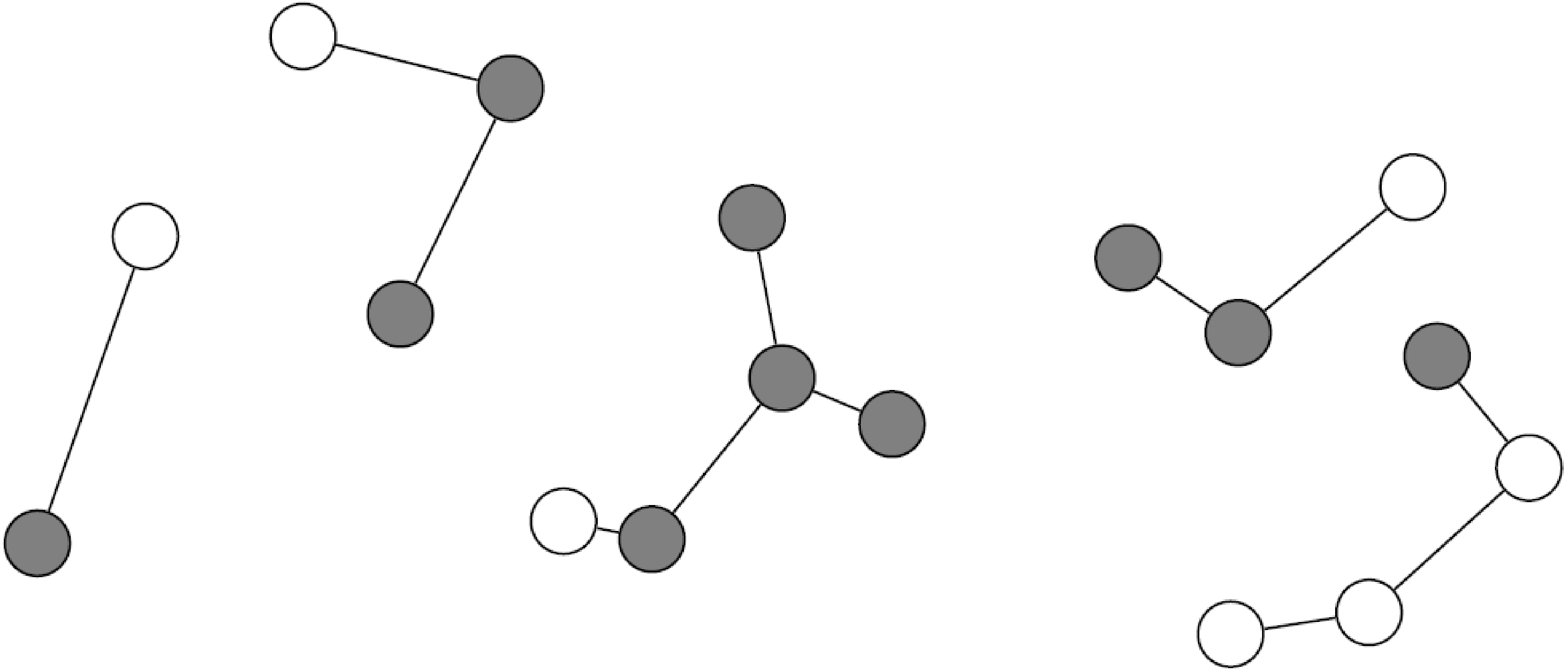
large

somewhat large

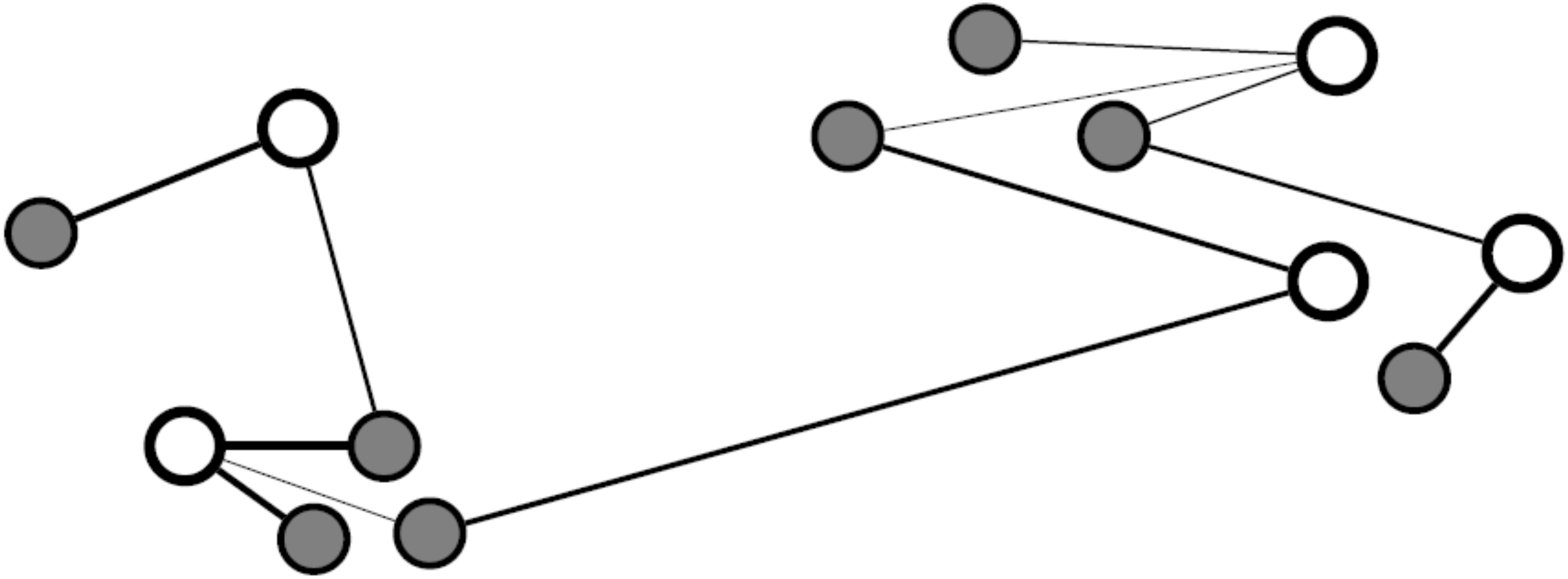
small

small

d(A,B) is the minimum total distance that nodes would have to be moved in order to render set A identical to set B, if nodes can be ‘merged’ along the way.

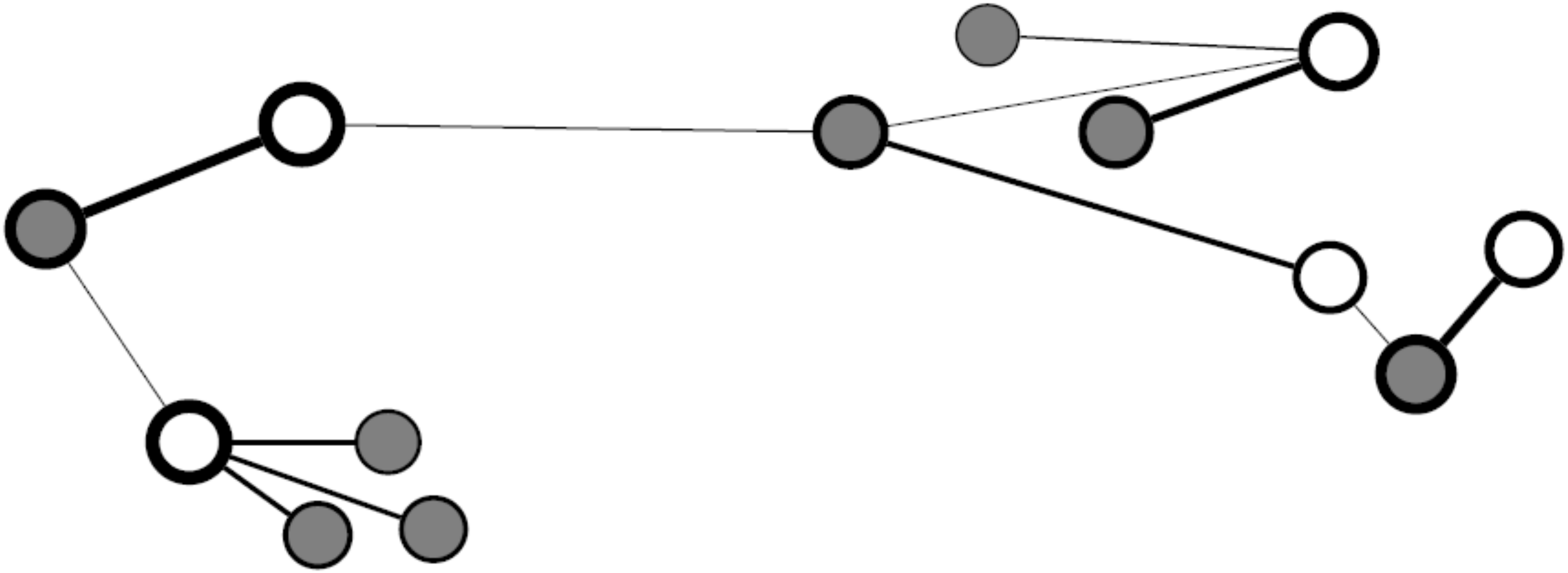


d(A,B) is the minimum total distance that nodes would have to be moved in order to render set A identical to set B (the Earth-mover's distance).



Above, we assume equally weighted nodes (weights are indicated by the thickness of the outline).

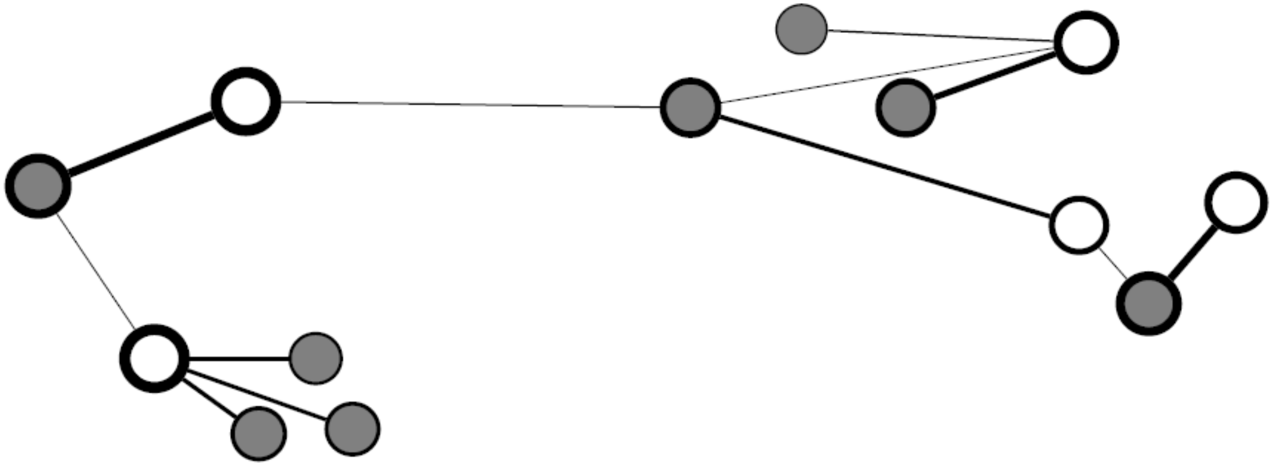
We also wish to determine weights that reflect the importance of sensors to the overall network (more isolated nodes are more important).



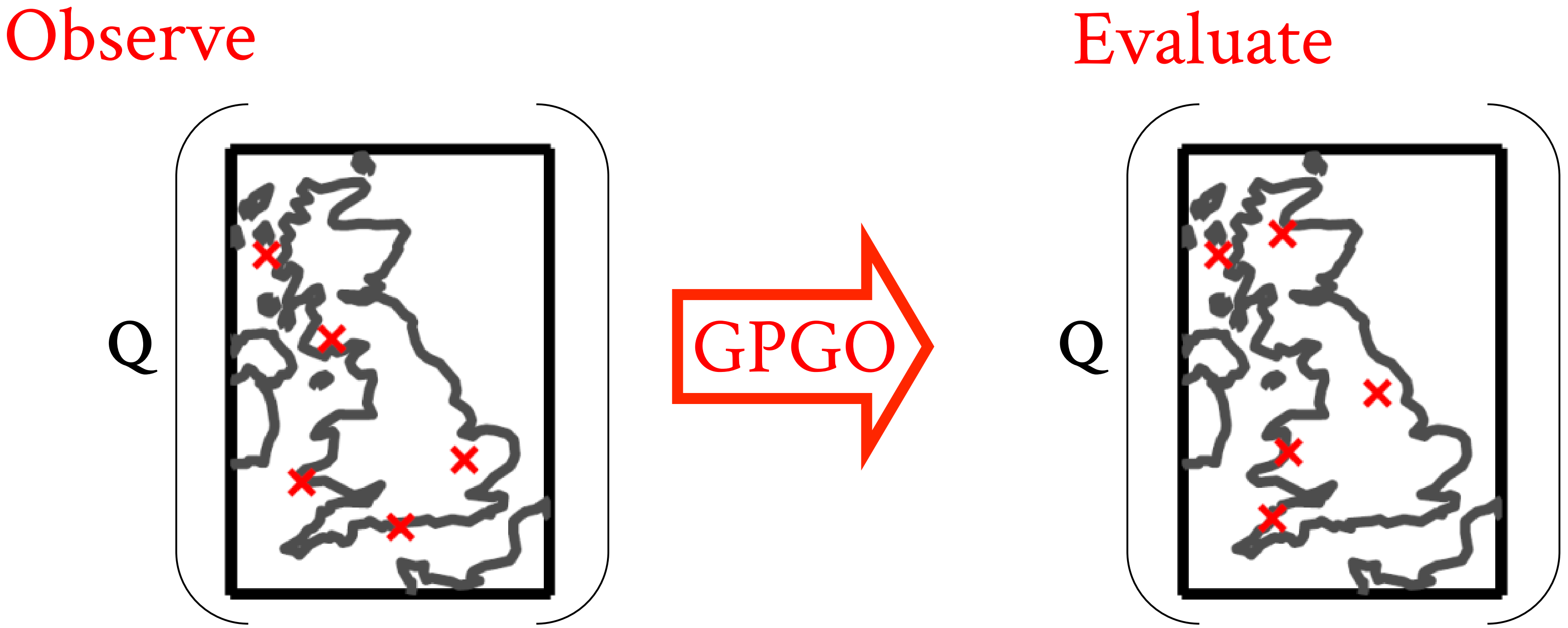
The mean prediction at a point $\boldsymbol{x}_\star$ given observations from set A is

$$\boldsymbol{\mu}(\boldsymbol{x}_\star) + \underbrace{\mathbf{K}\,(\boldsymbol{x}_\star, \boldsymbol{x}_A)\mathbf{V}\,(\boldsymbol{x}_A, \boldsymbol{x}_A)^{-1}}(\boldsymbol{z}_A - \boldsymbol{\mu}(\boldsymbol{x}_A))\,.$$

This term weights the observations from A: we use it as a measure of the importance of the nodes. Note that we integrate over $\boldsymbol{x}_\star$ and normalise.



We sequentially selected subsets of 5 sensors (from 50 around the UK) so as to optimize the quality of our predictions (Q) about air temperature from 1959 to 1984.



Observe

Evaluate

Q

GPGO

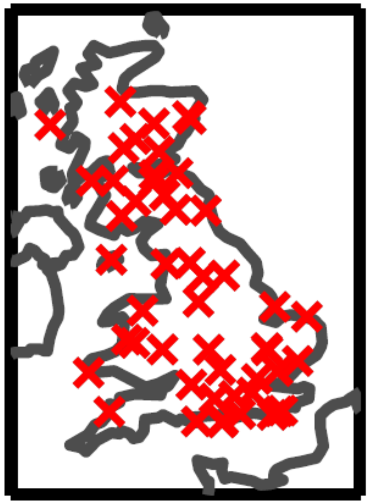
Q

A) All sensors

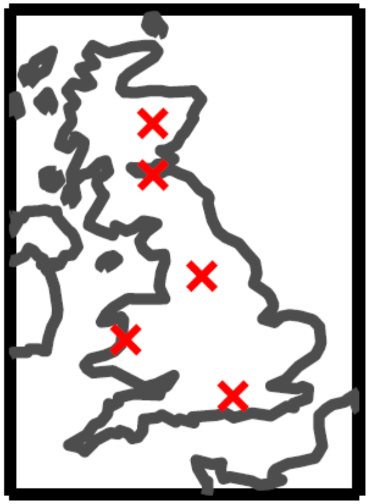
B) Sensors selected by greedy mutual information procedure (MI). MI is not adaptive to data!

C) Sensors we selected in 1969.

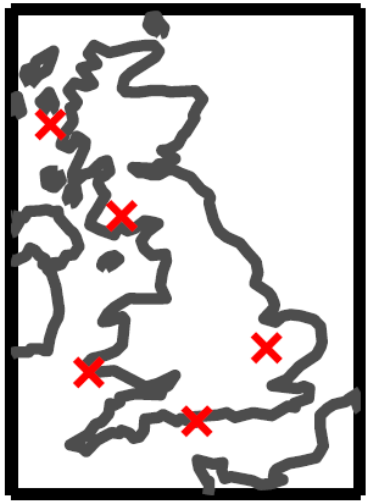
D) Sensors we selected in 1979.



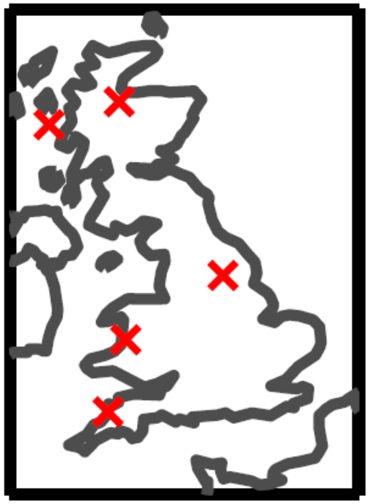
(A)



(B)

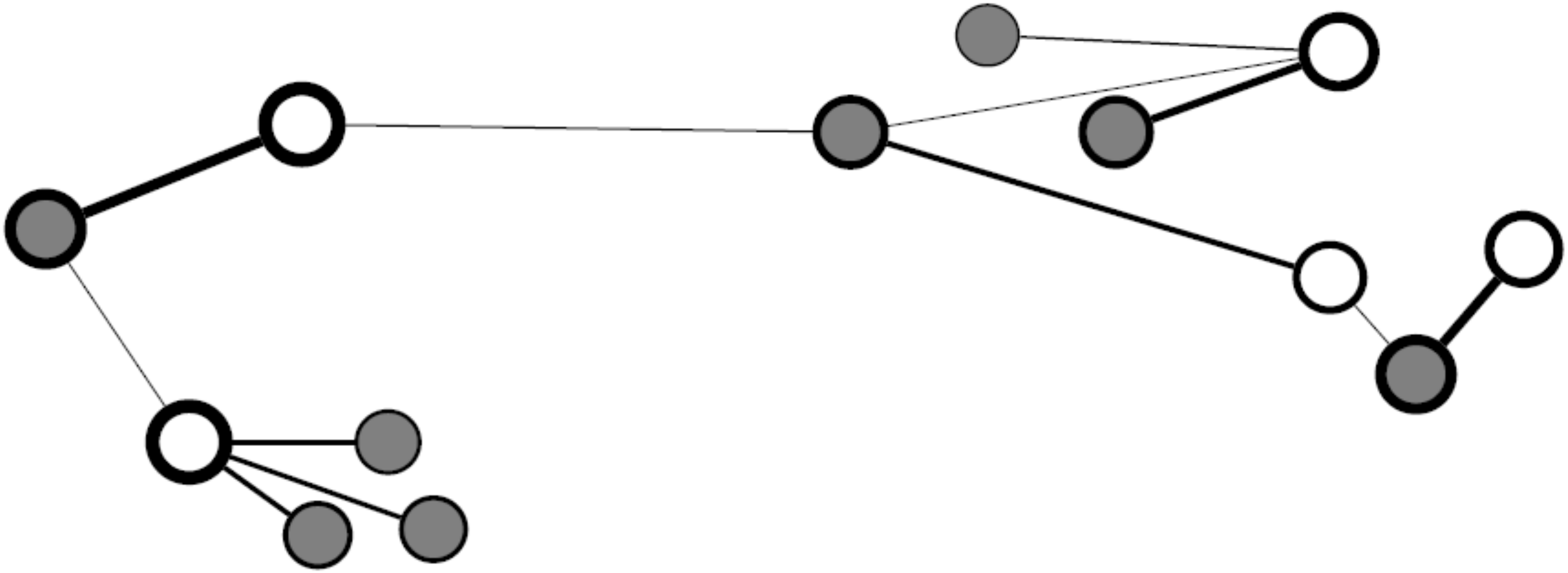


(C)

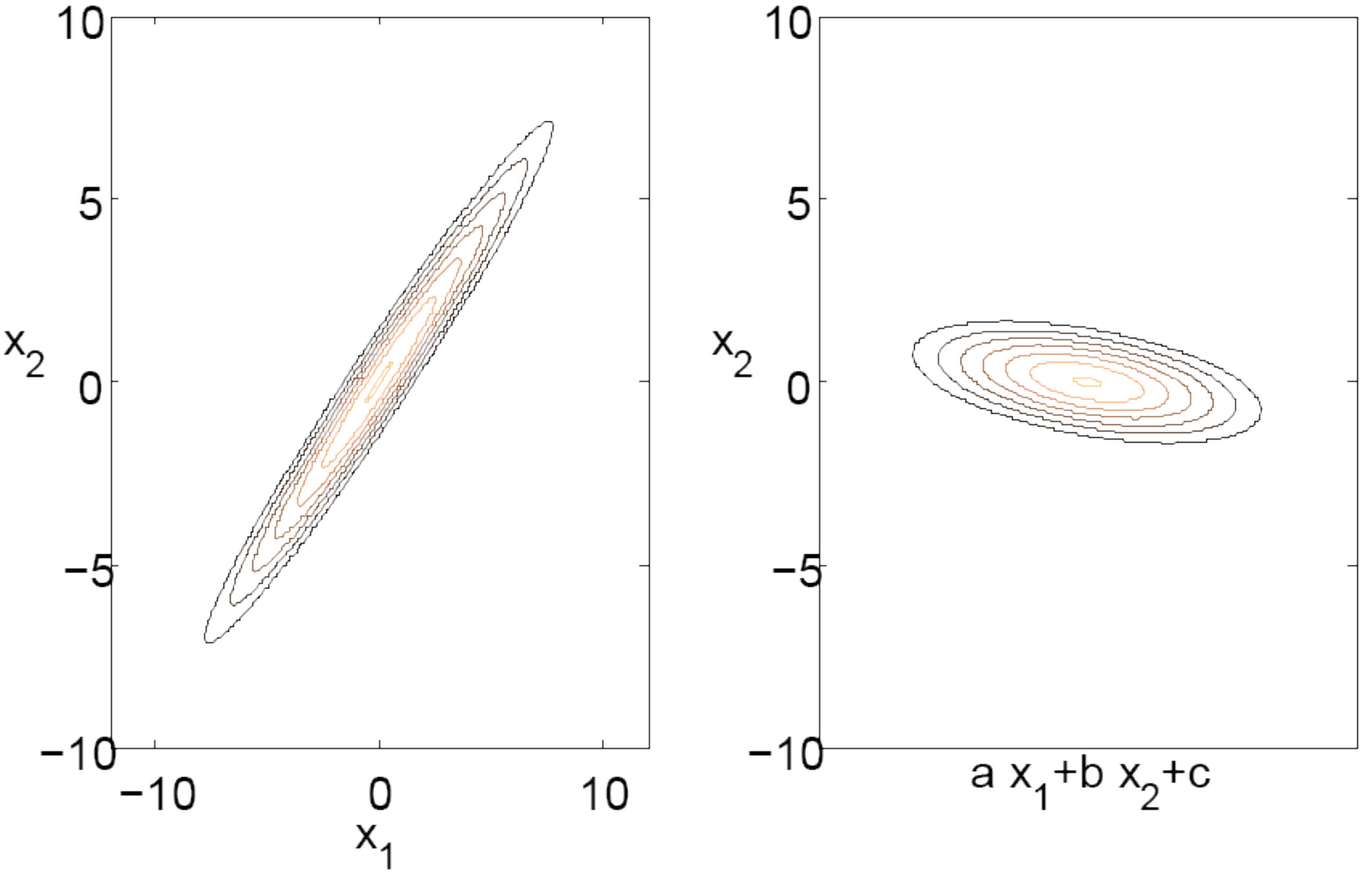


(D)

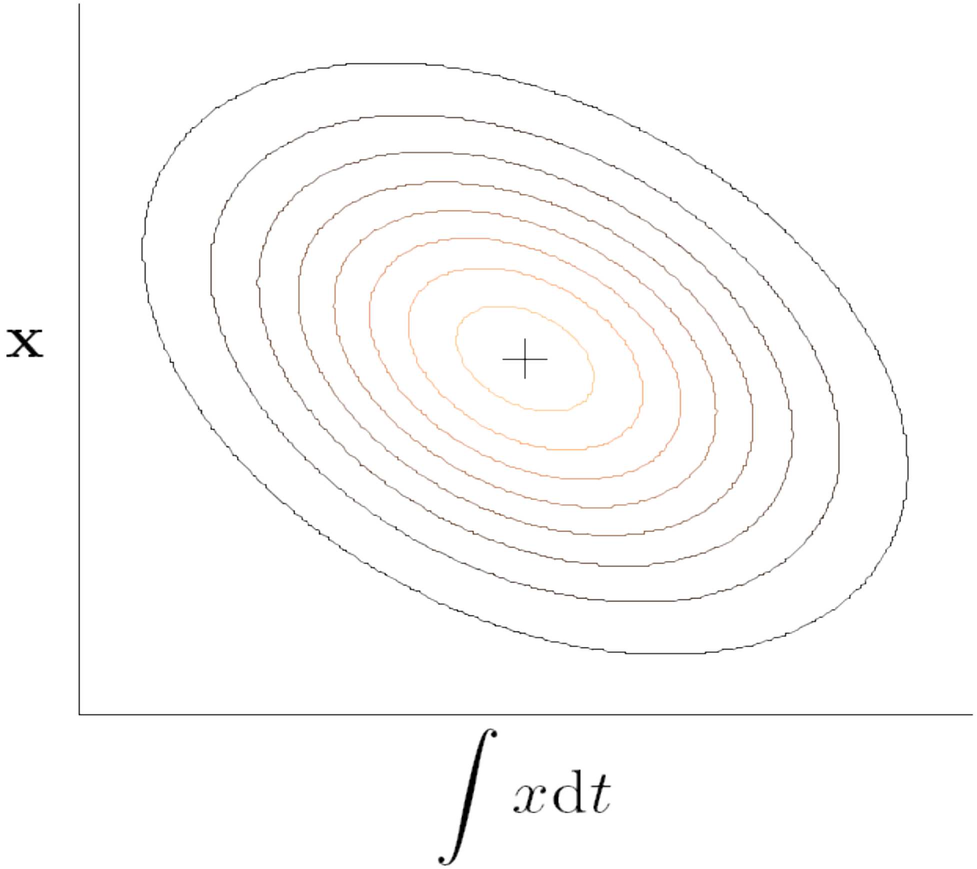
We could extend GPGO to additionally find the optimal number of set elements by defining the cost of additional elements.



Gaussian distributed variables are joint Gaussian with any affine transform of them.



A function over which we have a Gaussian process is joint Gaussian with any integral or derivative of it, as integration and differentiation are affine.



We can modify covariance functions to manage derivative or integral observations.

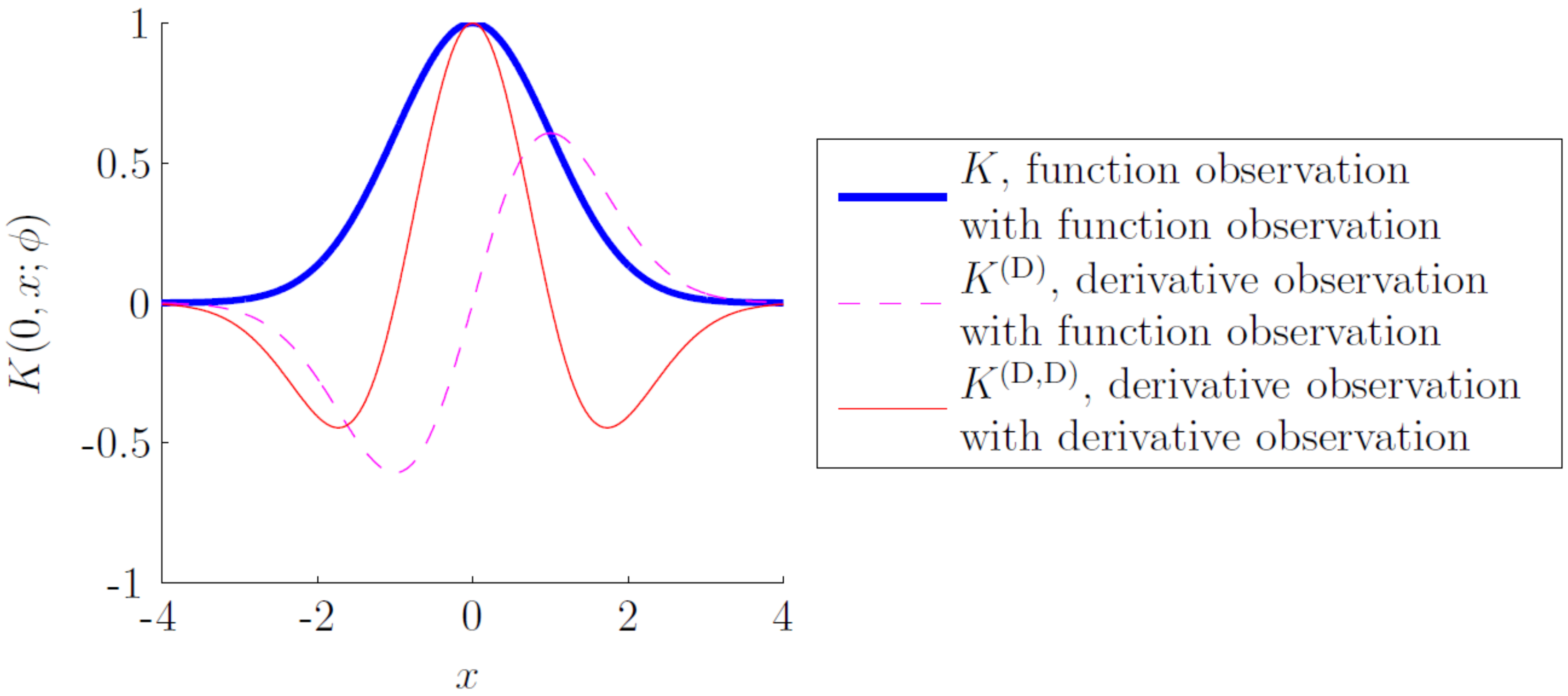
derivative observation at x_i and function observation at x_j.

$$K_D\left(x_i, x_j\right) = \frac{\partial}{\partial x} K\left(x, x_j\right)\Big|_{x=x_i}$$

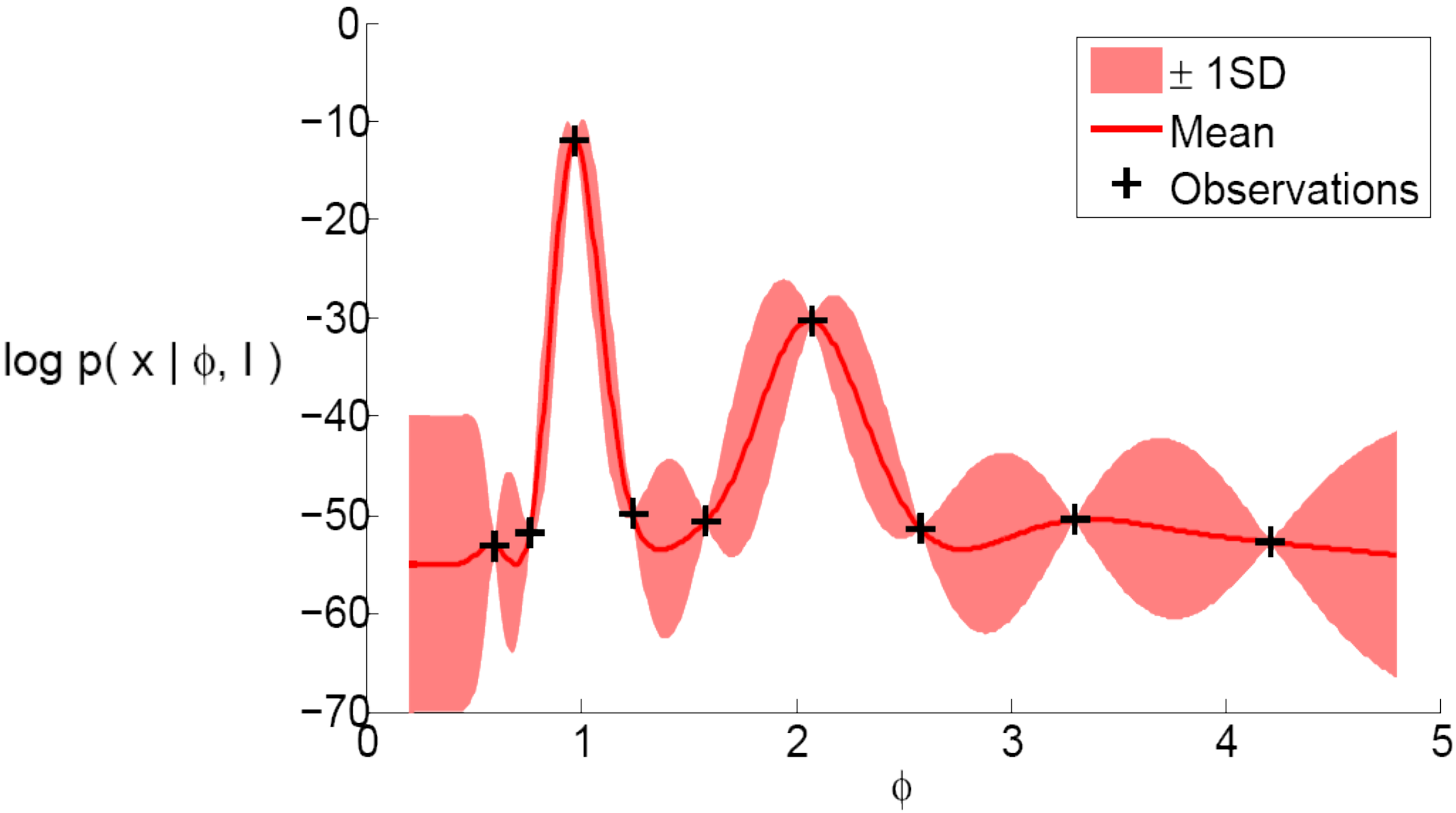
derivative observation at x_i and derivative observation at x_j.

$$K_{D,D}\left(x_i, x_j\right) = \frac{\partial}{\partial x'} \frac{\partial}{\partial x} K\left(x, x'\right)\Big|_{x=x_i}\Bigg|_{x'=x_j}$$

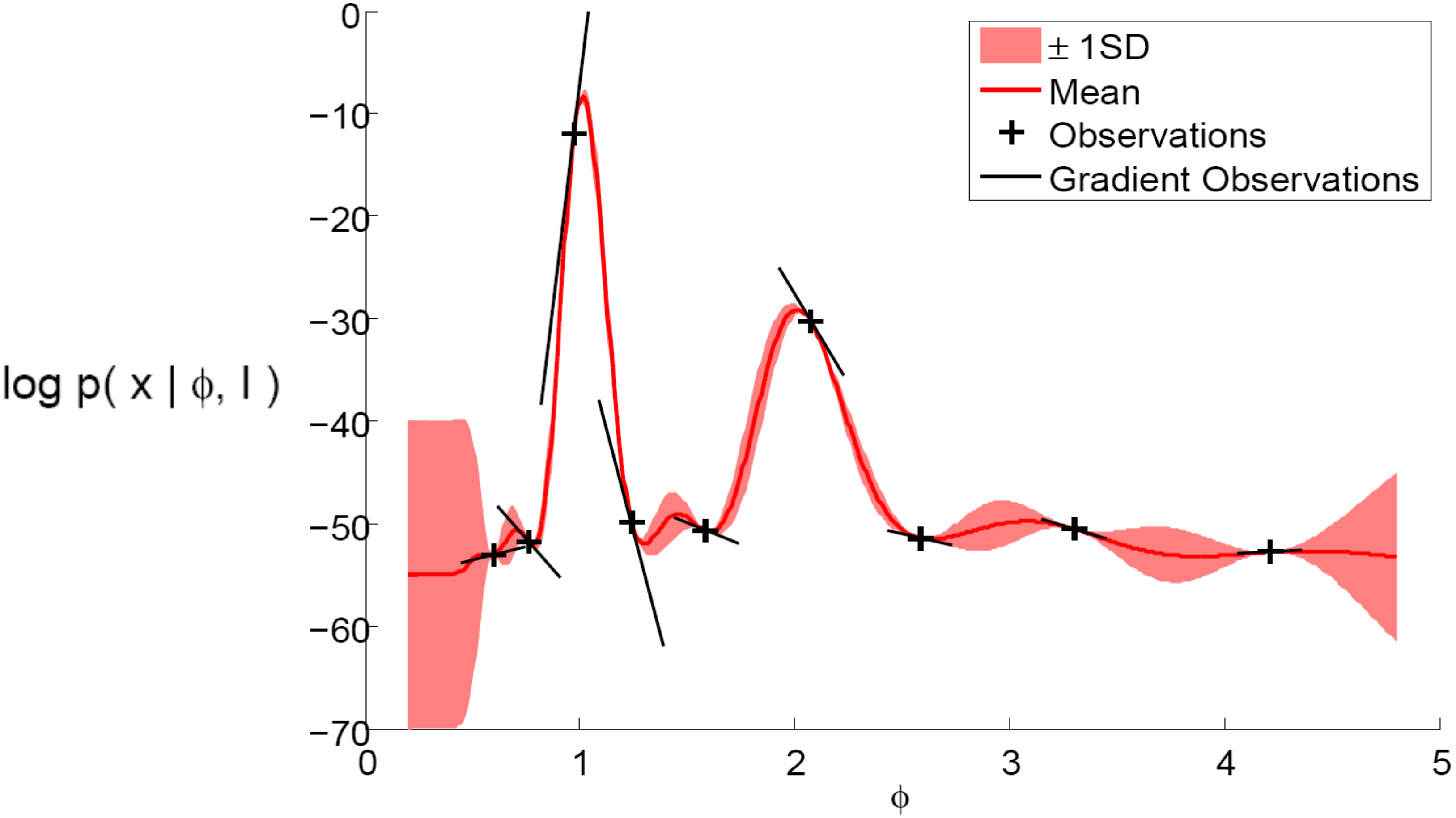
We can modify the squared exponential covariance to manage derivative observations.



The performance of Gaussian process global optimisation can be improved by including observations of the gradient of a function.



The performance of Gaussian process global optimisation can be improved by including observations of the gradient of a function.



Conditioning becomes an issue when we have multiple close observations, giving rows in the covariance matrix that are very similar.

$$\begin{pmatrix} 1 & 0.9999 & 0 & 0 \\ 0.9999 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.1 \\ 0 & 0 & 0.1 & 1 \end{pmatrix}$$

Too similar

Conditioning is a particular problem in optimisation, where we often take many samples around a local minimum to exactly determine it.

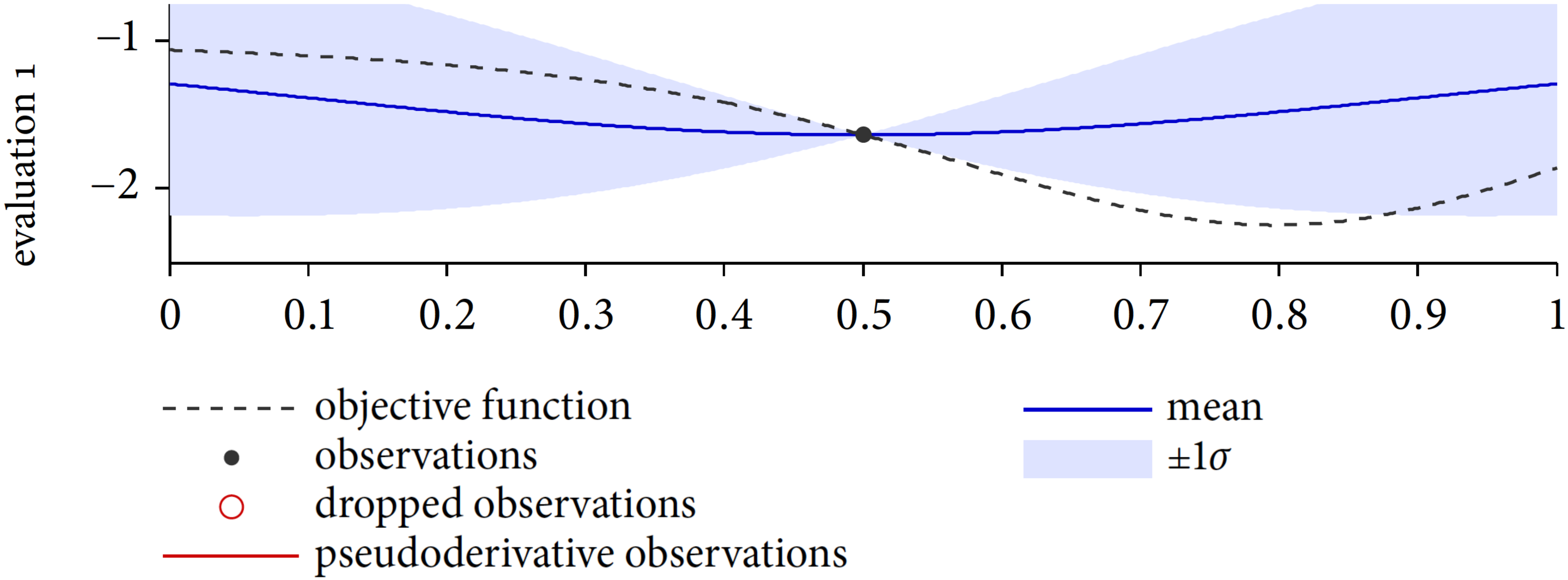
The usual solution to conditioning problems is to add a small positive quantity (*jitter*) to the diagonal of the covariance matrix.

$$\begin{pmatrix} 1.01 & 0.9999 & 0 & 0 \\ 0.9999 & 1.01 & 0 & 0 \\ 0 & 0 & 1.01 & 0.1 \\ 0 & 0 & 0.1 & 1.01 \end{pmatrix}$$

Sufficiently dissimilar

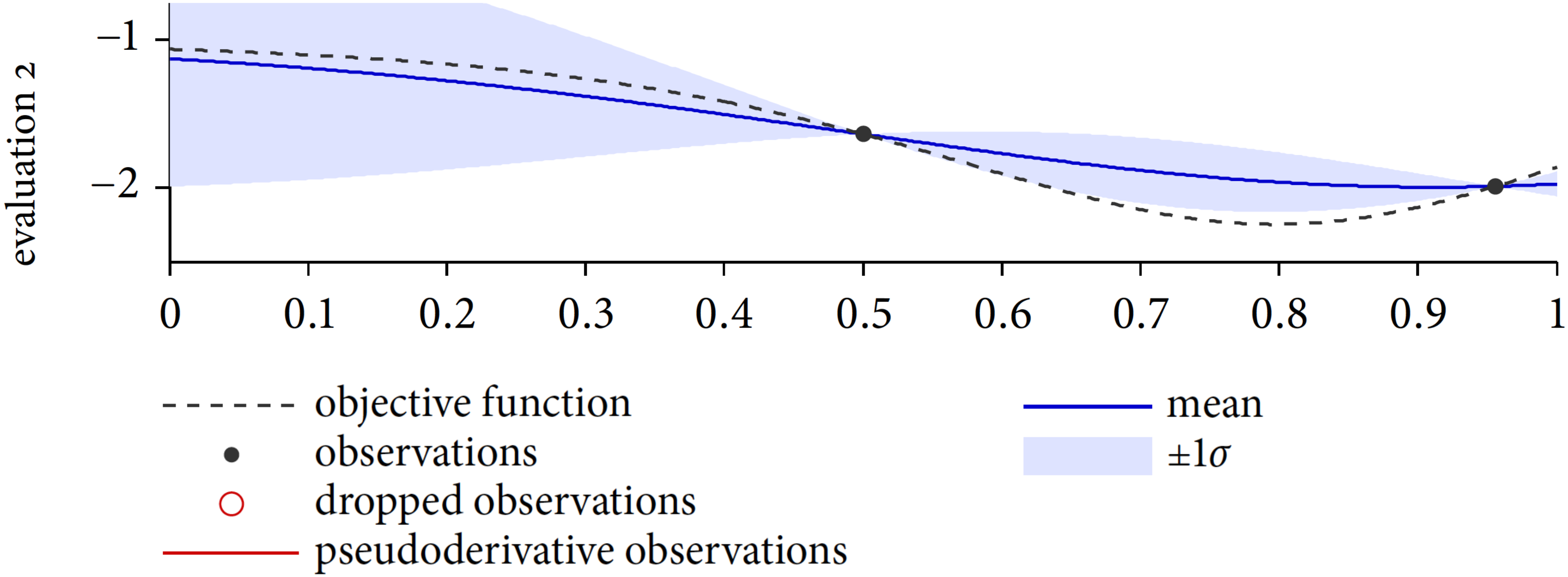
As jitter is effectively imposed noise, adding jitter to all diagonal elements dilutes the informativeness of our data.

We can readily incorporate observations of the derivative into the Gaussian process used in GPGO.



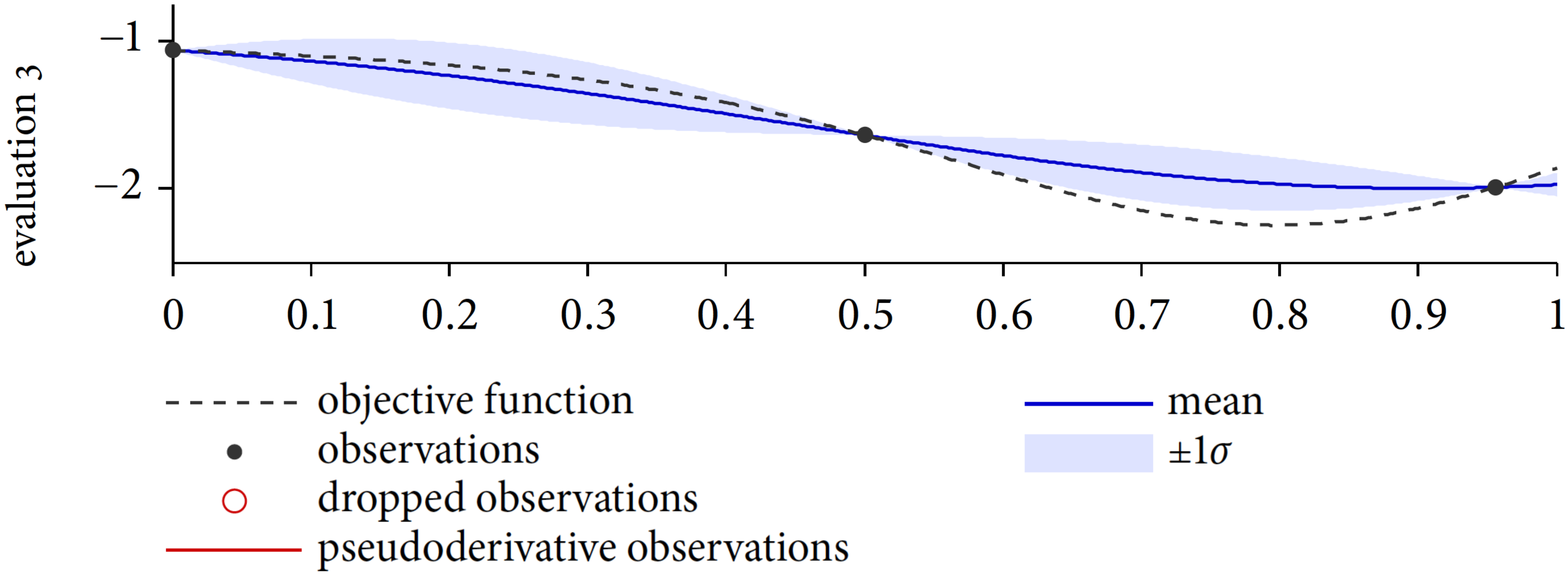
This also gives us a way to resolve conditioning issues (that result from the very close observations that are taken during optimisation).

We can readily incorporate observations of the derivative into the Gaussian process used in GPGO.



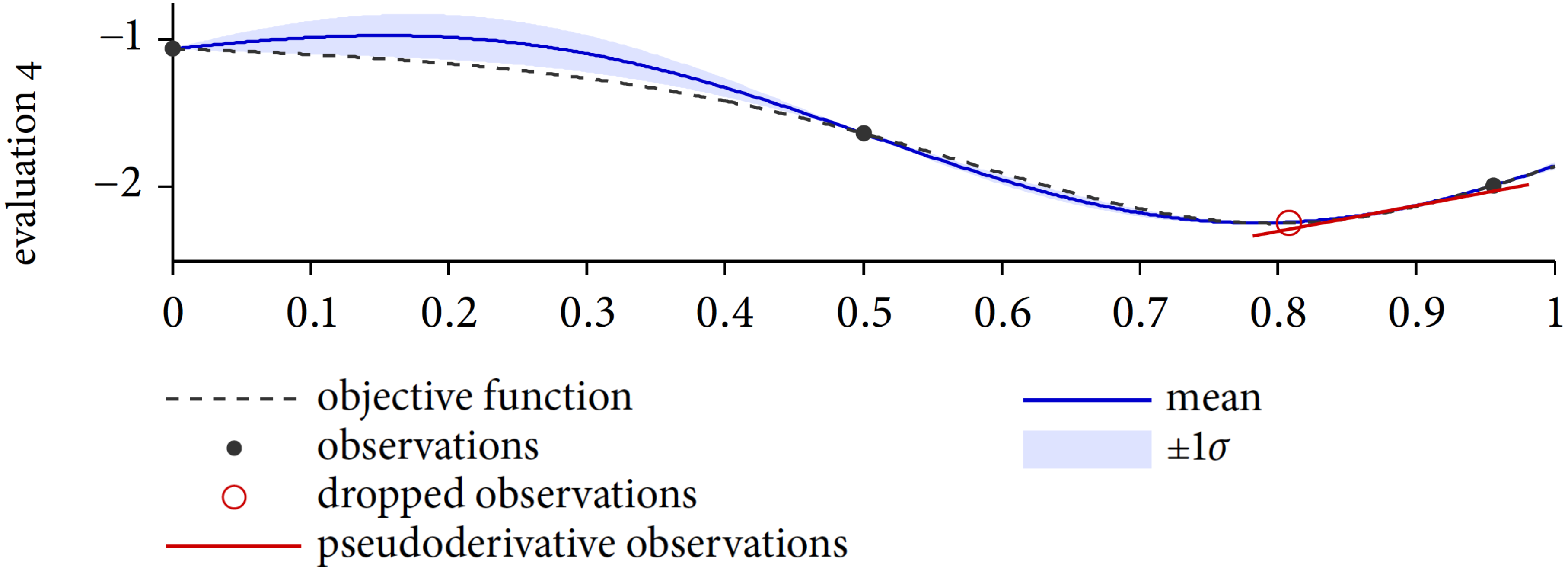
This also gives us a way to resolve conditioning issues (that result from the very close observations that are taken during optimisation).

We can readily incorporate observations of the derivative into the Gaussian process used in GPGO.



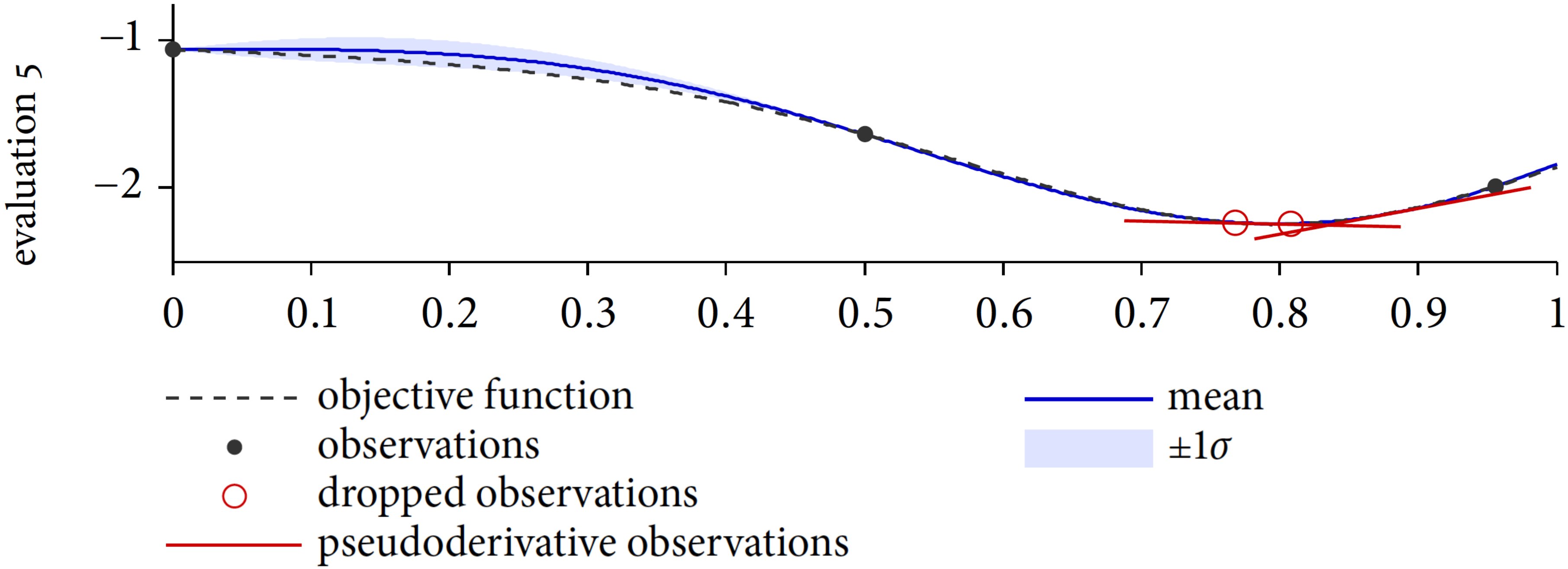
This also gives us a way to resolve conditioning issues (that result from the very close observations that are taken during optimisation).

We can readily incorporate observations of the derivative into the Gaussian process used in GPGO.



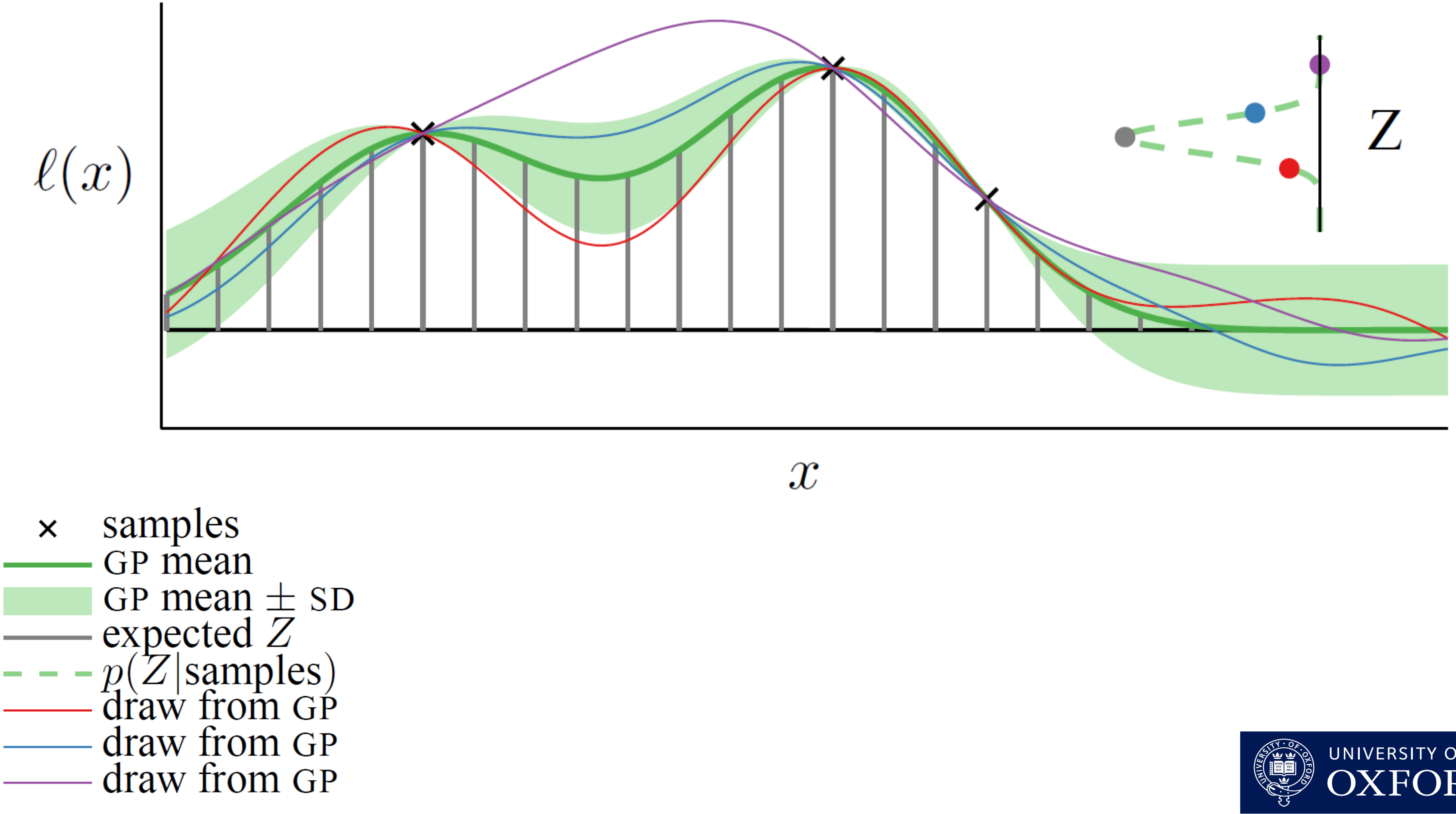
This also gives us a way to resolve conditioning issues (that result from the very close observations that are taken during optimisation).

We can readily incorporate observations of the derivative into the Gaussian process used in GPGO.

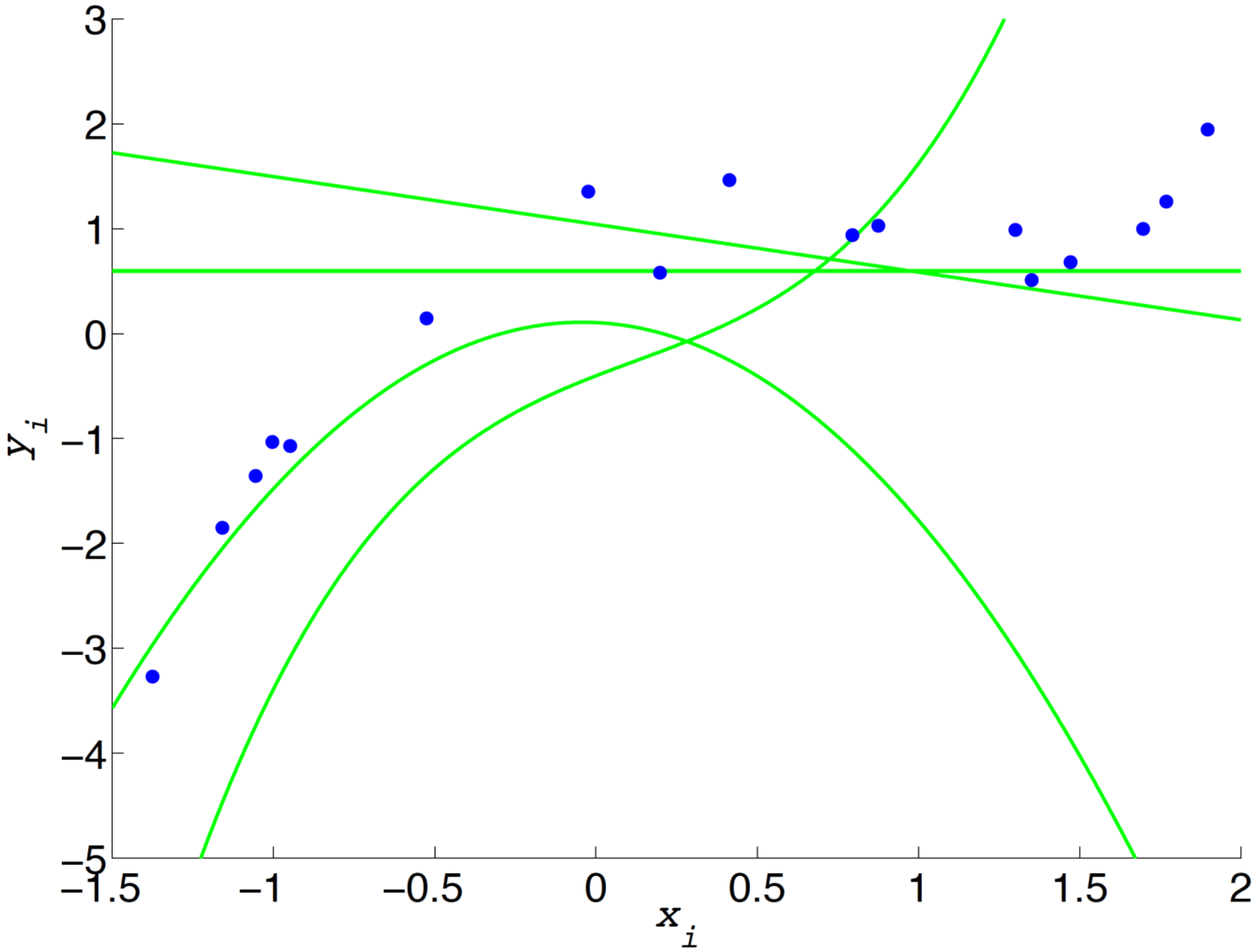


This also gives us a way to resolve conditioning issues (that result from the very close observations that are taken during optimisation).

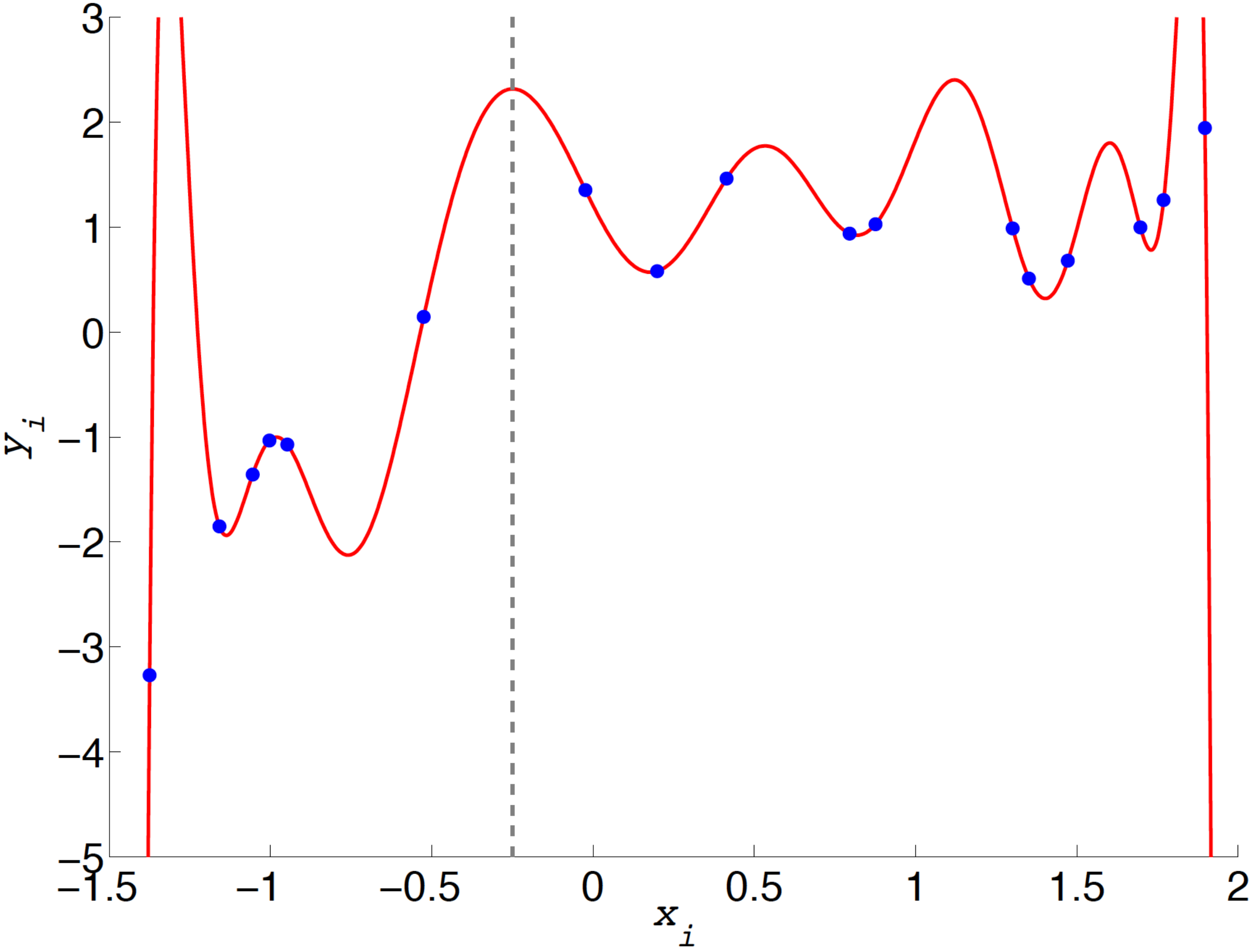
Likewise, we can use observations of an integrand ℓ in order to perform inference for its integral, Z: this is known as Bayesian Quadrature.



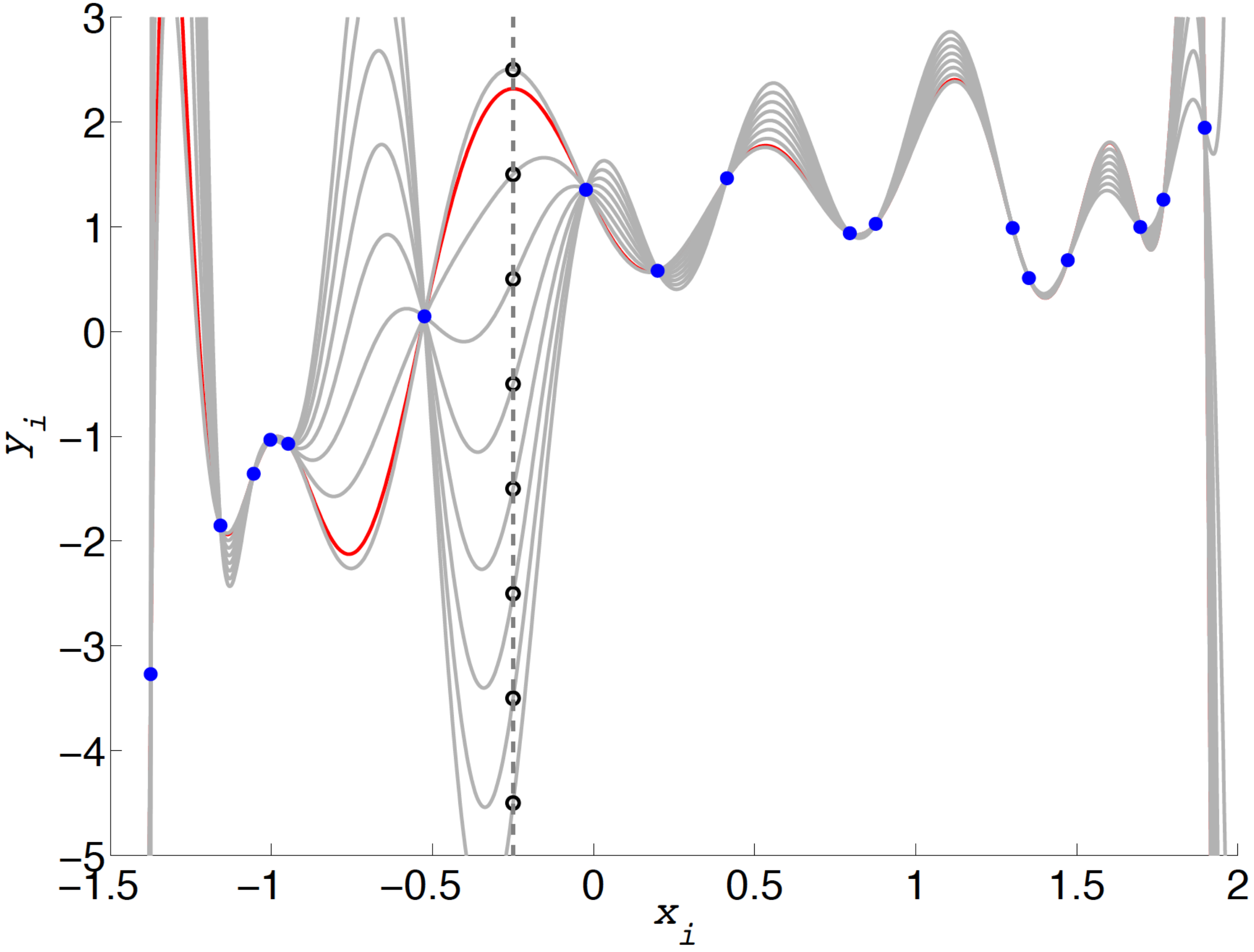
Modelling data is problematic due to parameters, such as the parameters of a polynomial model.



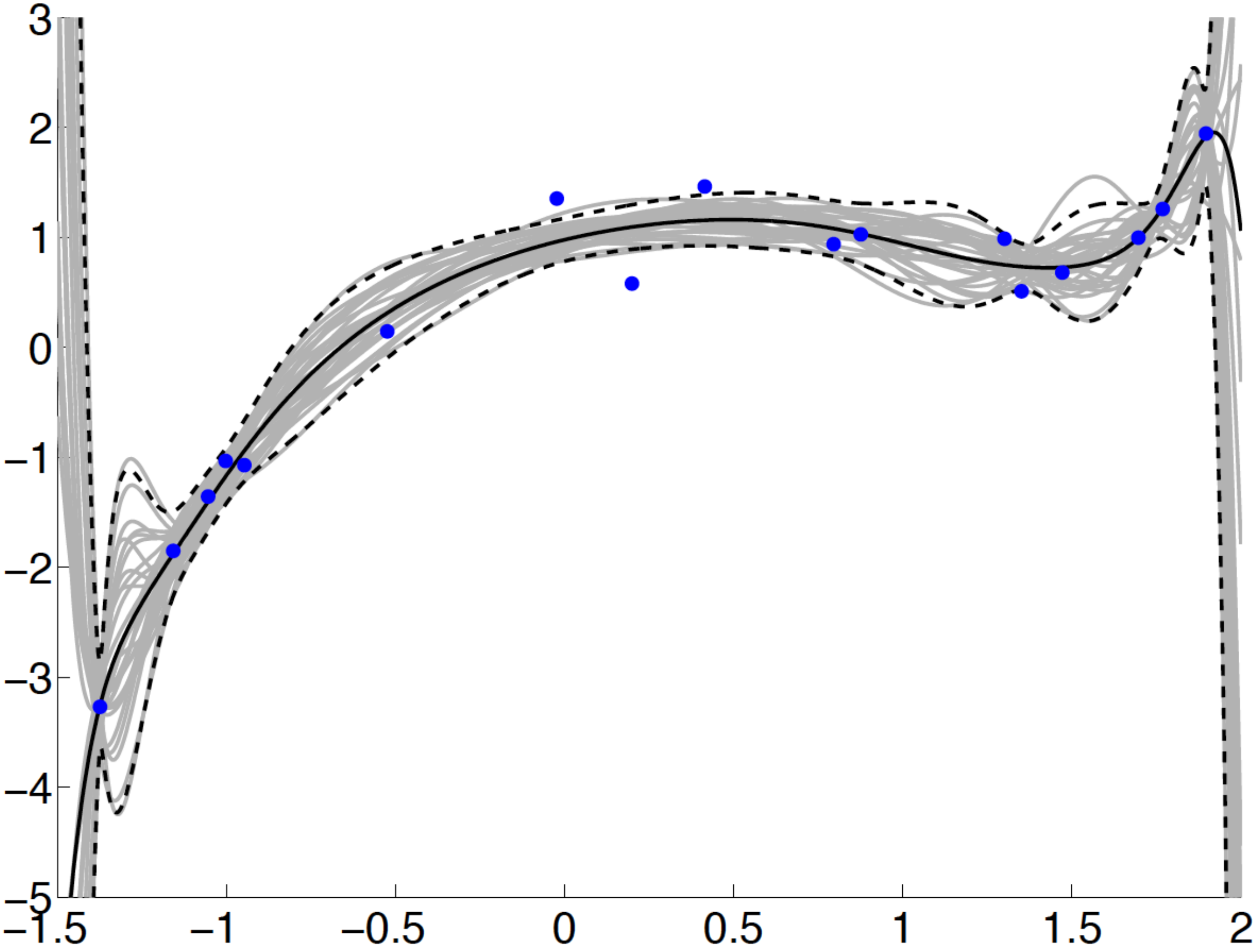
We could minimise errors (or maximise likelihood) to find a `best-fit’, at the risk of overfitting.



The Bayesian approach is to average over many possible parameters, requiring integration.

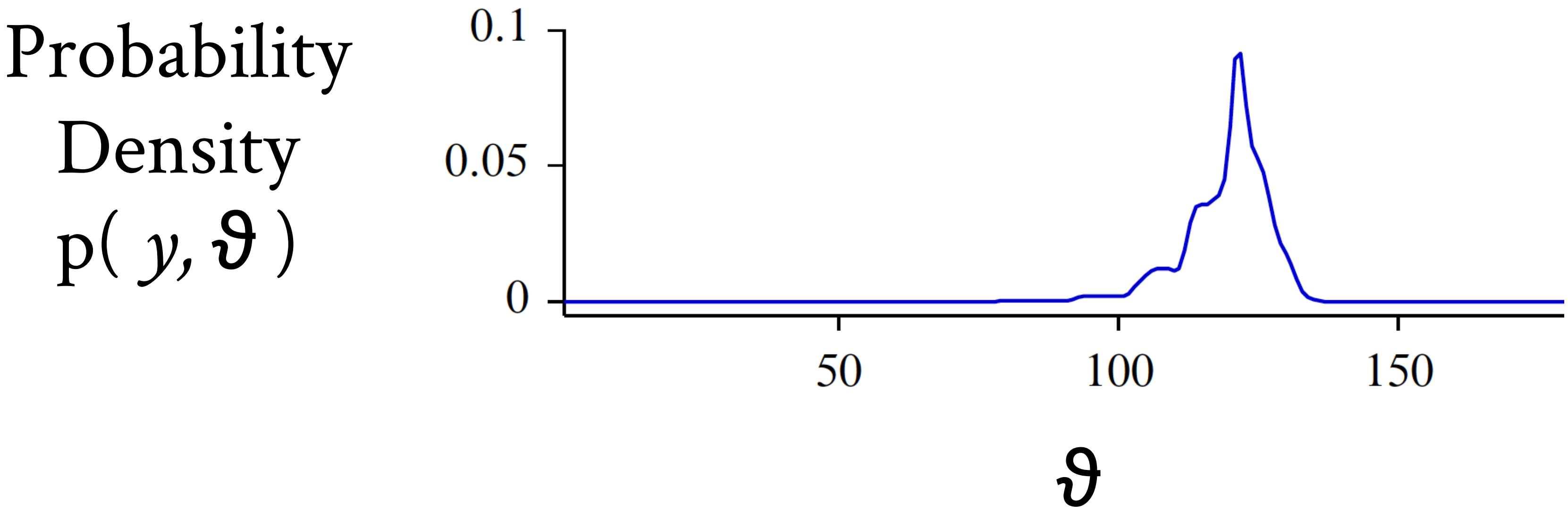


The Bayesian approach is to average over many possible parameters, requiring integration.

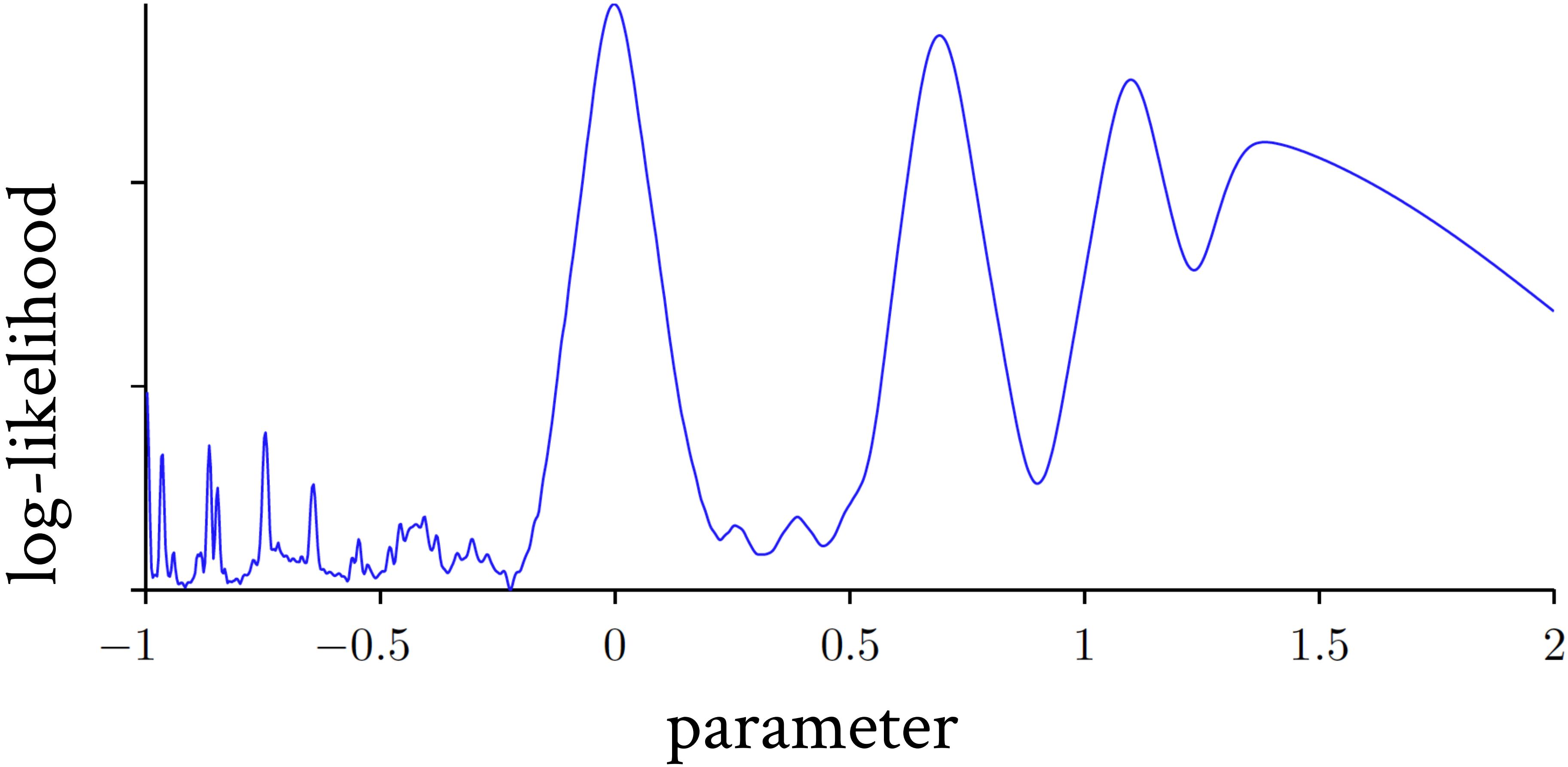


Inference requires integrating (or marginalising) over the many possible states of the world consistent with our data, which is often non-analytic.

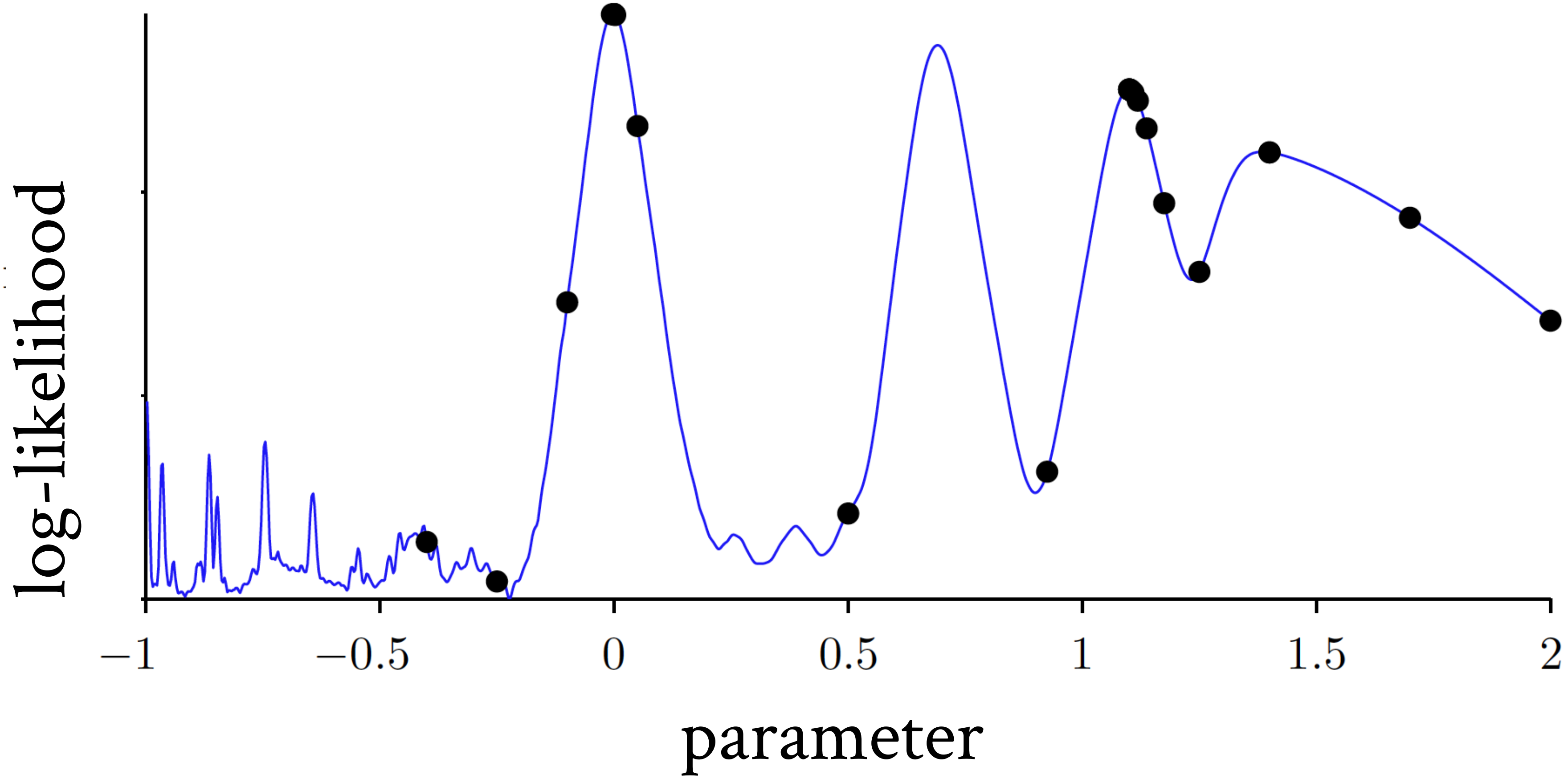
Probability(y) = the area under this curve:



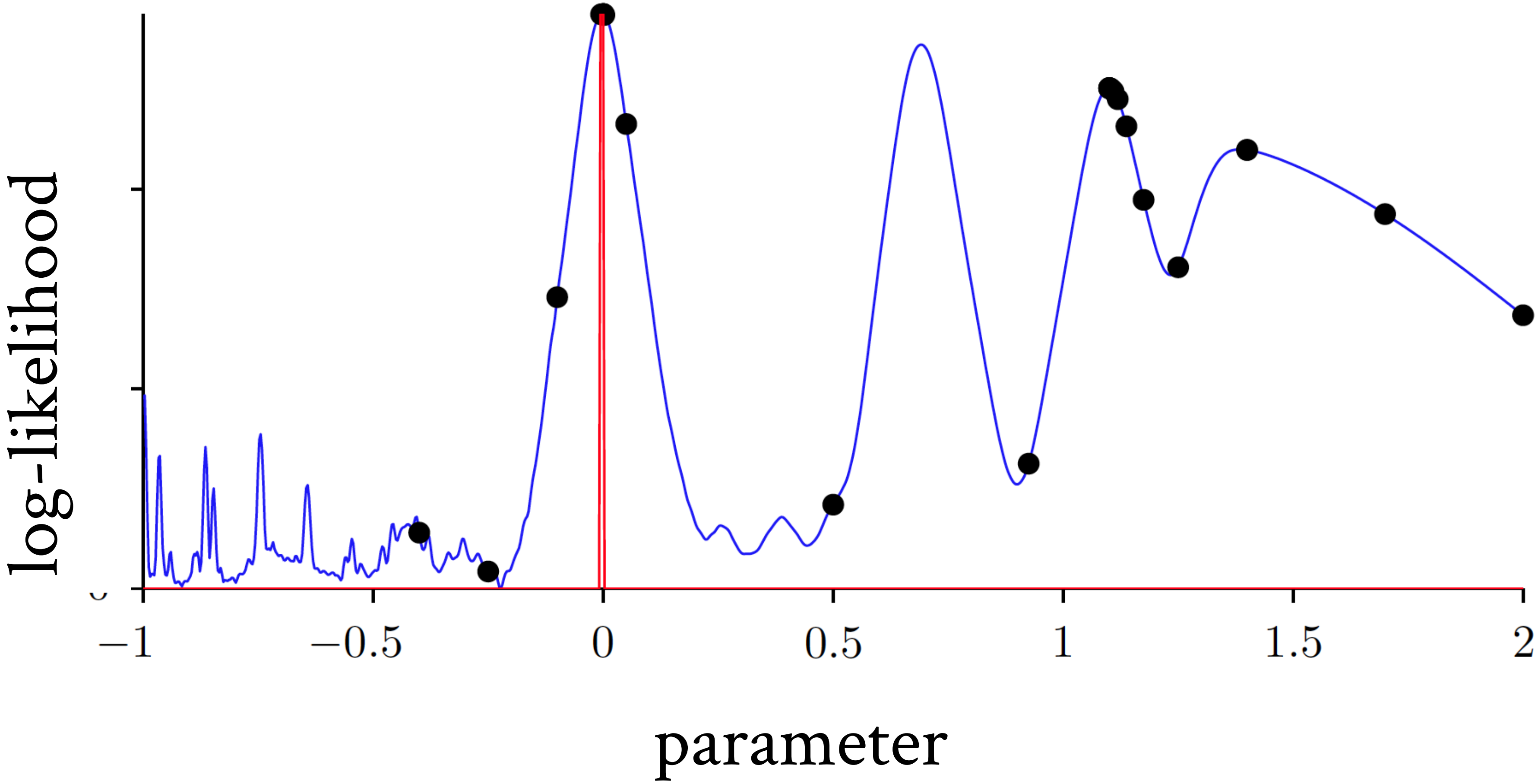
There are many different approaches to quadrature (numerical integration) for probabilistic integrals; integrand estimation is usually undervalued.



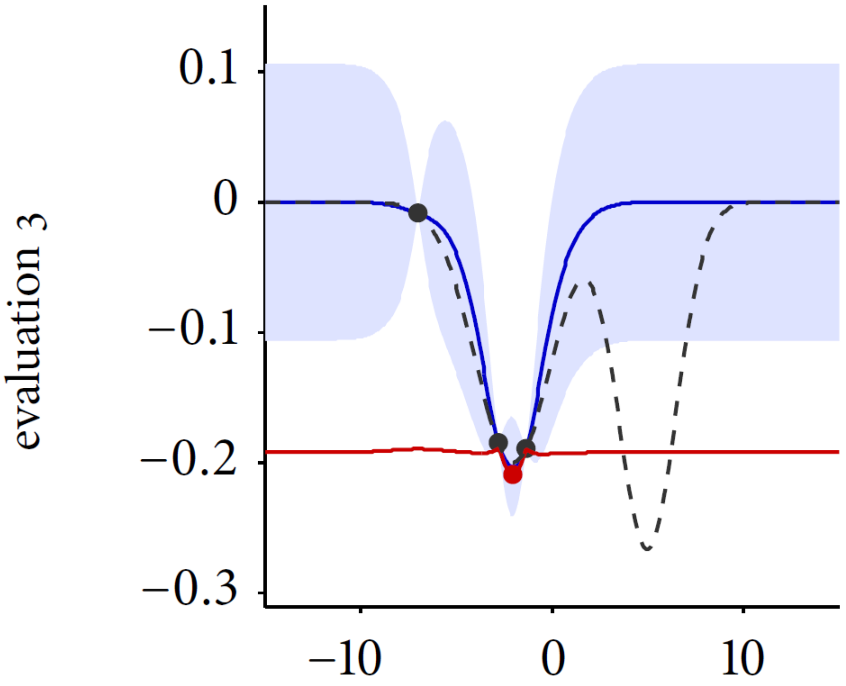
Optimisation (as in maximum likelihood), particularly using global optimisers, gives a reasonable heuristic for exploring the integrand.

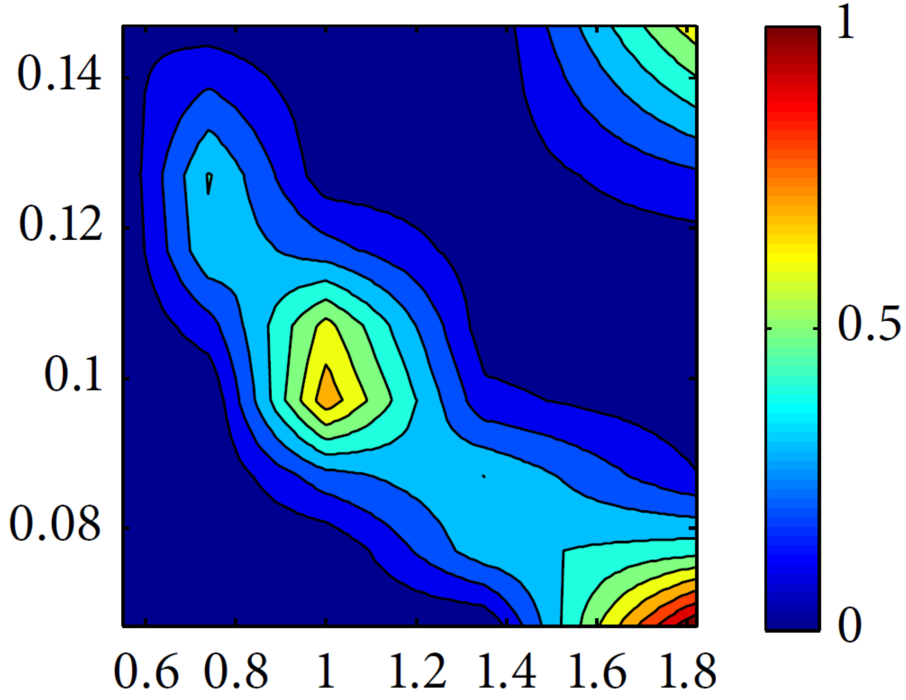


However, maximum likelihood is an unreasonable way of estimating a multi-modal or broad likelihood integrand: why throw away all those other samples?



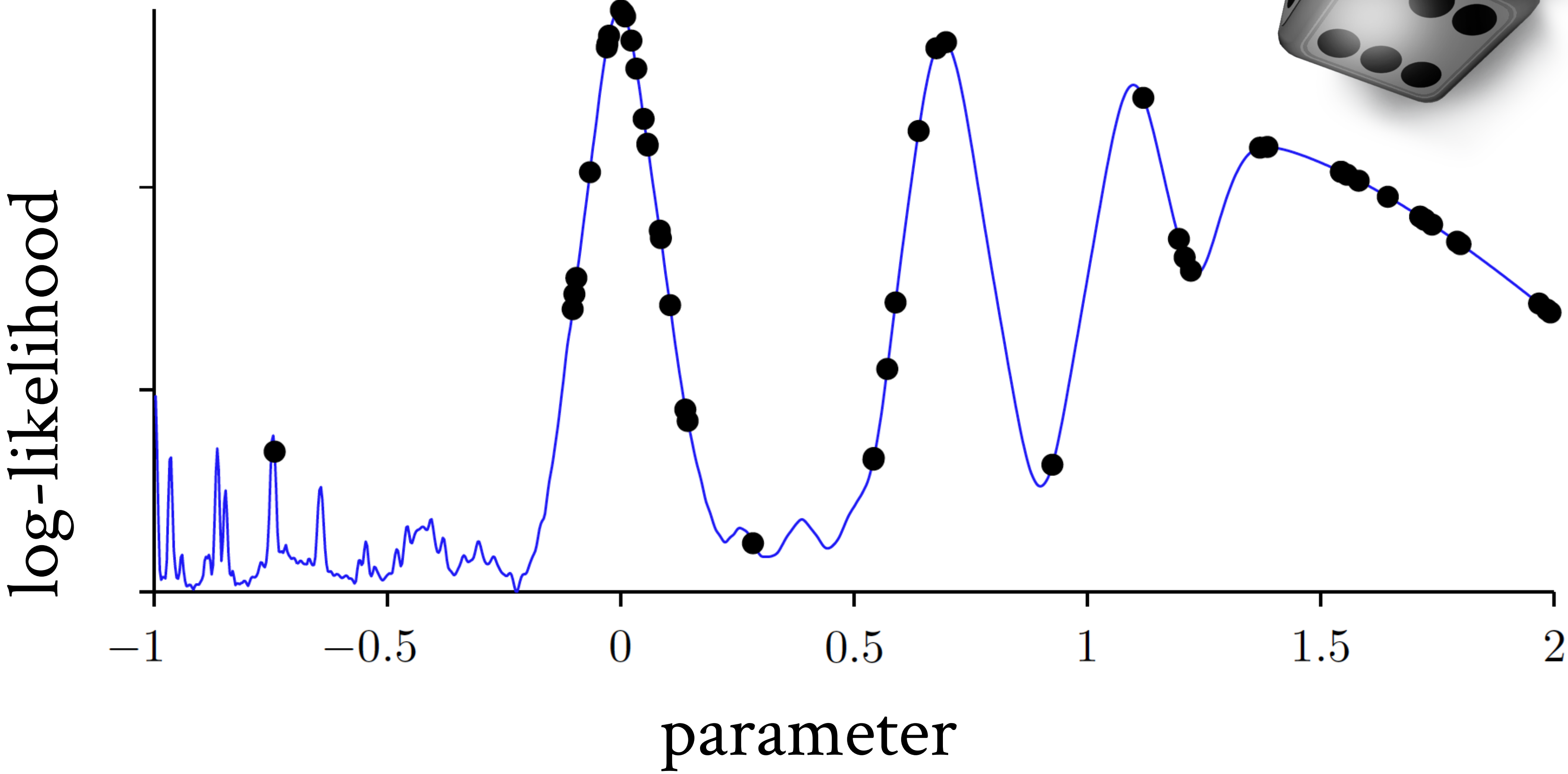
In particular, maximum likelihood is inadequate for the hyper-parameters of a Gaussian process used for optimisation.



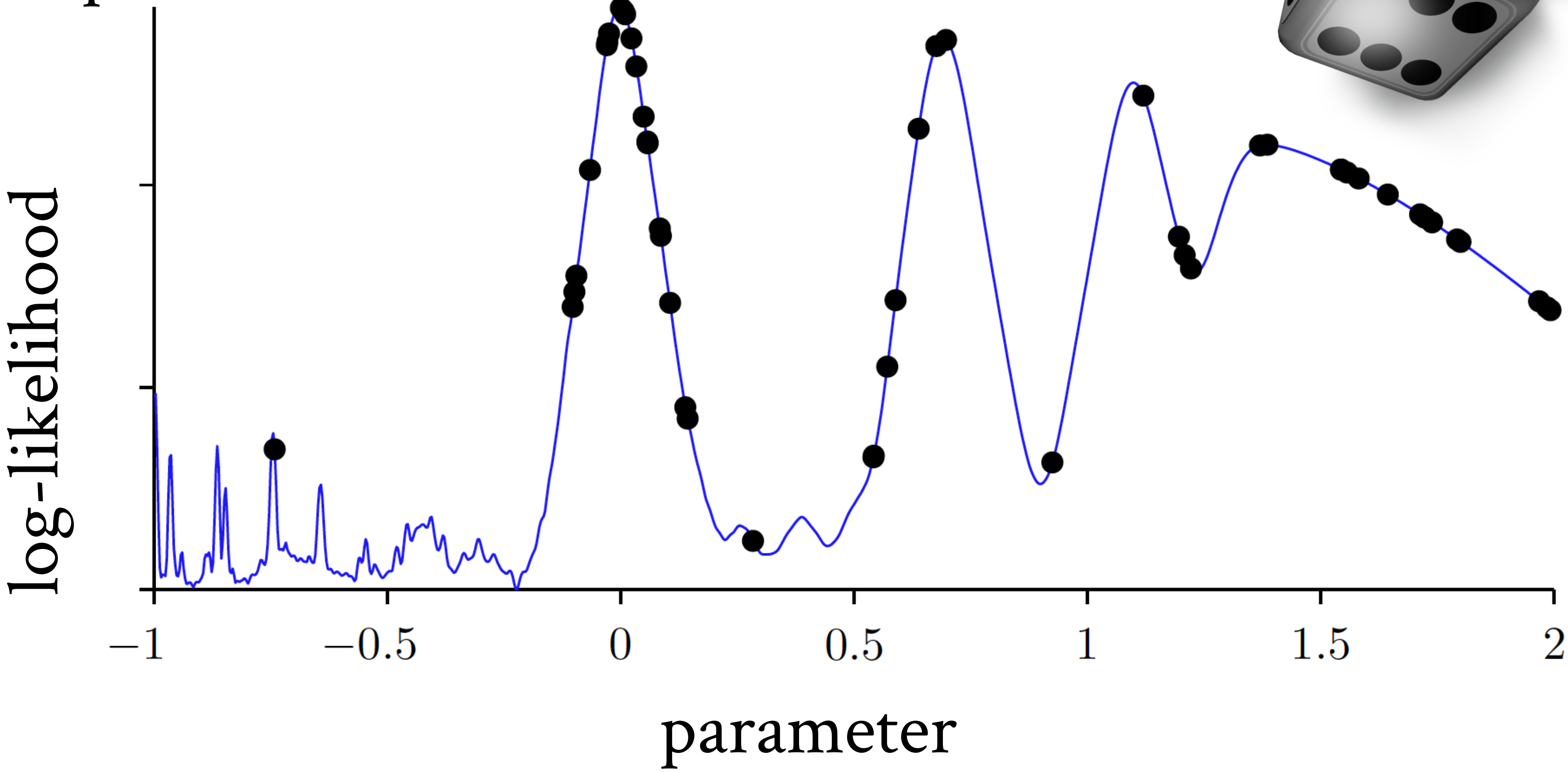


Approximating the posterior over hyperparameters (above right) as a delta function results in insufficient exploration of the objective function!

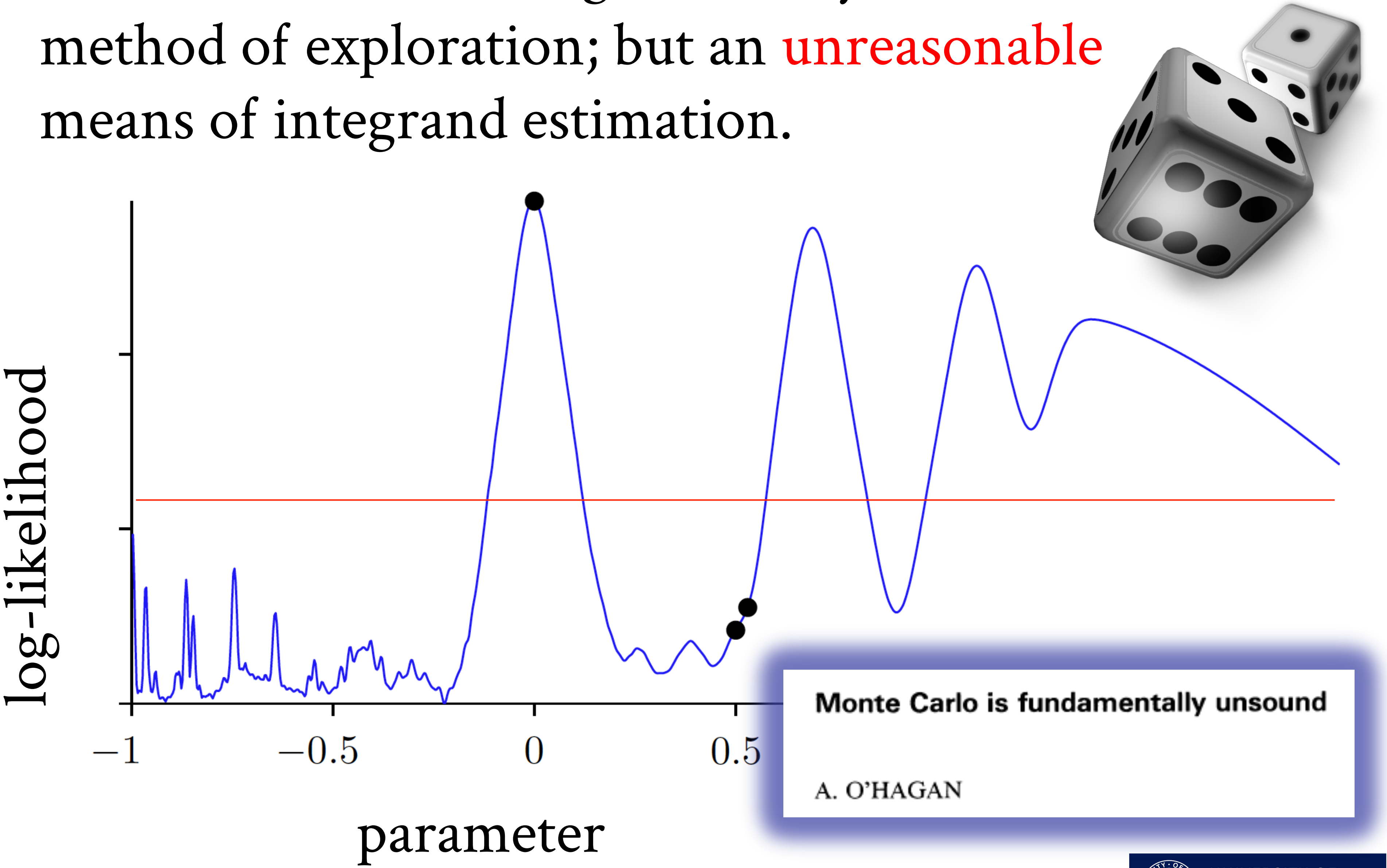
Monte Carlo schemes give even more powerful methods of exploration, that have revolutionised Bayesian inference.



These methods force us to be uncertain about our action, which is unlikely to result in actions that minimise a sensible expected loss.



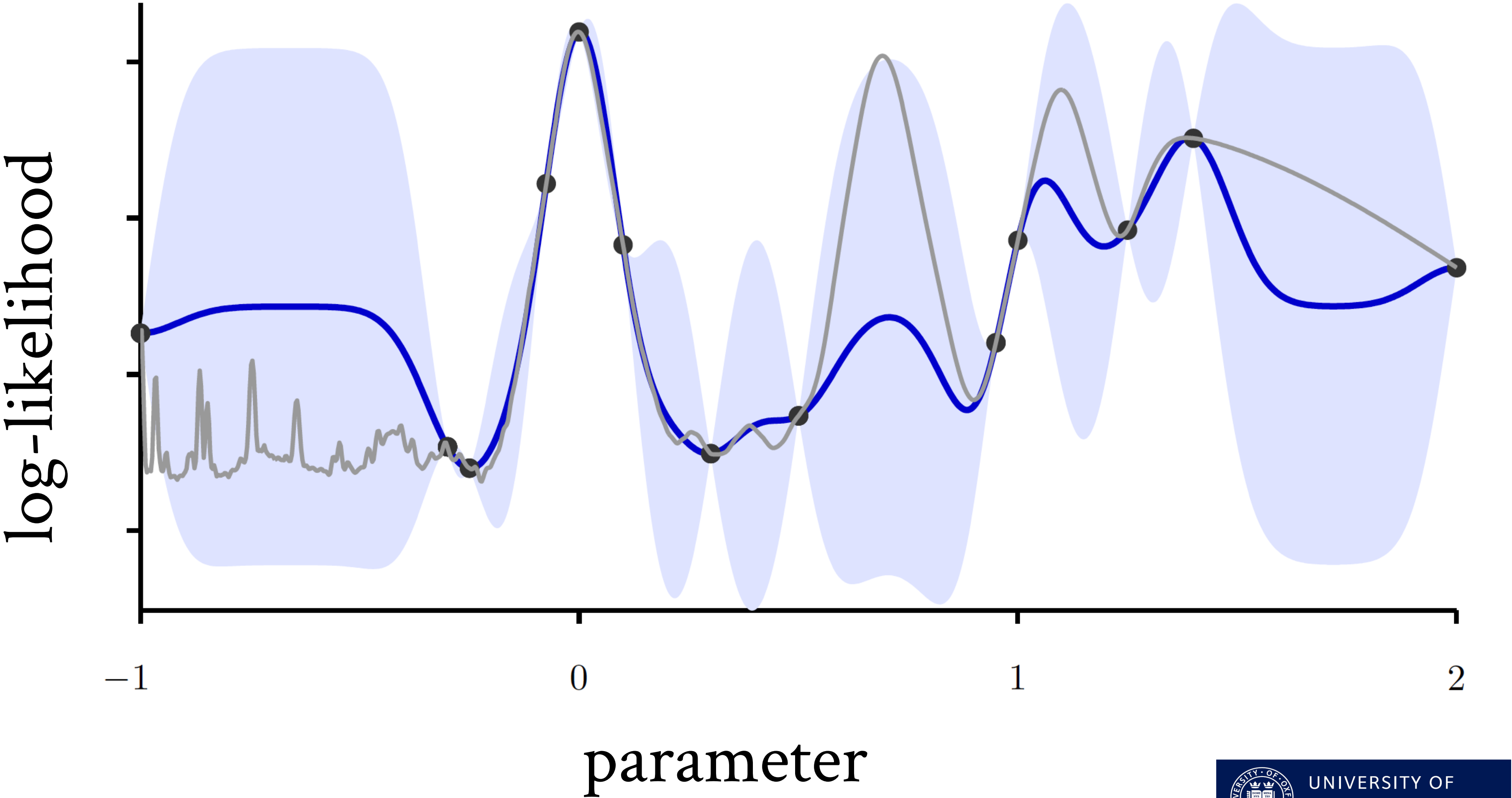
Monte Carlo schemes give a fairly reasonable method of exploration; but an unreasonable means of integrand estimation.



Monte Carlo is fundamentally unsound

A. O'HAGAN

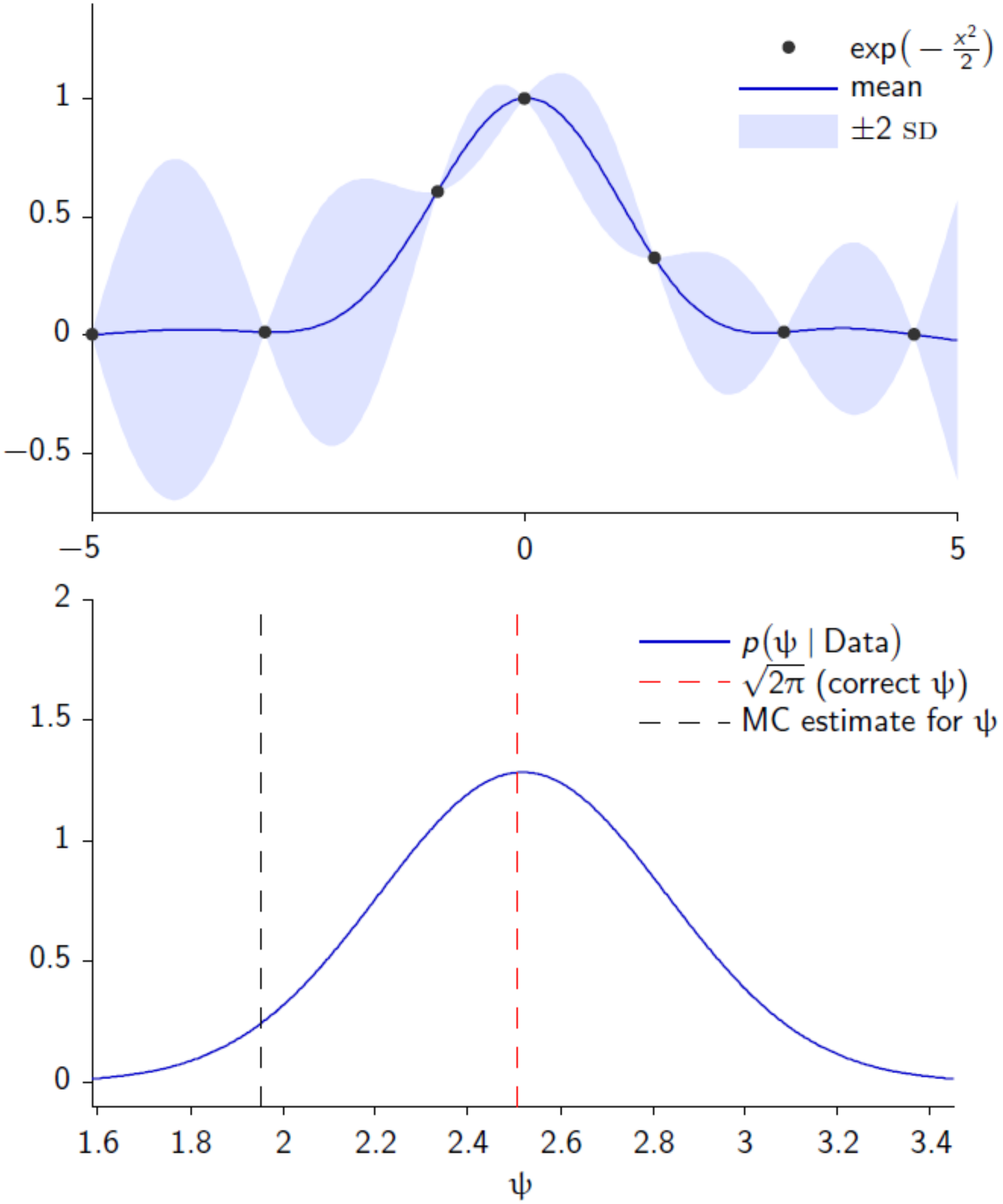
Bayesian quadrature (aka Bayesian Monte Carlo) gives a powerful method for estimating the integrand: a Gaussian process.



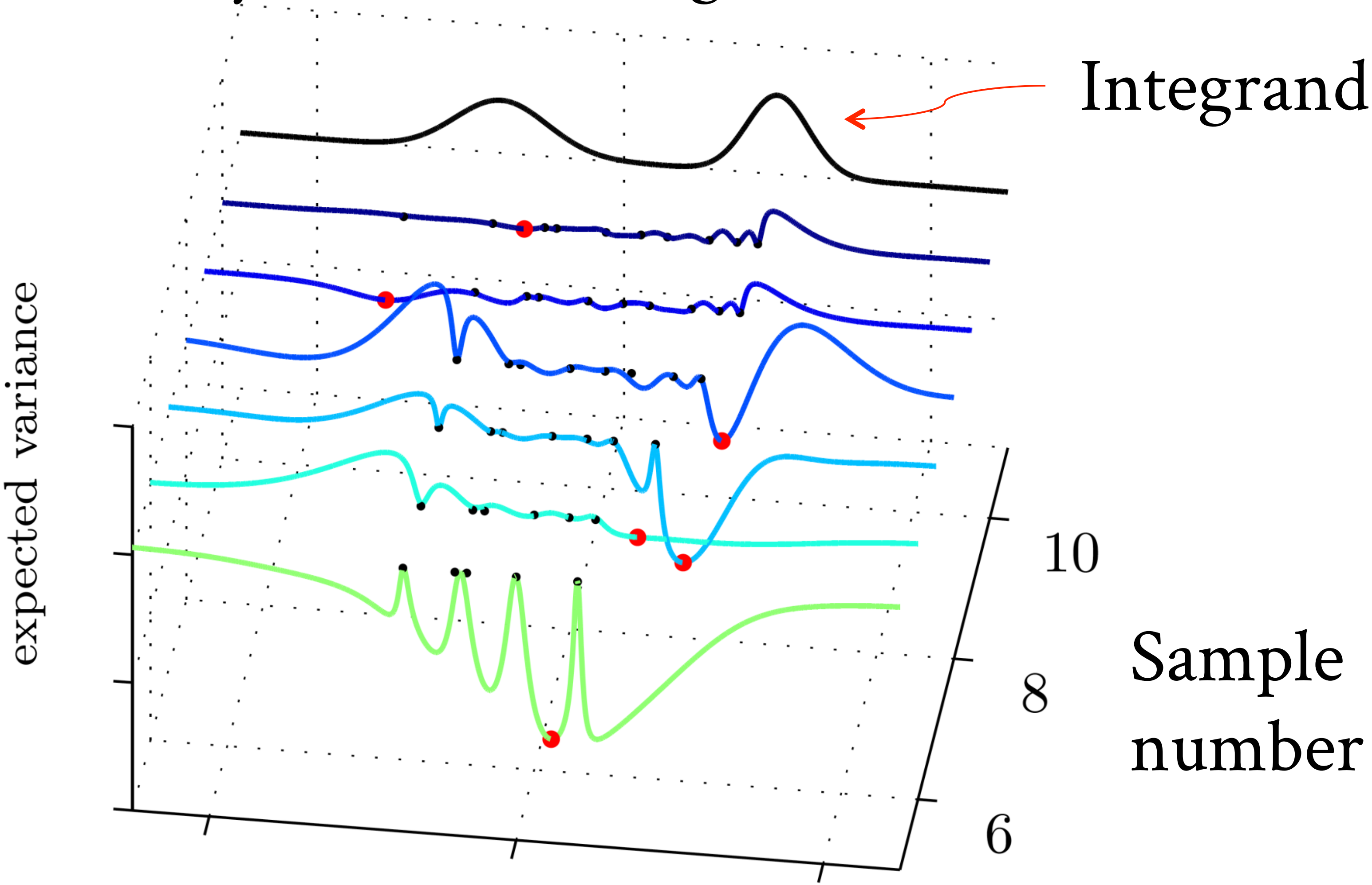
Consider the integral

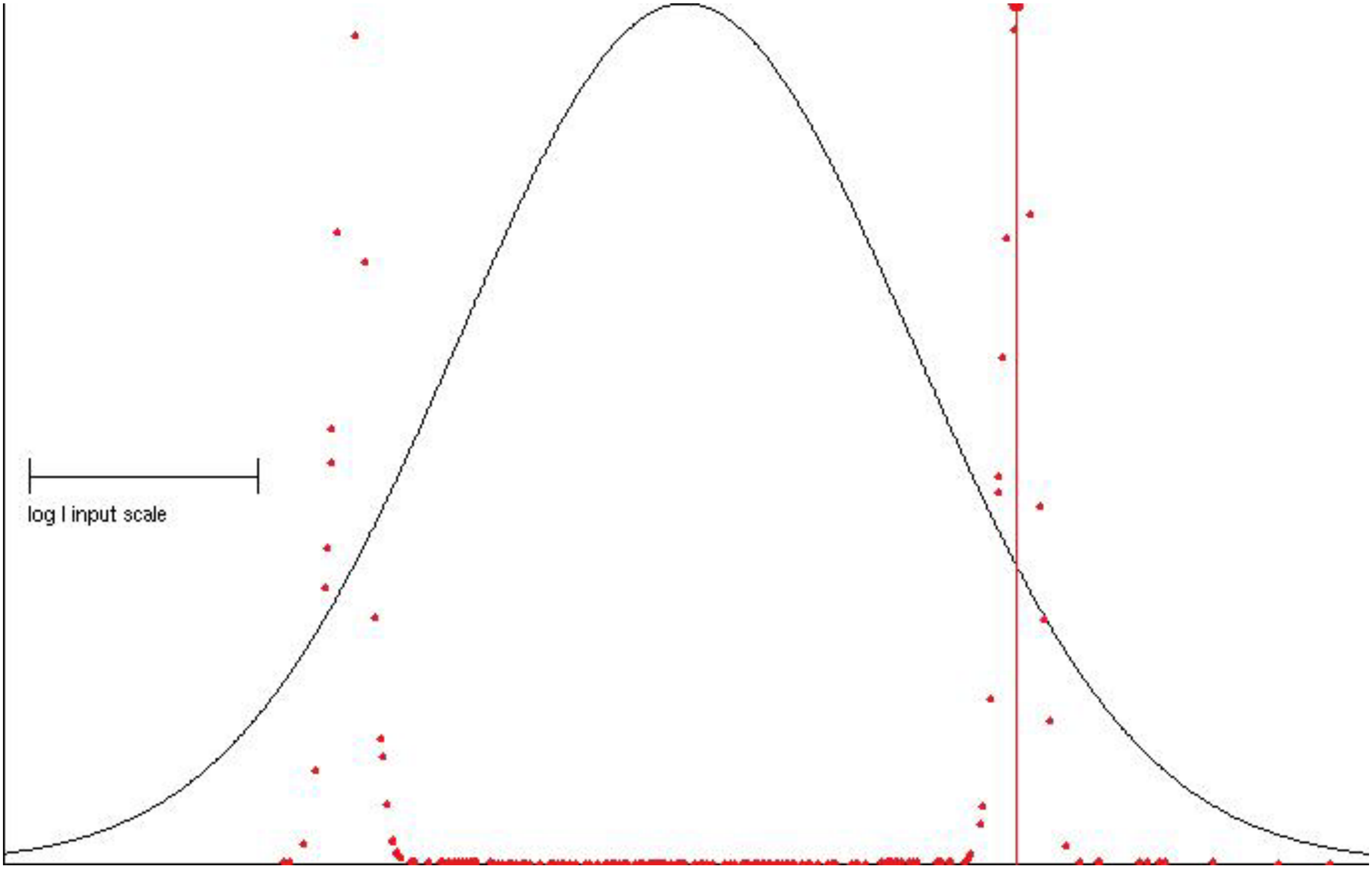
$$\psi = \int_{-5}^{5} \exp\left(-\frac{x^2}{2}\right) \mathrm{d}x$$

Bayesian quadrature achieves more accurate results than Monte Carlo, and provides an estimate of our uncertainty.

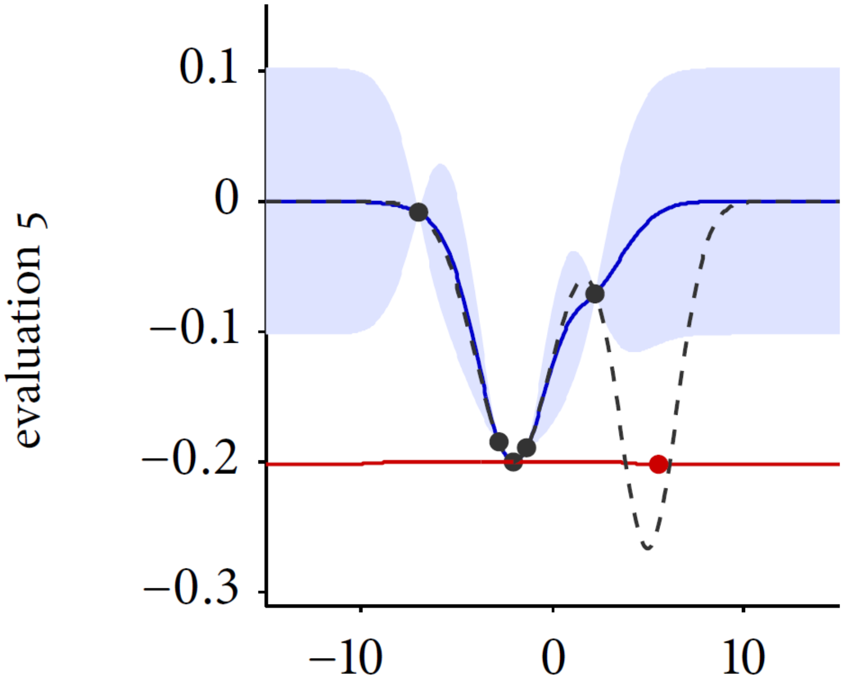


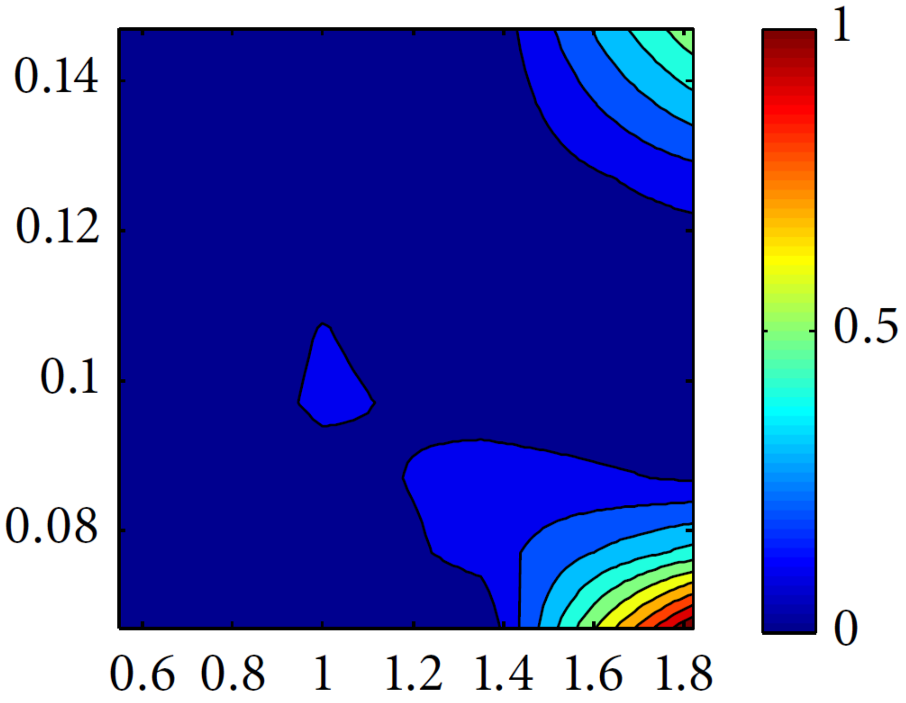
Doubly-Bayesian quadrature (BBQ) additionally explores the integrand so as to minimise the uncertainty about the integral.




log I input scale

Bayesian optimisation is a means of optimising expensive, multi-modal objective functions; Bayesian quadrature is a means of integrating over expensive, multi-modal integrands.





Thanks! I would like to also like thank my collaborators:

David Duvenaud,
Roman Garnett,
Zoubin Ghahramani,
Carl Rasmussen,
and Stephen Roberts.

References

http://www.robots.ox.ac.uk/~mosb