COMP 551 – Applied Machine Learning
Lecture 21: Bayesian optimisation

Associate Instructor: Herke van Hoof (herke.vanhoof@mcgill.ca)

Class web page: *www.cs.mcgill.ca/~jpineau/comp551*

Office hours & midterm prep

I'm available for questions:
today, 2:30 – 3:30, room 104N (next to Joelle's office)
wednesday, 9:00 – 10:00, room 104 N (next to Joelle's office)

Other resources:
Tutorial this evening: TR3120, 7–9pm
Last minute questions, Joelle Pineau, 8:30–11 am

Kaggle #3

#	△pub	Team Name	Kernel	Team Members	Score	Entries	Last
1	▲1	Definitely need a Titan X			0.98833	10	9d
2	▲2	chuanpu			0.98533	8	8d
3	▼2	Kineserne			0.98516	15	6d
4	▼1	Braavos			0.98299	20	8d
5	—	CAM CAM CAM			0.97488	23	8d
6	▲3	2ez			0.97433	6	9d
7	▼1	MeowdRed			0.97188	12	8d
8	—	MAE			0.97150	4	7d
9	▼2	AOM			0.97066	13	7d
10	▲1	Random Predict Baseline			0.96988	14	13d
11	▲5	xXxXx			0.96916	13	11d
12	—	Have no Titan X			0.96566	11	10d
13	▼3	rua			0.96499	21	7d
14	▲1	M			0.96483	2	12d
15	▲3	Nvidia Tesla K40c			0.96393	10	8d

Choosing between linear methods

linear regression	ridge regression	bayesian linear regression
↓	↓	↓
(kernel regression)	kernel ridge regression	Gaussian process

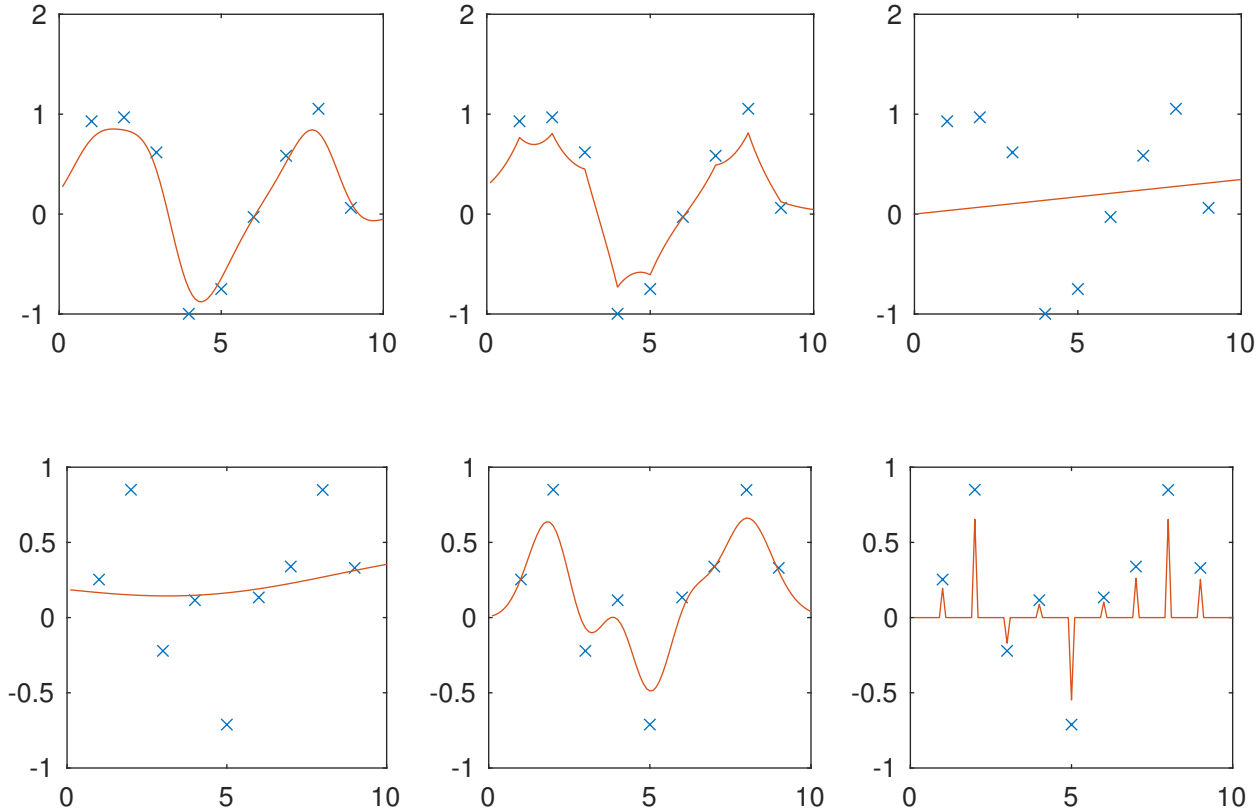
- Do we know good features?
- Do we care about uncertainty?
- How much data (in relation to number of features)?

Choosing between linear methods

	Few data, or care about uncertainty	Plenty of data
Know good features	Bayesian linear regression	Linear regression or ridge regression
Don’t know good features, or #features > #datapoints	Gaussian process regression	Kernel ridge regression

Hyperparameter optimisation for GPR & BLR

What is a good model? Many hyperparameters / kernels possible



Hyperparameter optimisation for GPR & BLR

What is a good model?
(Model = choice of features, kernel, hyperparameters)

When we make point estimates (select best parameters), we can select hyperparameters to maximise MSE on validation set

What is a good way to judge the 'goodness' of hyperparameters in Gaussian process regression and Bayesian linear regression?

- How to judge whether the predicted amount of 'uncertainty' is good?
- Should we only look at the mean function (best **w**) or can we use all predictions (posterior distribution of **w**)?

- We will look at one answer to these questions

Hyperparameter optimisation for GPR & BLR

What is a good model?

In Gaussian process regression and Bayesian linear regression, the model specifies a **prior** generative process over datapoints

Bayesian linear regression:

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma)$$

$$\mathbf{y} \sim \mathcal{N}(\mathbf{w}^T\mathbf{x}, \beta^{-1}\mathbf{I})$$

Gaussian process regression:

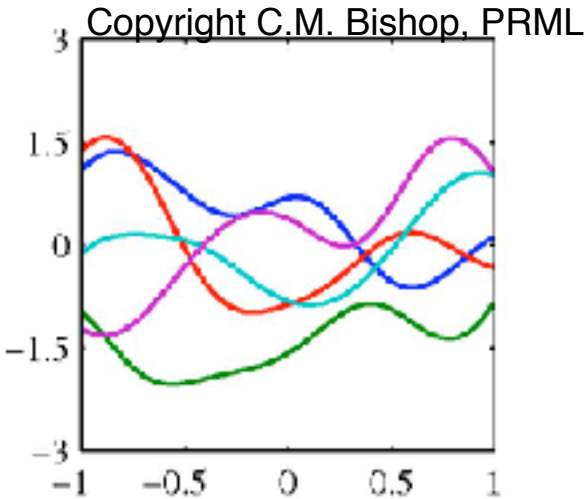
$$\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$$

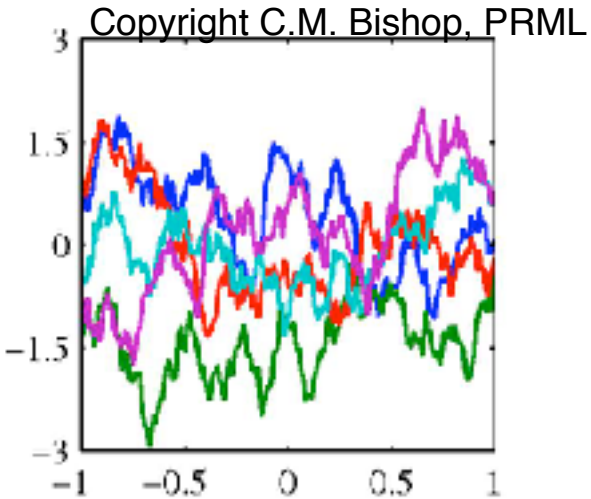
$$\mathbf{t} \sim \mathcal{N}(\mathbf{y}, \beta^{-1}\mathbf{I})$$

We can use this model to 'hallucinate' or guess what the data might look like (before seeing the actual data)

Hyperparameter optimisation for GPR & BLR

We can use generative models to 'hallucinate' or guess what the data might look like (before seeing the actual data)





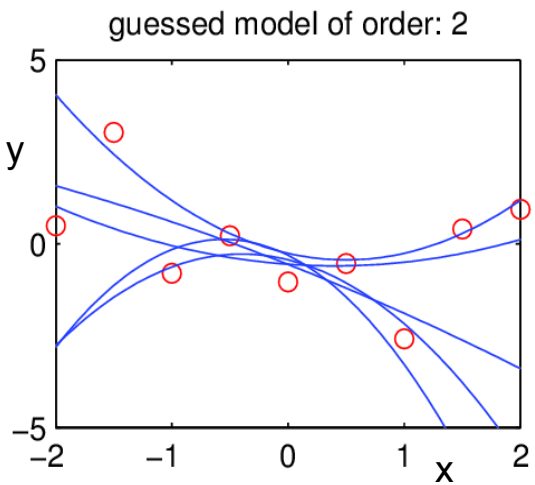
$$k(x, x') = \exp -\|x - x'\|^2$$

$$k(x, x') = \exp -|x - x'|$$

If we have chosen a good prior (good kernel & hyperparameters), the guesses will look similar to the real dataset

Hyperparameter optimisation for GPR & BLR

- If chosen a good prior (good kernel & hyperparameters), the guesses will look similar to the real dataset
- Can compare random guesses (random **w** from prior) to dataset:



$$y = w_0 + w_1 x + w_2 x^2$$

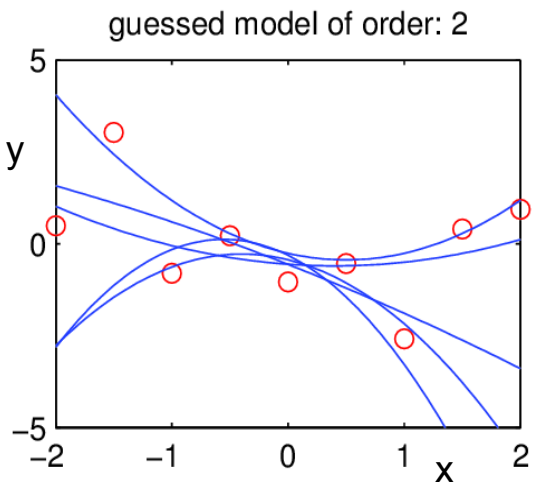
1) Sample random **w** from prior

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma)$$

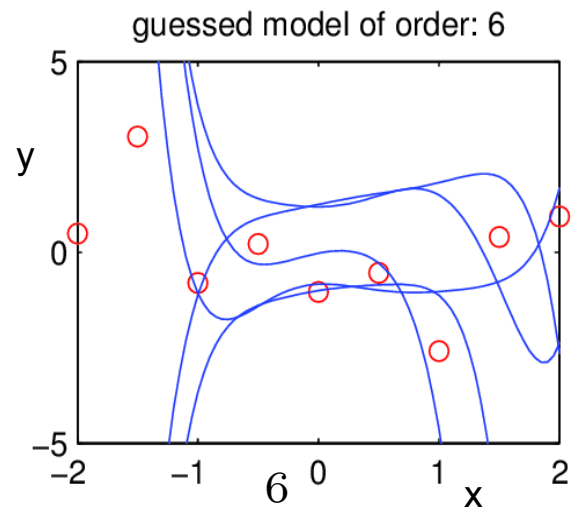
2) plot mean prediction

Hyperparameter optimisation for GPR & BLR

- If chosen a good prior (good kernel & hyperparameters), the guesses will look similar to the real dataset
- Can compare random guesses (random **w** from prior) to dataset:



$$y = w_0 + w_1 x + w_2 x^2$$



$$y = \sum_{i=0}^{6} w_i x^i$$

Hyperparameter optimisation for GPR & BLR

- Compare random guesses to dataset
- We want guesses to be good **on average**
  - What makes a guess **'good'**?

    The dataset should be likely for this guess: take likelihood

    $$p(y|\mathbf{x}, \mathbf{w}) = \mathcal{N}(\mathbf{w}^T\mathbf{x}, \beta^{-1})$$

  - How do we take the **average?**

    Consider all possible values **w** according to prior

    $$p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$$

Hyperparameter optimisation for GPR & BLR

- Compare random guesses to dataset
- We want guesses to be good **on average**

$$p(y|\mathbf{x}, \mathbf{w}) = \mathcal{N}(\mathbf{w}^T\mathbf{x}, \beta^{-1})$$

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$$

$$\int p(\mathbf{w})p(y|\mathbf{w})d\mathbf{w}$$

consider all **w**

'goodness' of this value of **w**

'weight' or contribution of this value of **w**

Hyperparameter optimisation for GPR & BLR

- Compare random guesses to dataset
- We want guesses to be good **on average**
- Make dependency on model explicit. Model *M* is defined by: feature choice, kernel choice, hyperparameter.

$$\int p(\mathbf{w}|\mathcal{M})p(y|\mathbf{w},\mathcal{M})d\mathbf{w}$$

Hyperparameter optimisation for GPR & BLR

- Compare random guesses to dataset
- We want guesses to be good **on average**
- Make dependency on model explicit. Model *M* is defined by:

  feature choice, kernel choice, hyperparameter.

$$\int p(\mathbf{w}|\mathcal{M})p(y|\mathbf{w},\mathcal{M})d\mathbf{w} \boxed{= p(y|\mathcal{M})}$$

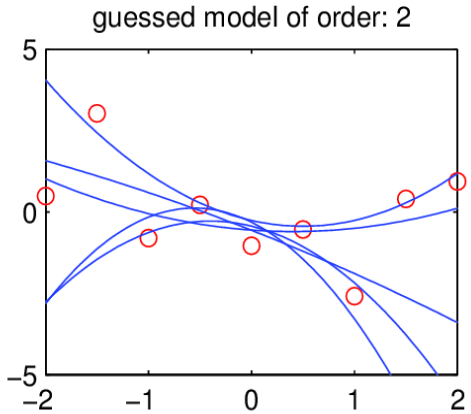
- Likelihood of data after marginalizing model hyperparameters:

  marginal likelihood

Hyperparameter optimisation for GPR & BLR

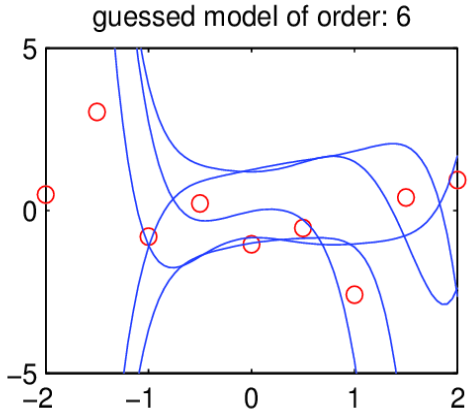
- Let's consider this example one more time

Marginal likelihood on train set: relatively high



$$y = w_0 + w_1 x + w_2 x^2$$

Marginal likelihood on train set: relatively low

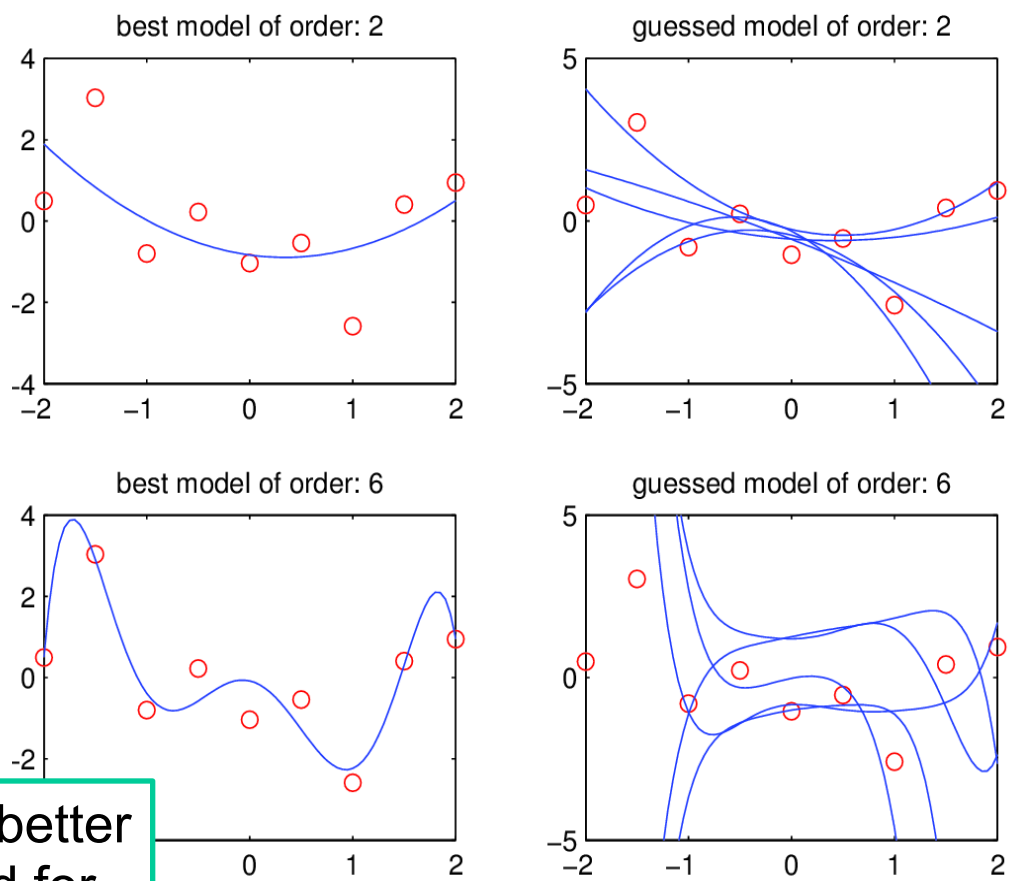


$$y = \sum_{i=0}^{6} w_i x^i$$

Hyperparameter optimisation for GPR & BLR

- Compare to MSE of best **w** on train set:

order 2: better marginal likelihood



$$y = w_0 + w_1 x + w_2 x^2$$

$$y = \sum_{i=0}^{6} w_i x^i$$

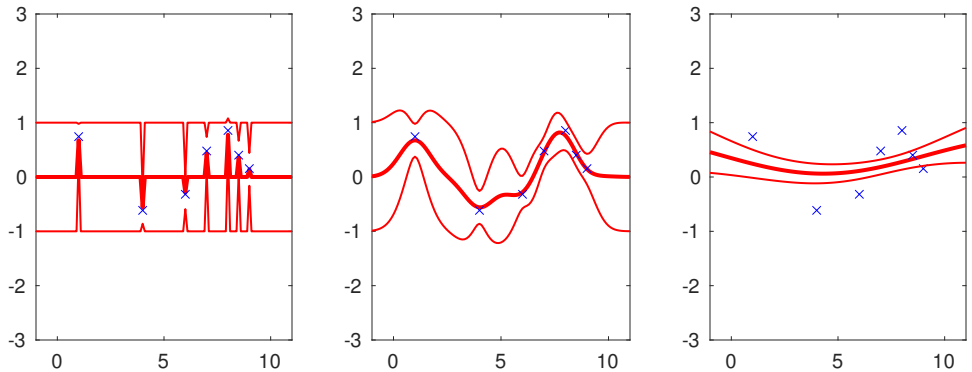
order 6: better likelihood for best **w**

Hyperparameter optimisation for GPR & BLR

- Example was for selecting the feature representation
- More general: maximise train-set likelihood of **best solution**

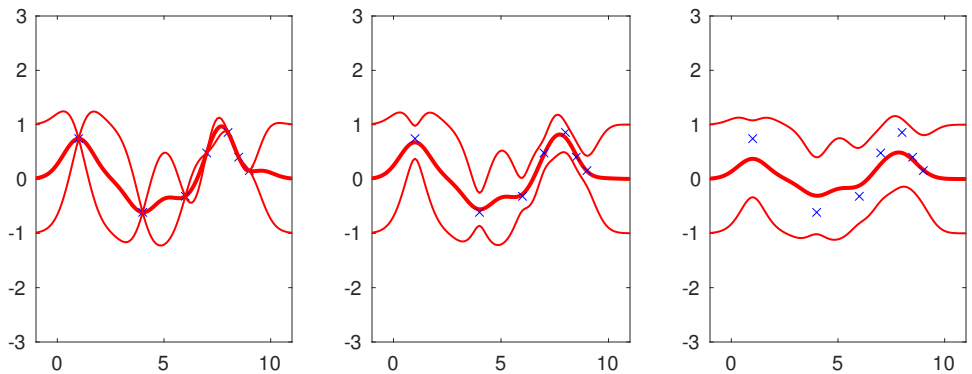
  leads to complex solution that overfit

Hyperparameter optimisation for GPR & BLR



small lengthscale overfit

large lengthscale overfit



assume low noise, overfit

assume high noise, overfit

Hyperparameter optimisation for GPR & BLR

- Example was for selecting the feature representation
- More general: maximise train-set likelihood of **best solution** leads to complex solution that overfit
  - More features, small assumed noise, narrow lengthscale,
  - Best solution fits data well, but does not generalise
  - Typical overfitting. One solution: cross validation

Hyperparameter optimisation for GPR & BLR

- Example was for selecting the feature representation
- More general: maximise train-set likelihood of **best solution** leads to complex solution that overfit
- Maximise train-set likelihood **averaged over all possible solutions (marginal likelihood)**, inherently discourages complexity
  - Complex models have a larger variance. Thus, some solutions will fit the data well and others terribly.
  - Holds as long as the number of hyperparameters is reasonably small in relation to number of datapoints

Hyperparameter optimisation for GPR & BLR

- How to calculate the maximum marginal likelihood?
- Take BLR model (GPR is analogous)

$$\int p(\mathbf{w}|\mathcal{M})p(y|\mathbf{w},\mathcal{M})d\mathbf{w} = p(y|\mathcal{M})$$

- Marginal likelihood is predictive distribution for empty dataset - we know how to evaluate it from lecture 20!

$$p(\mathbf{y}|\mathcal{M}) = \mathcal{N}(\mathbf{0}, \underbrace{\beta^{-1}\mathbf{I} + \mathbf{x}^T\Sigma\mathbf{x}}_{\mathbf{C}_N}) = \frac{1}{\sqrt{|2\pi\mathbf{C}_N|}} \exp -\frac{1}{2}\mathbf{y}^T\mathbf{C}_N^{-1}\mathbf{y}$$

- Easier to maximise the log:

$$\log p(\mathbf{y}|\mathcal{M}) = -\frac{1}{2}\log|\mathbf{C}_N| - \frac{1}{2}\mathbf{y}^T\mathbf{C}_N^{-1}\mathbf{y} - \frac{N}{2}\log(2\pi)$$

Hyperparameter optimisation for GPR & BLR

- Marginal likelihood can be used as objective to select e.g. between different kernels, or different numbers of features to include (e.g. order of polynomial)
  - In this case, can simply take the model with the best marginal likelihood
- It can also be used to optimise continuous parameters:
  - the kernel hyperparameters (for GPR)
  - possibly hyperparameters for features
  - the noise term β^{-1}

Hyperparameter optimisation for GPR & BLR

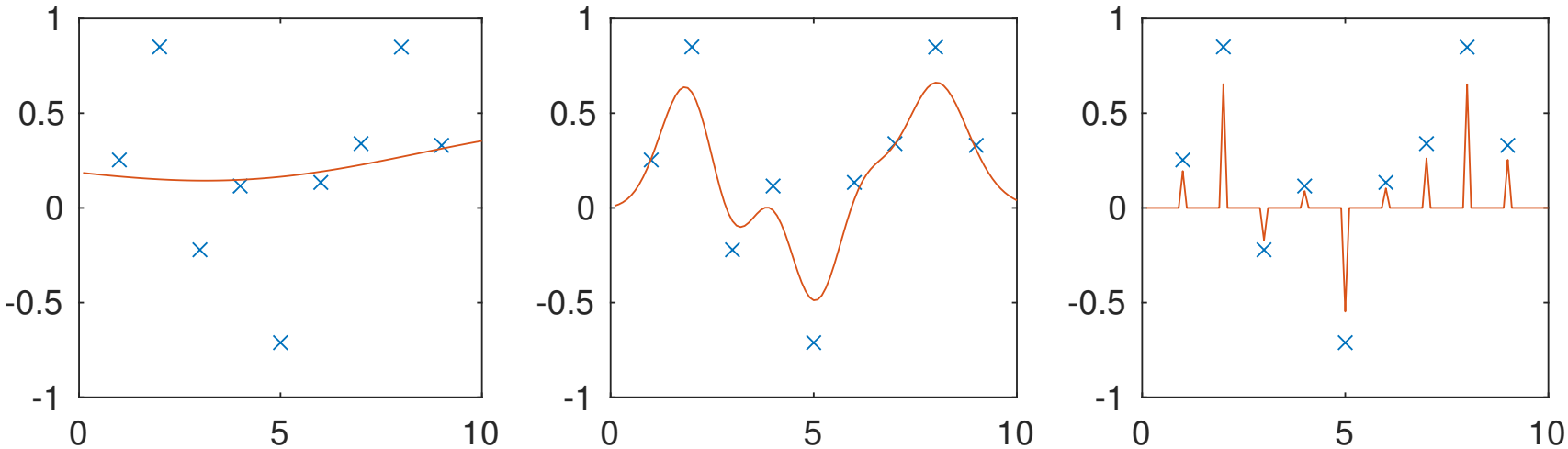
- How to maximise the marginal likelihood?

$$\log p(\mathbf{y}|\mathcal{M}) = -\frac{1}{2}\log|\mathbf{C}_N| - \frac{1}{2}\mathbf{y}^T\mathbf{C}_N^{-1}\mathbf{t} - \frac{N}{2}\log(2\pi)$$

- Can find a maximum using gradient ascent
- Local optima possible!
- If we have many hyperparameters, might still overfit
  - In that case, marginal likelihood can be calculated on validation set or e.g. LOOCV
  - There is an efficient way to calculate LOOCV

Hyperparameter optimisation for GPR & BLR

- Example: **automatic relevance determination** (ARD)
- Length scale tells how fast we expect the GP to change



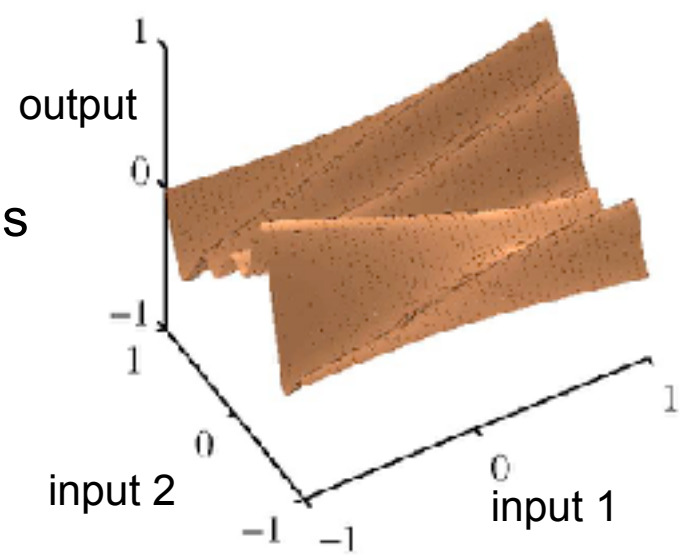
$\sigma = 10$ $\sigma = 1$ $\sigma = 0.1$

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

Hyperparameter optimisation for GPR & BLR

- Example: **automatic relevance determination** (ARD)
- Length scale tells how fast we expect the GP to change
- High length scale: knowing this feature exactly might not be so important. Low **relevance.**
- Can use a different length scale for each input dimension
- Optimisation shows which dimensions have high relevance and which low (inverse of length scale)

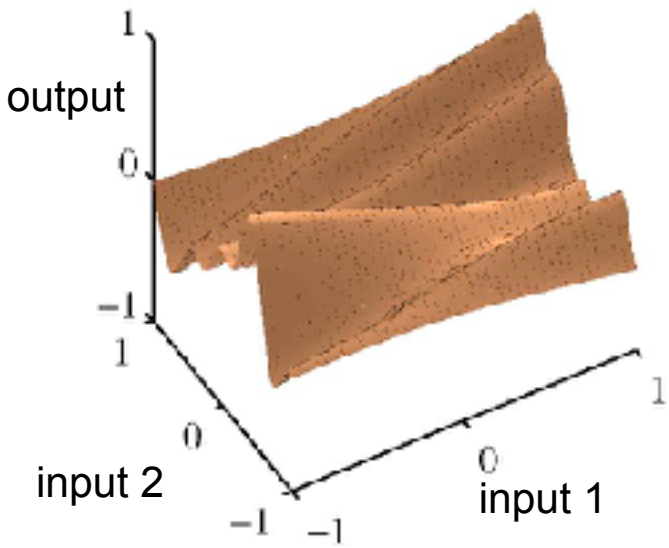
Copyright C.M. Bishop, PRML



Hyperparameter optimisation for GPR & BLR

- E.g. for data from function shown, optimisation would yield:
  - High relevance (short lengthscale) for input 2
  - Low relevance (long lengthscale) for input 1
- Using these relevance values usually leads to better predictions and generalisation

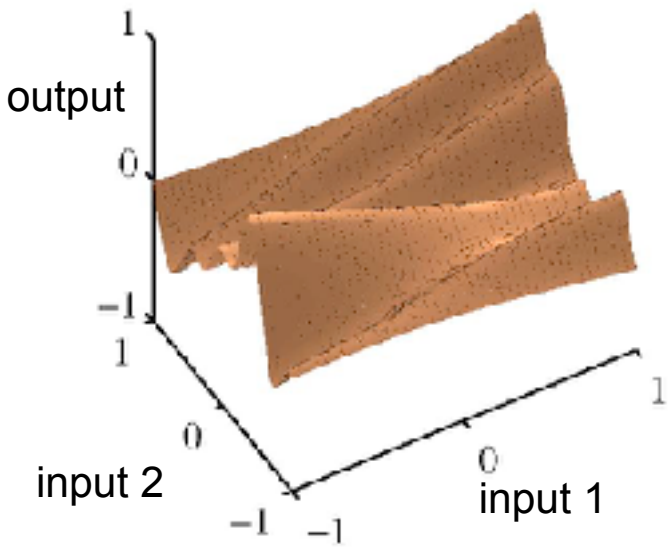
Copyright C.M. Bishop, PRML



Hyperparameter optimisation for GPR & BLR

- Any questions about hyperparameter optimisation for Gaussian process regression and Bayesian linear regression?

Copyright C.M. Bishop, PRML



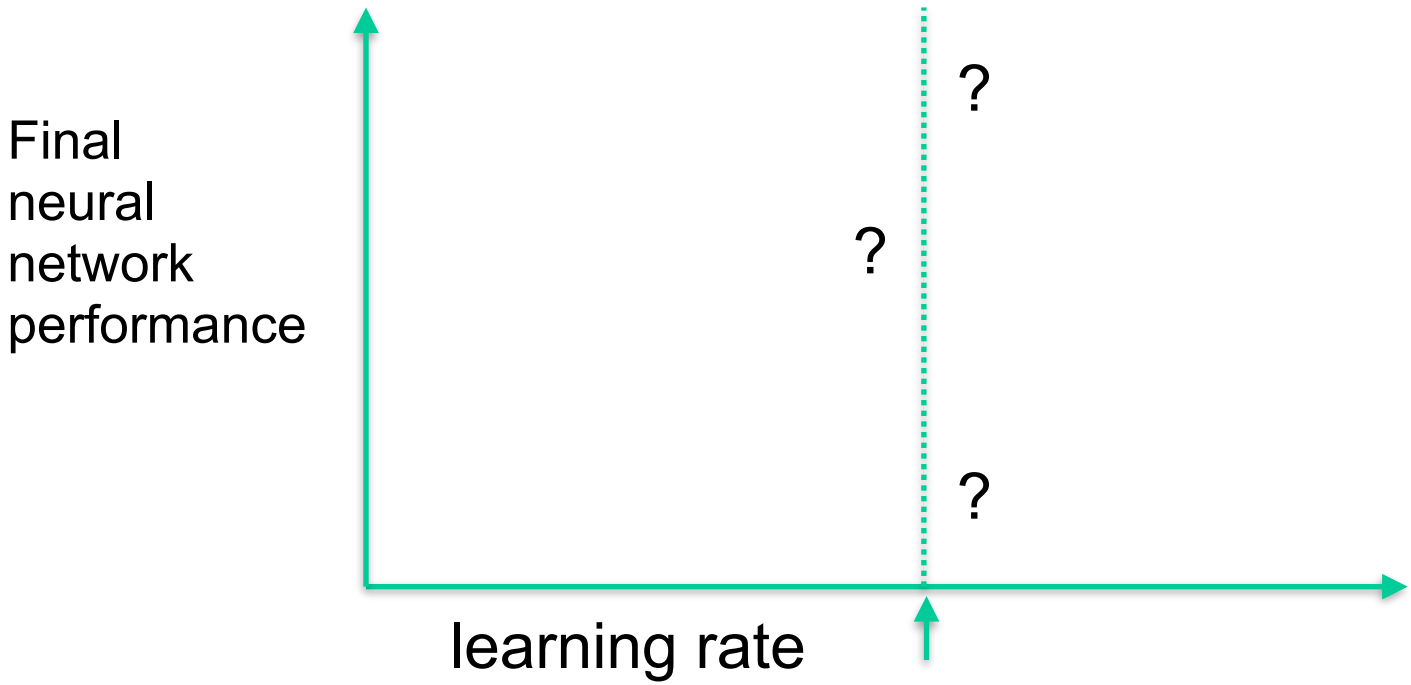
Optimisation of unknown function

Optimisation of unknown function

- Problem: find value x for which function f(x) is maximal
  - f(x) is a ‘black box’ function: we only know the value f(x) for small set of points x that we evaluate
  - evaluating f(x) is relatively expensive
    - Say, x describes advertising strategy and f(x) #sales
    - x NN hyperparameters, f(x) cross-validation performance
    - x describes vehicle design, f(x) result of evaluation in time-consuming simulation

Optimisation of unknown function

- Problem: find value x for which function f(x) is maximal
- Example of black box function



Optimisation of unknown function

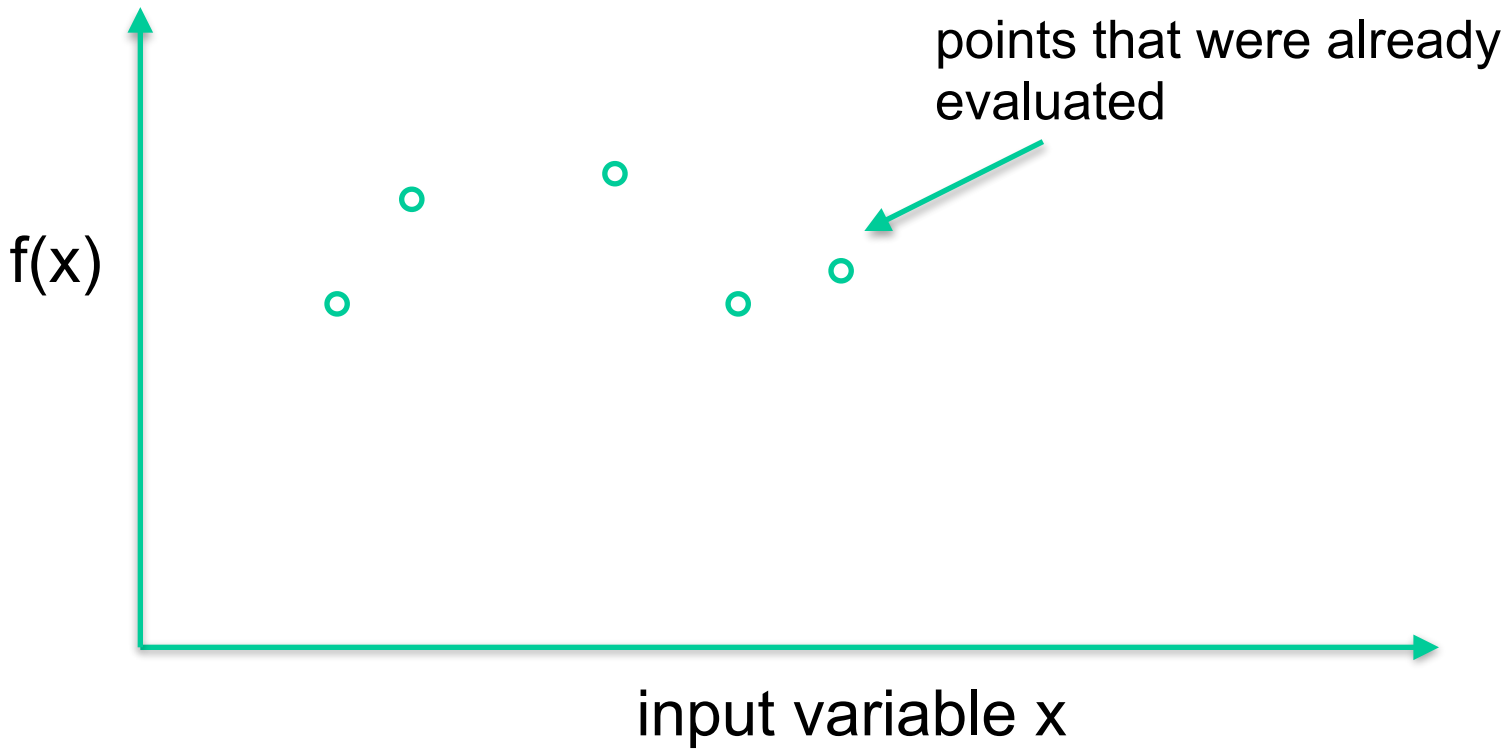
- Problem: find value x for which function f(x) is maximal
  - f(x) is a ‘black box’ function: we only know the value f(x) for small set of points x that we evaluate
  - evaluating f(x) is relatively expensive
  - f(x) might have local optima
  - derivatives might not be known
- How can we approach this problem?

Optimisation of unknown function

- So far, we have mainly done gradient ascent
- but gradient ascent might need many function evaluations (costly), can get stuck in local minima, requires an estimate of the gradient
- Can we do better?

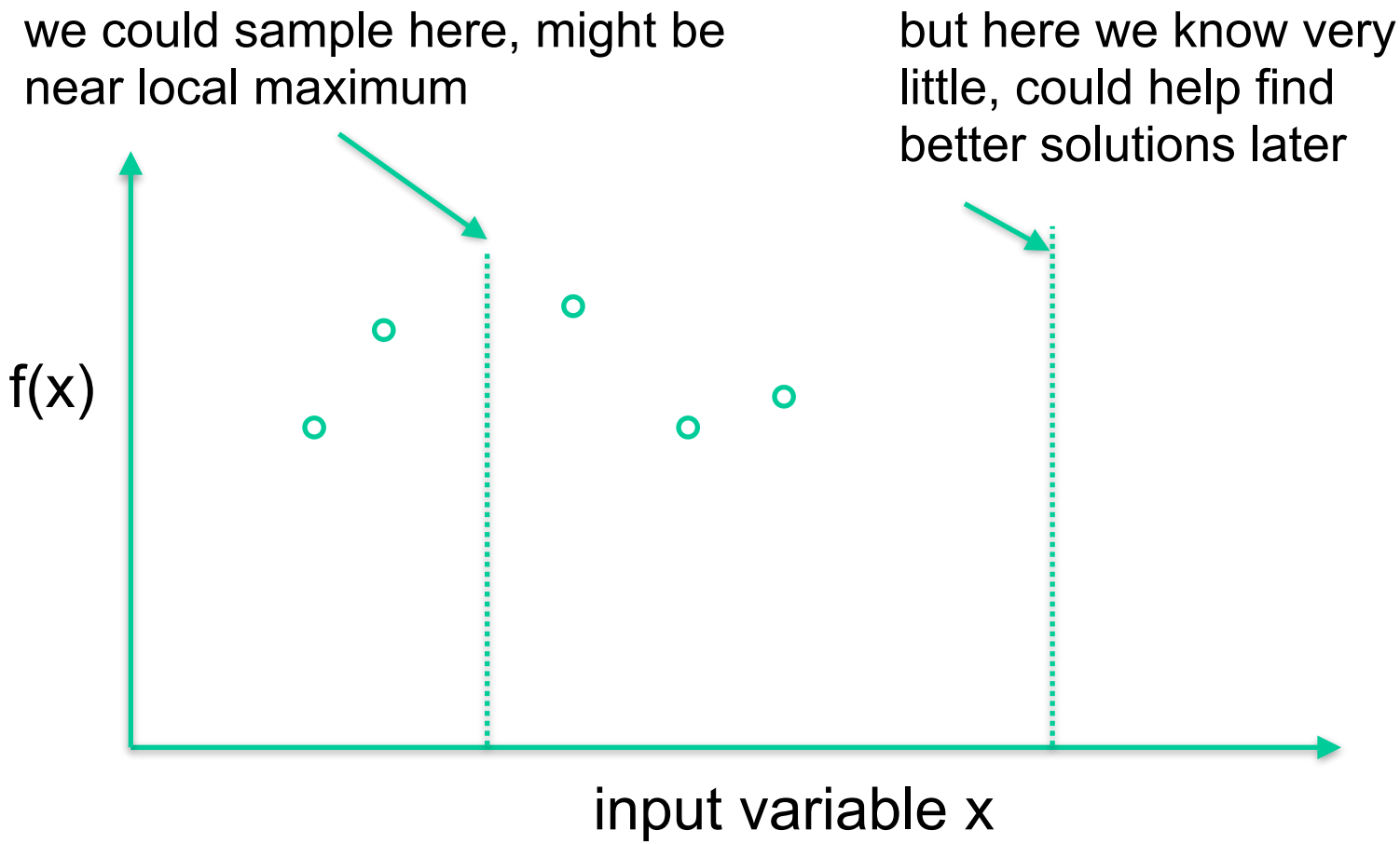
Optimisation of unknown function

- How might a problem look like?
- Where to sample next, if we have a budget for, say, 10 samples?



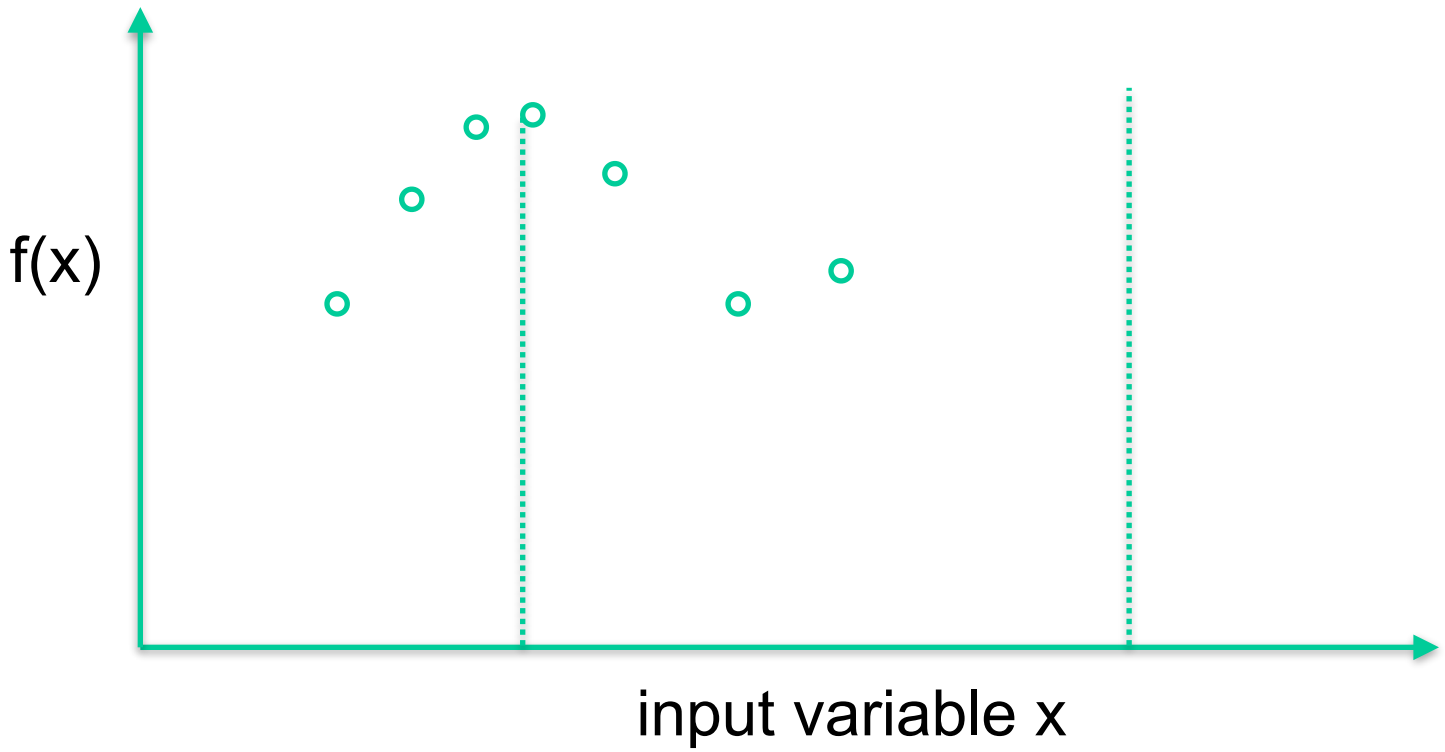
Optimisation of unknown function

- How might a problem look like?



Optimisation of unknown function

- How might a problem look like?
- How about now?

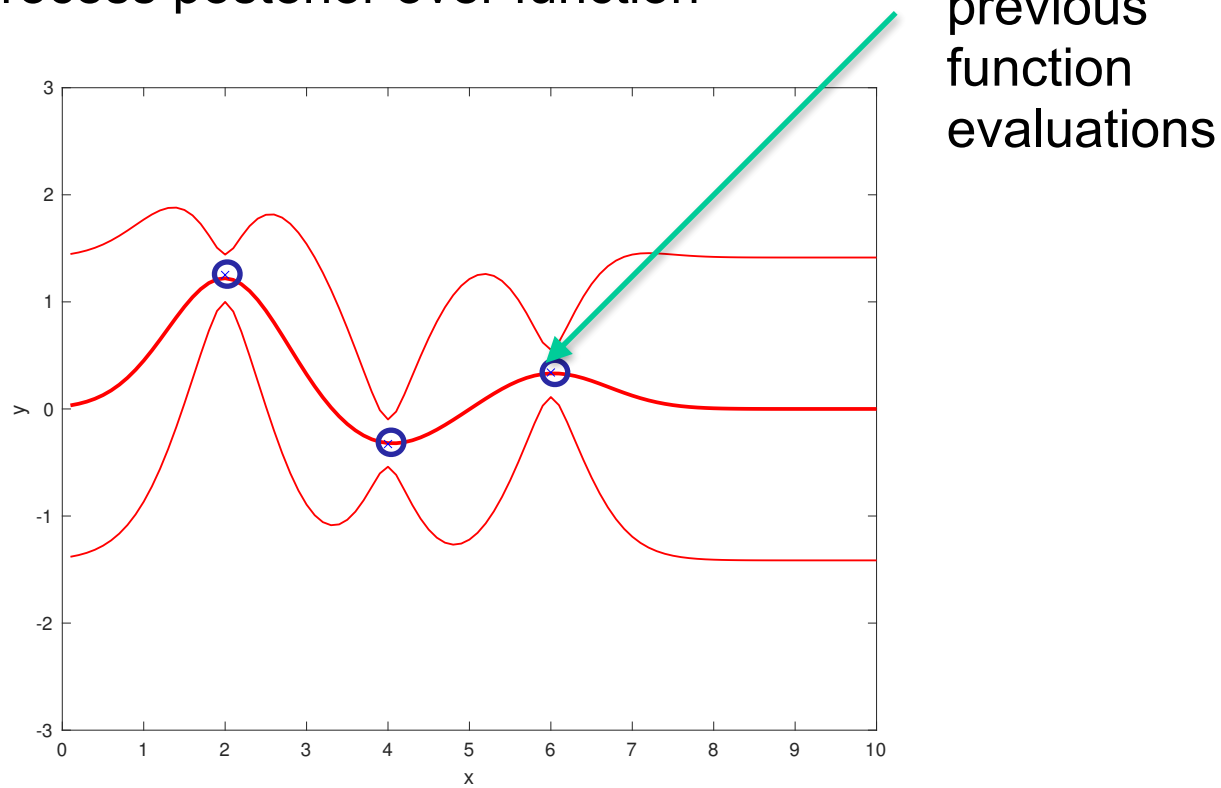


Optimisation of unknown function

- Idea: to make a good decision we should imagine what the whole function should look like
- It seems important to take into account how certain we are for various input values **x**
- A **Gaussian process** might do the job here!
- This implies Bayesian point of view: **Bayesian optimisation**

Bayesian optimisation

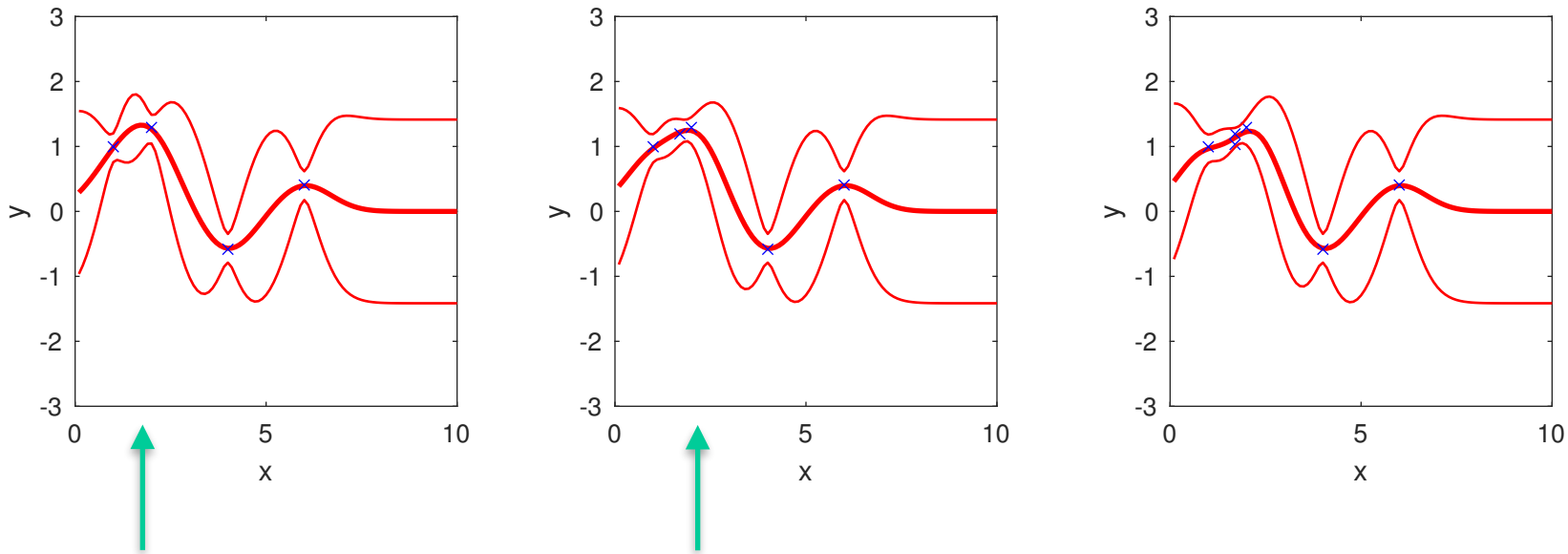
- Gaussian process posterior over function



- Where to sample next?

Bayesian optimisation

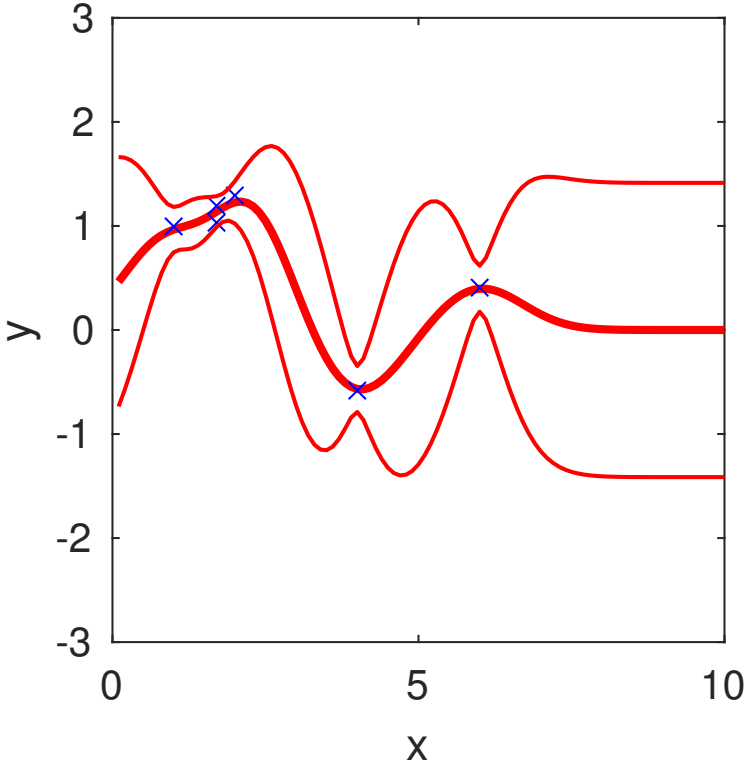
- Where to sample next?
- What happens if we simply sample where mean is highest?

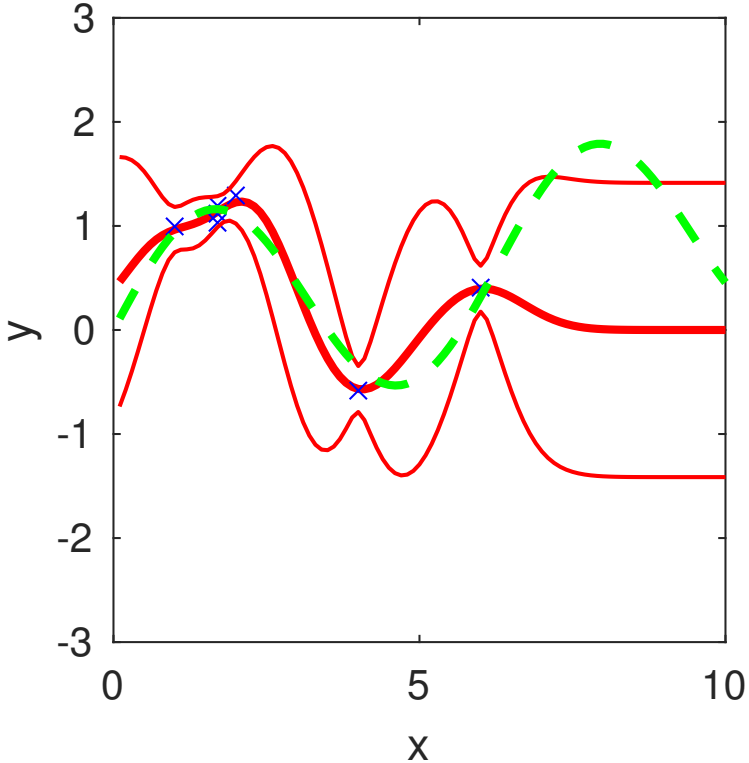


sample location

Bayesian optimisation

- We don't sample on the right at all!
- We might miss the real maximum





Bayesian optimisation

- Where to sample next?
- Two objectives:
  - **Exploitation**: sample where we think high values are

    If we know the samples will be low, it does not make sense to sample there

    Maybe: sample highest Gaussian process mean?
  - **Exploration:** If we always sample where we think the highest value is, we might miss other values

    Maybe: sample where uncertainty is highest?

Bayesian optimisation

- Several strategies exist for combining these two objectives
- Strategies for ‘rating’ new sample: **acquisition function**
- Very straightforward: upper confidence bound

$$a_{\mathrm{UCB}}(\mathbf{x}^*; \mathcal{D}) = \mu(\mathbf{x}^*; \mathcal{D}) + \kappa\sigma(\mathbf{x}^*; \mathcal{D})$$

predicted mean given data so far

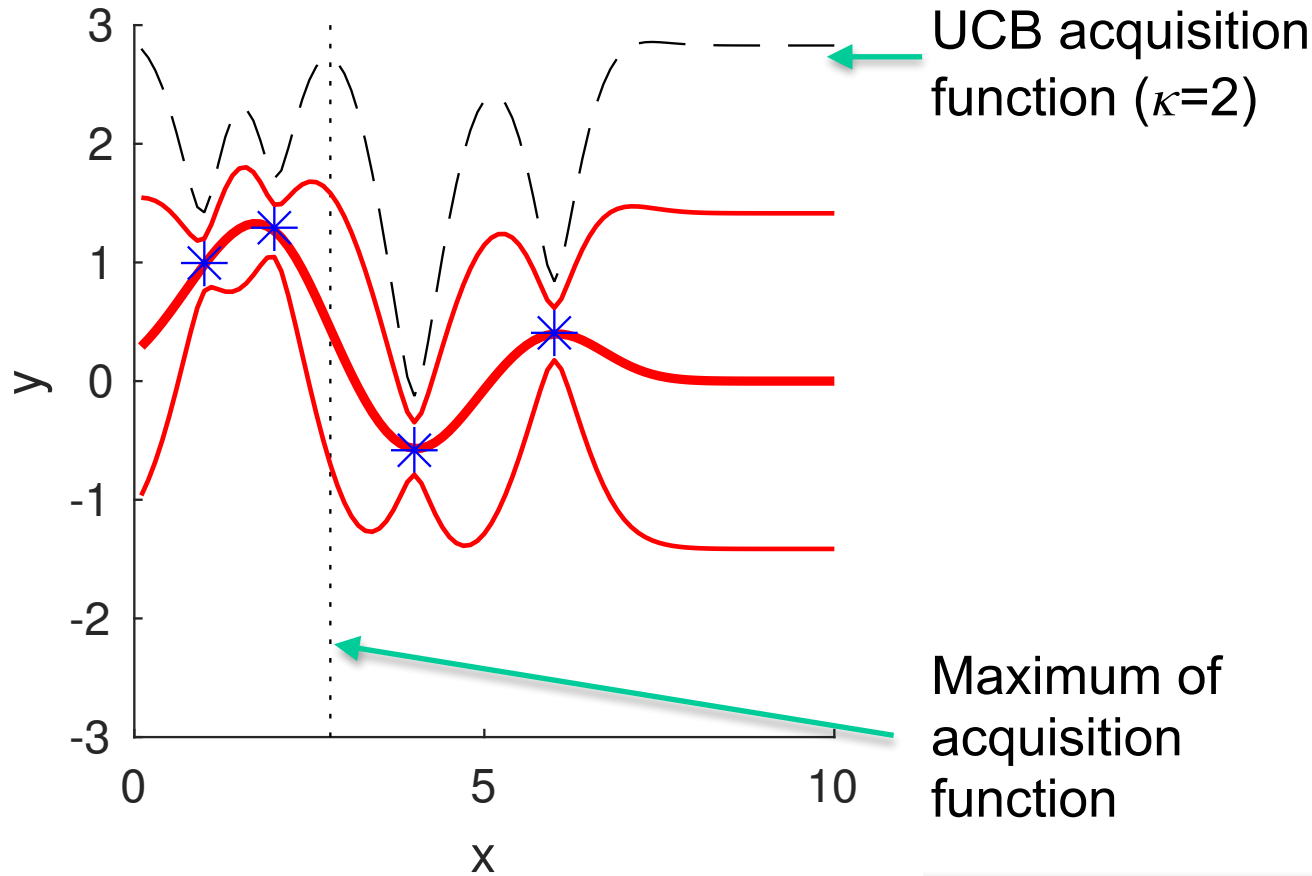
trade-off parameter

predicted standard deviation given data so far

- Acquisition functions gives a ‘score’ to each sample point
- Upper confidence bound has good theoretical properties

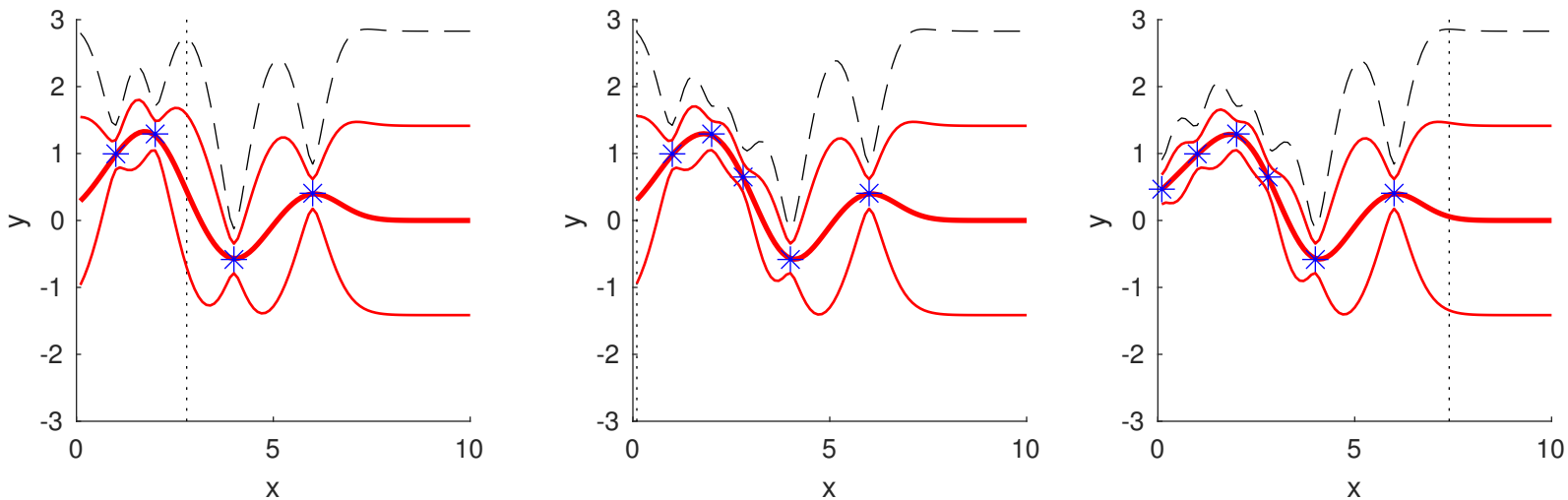
Bayesian optimisation

- Upper confidence bound acquisition function



Bayesian optimisation

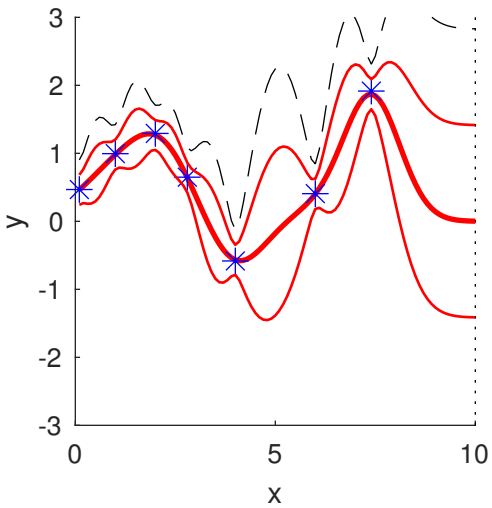
- Upper confidence bound acquisition function



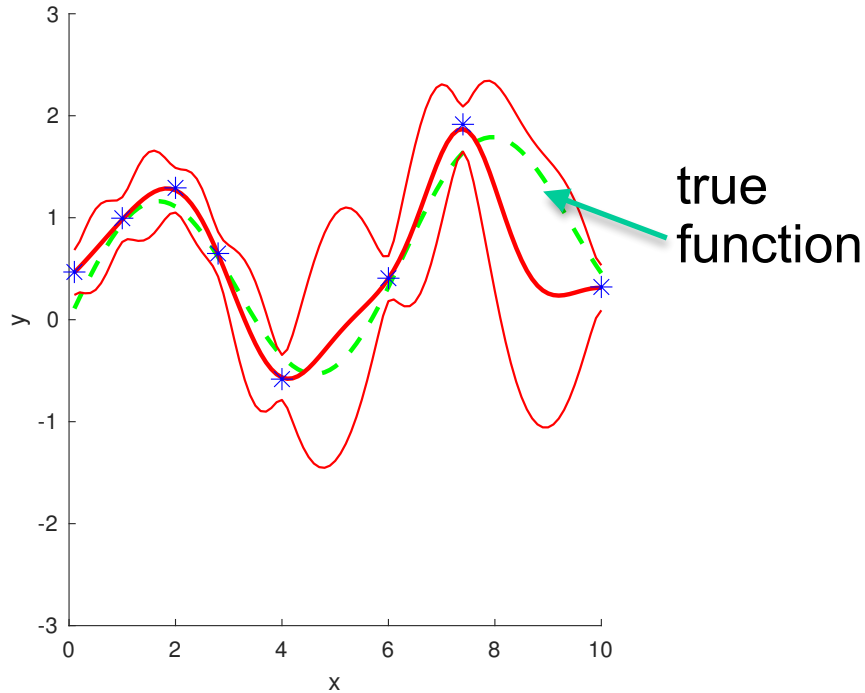
first sample second sample third sample

Bayesian optimisation

- We now explore sufficiently well go get close to the maximum



fourth sample



fifth sample,
comparison to true function

Bayesian optimisation

- Different acquisition functions exist:
  - Probability of improvement
    - Probability sampled value > current maximum?
    - Sometimes too greedy
  - Expected improvement
    - Weights probability with amount of improvement
    - Can be overly greedy
  - Upper confidence bound
    - Strong theoretical properties
    - Need to set tuning parameter κ

Bayesian optimisation

- Pros
  - Attempt at global optimisation
  - Need relatively few samples to get close to optimum
  - Software packages available
- Cons
  - Computational expensive
    - Need to fit a GP and hyperparameters in every iteration
    - Need to maximise non-convex acquisition function
  - Sensitive to choice of model (kernel, hyperparameters)
  - Only works well with few input (up to ~10 dimensions)

Bayesian hyperparameter optimisation

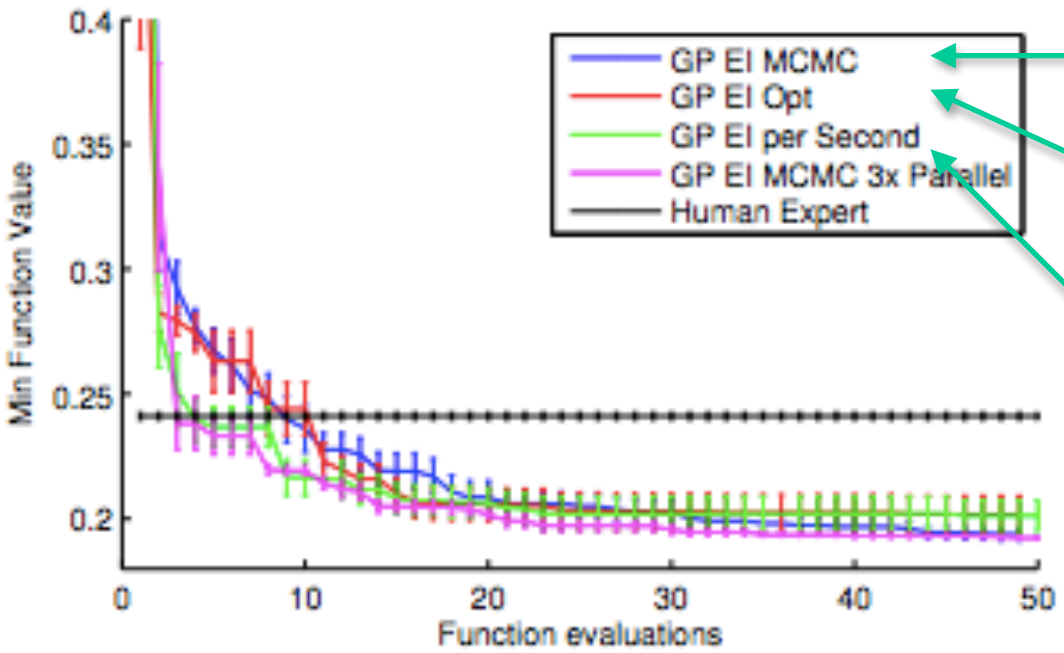
- One application of Bayesian optimisation is hyperparameter optimisation
- Example: Tune learning rate in deep neural net
  - Nonconvex function with local optima
  - Evaluating a learning rate is expensive: we must train the network with that rate to know how good it is

Bayesian hyperparameter optimisation

- One application of Bayesian optimisation is hyperparameter optimisation
- Practical issues [Snoek & Larochelle, 2012]
  - ‘Standard’ Gaussian kernel too smooth
  - Taking only the best hyperparameters ignores some plausible functions: take average over plausible parameters
  - Some hyperparameters are more costly to evaluate than others: e.g. small networks vs large network
    - Use ‘improvement per second’ as acquisition function

Bayesian hyperparameter optimisation

- Results [Snoek & Larochelle, 2012]
- Code available: https://github.com/HIPS/Spearmint



(a)

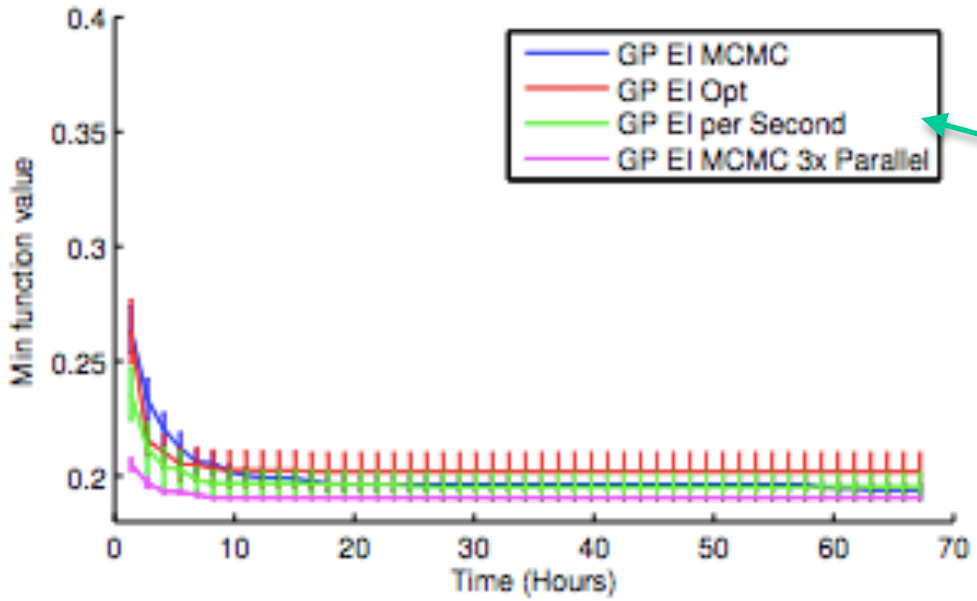
Average over hyperparameters

Optimisation of hyperparameters

Take cost into account

Bayesian hyperparameter optimisation

- Results [Snoek & Larochelle, 2012]
- Code available: https://github.com/HIPS/Spearmint



(b)

Take cost into account

Recap

- Hyperparameter optimisation for GP regression and Bayesian linear regression using **marginal likelihood**
  - Pick model & hyperparameters such that fit is good on average over all parameter vectors
  - Is an **objective function**
- **Bayesian optimisation** for optimising expensive ‘black box’ functions
  - Fit unknown objective function with GPR
  - Can be used for optimising hyperparameters of learners
  - Is an **optimisation method** that needs an objective function

What you should know

- Key idea of hyperparameter optimisation with **marginal likelihood**
- Difference between marginal likelihood and likelihood of best model
- Key idea of **Bayesian optimisation**
- Pros and cons of Bayesian optimisation