COMP 551 – Applied Machine Learning
Lecture 19: Bayesian Linear Regression

Associate Instructor: Ryan Lowe (ryan.lowe@mail.mcgill.ca)

Slides mostly by: Herke van Hoof (herke.vanhoof@mcgill.ca)

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

Announcements

- Assignment 2 grades are online!
  - See TAs if you have questions about your grade
  - Will try to organize 'joint office hour' with all TAs who graded assignment 2 (will be announced)

- Project 4 Kaggle deadline March 21st!
  - **Report only deadline extended 1 day, to March 22nd.**

Announcements

Public Leaderboard Private Leaderboard

This leaderboard is calculated with approximately 30% of the test data.
The final results will be based on the other 70%, so the final standings may be different.

Raw Data Refresh

#	△1w	Team Name	Kernel	Team Members	Score	Entries	Last
1	▲1	Sigma Mu			0.98399	20	18h
2	▲5	haoh.bo			0.98066	9	2d
3	▲11	AI geeks			0.97666	23	1d
4	▼3	KCM			0.97333	7	9d
5	▼1	Team Biceps			0.97299	12	3h
6	▼3	ASDFSWAG			0.96966	8	10d
7	▲4	2 g b			0.96733	11	1d
8	new	spicyAway			0.96733	4	2d
9	▼3	ApplicationMemoryError			0.96733	11	1h
10	▲15	Happy Decision Tree Friends			0.96666	8	4h

Recall: Bayesian terminology

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$$

- **Likelihood** $p(\mathcal{D}|\mathbf{w})$: our model of the data. Given our weights, how do we assign probabilities to dataset examples?
- **Prior** $p(\mathbf{w})$: before we see any data, what do we think about our parameters?
- **Posterior** $p(\mathbf{w}|\mathcal{D})$: our distribution over weights, given the data we've observed *and our prior*
- **Marginal likelihood** $p(\mathcal{D})$: also called the normalization constant. Does not depend on **w**, so not usually calculated explicitly

Recall: Conjugate priors

- A *prior* $p(\mathbf{w})$ is *conjugate* to a *likelihood* function $p(\mathcal{D}|\mathbf{w})$ if **the posterior is in the same family as the prior**
- In other words, if *prior * likelihood* gives you the same form as the prior with different parameters, it's a conjugate prior
  - Ex 1: the Gaussian distribution is a conjugate prior to a Gaussian likelihood
  - Ex 2: the Beta distribution is conjugate to a Bernoulli likelihood
- **Why?** *Want simple form for our posterior!* Don't want it to get more complicated every time you add more data

Bayesian linear regression

- Previous examples (coin flip, learning the mean of a Gaussian) only had outputs *y*, no inputs *x*
- How can we learn to make predictions that are input-dependent?
- Can use an extension of linear regression: **Bayesian linear regression**

Recall: Steps for Bayesian inference

- Given a dataset $\mathcal{D}$, how do we make predictions for a new input?

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$$

- **Step 1**: Define a model that represents your data (the **likelihood**): $p(\mathcal{D}|\mathbf{w})$
- **Step 2:** Define a **prior** over model parameters: $p(\mathbf{w})$
- **Step 3:** Calculate **posterior** using Bayes' rule: $p(\mathbf{w}|\mathcal{D}) = \dfrac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$
- **Step 4:** Make **prediction** by integrating over model parameters:

$$p(y^*|\mathbf{x}^*, \mathcal{D}) = \int_{\mathbb{R}^N} p(\mathbf{w}|\mathcal{D})p(y^*|\mathbf{x}^*, \mathbf{w})d\mathbf{w}$$

Bayesian linear regression

- We take a *specific form of the likelihood and the prior*:
  - Step 1: Likelihood

$$p(y|\mathbf{x}, \mathbf{w}) = \mathcal{N}(\mathbf{w}^T\mathbf{x}, \sigma^2)$$

Output *y* close to learned linear function **w*x** , with some noise

  - Step 2: Conjugate prior

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \alpha^{-1}\mathbf{I})$$

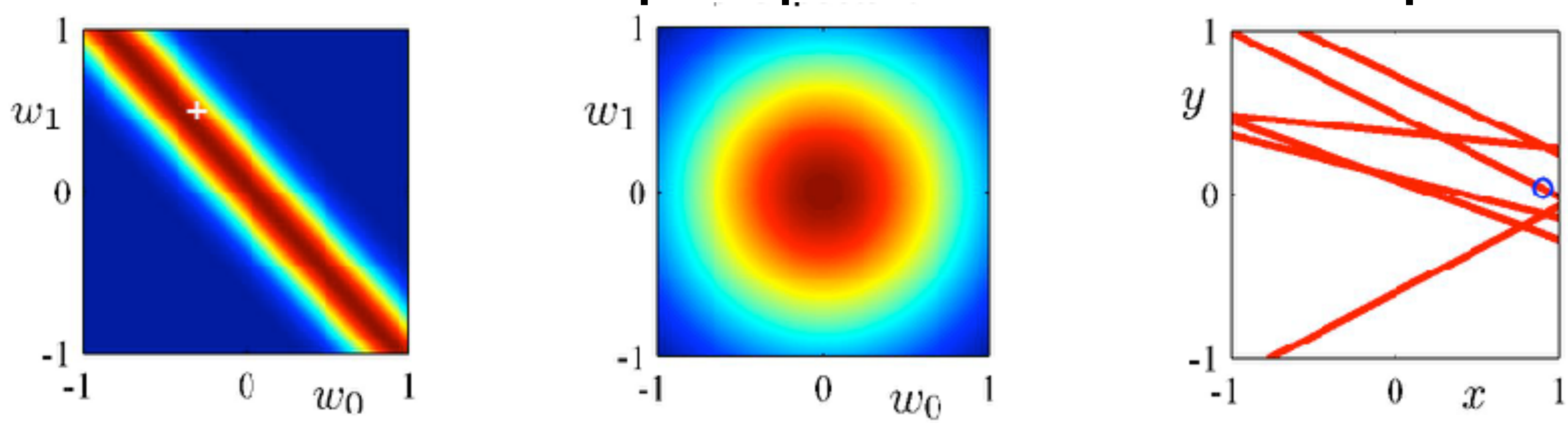
Prefer small weights.
(assuming no other info)

  - Prior precision α and noise variance σ^2 considered known
  - Linear regression where we **learn a distribution over the parameters**

Visualizing inference

- Start with simple example (one feature *x*): $y = w_0 + w_1 x + \epsilon$
- How can we visualize what's happening in Step 3? (finding $p(\mathbf{w}|\mathcal{D})$)

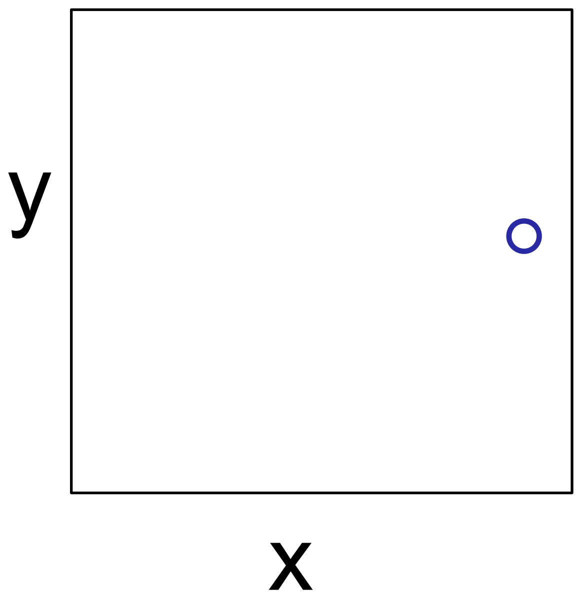
likelihood **prior/ posterior** **data space**



For different w0, w1, how likely is this data point?	How likely are different (w0, w1) given data so far?	Shows data points and sample functions for data so far

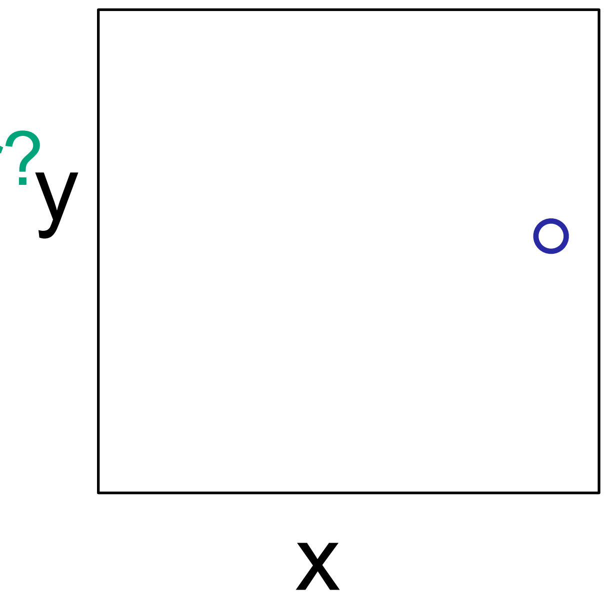
Visualizing inference

- Goal: fit lines $y = w_0 + w_1 x + \epsilon$
- Bayes theorem: $p(\mathbf{w}|\mathcal{D}) = \dfrac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$



Visualizing inference

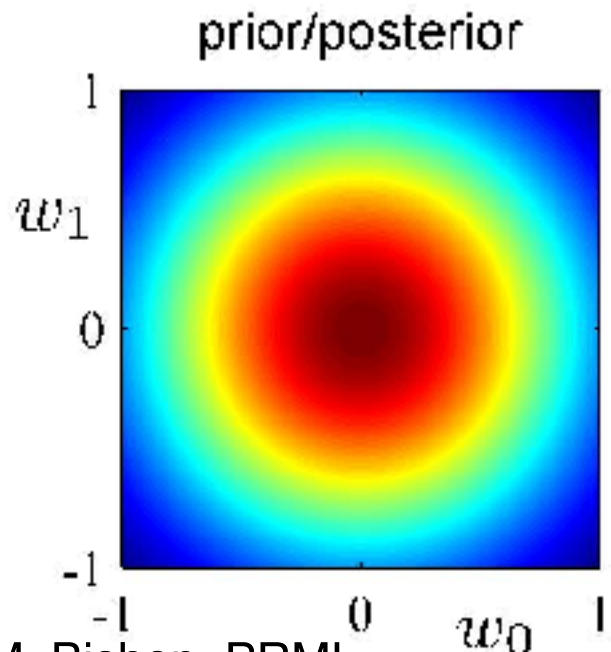
- Goal: fit lines $y = w_0 + w_1 x + \epsilon$



What prior?

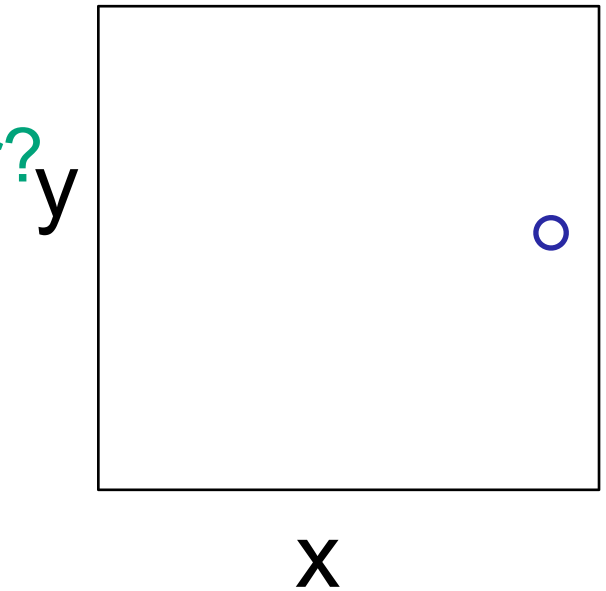
- Bayes theorem: $$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$$

- Similar to ridge regression, expect good **w** to be small



Visualizing inference

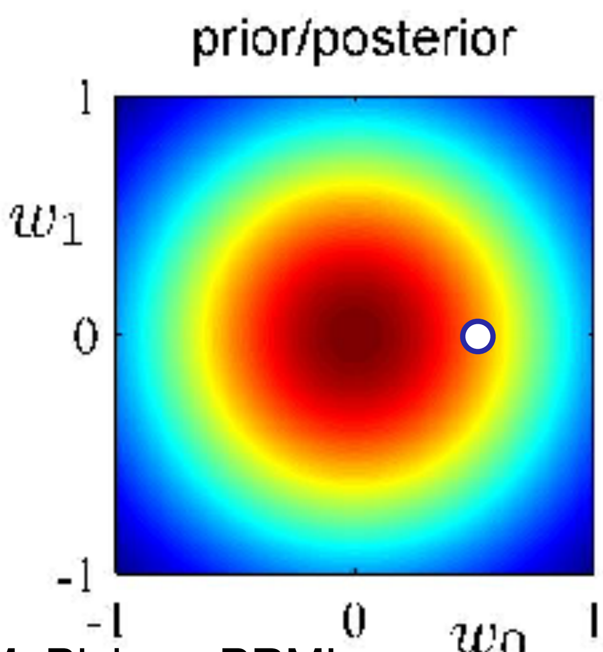
- Goal: fit lines $y = w_0 + w_1 x + \epsilon$

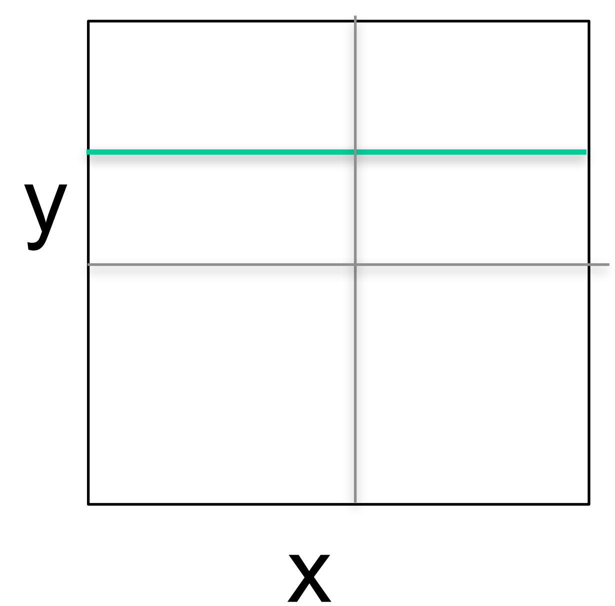


What prior?

- Bayes theorem: $p(\mathbf{w}|\mathcal{D}) = \dfrac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$

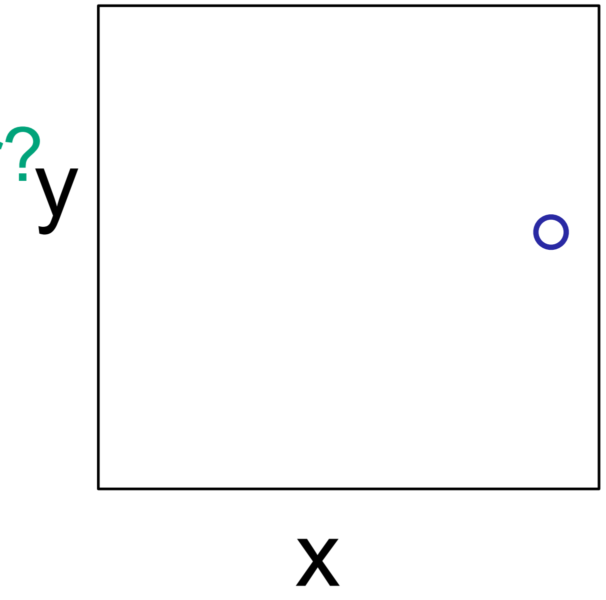
- Similar to ridge regression, expect good **w** to be small





Visualizing inference

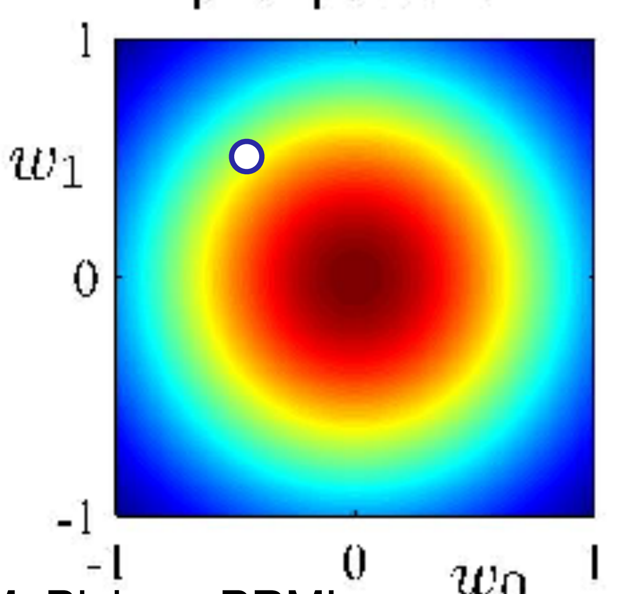
- Goal: fit lines $y = w_0 + w_1 x + \epsilon$

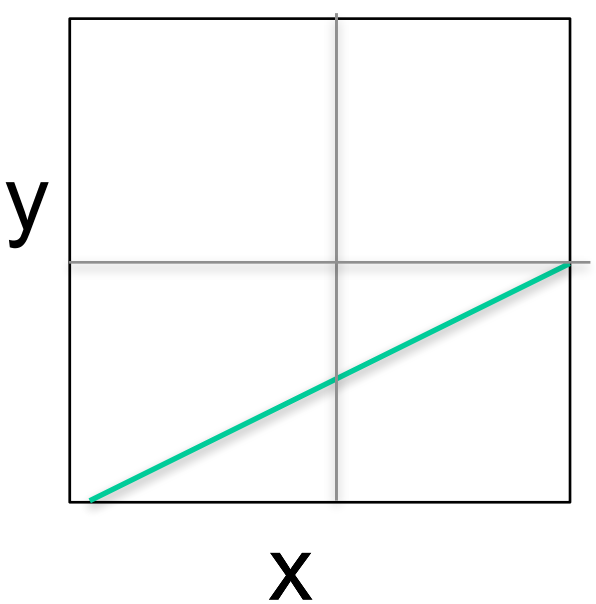


What prior?

- Bayes theorem: $p(\mathbf{w}|\mathcal{D}) = \dfrac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$

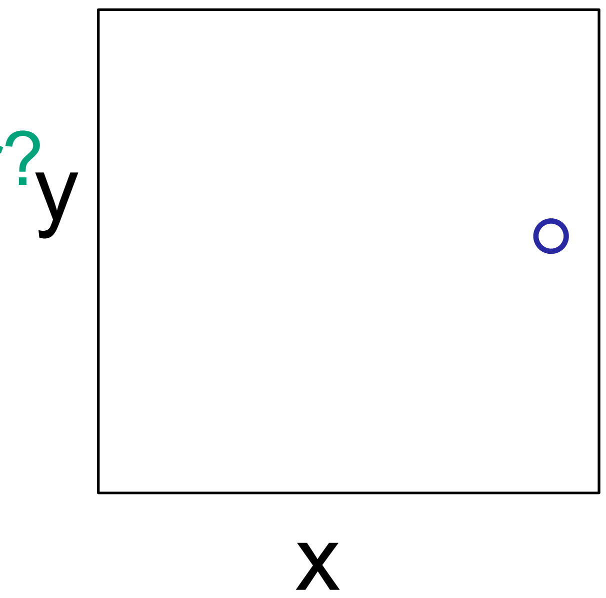
- Similar to ridge regression, expect good **w** to be small





Visualizing inference

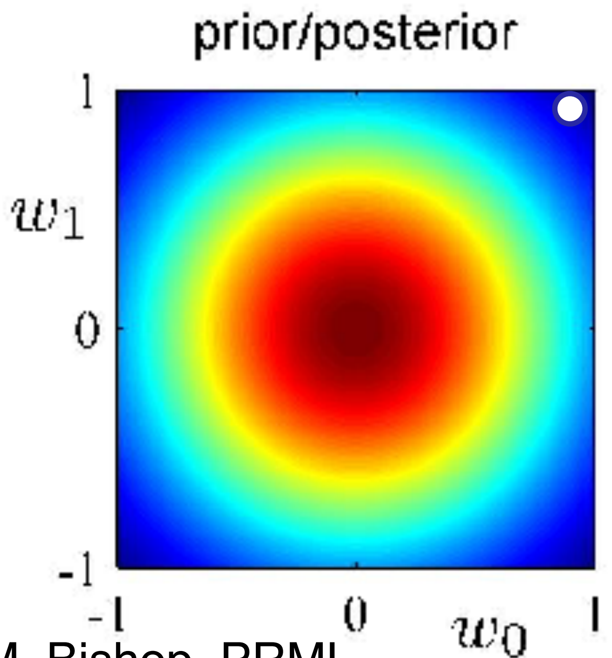
- Goal: fit lines $y = w_0 + w_1 x + \epsilon$

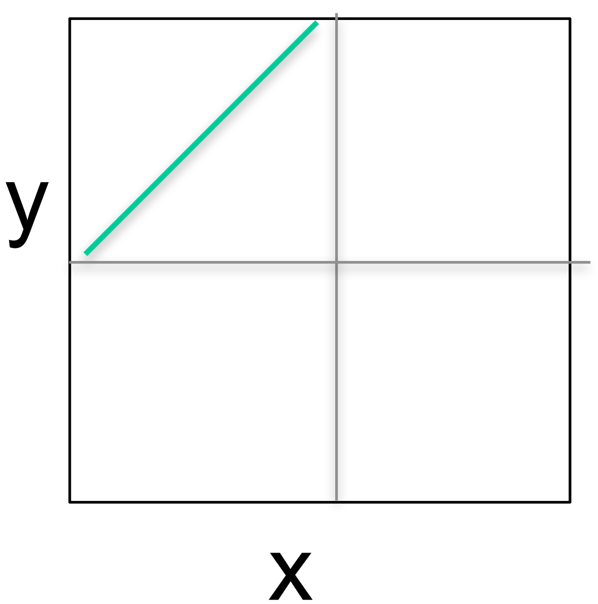


What prior?

- Bayes theorem: $p(\mathbf{w}|\mathcal{D}) = \dfrac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$

- Similar to ridge regression, expect good **w** to be small



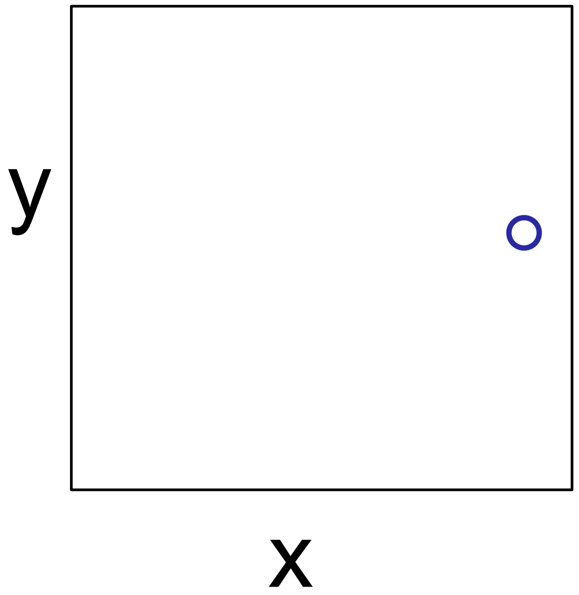


Visualizing inference

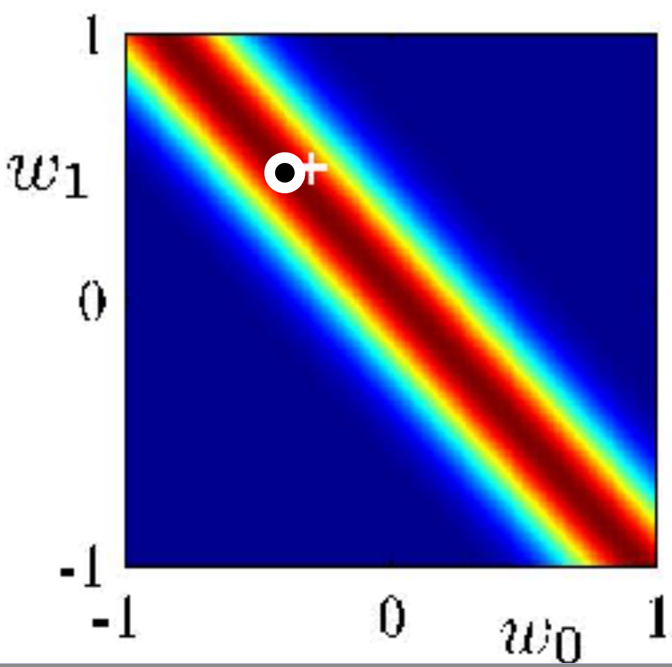
- Goal: fit lines $y = w_0 + w_1 x + \epsilon$

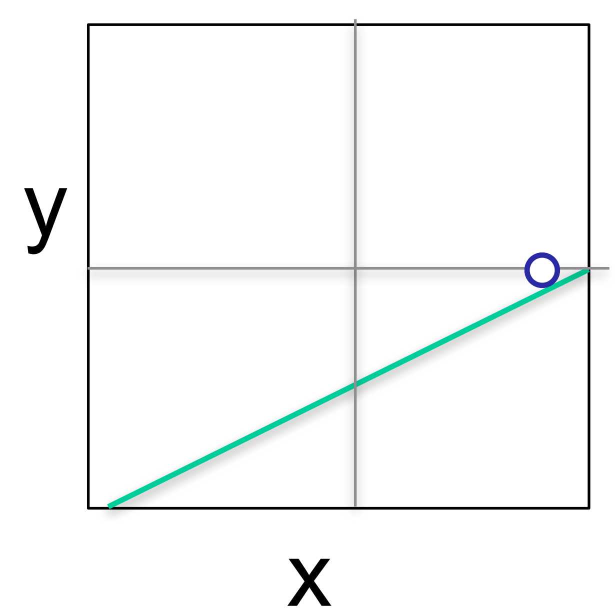
What likelihood?

- Bayes theorem: $p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$



- Good lines should pass ‘close by’ datapoint





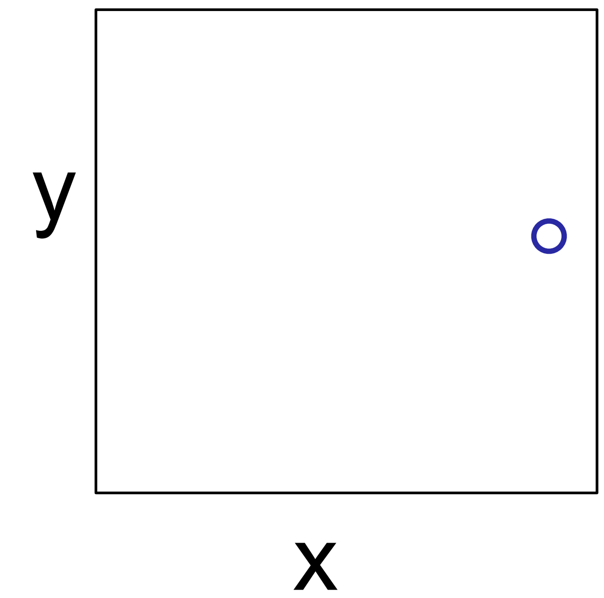
Copyright C.M. Bishop, PRML

Visualizing inference

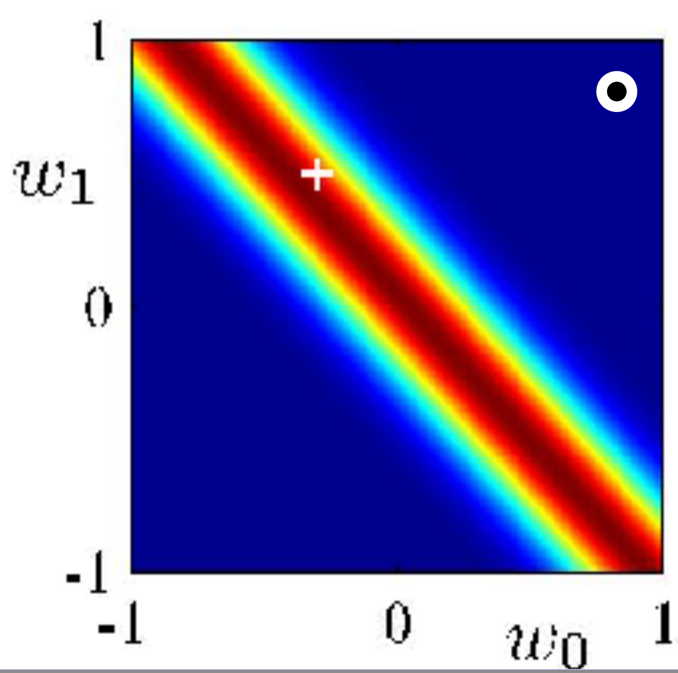
- Goal: fit lines $y = w_0 + w_1 x + \epsilon$

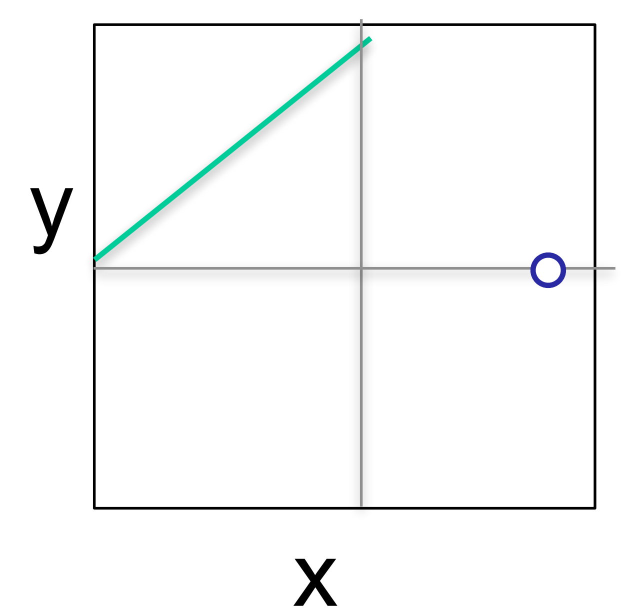
What likelihood?

- Bayes theorem: $p(\mathbf{w}|\mathcal{D}) = \dfrac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$



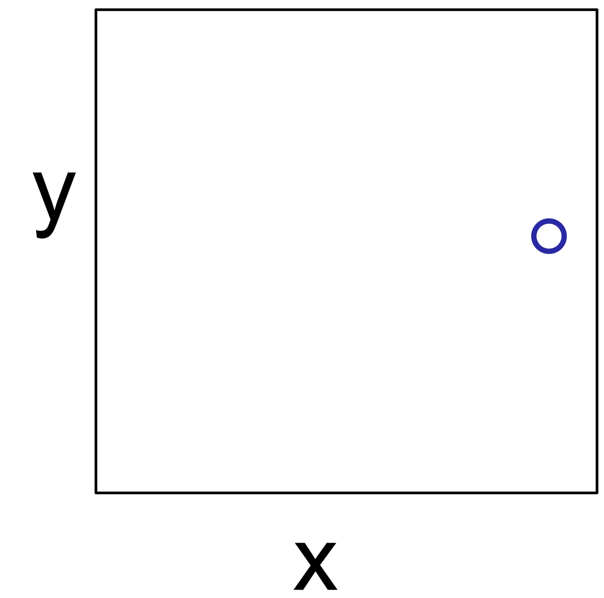
- Good lines should pass ‘close by’ datapoint



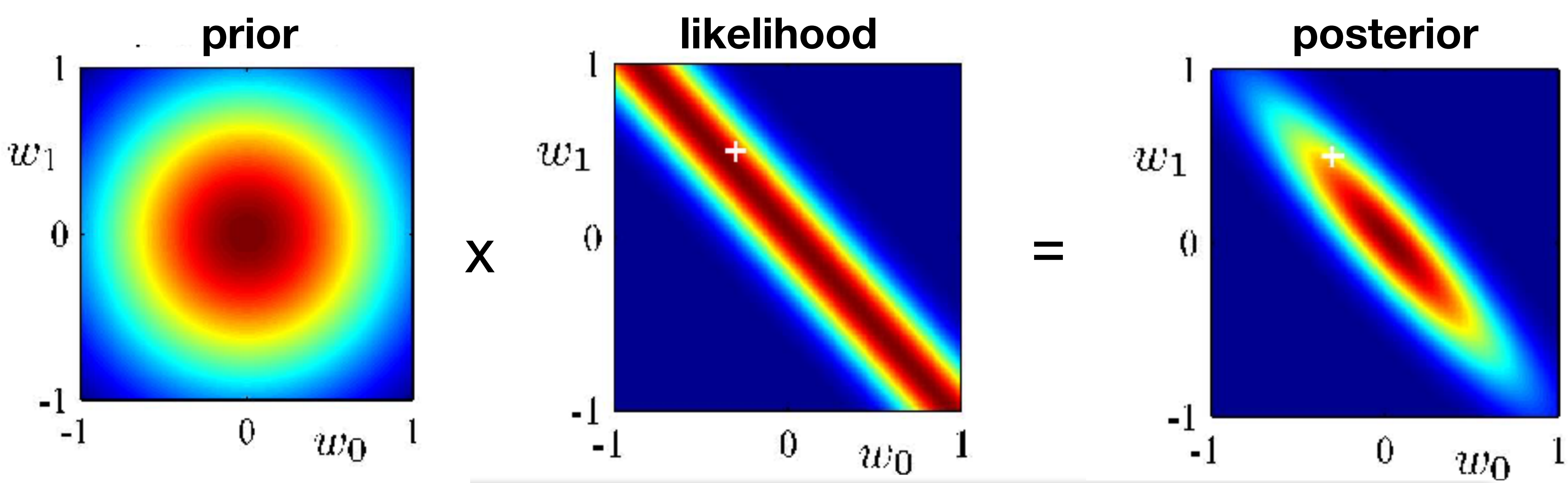


Visualizing inference

- Goal: fit lines $y = w_0 + w_1 x + \epsilon$

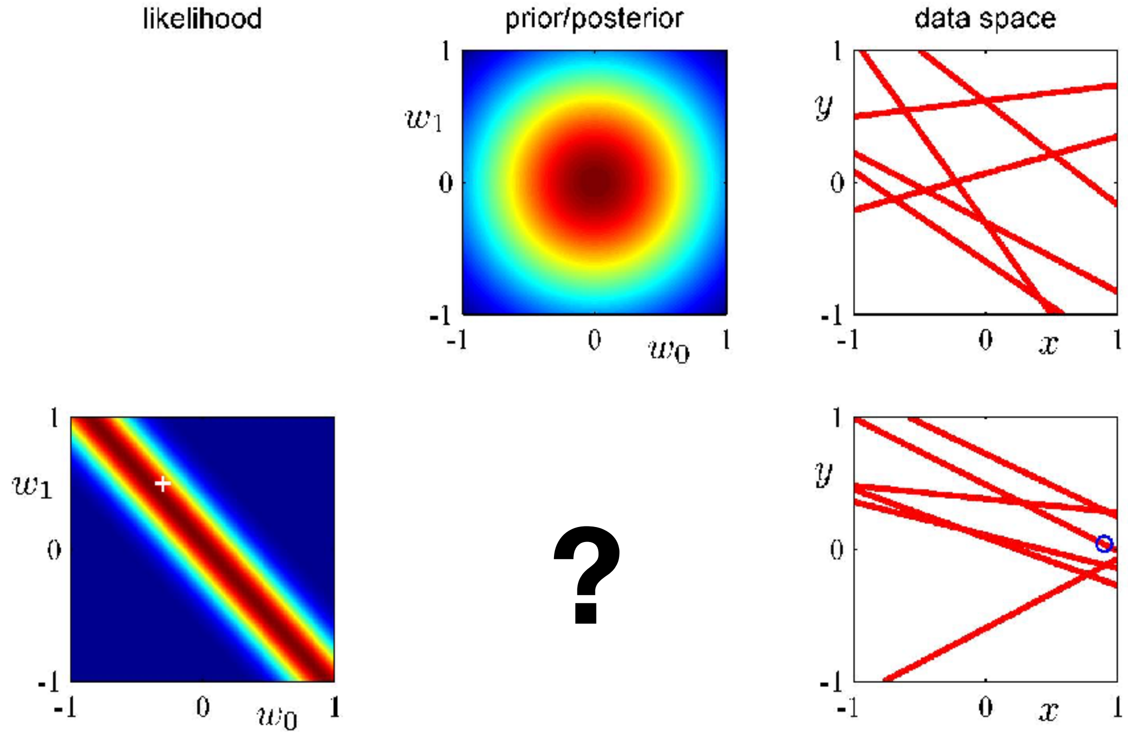


- Bayes theorem: $p(\mathbf{w}|\mathcal{D}) = \dfrac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$
- For all values of **w**, multiply prior and likelihood (and re-normalize)

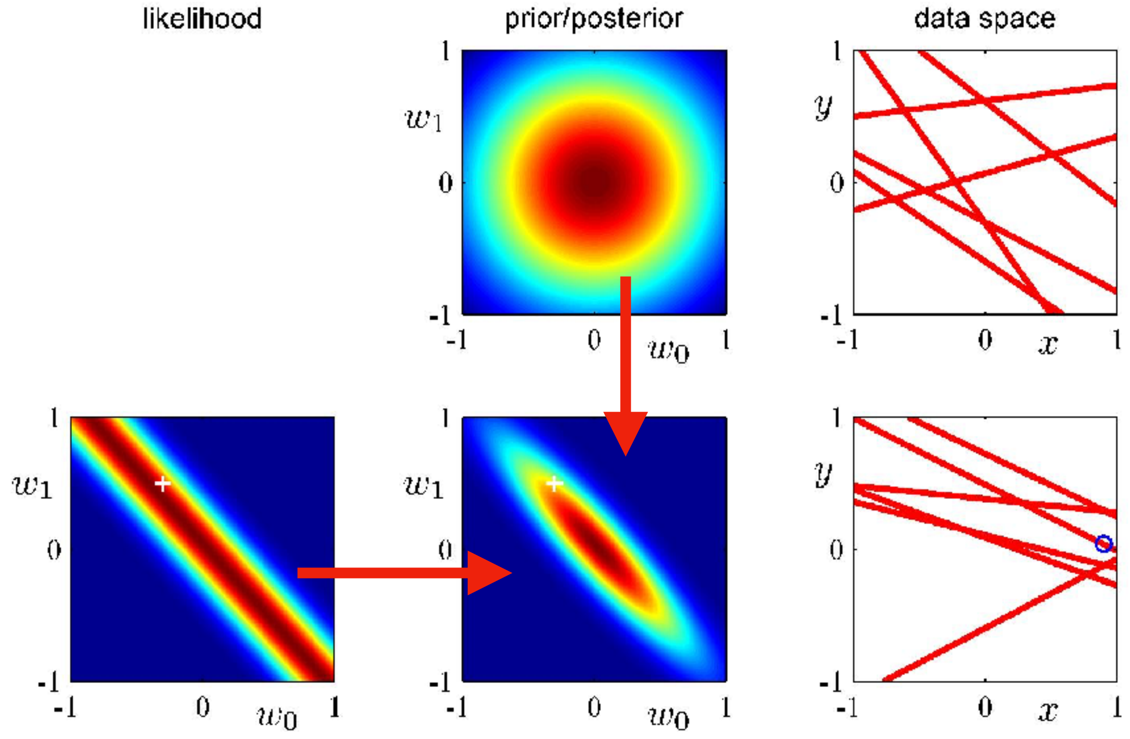


Copyright C.M. Bishop, PRML

Bayesian linear regression: inference

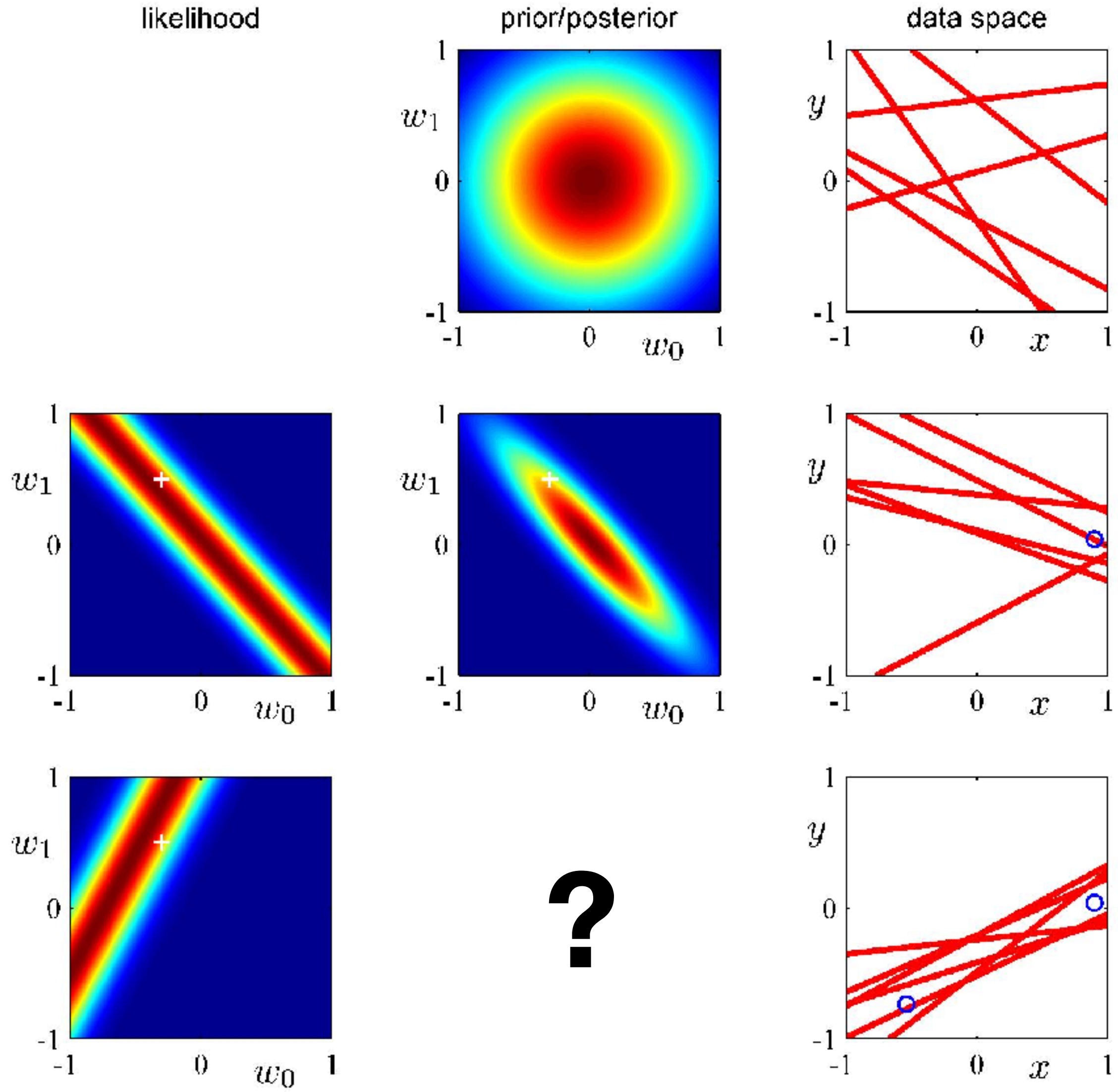


Bayesian linear regression: inference



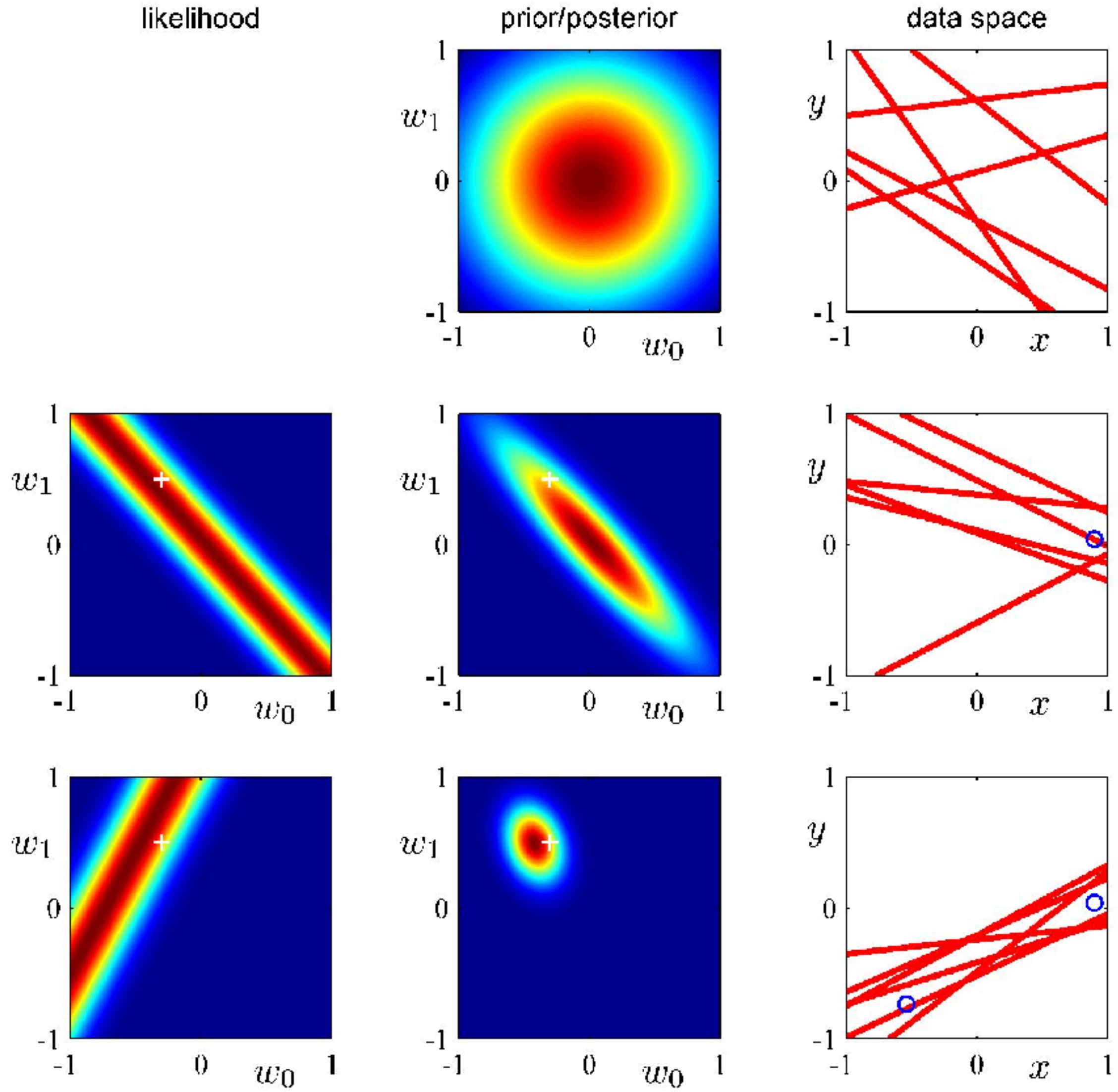
Copyright C.M. Bishop, PRML

Bayesian linear regression: inference



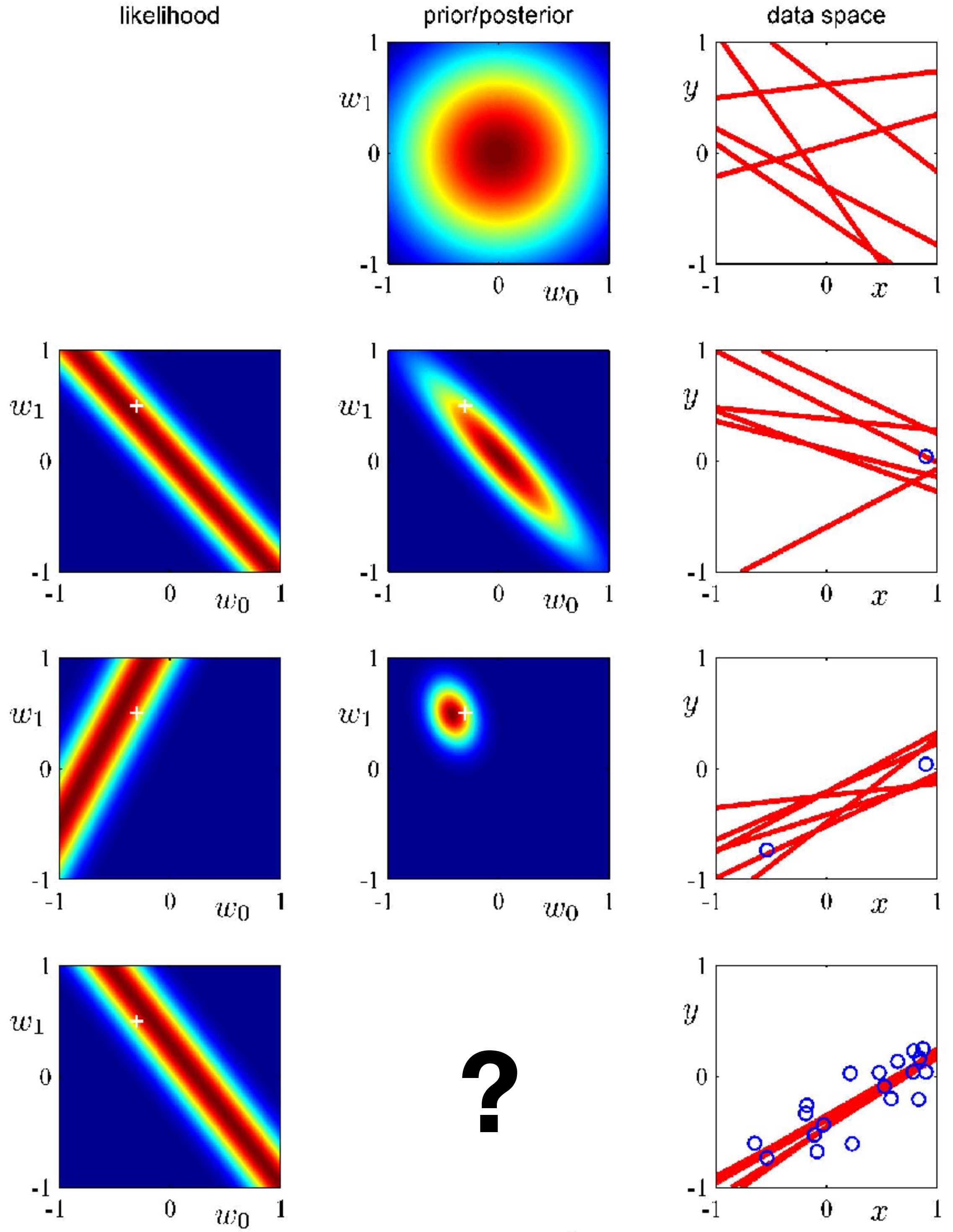
Copyright C.M. Bishop, PRML

Bayesian linear regression: inference

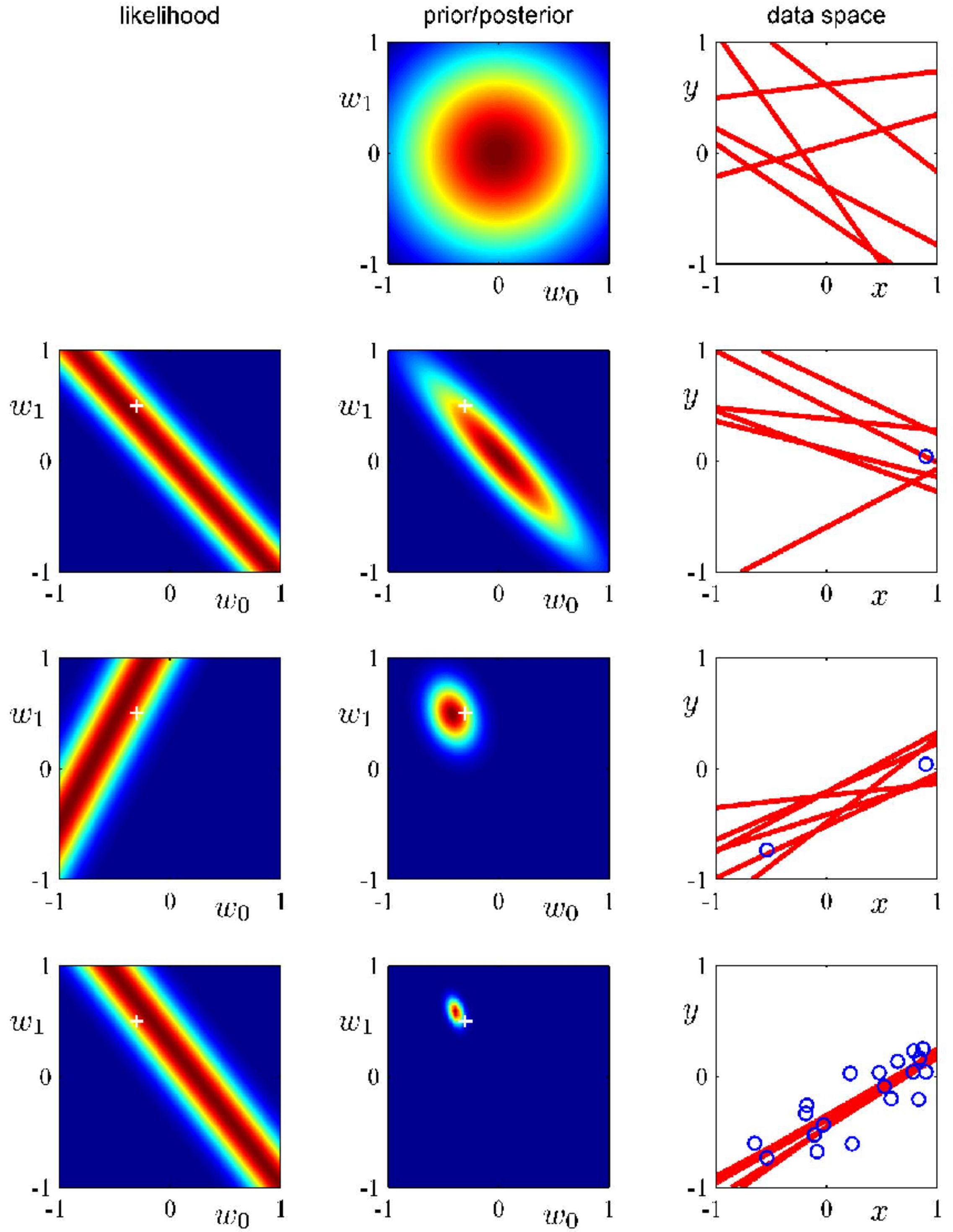


Copyright C.M. Bishop, PRML

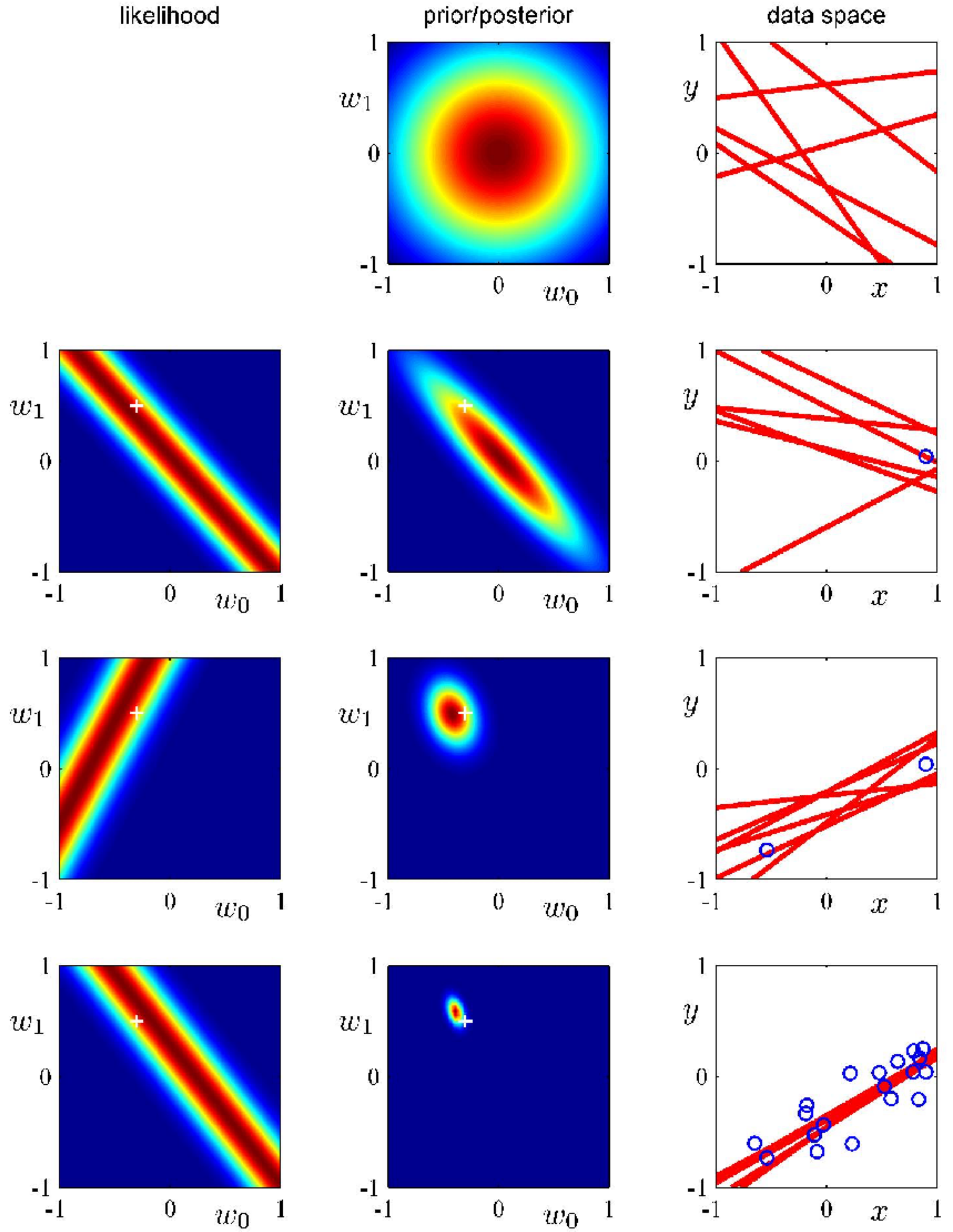
Bayesian linear regression: inference



Bayesian linear regression: inference



Bayesian linear regression: inference



Copyright C.M. Bishop, PRML

As new data points are added, posterior converges on true value of parameters

Step 3: calculate posterior

- Can calculate posterior by multiplying prior and likelihood:

$$p(\mathbf{w}|\mathcal{D}) = \mathcal{N}(\sigma^{-2}\mathbf{S}_N\mathbf{X}^T\mathbf{y}, \mathbf{S}_N)$$

$$\mathbf{S}_N = (\alpha\mathbf{I} + \sigma^{-2}\mathbf{X}^T\mathbf{X})^{-1}$$

(derivation similar to case with no inputs — slide 30 of lecture 18)

- $\mathbf{X}$ has one input per row, $\mathbf{y}$ has one target output per row
- If prior precision α goes to 0, mean becomes maximum likelihood solution (ordinary linear regression)
- Infinitely wide likelihood variance σ^2, or 0 datapoints, means distribution reduces to prior

Aside: finding the MAP

- We can investigate the maximum of the posterior (MAP)
- Log-transform posterior: log is sum of prior + likelihood

$$\max \log p(\mathbf{w}|\mathbf{y})$$

$$= \max -\frac{\sigma^{-2}}{2}\sum_{n=1}^{N}(y_n - \mathbf{w}^T\mathbf{x}_n)^2 - \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} + \text{const.}$$

Aside: finding the MAP

- We can investigate the maximum value of the posterior **(MAP)**
- Calculate in log space: *log posterior = log prior + log likelihood*

$$\max \log p(\mathbf{w}|\mathbf{y})$$

$$= \max -\frac{\sigma^{-2}}{2} \sum_{n=1}^{N} (y_n - \mathbf{w}^T \mathbf{x}_n)^2 - \frac{\alpha}{2} \mathbf{w}^T \mathbf{w} + \text{const.}$$

<u>*Recall*</u>*:*

$$\min \sum_{n=1}^{N} (y_n - \mathbf{w}^T \mathbf{x}_n)^2 + \lambda \mathbf{w}^T \mathbf{w}$$

Ridge regression, Lecture 4 (linear regression)

- Same objective function as for ridge regression!
- Penalty term: $\lambda = \alpha \sigma^2$

prior precision

likelihood variance

<u>Note:</u> since posterior is Gaussian, MAP = mean of posterior

Step 4: prediction

- Prediction for new datapoint:

$$p(y^*|\mathbf{x}^*, \mathcal{D}) = \int_{\mathbb{R}^N} p(\mathbf{w}|\mathcal{D})p(y^*|\mathbf{x}^*, \mathbf{w})d\mathbf{w}$$

- For Gaussians, can compute solution analytically:

$$p(y^*|\mathcal{D}) = \mathcal{N}(\sigma^{-2}\mathbf{x}^{*T}\mathbf{S}_N\mathbf{X}^T\mathbf{y}, \sigma^2 + \mathbf{x}^T\mathbf{S}_N\mathbf{x})$$

mean from before

from weight uncertainty

new input

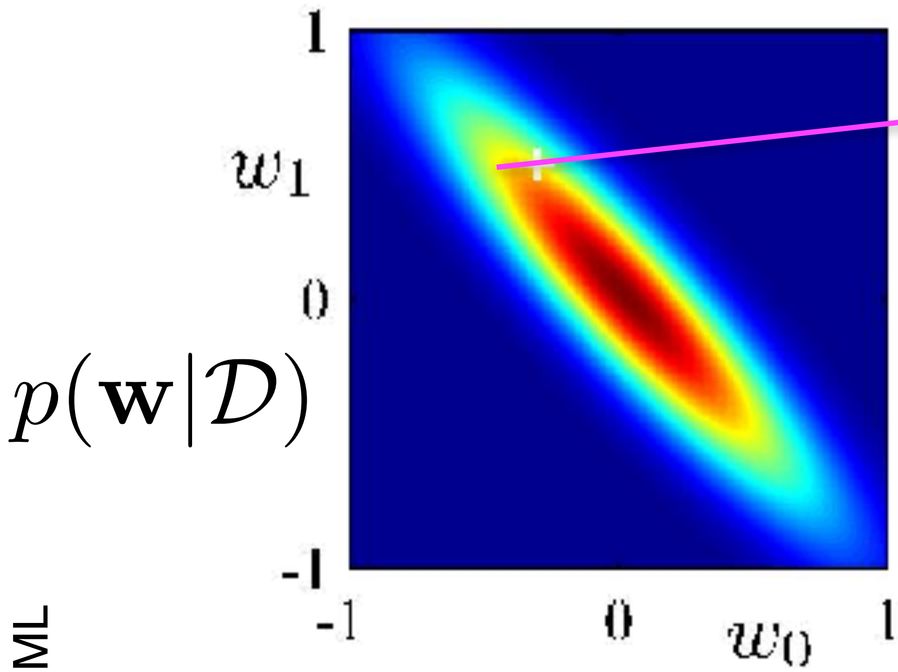
from observation noise

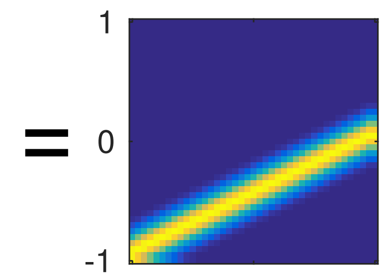
- Variance tends to go down with more data until it reaches σ^2

Step 4: prediction

- Every **w** makes a prediction, weighted by posterior

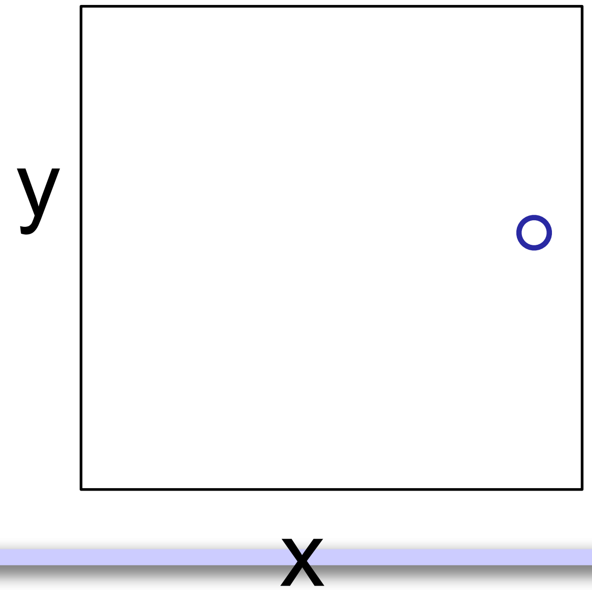
$$\int_{\mathbb{R}^N} p(\mathbf{w}|\mathcal{D})p(y^*|\mathbf{x}^*, \mathbf{w})d\mathbf{w}$$





x medium

$p(\mathbf{w}|\mathcal{D})$

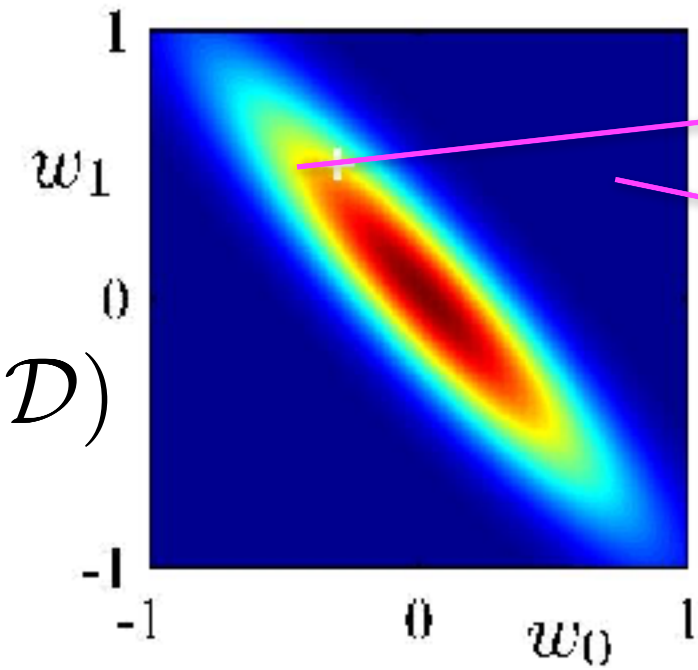


Step 4: prediction

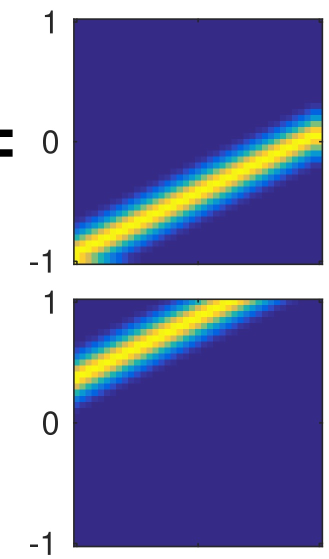
$$\int_{\mathbb{R}^N} p(\mathbf{w}|\mathcal{D})p(y^*|\mathbf{x}^*,\mathbf{w})d\mathbf{w}$$

- Every **w** makes a prediction, weighted by posterior

$p(\mathbf{w}|\mathcal{D})$



=

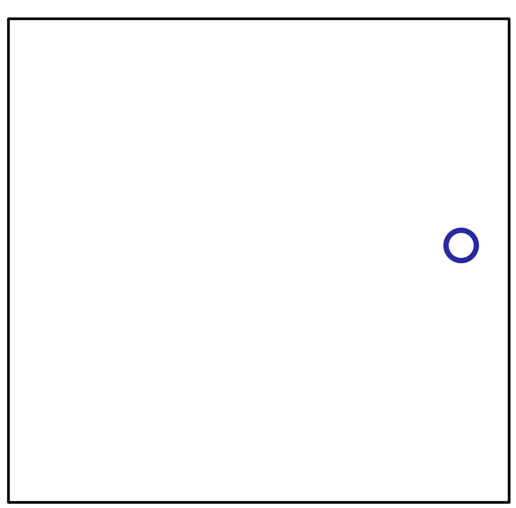


+

x medium $p(\mathbf{w}|\mathcal{D})$

x small $p(\mathbf{w}|\mathcal{D})$

y

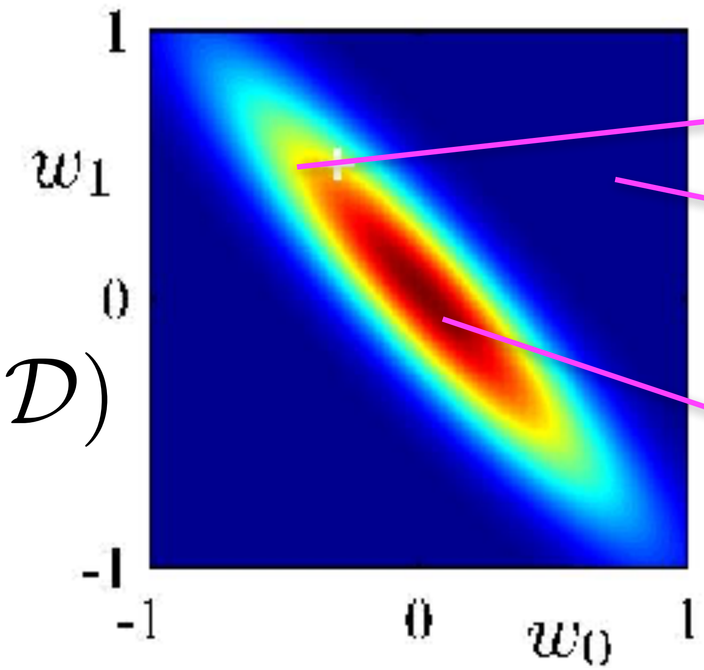


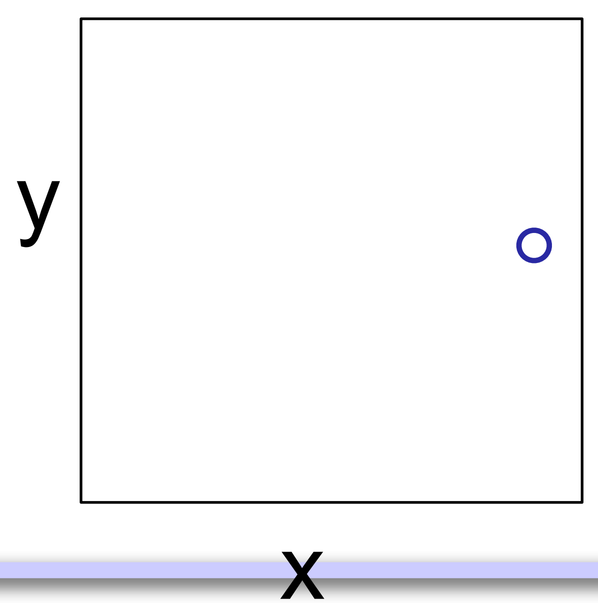
x

Step 4: prediction

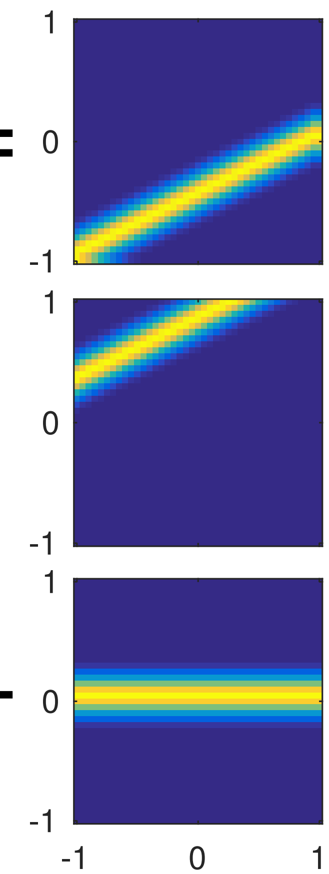
- Every **w** makes a prediction, weighted by posterior

$p(\mathbf{w}|\mathcal{D})$





$$\int_{\mathbb{R}^N} p(\mathbf{w}|\mathcal{D})p(y^*|\mathbf{x}^*,\mathbf{w})d\mathbf{w}$$



= [model 1] x medium $p(\mathbf{w}|\mathcal{D})$

\+ [model 2] x small $p(\mathbf{w}|\mathcal{D})$

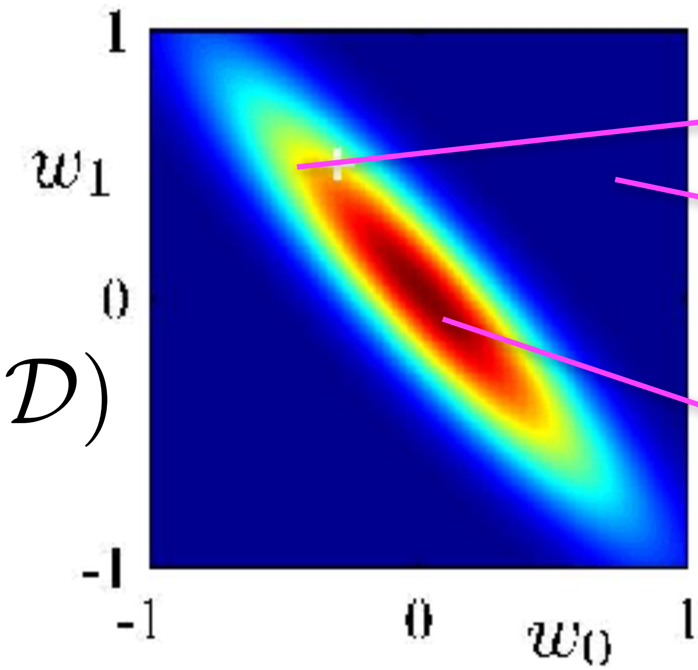
\+ [model 3] x large $p(\mathbf{w}|\mathcal{D})$

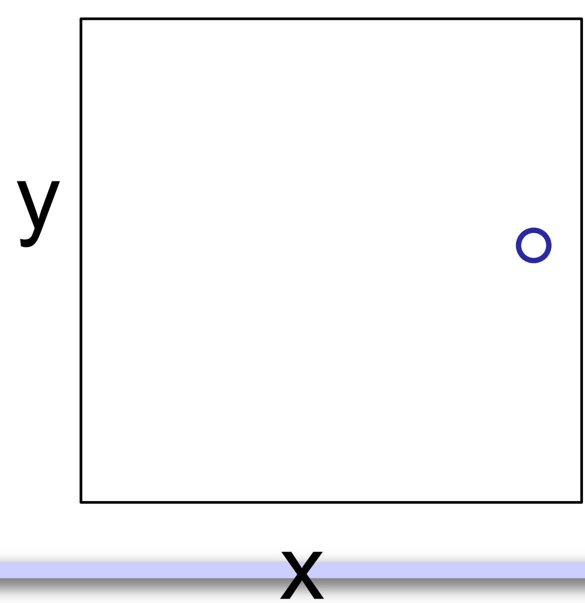
\+ many other models

Step 4: prediction

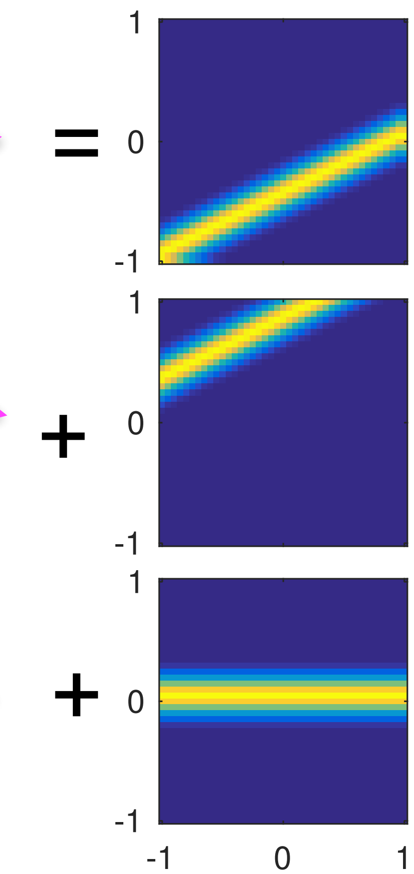
- Every **w** makes a prediction, weighted by posterior

$p(\mathbf{w}|\mathcal{D})$



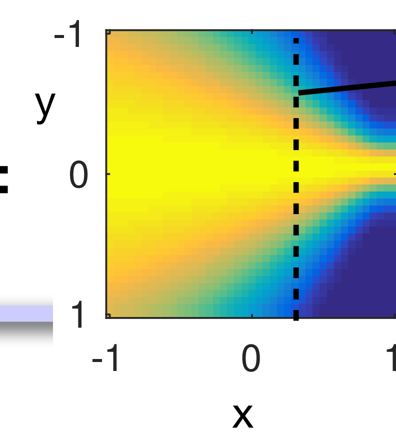


$p(y^*, \mathbf{x}^*|\mathcal{D})$



\+ many other models

=

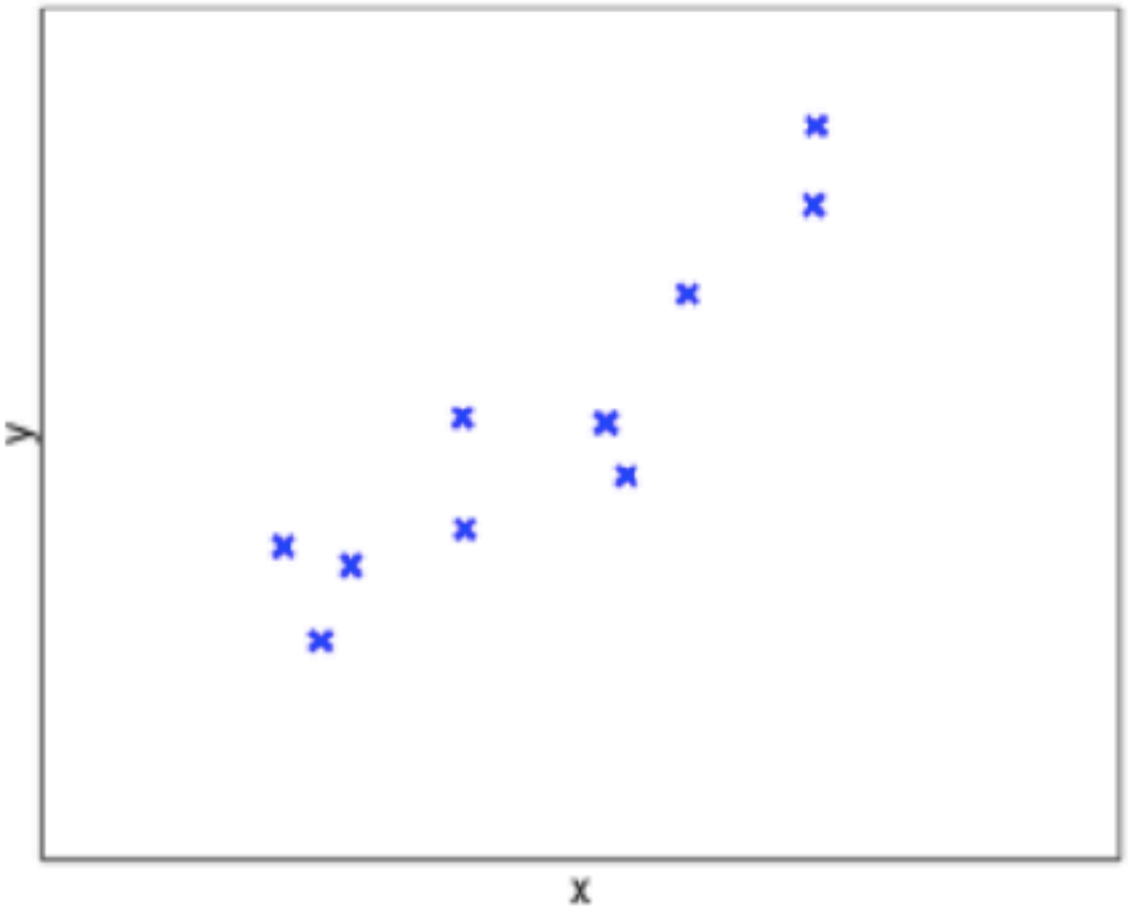


Each vertical line defines $p(y^*|\mathbf{x}^*, \mathcal{D})$ for that x*

Bayesian linear regression

- **Like ordinary linear regression, can use non-linear basis**

$$f_w(x) = w_0 + w_1 x + w_2 x^2$$

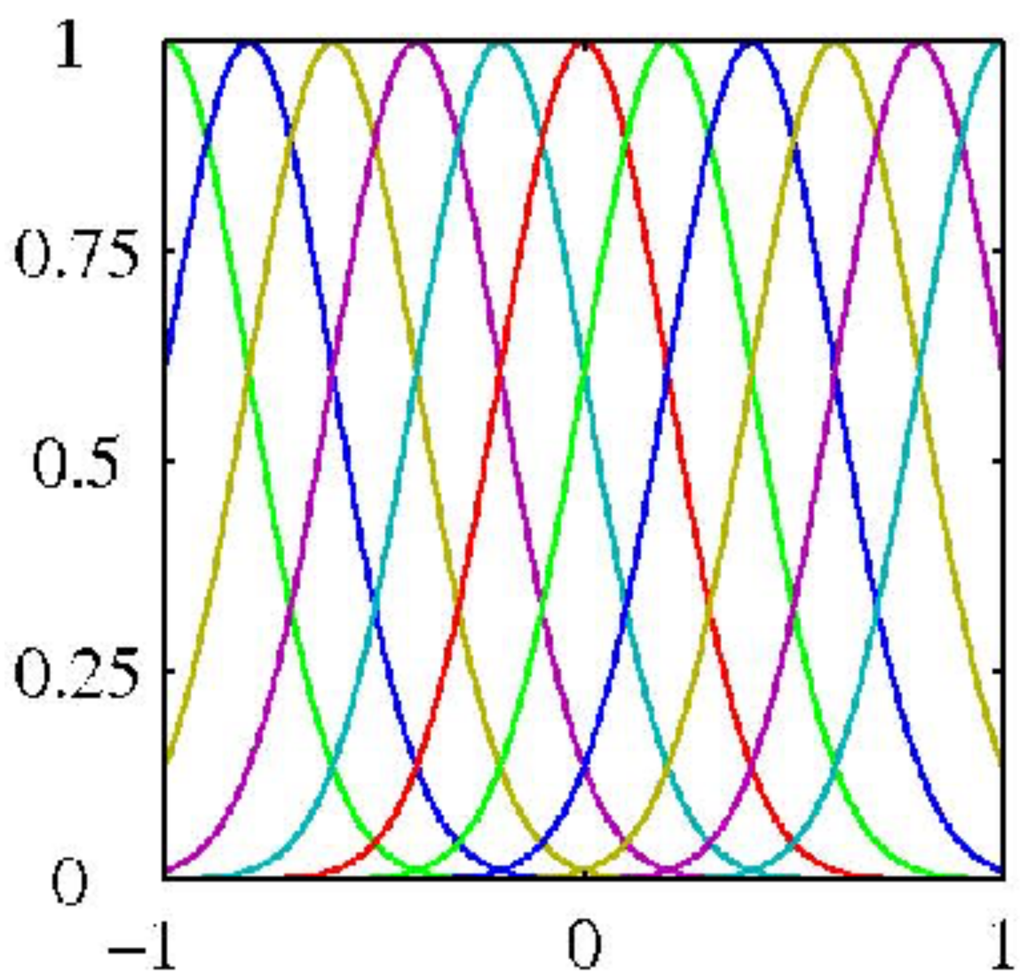


$$X = \begin{array}{c} \begin{array}{ccc} x^2 & x & \end{array} \\ \begin{bmatrix} 0.75 & 0.86 & 1 \\ 0.01 & 0.09 & 1 \\ 0.73 & -0.85 & 1 \\ 0.76 & 0.87 & 1 \\ 0.19 & -0.44 & 1 \\ 0.18 & -0.43 & 1 \\ 1.22 & -1.10 & 1 \\ 0.16 & 0.40 & 1 \\ 0.93 & -0.96 & 1 \\ 0.03 & 0.17 & 1 \end{bmatrix} \end{array} \qquad Y = \begin{bmatrix} 2.49 \\ 0.83 \\ -0.25 \\ 3.10 \\ 0.87 \\ 0.02 \\ -0.12 \\ 1.81 \\ -0.83 \\ 0.43 \end{bmatrix}$$

Lecture 4, linear regression

Bayesian linear regression

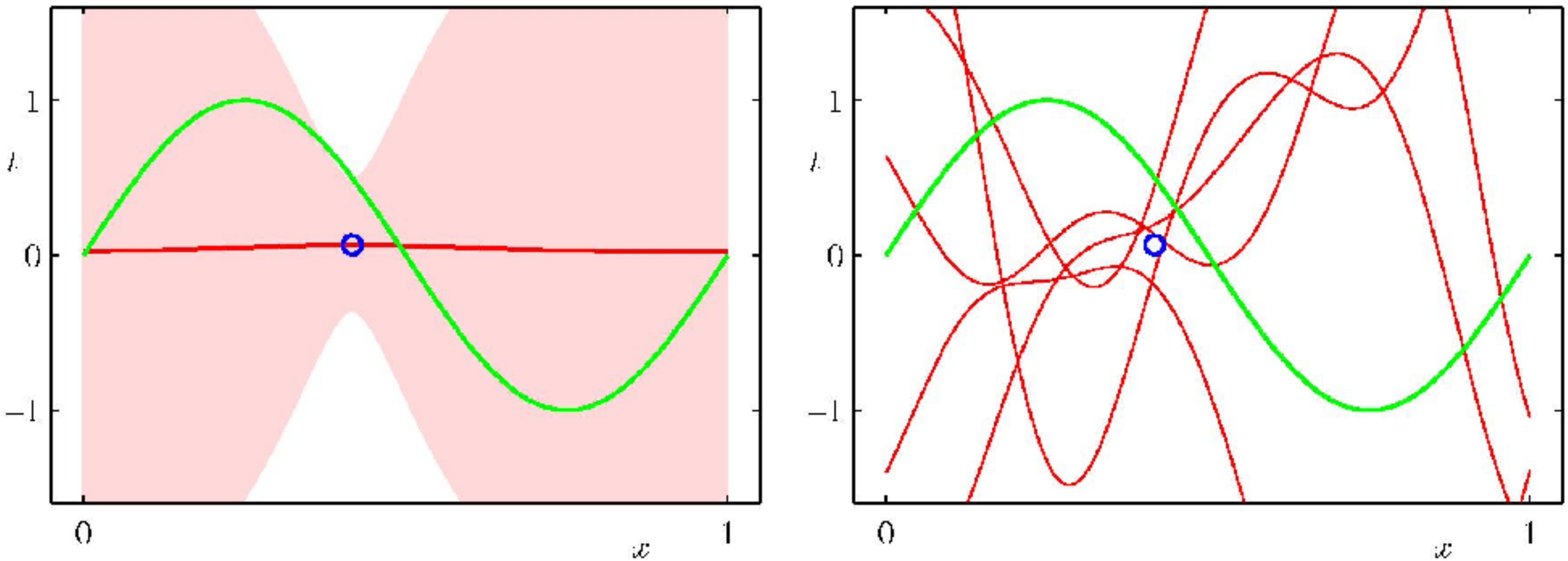
- **Like ordinary linear regression, can use non-linear basis**



$$\hat{y} = \sum_{i=1}^{M} \mathbf{w}_i \phi_i(\mathbf{x})$$

Bayesian linear regression: polynomial bases

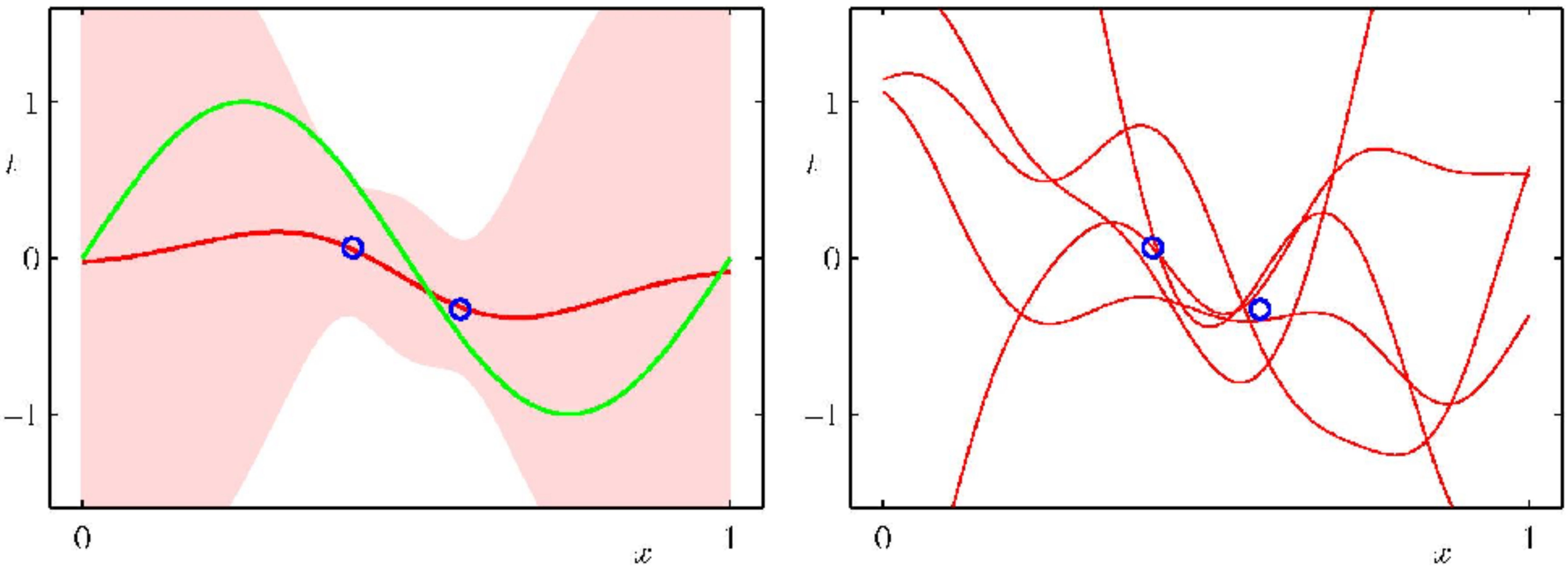
- Example: Bayesian linear regression with polynomial bases



- <u>Green line:</u> true function. <u>Blue circles:</u> data points.

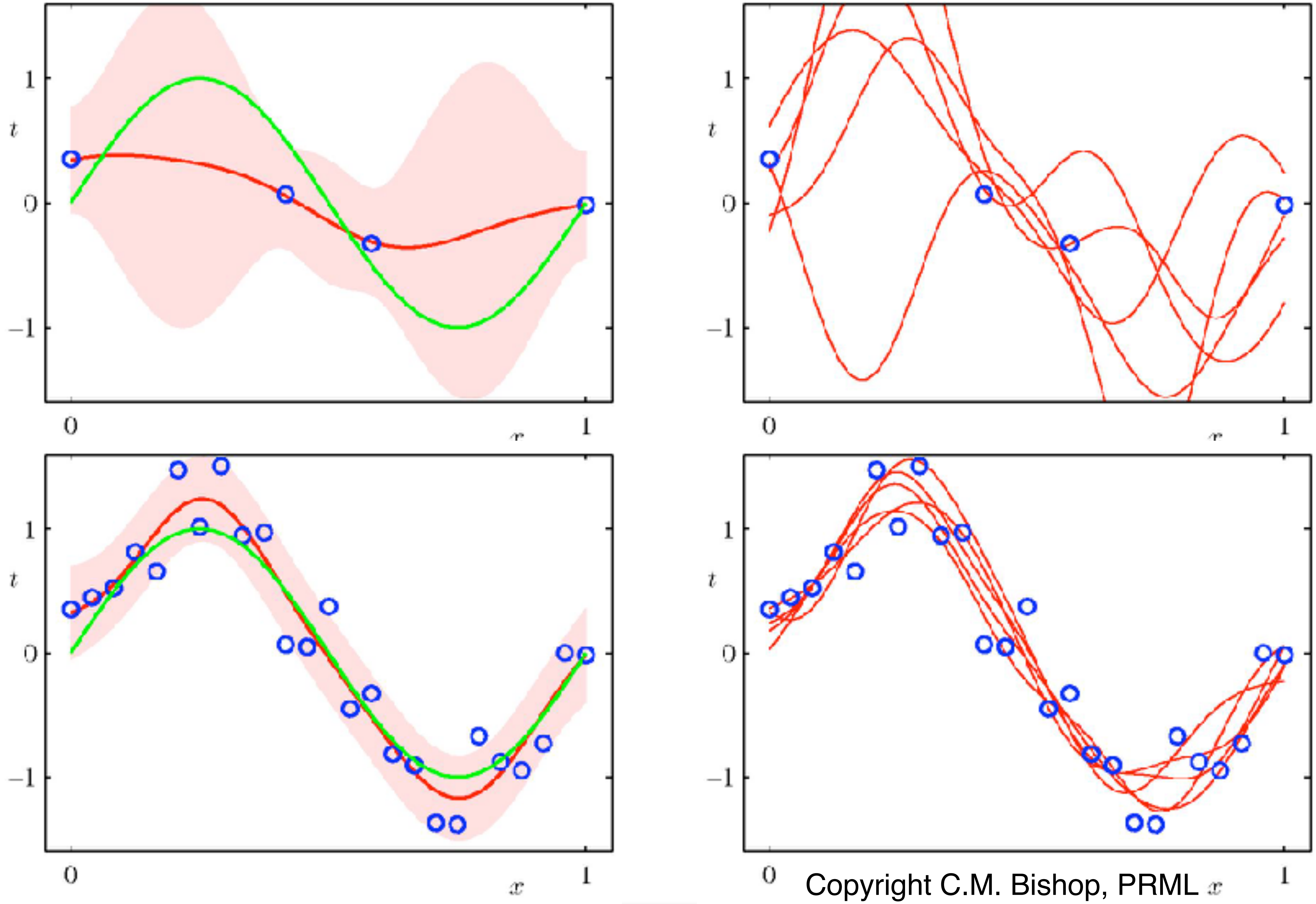
  <u>Red line:</u> MAP prediction. <u>Shaded red:</u> posterior predictive distribution.

Bayesian linear regression: polynomial bases

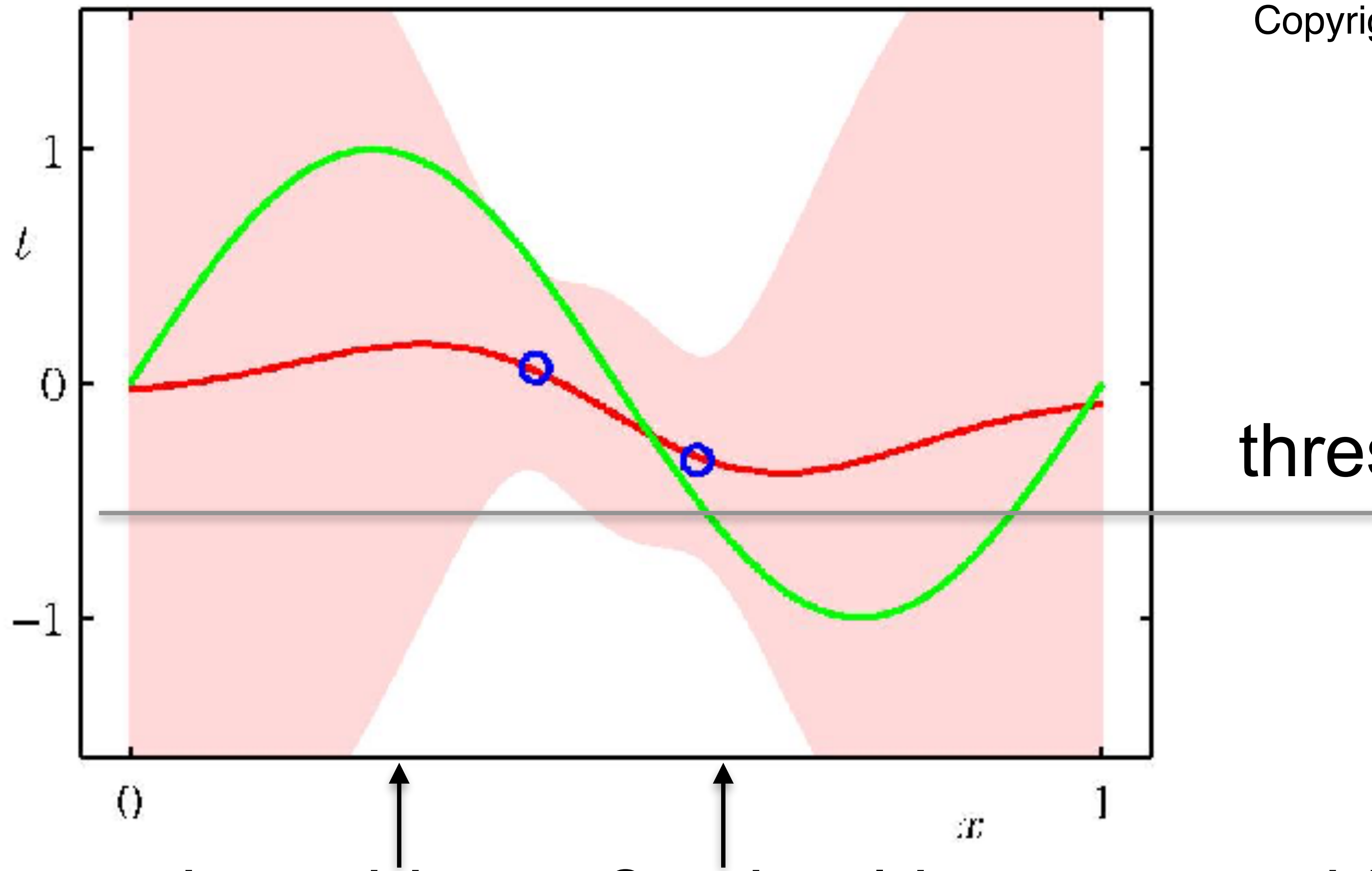


Copyright C.M. Bishop, PRML

Bayesian linear regression: polynomial bases



Recall: inspection task example



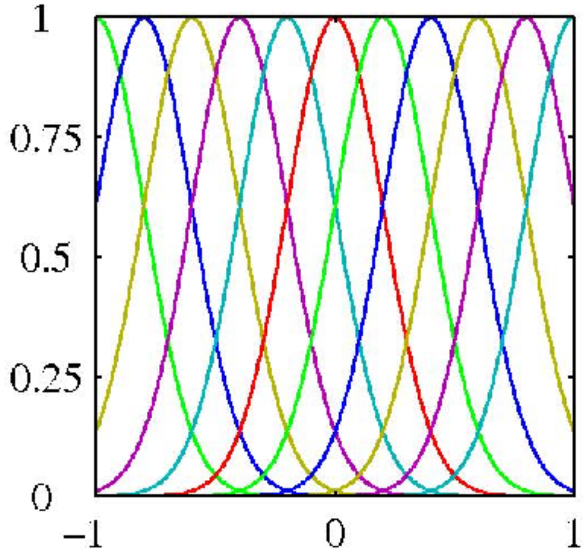
Could produce this graph using Bayesian linear regression

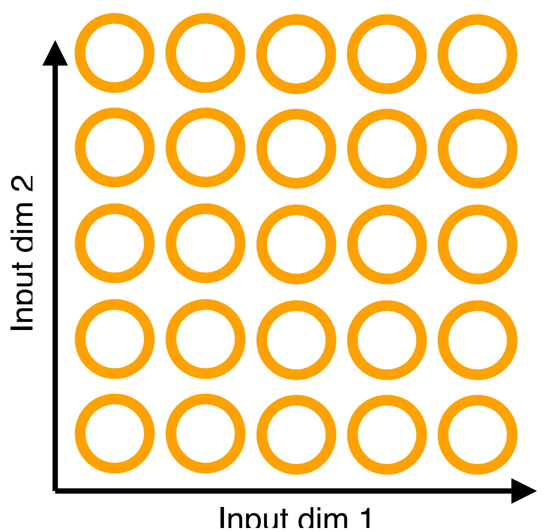
Beyond linear regression

- Non-linear data sets can be handled by using non-linear features
- Features specify the class of functions we consider (hypothesis class)

$$\hat{y} = \sum_{i=1}^{M} \mathbf{w}_i \phi_i(\mathbf{x})$$

- What if we do not know good features?
- Some features (polynomial, RBF) work for many problems



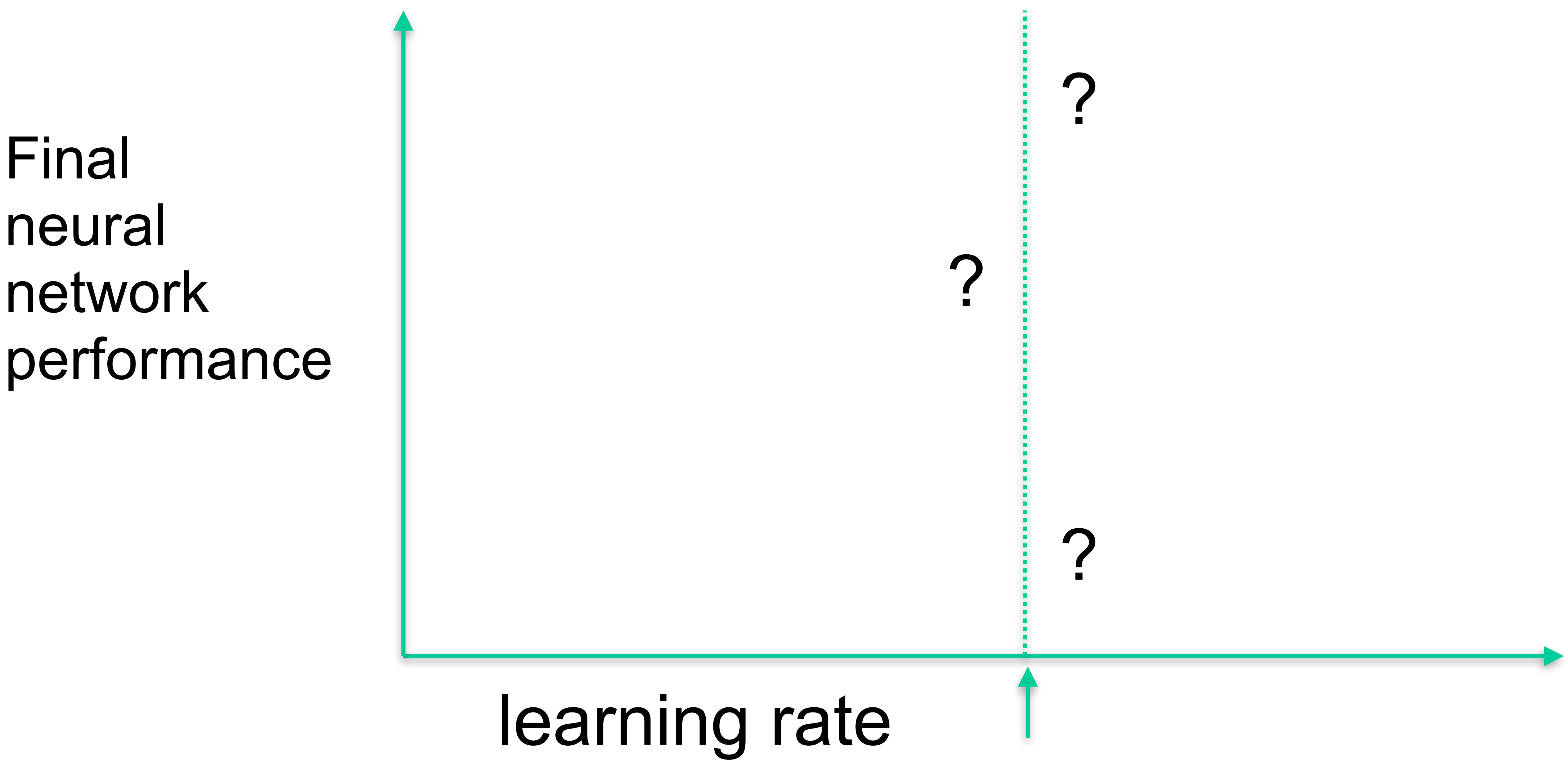


Application: black-box optimization

- **Problem:** find value x for which function f(x) is maximized
- **Constraints:**
  - 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
- Example: finding the hyperparameters of a neural network
- **How can we approach this problem?**

Black-box optimization

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

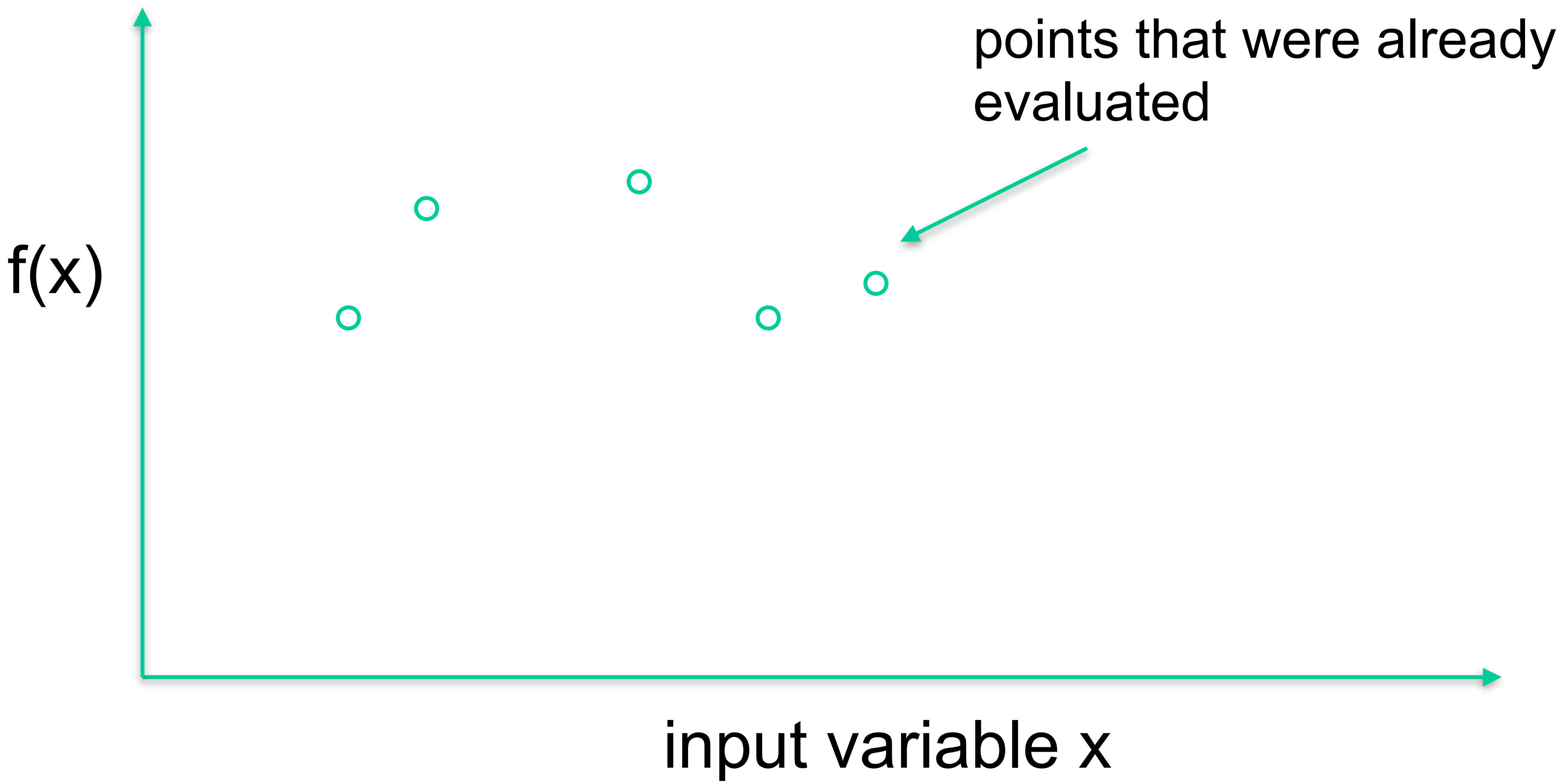


Black-box optimization

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

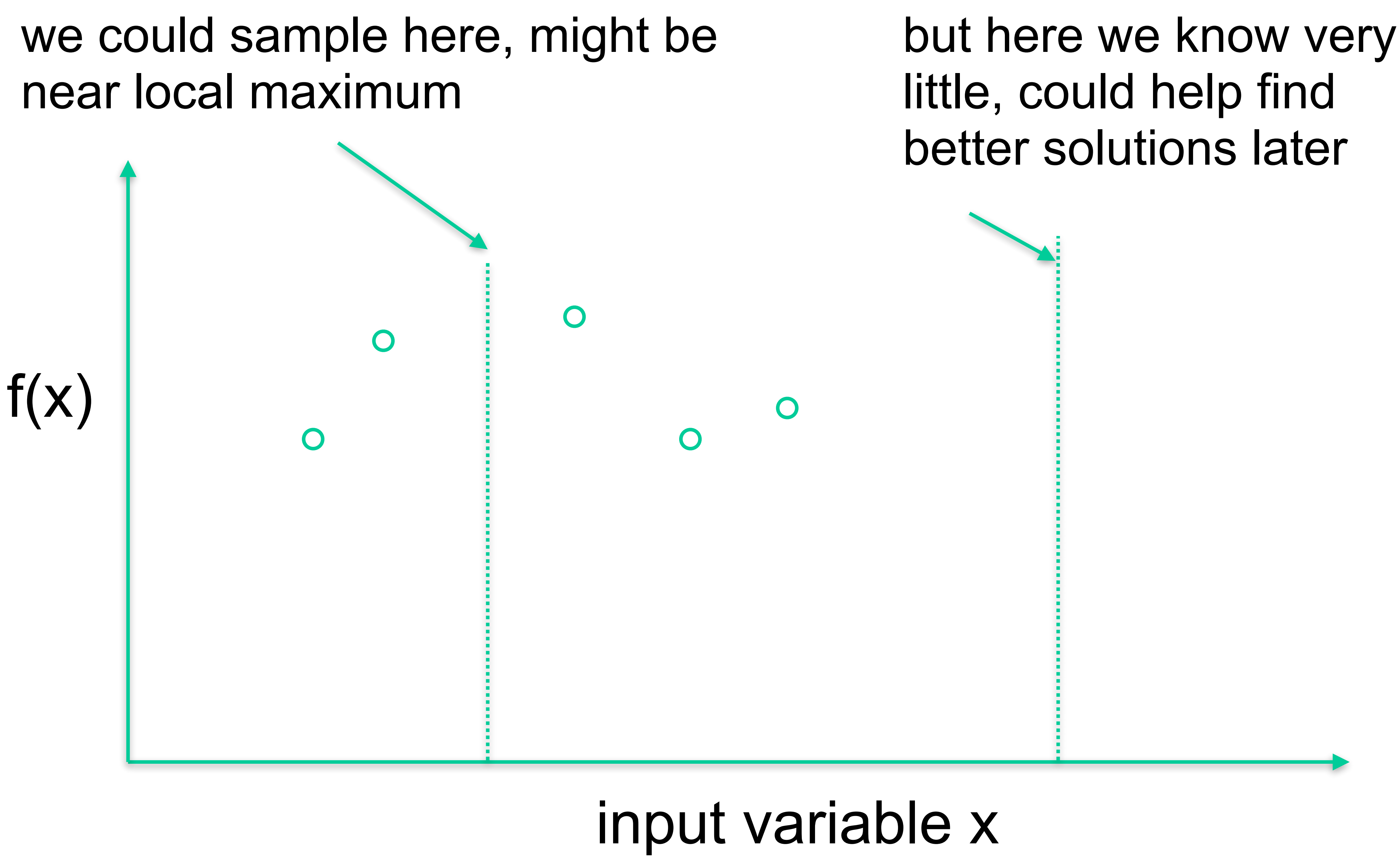
Black-box optimization

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



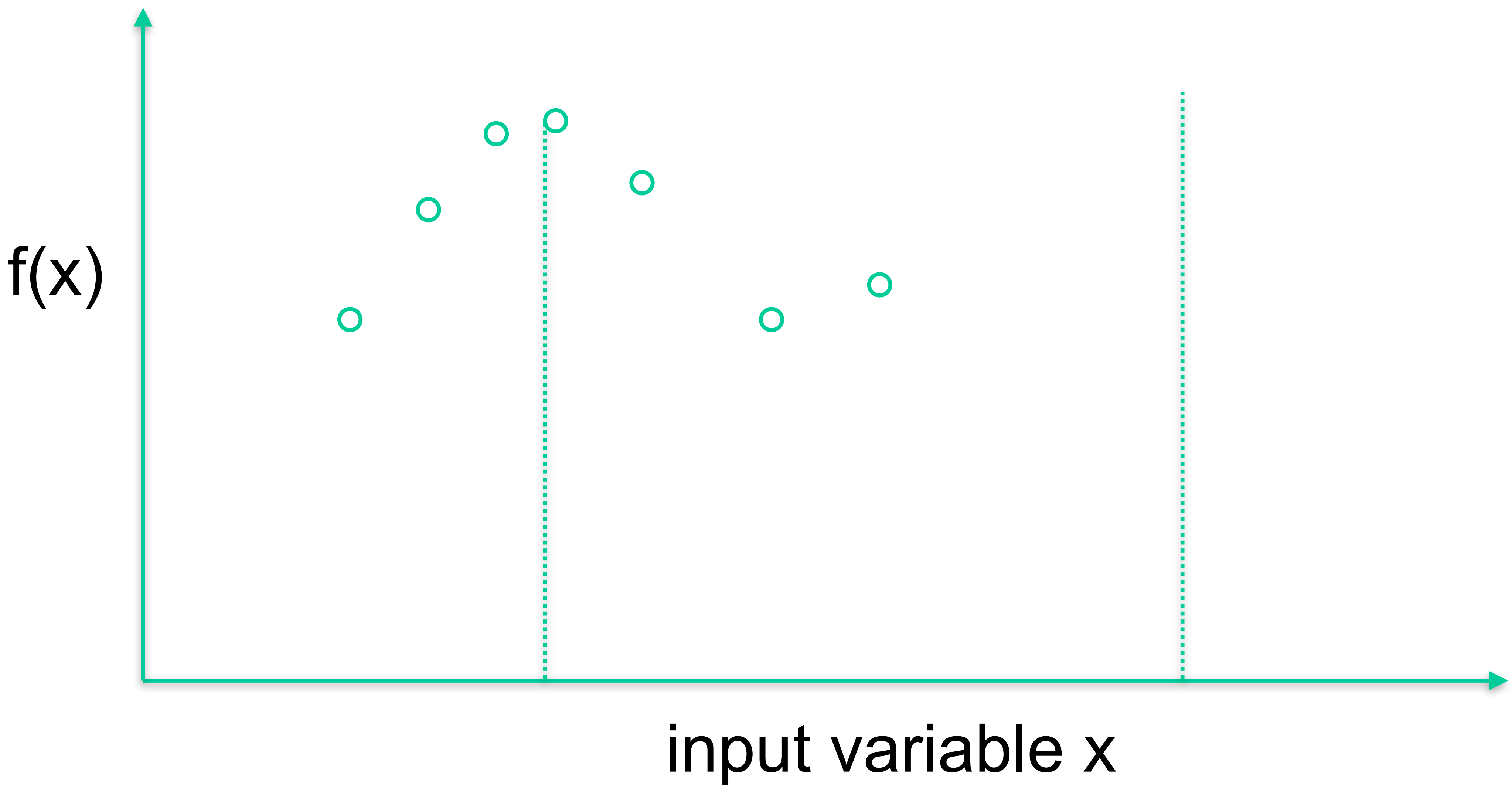
Black-box optimization

- How might a problem look like?



Black-box optimization

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

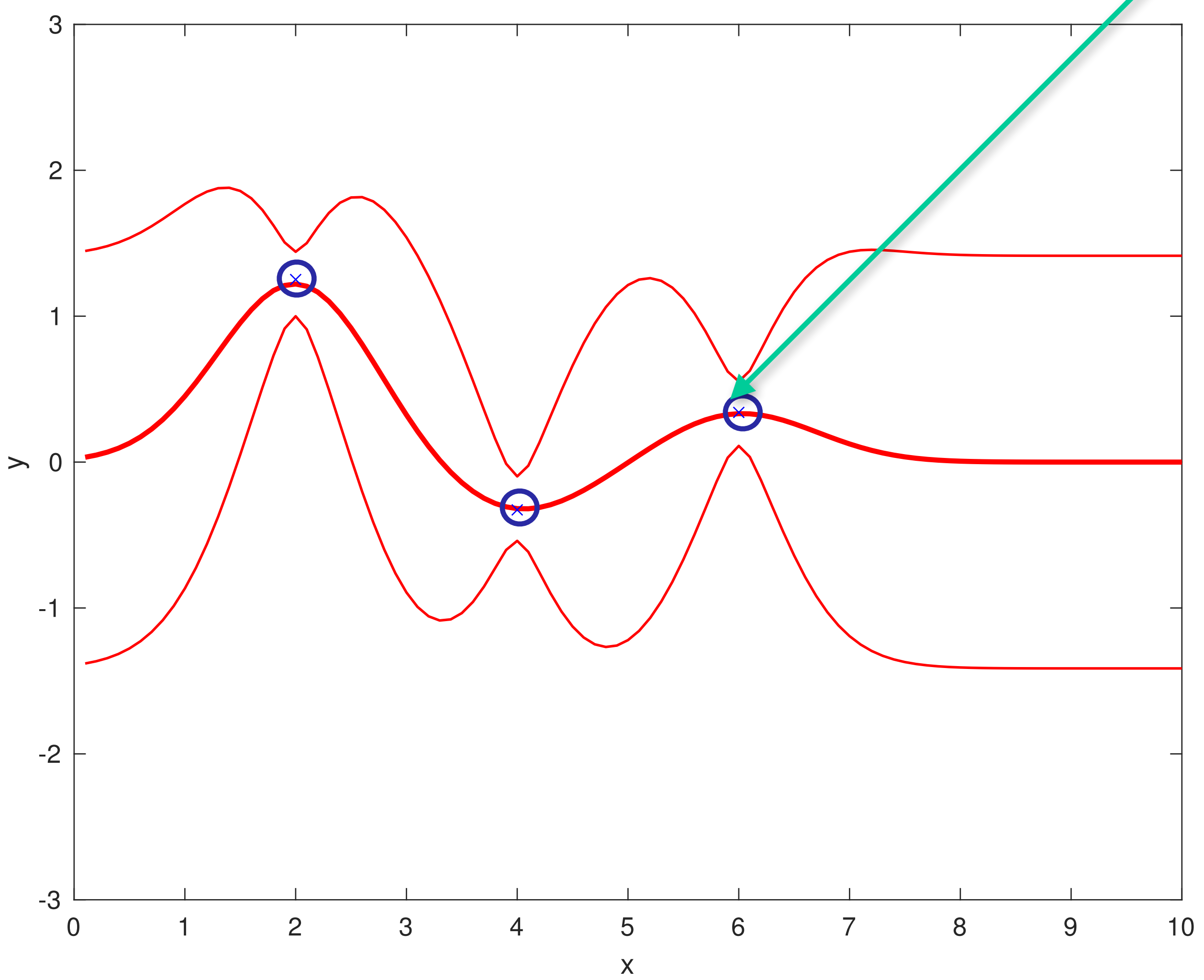


Bayesian optimization

- <u>**Idea:**</u> 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**
- **Bayesian linear regression** might do the job here!
- This implies Bayesian point of view: **Bayesian optimisation** (a method to do black-box optimization)

Bayesian optimisation

- Bayesian posterior over function

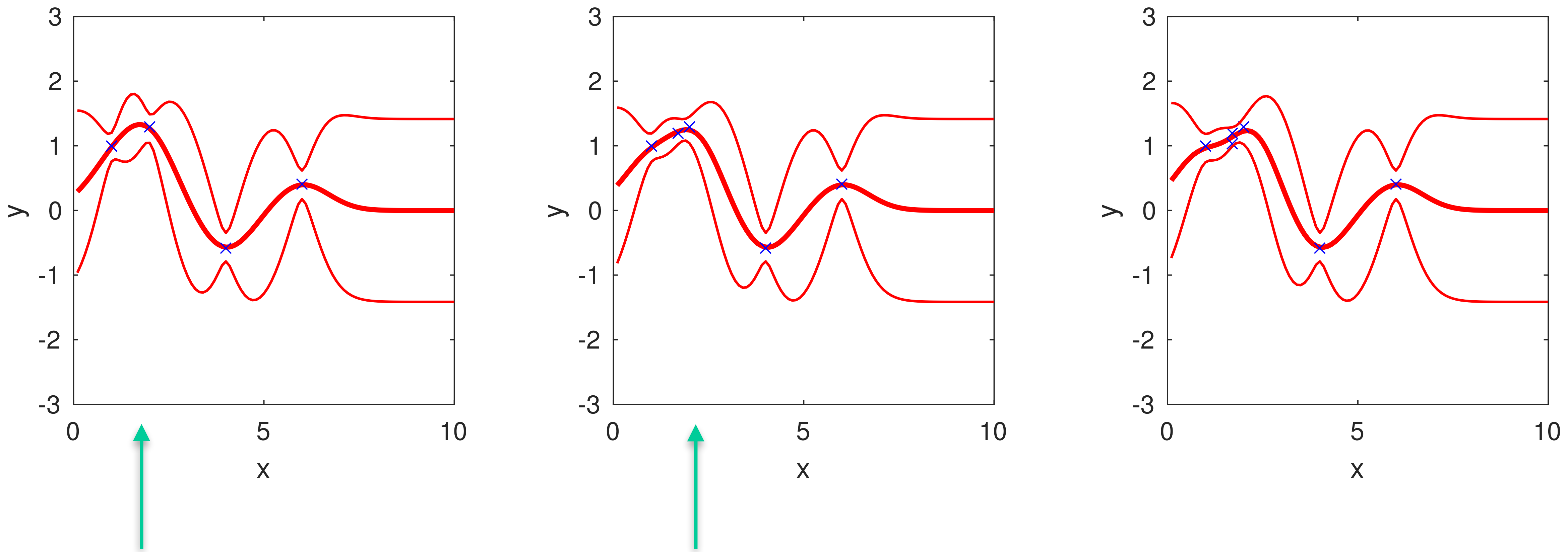


previous function evaluations

- Where to sample next?

Bayesian optimisation

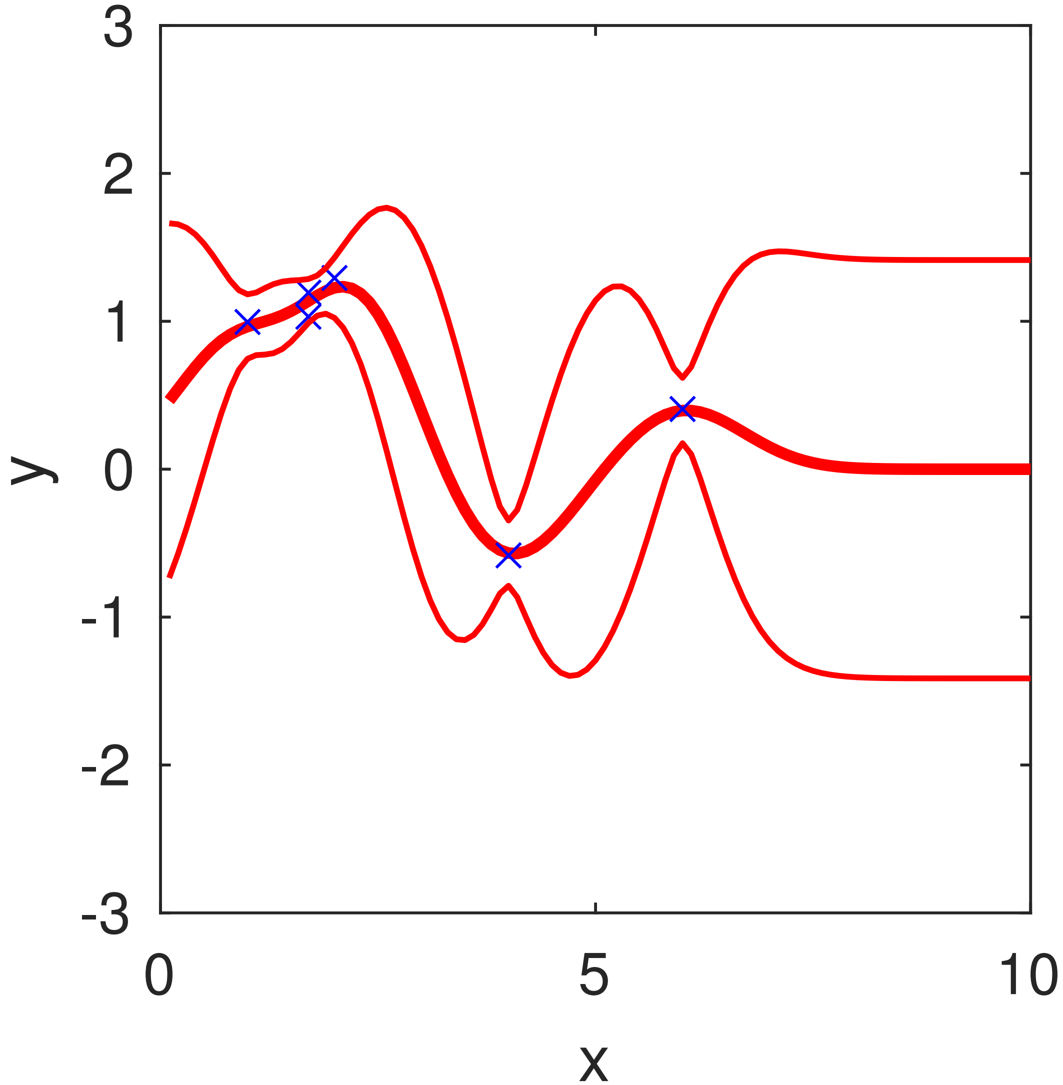
- Where to sample next?
- What happens if we simply sample where mean is highest?

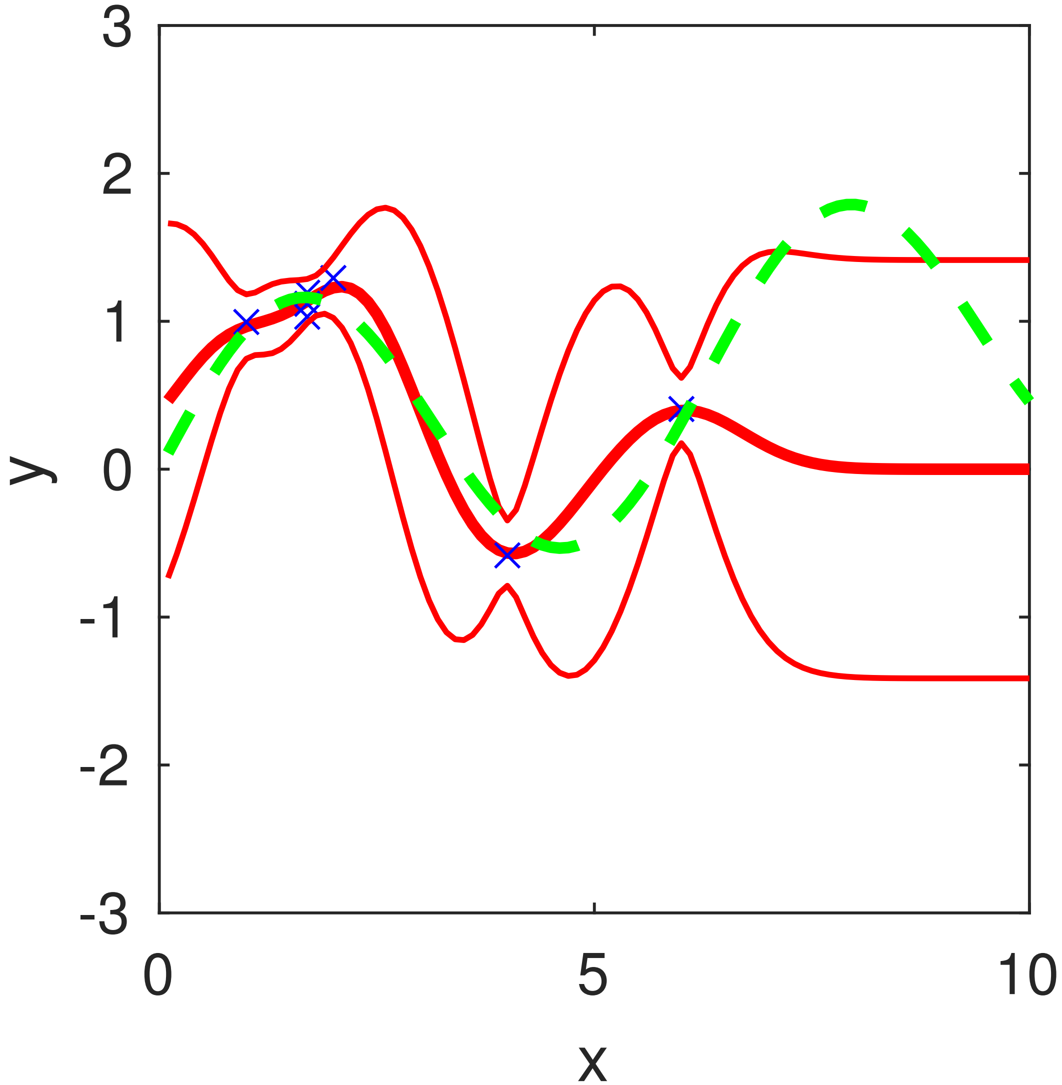


sample location

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 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
- Can give 'score' to possible examples using **acquisition function**
- Very straightforward method: **upper confidence bound (UCB)**

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

predicted mean given data so far

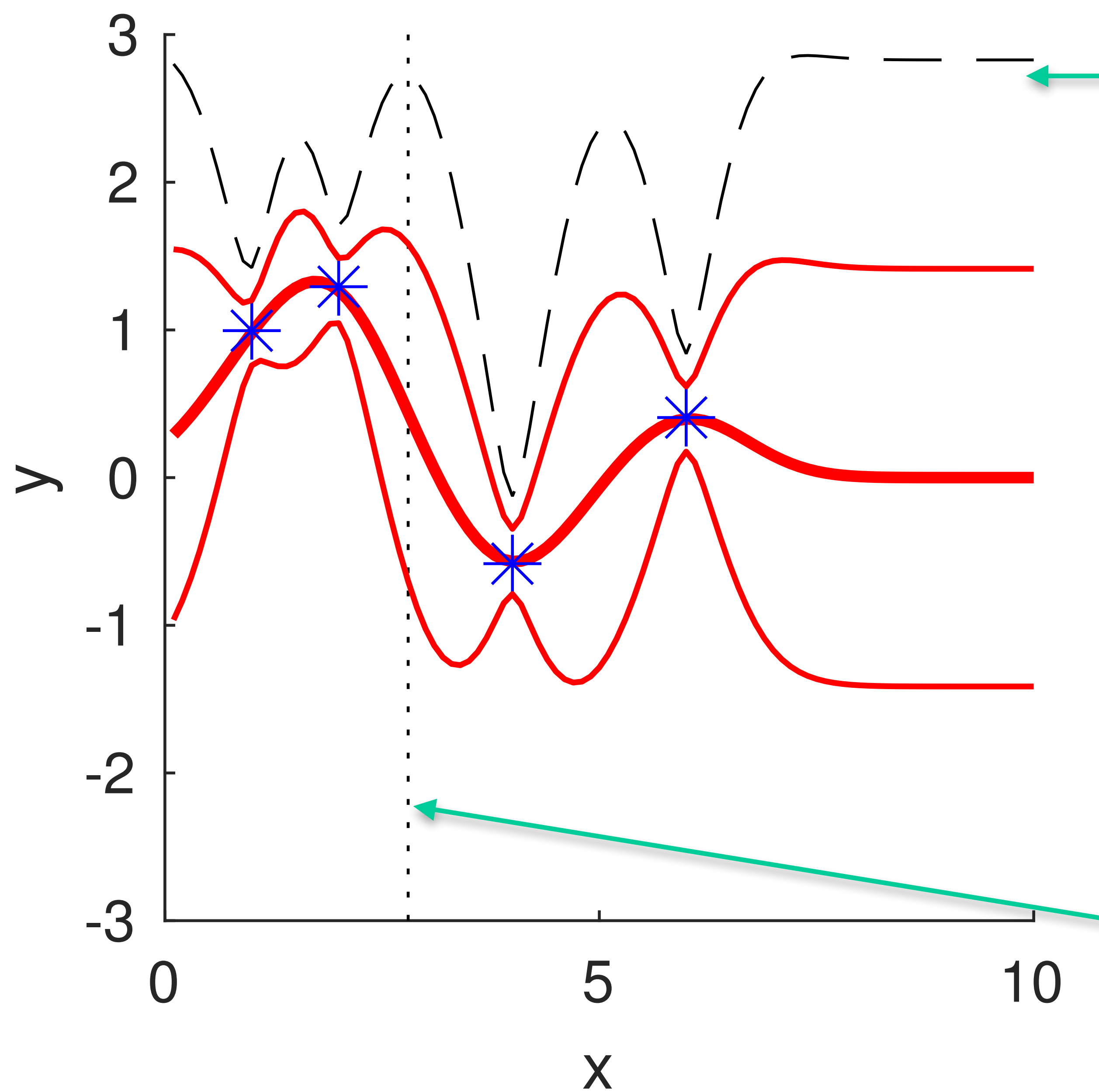
trade-off parameter

predicted standard deviation given data so far

- Acquisition functions gives a 'score' to each sample point
- UCB has good theoretical properties

Bayesian optimisation

- Upper confidence bound acquisition function

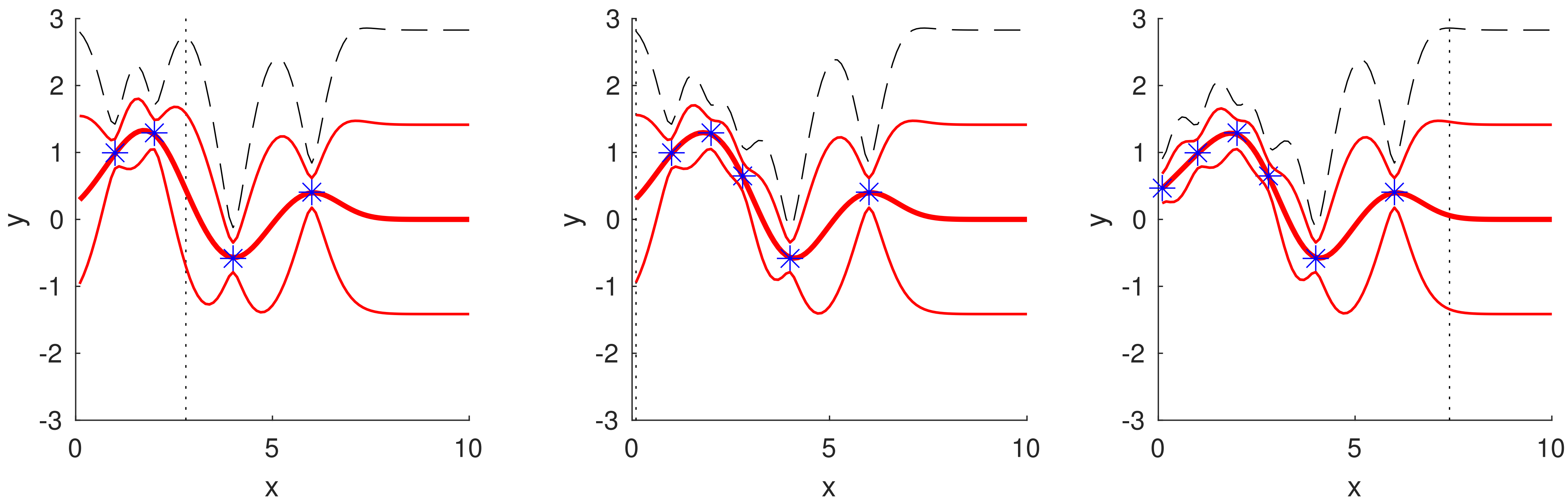


UCB acquisition function (κ=2)

Maximum of acquisition function

Bayesian optimisation

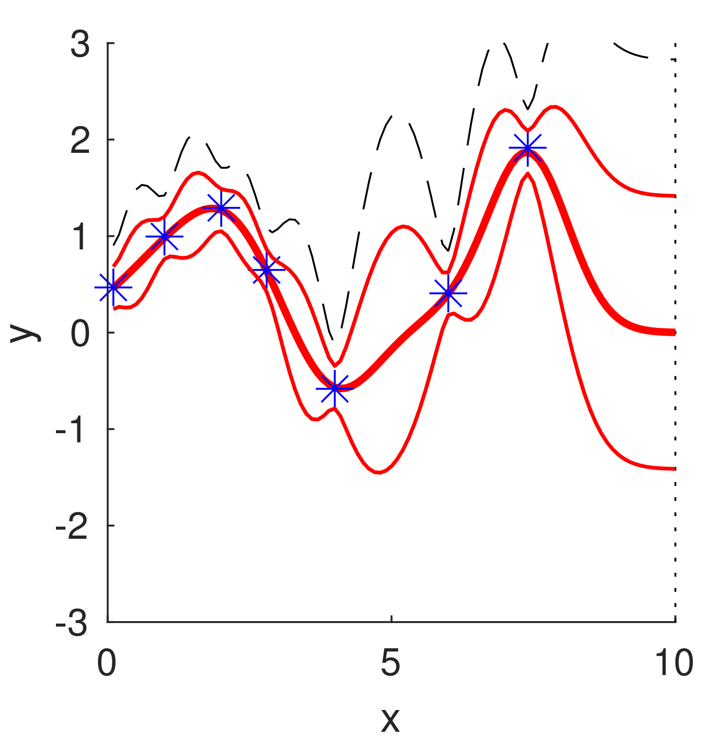
- Upper confidence bound acquisition function



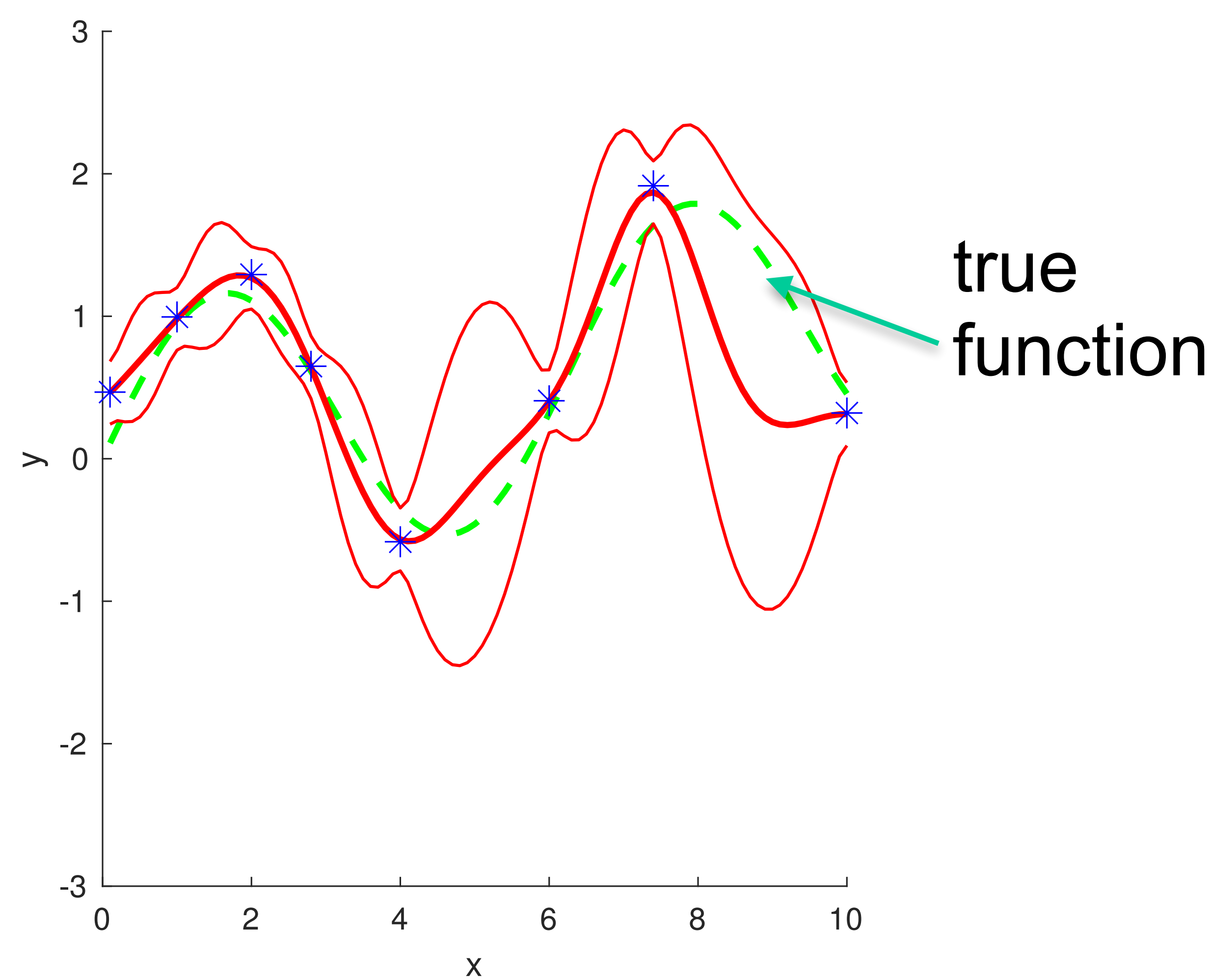
first sample second sample third sample

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 model and hyperparameters in every iteration
    - Need to maximise non-convex acquisition function
  - Sensitive to choice of model
  - 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

Inference vs. Learning

- Different (overlapping!) communities use different terminology, can be confusing
- In *traditional machine learning*:
  - **Learning:** adjusting the parameters of your model to fit the data (by optimization of some cost function)
  - **Inference:** given your model + parameters and some data, make some prediction (e.g. the class of an input image)
- In *Bayesian statistics*, inference is to say something about the process that generated some data **(includes parameter estimation)**
- Take-away: in an ML problem, we can find a good value of params by optimization (*learning*) or calculate a distribution over params (*inference*)

Why Bayesian probabilities?

- **Maximum likelihood estimates can have large variance**
  - Overfitting in e.g. linear regression models
  - MLE of coin flip probabilities with three sequential ‘heads’

Why Bayesian probabilities?

- Maximum likelihood estimates can have large variance
- **We might desire or need an estimate of uncertainty**
  - Can use uncertainty in decision making
  - Can use uncertainty to decide which data to acquire (active learning, experimental design)

Why Bayesian probabilities?

- Maximum likelihood estimates can have large variance
- We might desire or need an estimate of uncertainty
- **Have small dataset, unreliable data, or small batches of data**
  - Account for reliability of different pieces of evidence
  - Possible to update posterior incrementally with new data
  - Variance problem especially bad with small data sets

Why Bayesian probabilities?

- Maximum likelihood estimates can have large variance
- We might desire or need an estimate of uncertainty
- Have small dataset, unreliable data, or small batches of data
- **Use prior knowledge in a principled fashion**

Why Bayesian probabilities?

- Maximum likelihood estimates can have large variance
- We might desire or need an estimate of uncertainty
- Have small dataset, unreliable data, or small batches of data
- Use prior knowledge in a principled fashion
- **In practice, using prior knowledge and uncertainty particularly makes difference with small data sets**

Why not Bayesian probabilities?

- Prior induces bias
- Misspecified priors: if prior is wrong, posterior can be far off
- Prior often chosen for mathematical convenience, not actually knowledge of the problem
- In contrast to frequentist probability, uncertainty is subjective, different between different people / agents

Beyond linear regression

- Relying on features can be problematic
- We tried to avoid using features before…
  - Lecture 8, instance based learning. Use distances!
  - Lecture 12, support vector machines. Use kernels!

- **<u>Next class:</u>** extend regression to nonparametric models
  - *Gaussian processes!*

What you should know

- Bayesian terminology (prior, posterior, likelihood, etc.)
- Conjugate priors, what they mean, showing a distribution is a conjugate prior
- Bayesian linear regression and its properties
- When and why to use Bayesian methods
- Core concepts behind Bayesian optimization