COMP 551 – Applied Machine Learning
Lecture 20: Gaussian processes

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

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

Last week's Quiz

The neural auto encoder is analogous to the PCA under which conditions?

Linear layer / Non-linear layer / **Single hidden layer** / Cross-entropy loss function / **Squared-error loss function** / L1-regularization

Which of the following statements are True:

Dropout reduces overfitting by reducing computation.
Dropout reduces overfitting by increasing the model capacity.
Dropout reduces overfitting by reducing noise.
Dropout reduces overfitting by model averaging.

CNNs are effective for computer vision task for which reasons:

They have a built-in ability to exploit local regularities.
They can scale to high-dimensional inputs.
They can be trained with less data than feed-forward neural networks.
They are invariant to translations.

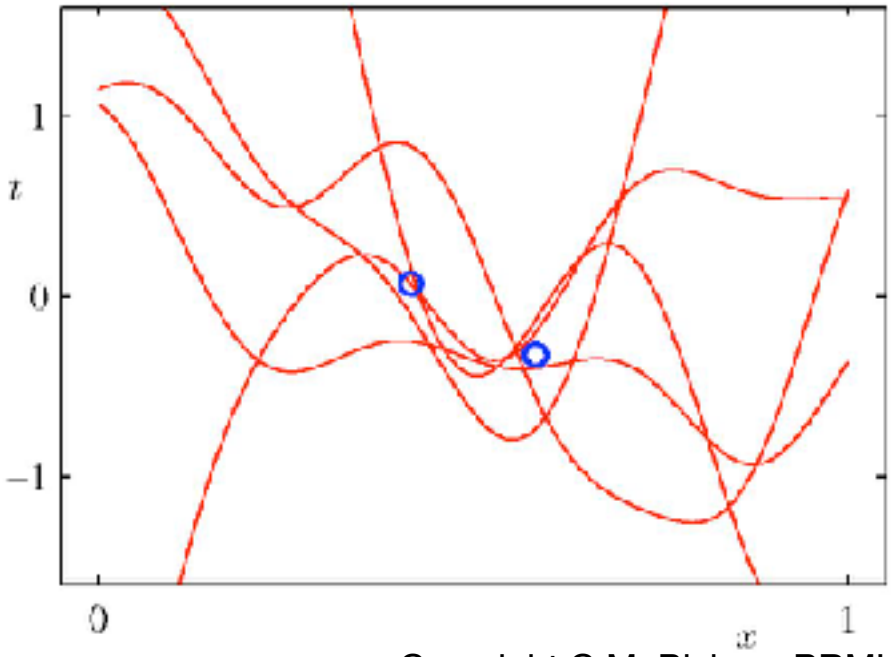
Today's Quiz

Today's Goals

- Why do we need uncertainty in regression?
- How can we quantify uncertainty in regression?

- State-of-art algorithms for regression:
  - Kernel ridge regression
  - Gaussian process regression

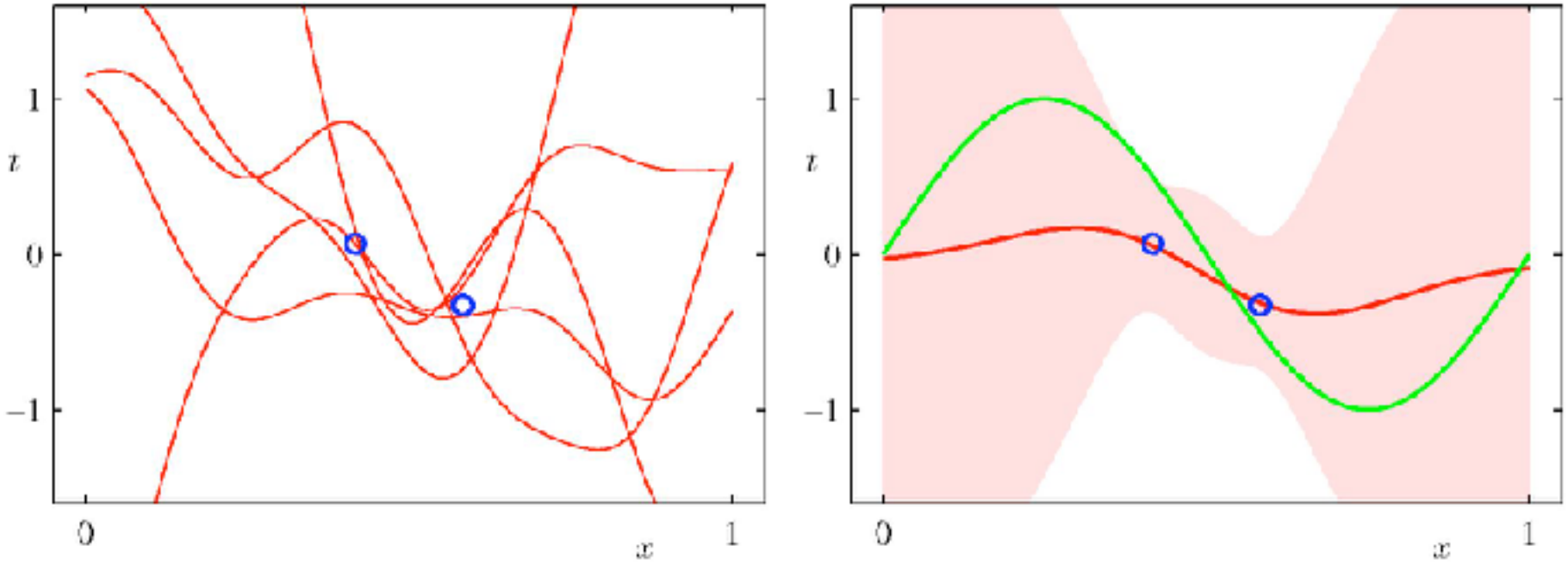
Bayesian linear regression - part II



Copyright C.M. Bishop, PRML

- Regression with (extremely) small and noisy dataset
- Many functions are compatible with data

Bayesian linear regression - part II



Copyright C.M. Bishop, PRML

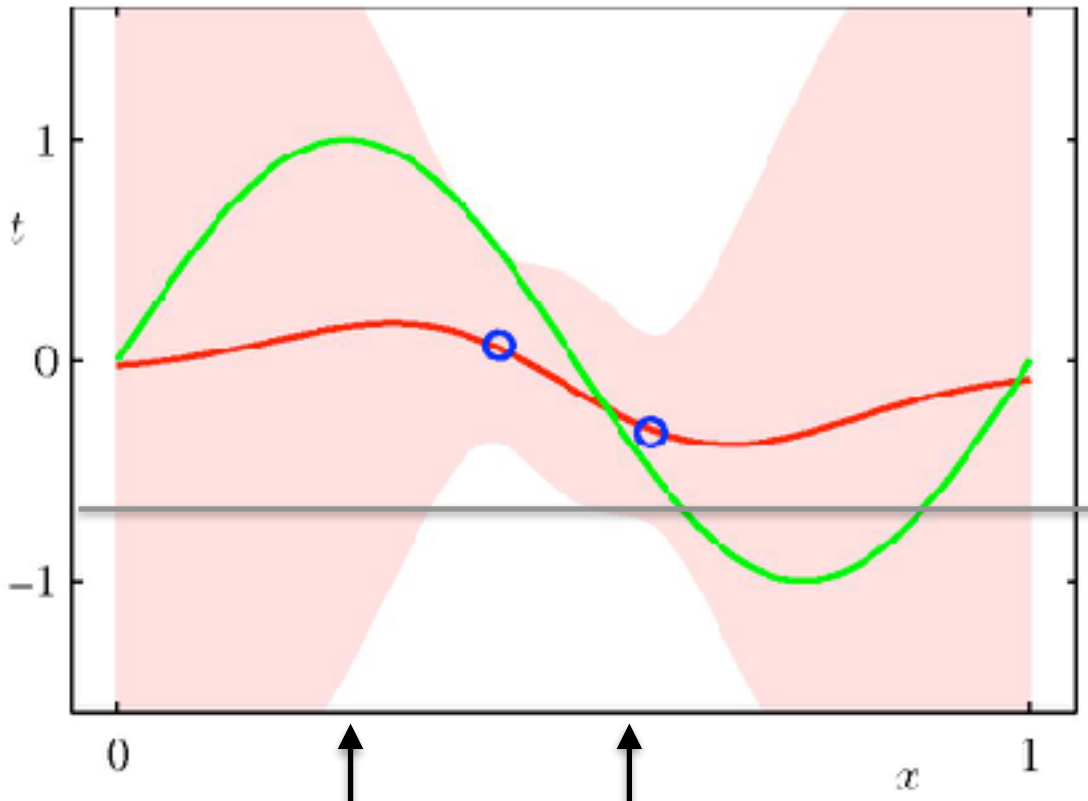
- Quantify the uncertainty using probabilities

  (e.g. Gaussian mean and variance for every input x)

Decision making

- What to do with the predictive distribution?
- Knowing uncertainty of output helpful in decision making
- Consider inspecting task.
  - x: some measurement
  - y: predicted breaking strength
- Parts which are to weak (breaking strength < t) are rejected
  - Falsely rejecting a part incurs a small cost (c=1)
  - Falsely accepting a part can cause more damage down the line (expected cost c=100)

Decision making



threshold

should we accept this part?

how about this one?

Copyright C.M. Bishop, PRML

Determining uncertainty

- To make good decisions, sometimes need to know uncertainty
- Sources of uncertainty:
  - We do not know the parameters **w**, especially in areas where we have little data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$

$$p(\mathbf{w}|\mathcal{D}) = \hat{\mathbf{w}} + \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$$

  - Even if we knew the parameters **w** of the underlying function, individual parts might be slightly offset from this function

$$p(y|\mathbf{x}, \mathbf{w}) = f(\mathbf{w}, \mathbf{x}) + \mathcal{N}(0, \sigma^2)$$

Determining uncertainty

- To make good decisions, sometimes need to know uncertainty
- Sources of uncertainty:
  - We do not know the parameters **w**, especially in areas where we have little data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$

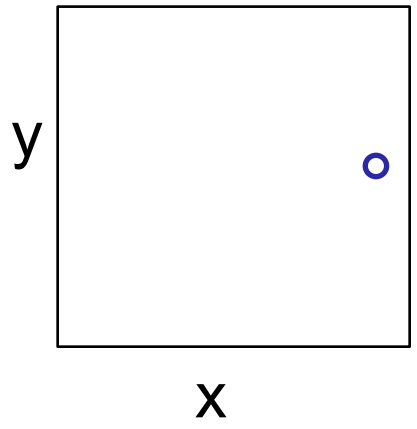
    1: determine $p(\mathbf{w}|\mathcal{D}) = \hat{\mathbf{w}} + \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$
  - Even if we knew the parameters **w** of the underlying function, individual parts might be slightly offset from this function

    $p(y|\mathbf{x}, \mathbf{w}) = f(\mathbf{w}, \mathbf{x}) + \mathcal{N}(0, \sigma^2)$

    *2: combine these predictions for all **w***

Step 1: Determine posterior

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

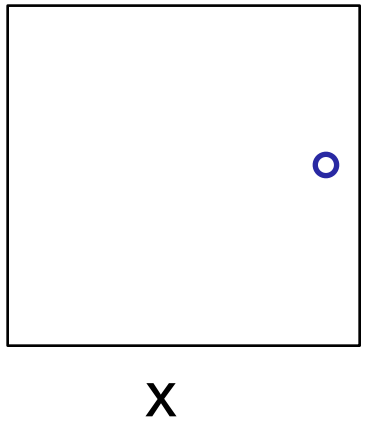


Step 1: Determine posterior

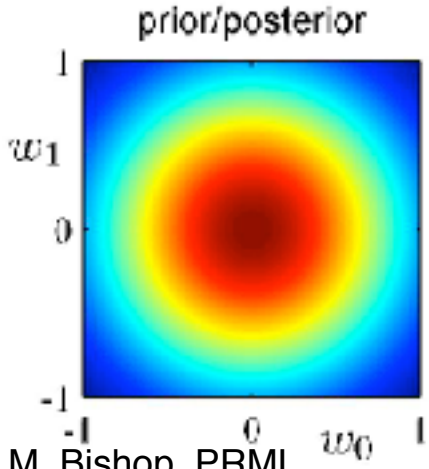
- Goal: fit lines $y = w_0 + w_1 x + \epsilon$

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

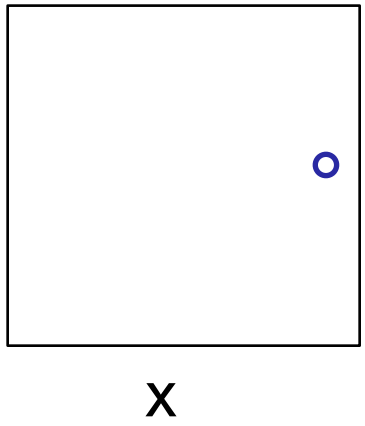


Step 1: Determine posterior

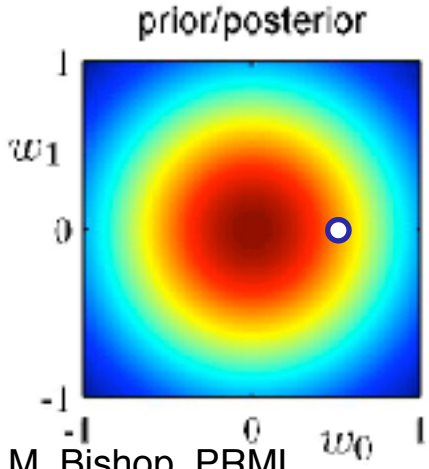
- Goal: fit lines $y = w_0 + w_1 x + \epsilon$

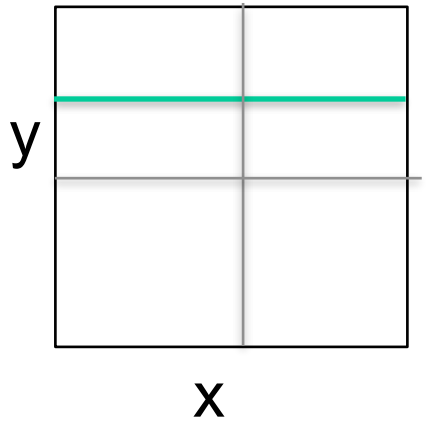
What prior?

- 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



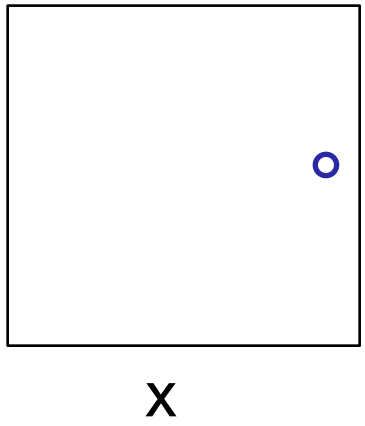


Step 1: Determine posterior

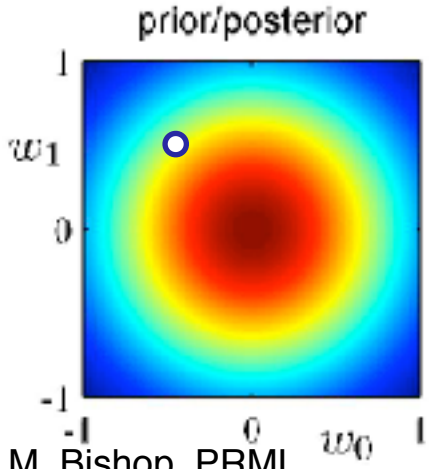
- Goal: fit lines $y = w_0 + w_1 x + \epsilon$

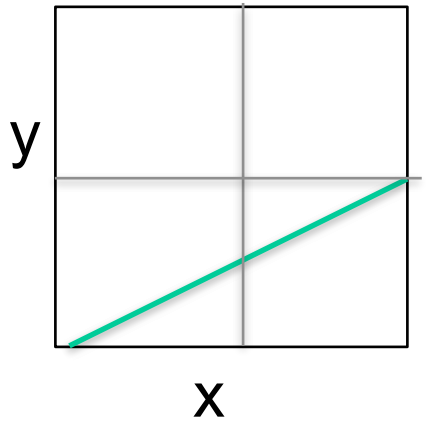
What prior?

- 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



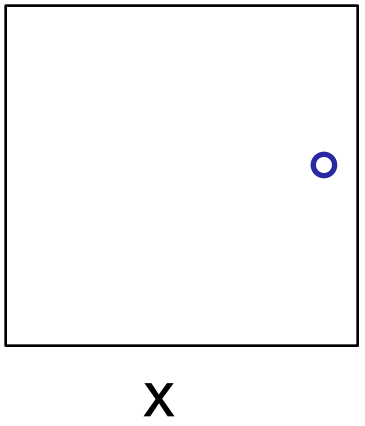


Step 1: Determine posterior

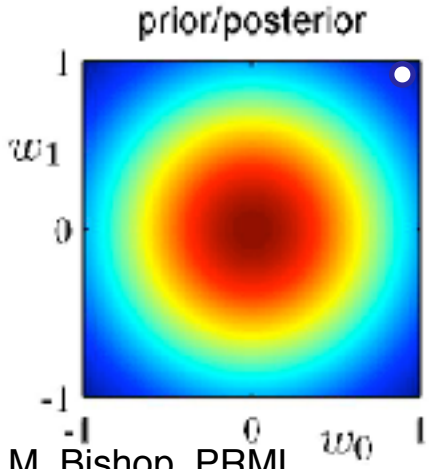
- Goal: fit lines $y = w_0 + w_1 x + \epsilon$

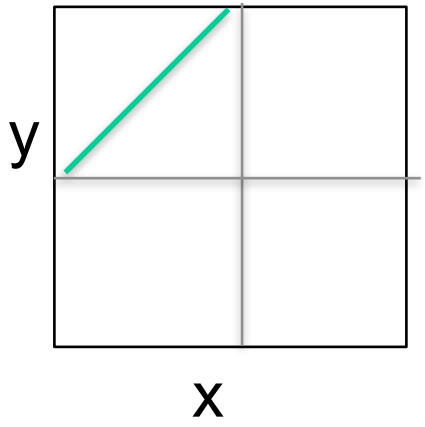
What prior?

- 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



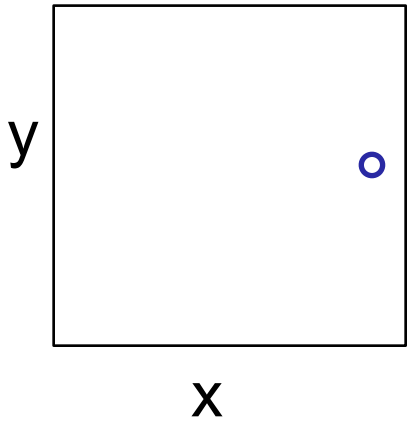


Step 1: Determine posterior

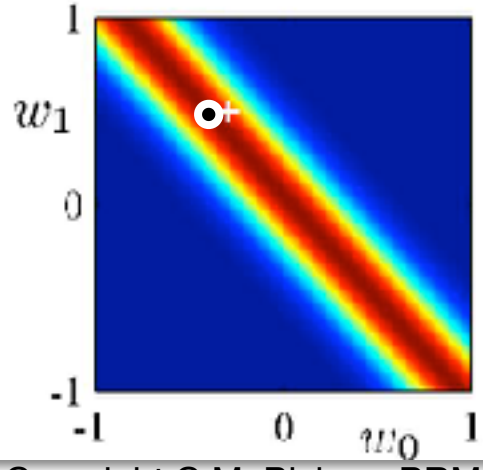
- Goal: fit lines $y = w_0 + w_1 x + \epsilon$

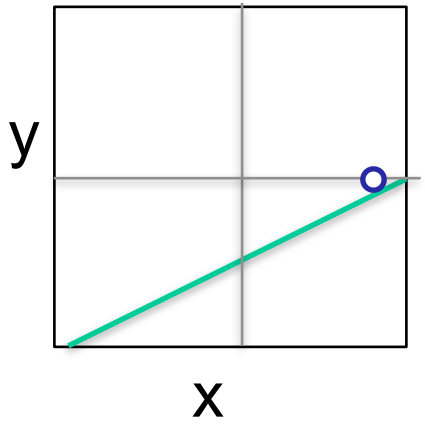
What likelihood?

- 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



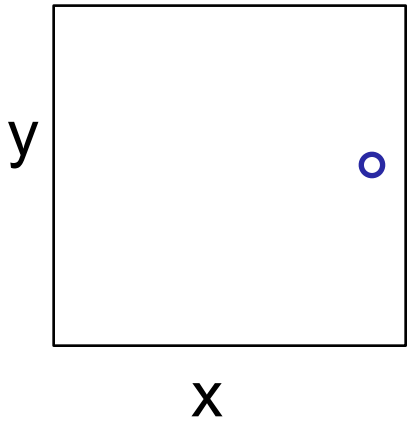


Step 1: Determine posterior

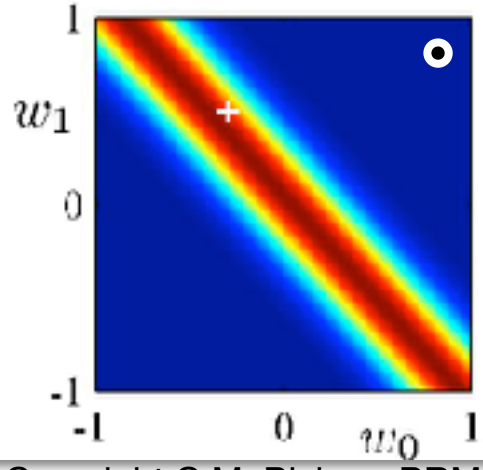
- Goal: fit lines $y = w_0 + w_1 x + \epsilon$

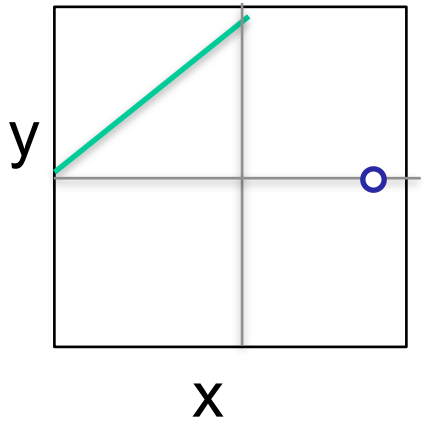
What likelihood?

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



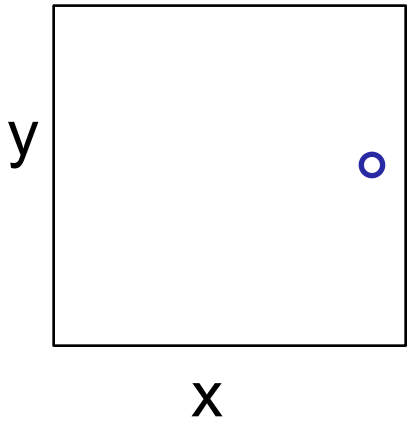
- Good lines should pass 'close by' datapoint



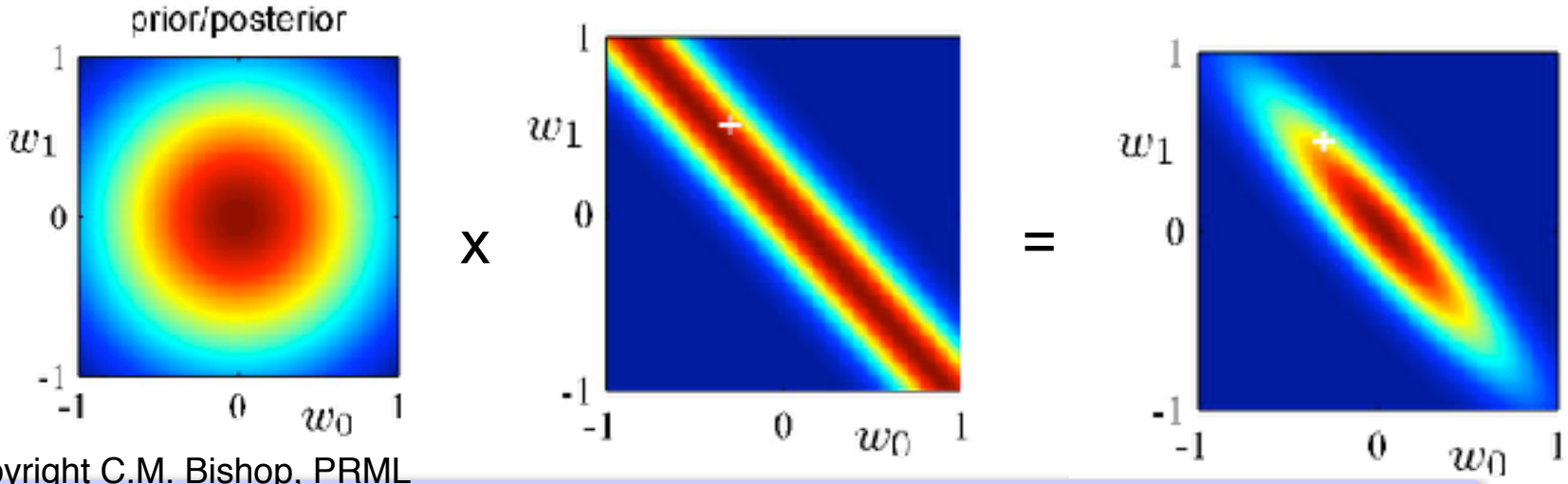


Step 1: Determine posterior

- 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)



Determining uncertainty

- To make good decisions, sometimes need to know uncertainty
- Sources of uncertainty:
  - We do not know the parameters **w**, especially in areas where we have little data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$

    1: determine $p(\mathbf{w}|\mathcal{D}) = \hat{\mathbf{w}} + \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$
  - Even if we knew the parameters **w** of the underlying function, individual parts might be slightly offset from this function

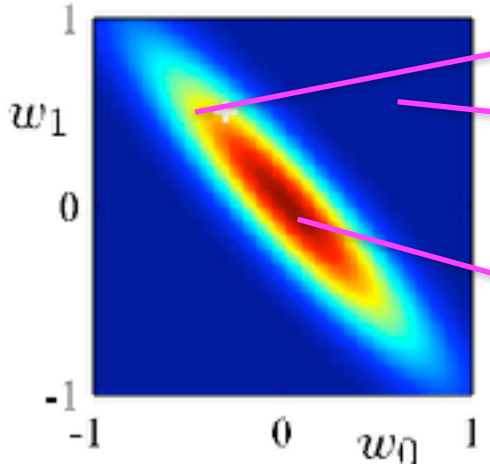
    $p(y|\mathbf{x}, \mathbf{w}) = f(\mathbf{w}, \mathbf{x}) + \mathcal{N}(0, \sigma^2)$

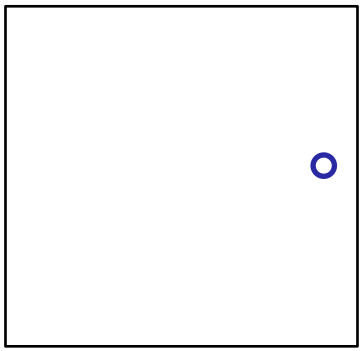
    *2: combine these predictions for all **w***

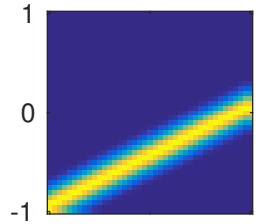
Step 2: Combine predictions

- Every **w** makes a prediction

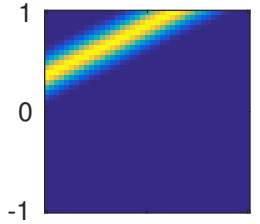
$$y = w_0 + w_1 x + \epsilon$$



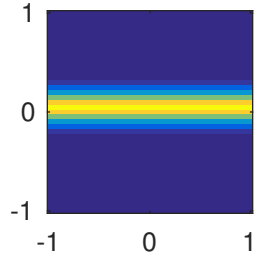




x medium weight



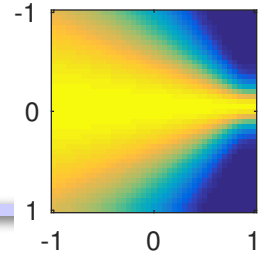
x low weight



x high weight

…

+



Bayesian linear regression in general

- Model:
  - Likelihood
    $$p(y|\mathbf{x}, \mathbf{w}) = \mathcal{N}(\mathbf{w}^T\mathbf{x}, \sigma^2)$$
  - Conjugate prior
    $$p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \alpha^{-1}\mathbf{I})$$
  - Prior precision α and noise variance σ^2 considered known
  - Linear regression with uncertainty about the parameters

Bayesian linear regression: inference

- Some algebra on the model definitions gives the solution

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

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

- $\mathbf{X}$ 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

Bayesian linear regression: inference

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

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

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

Bayesian linear regression: inference

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

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

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

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

Ridge regression, Lecture 4 (linear regression)

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

prior precision

likelihood variance

Bayesian linear regression: prediction

- Prediction for new datapoint:

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

- Convolution of two 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 **w** from before

from weight uncertainty

new input

from observation noise

- Variance tends to go down with more data until it reaches σ^2
- Corresponds to sources of uncertainty discussed before

Beyond linear regression

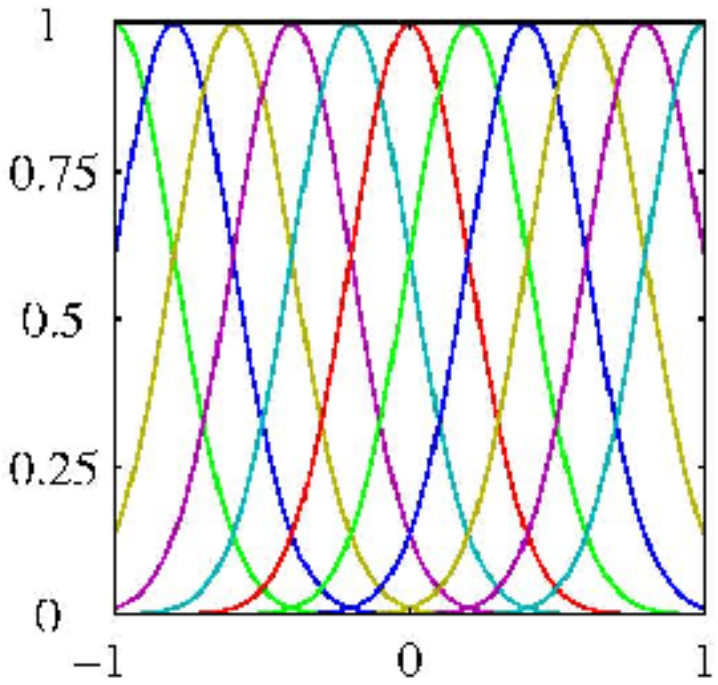
- Non-linear data sets can be handled by using non-linear features
- Features specify the class of functions we consider (hypothesis class)

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

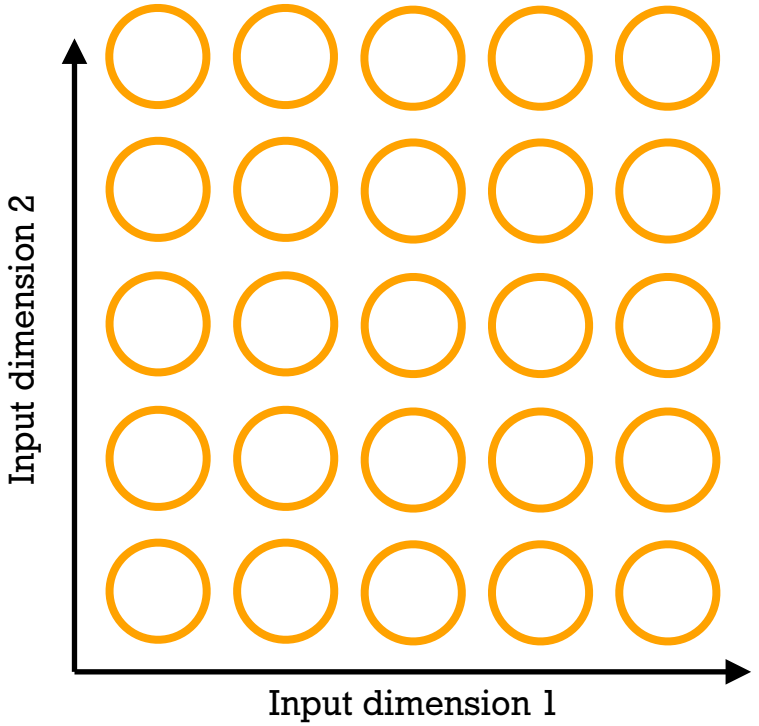
- What if we do not know good features?

Beyond linear regression

- Certain features work with many problems



Copyright C.M. Bishop, PRML



Beyond linear regression

- Scaling with number of inputs
  - Grid of radial basis functions

    k RBFs per dimension, m-dimensional input?

  - Polynomial expansion

    order k polynomial, m-dimensional input?

Beyond linear regression

- Scaling with number of inputs
  - Grid of radial basis functions

    k RBFs per dimension, m-dimensional input?

    k^m features
  - Polynomial expansion

    order k polynomial, m-dimensional input?

    m^k features (+ lower order terms)

Beyond linear regression

- Relying on features can be problematic
- We tried to avoid using features before…

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!

- We can use a similar approach with (Bayesian) linear regression

Kernels (recap)

- Kernel is a function of two arguments which corresponds to a dot product in some feature space

$$k(\mathbf{x}, \mathbf{y}) = \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{y})$$

- Advantage of using kernels:
  - Sometimes evaluating k is cheaper than evaluating features and taking the dot product $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y})^d$
  - Sometimes k corresponds to an inner product in a feature space with infinite dimensions $k(\mathbf{x}, \mathbf{y}) = \exp(-(\mathbf{x} - \mathbf{y})^2)$

Kernels (recap)

- Kernelize algorithm:
  - Try to formulate algorithm so feature vectors only ever occur in inner products
  - Replace inner products by kernel evaluations **(kernel trick)**

- Sidenote: different kernel definitions are used in different methods. Here, ’Mercer kernels’ are used.

Kernelizing the mean function

- Inspect solution mean from Bayesian linear regression

$$p(y^*|\mathcal{D}) = \mathcal{N}(\sigma^{-2}\mathbf{x}^{*T}\mathbf{S}_N\mathbf{X}^T\mathbf{y}, \sigma^2 + \mathbf{x}^T\mathbf{S}_N\mathbf{x}) \quad (1)$$

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

- Vector $\mathbf{y}$ concatenates training outputs
- Matrix $\mathbf{X}$ has one column for each feature (length N)

  one row for each datapoint (length M)
- Mean prediction is

$$y^* = \sigma^{-2}\mathbf{x}^{*T}(\alpha\mathbf{I} + \sigma^{-2}\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Kernelizing the mean function

- Step 2: Reformulate to only have inner products of features

$$y^* = \sigma^{-2}\mathbf{x}^{*T}(\alpha\mathbf{I} + \sigma^{-2}\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

If $\mathbf{P}, \mathbf{R}$ are positive definite, then

$$(\mathbf{P}^{-1} + \mathbf{B}^T\mathbf{R}^{-1}\mathbf{B})^{-1}\mathbf{B}^T\mathbf{R}^{-1} = \mathbf{P}\mathbf{B}^T(\mathbf{B}\mathbf{P}\mathbf{B}^T + \mathbf{R})^{-1}$$

$$y^* = \underbrace{\sigma^{-2}\mathbf{x}^{*T}\mathbf{X}^T}_{\mathbf{k}(\mathbf{x}^*)^T}(\alpha\mathbf{I} + \underbrace{\sigma^{-2}\mathbf{X}\mathbf{X}^T}_{\mathbf{K}})^{-1}\mathbf{y}$$

element i,j of this matrix is $\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$

element i of this vector is $\phi(\mathbf{x}_i)^T\phi(\mathbf{x}^*)$

Kernelizing the mean function

- Step 2: Reformulate to only have inner products of features

$$y^* = \sigma^{-2}\mathbf{x}^{*T}(\alpha\mathbf{I} + \sigma^{-2}\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

features x #features

datapoints x #datapoints

$$y^* = \underbrace{\sigma^{-2}\mathbf{x}^{*T}\mathbf{X}^T}_{\mathbf{k}(\mathbf{x}^*)^T}(\alpha\mathbf{I} + \underbrace{\sigma^{-2}\mathbf{X}\mathbf{X}^T}_{\mathbf{K}})^{-1}\mathbf{y}$$

element i,j of this matrix is $\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$

element i of this vector is $\phi(\mathbf{x}_i)^T\phi(\mathbf{x}^*)$

Kernelizing the mean function

- Step 3: Replace inner products by kernel evaluations

$$y^* = \mathbf{k}(\mathbf{x}^*)^T(\alpha\mathbf{I} + \mathbf{K})^{-1}\mathbf{y}$$

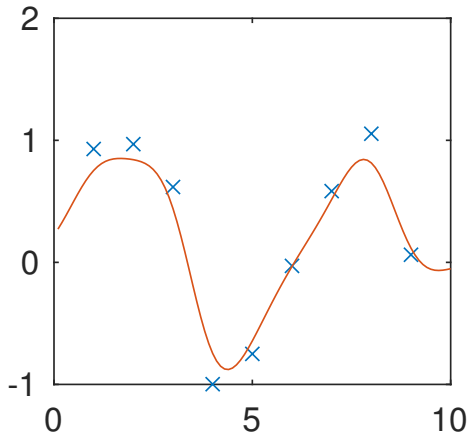
element i,j of this matrix is $k(\mathbf{x}_i, \mathbf{x}_j)$

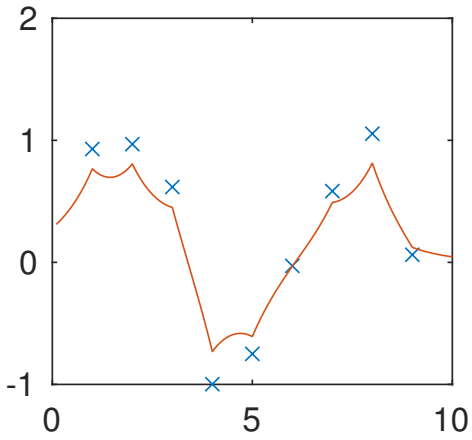
element i of this vector is $k(\mathbf{x}_i, \mathbf{x}^*)$

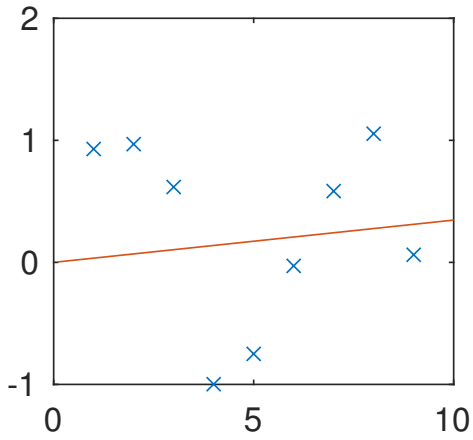
- Remember: Mean function is same as ridge regression
- This is **kernel** ridge regression

Kernel ridge regression

- Choosing a kernel







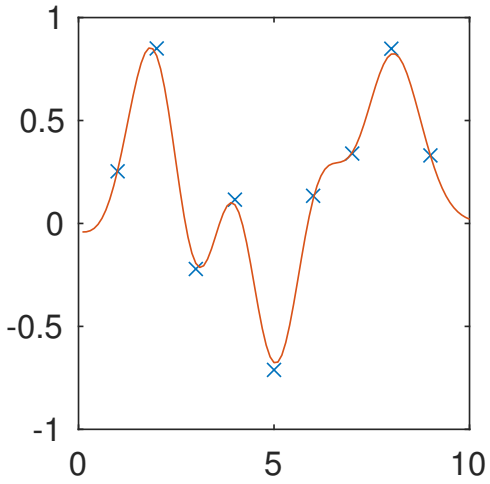
$$k(x,y) = \exp -(x-y)^2$$

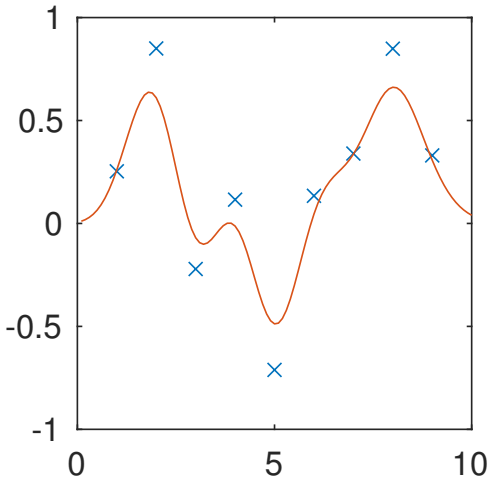
$$k(x,y) = xy$$

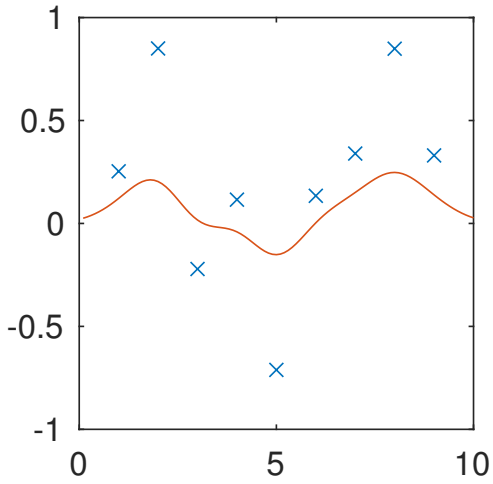
$$k(x,y) = \exp -|x-y|$$

Kernel ridge regression

- Setting parameters



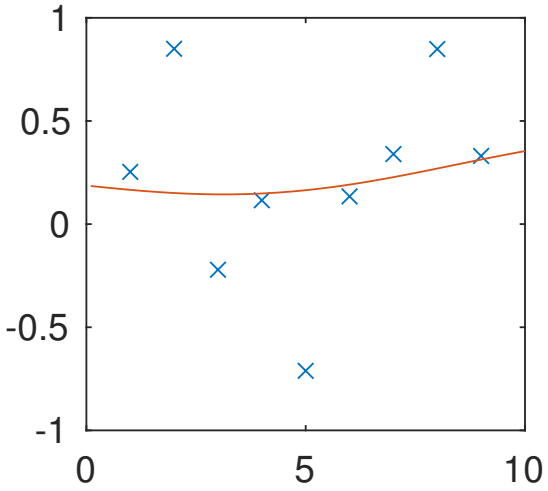




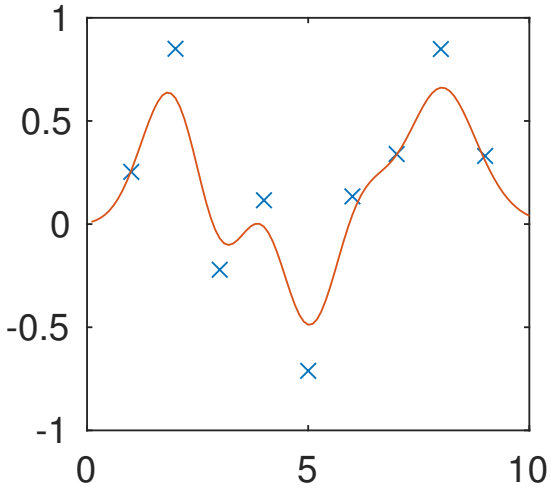
$\lambda = 0.03$ $\lambda = 0.3$ $\lambda = 3$

Kernel ridge regression

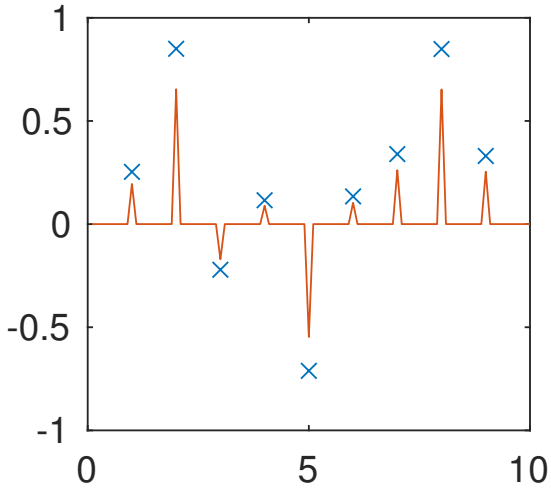
- Setting parameters



$\sigma = 10$



$\sigma = 1$

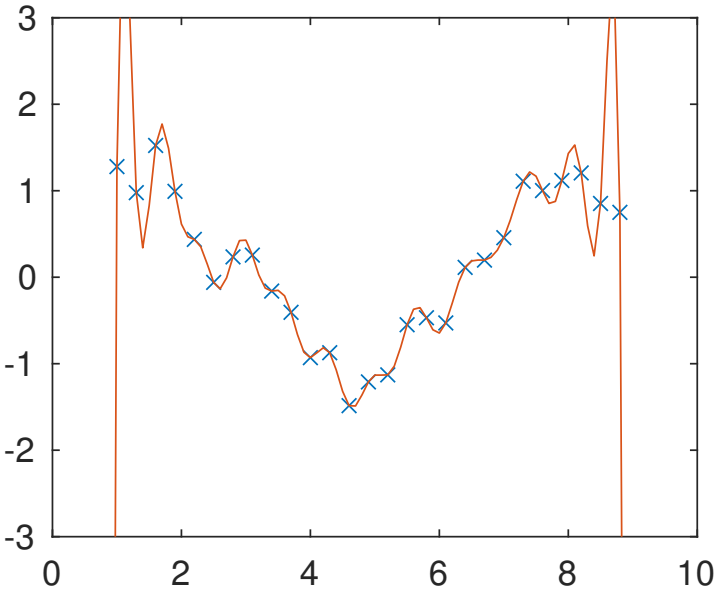


$\sigma = 0.1$

$$k(x, y) = \exp -\frac{(x - y)^2}{\sigma^2}$$

Why does it work

- We still have #features = #datapoints, so regularisation critical!



$$\lambda = 0$$

Kernel regression: Practical issues

- Compare ridge regression: $\mathbf{w} = (\lambda \mathbf{I} + \mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$

  inverse $O(d^3)$ matrix-vector product $O(d^2 N)$

  prediction $O(d)$

  memory $O(d)$

- Kernel ridge regression: $y^* = \mathbf{k}(\mathbf{x}^*)^T (\alpha \mathbf{I} + \mathbf{K})^{-1} \mathbf{y}$

  inverse, product $O(N^3)$

  prediction $O(N)$

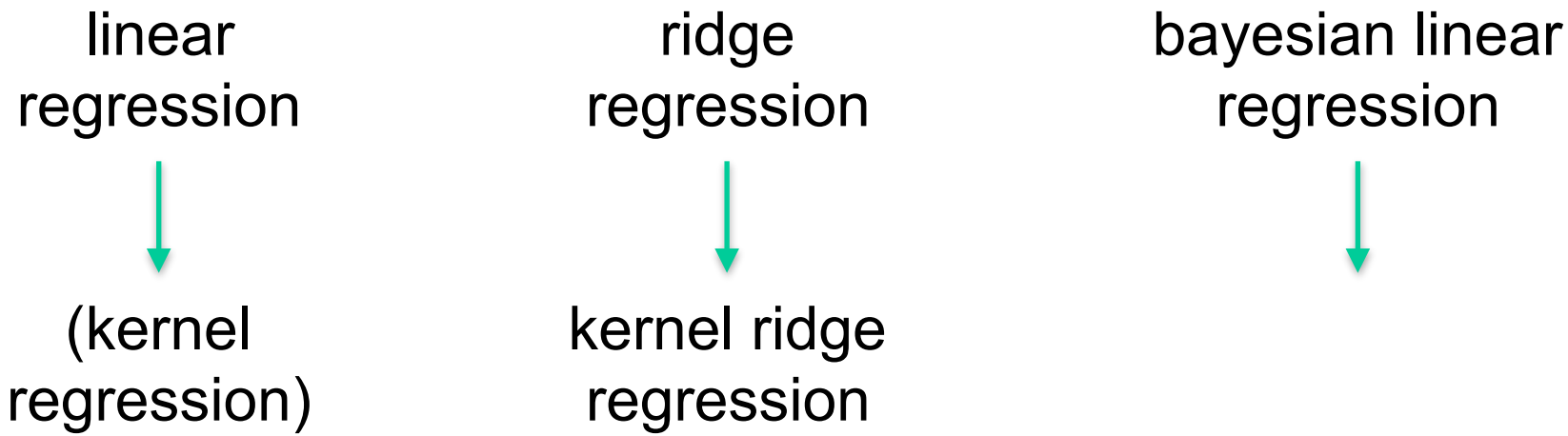
  memory $O(N)$

Kernel regression: Practical issues

- If we have a **small set** of **good features** it's faster to do regression in feature space
- However, if no good features are available (or we need a very big set of features), kernel regression might yield better results
- Often, it is easier to pick a kernel than to choose a good set of features

Kernelizing Bayesian linear regression

- We have now kernelized ridge regression
- Could we kernelize Bayesian linear regression, too?



Kernelizing Bayesian linear regression

- We have now kernelized ridge regression
- Could we kernelize Bayesian linear regression, too?
- Yes, and this is called a Gaussian process regression (GPR)

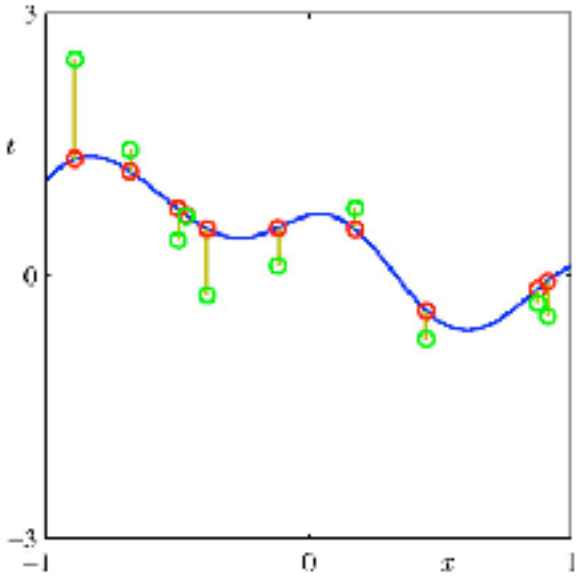
linear regression	ridge regression	bayesian linear regression
↓	↓	↓
(kernel regression)	kernel ridge regression	Gaussian process

Deriving GP equations

- Model:
  - We are interested in the function values $y_1, y_2, \ldots$, at a set of points $\mathbf{x}_1, \mathbf{x}_2, \ldots$. We observe target values for the training set, but we assume these are noisy $t_n = y_n + \epsilon$
  - Prior: $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$

    With **y** a vector of function values

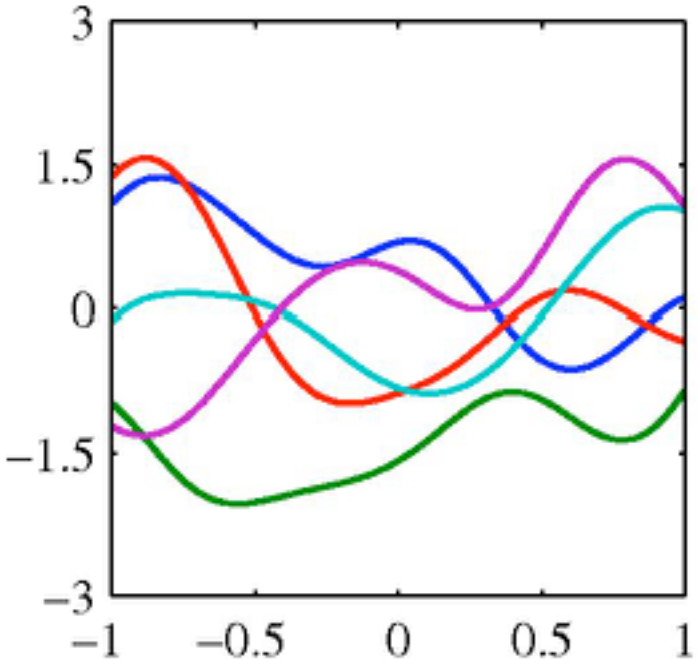
    and **K** the kernel matrix
  - Likelihood: $\mathbf{t} \sim \mathcal{N}(\mathbf{y}, \beta^{-1}\mathbf{I})$



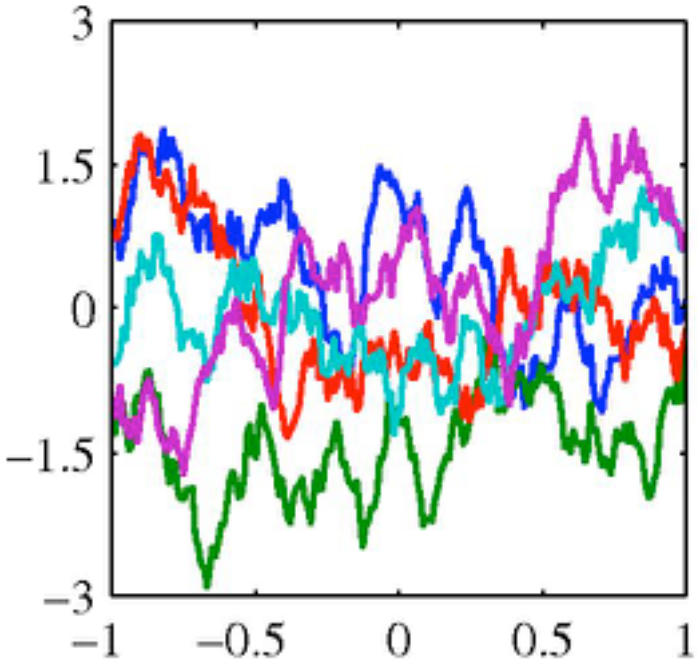
Copyright C.M. Bishop, PRML

Examples from the prior

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



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



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

GP Regression

- Prior and likelihood are Gaussian
- Again obtain a closed form solution

$$\mathbb{E}[y^*] = \mathbf{y}^T(\mathbf{K} + \beta^{-1}\mathbf{I})^{-1}\mathbf{k}(\mathbf{x}^*)$$

kernel ridge regression

$$\mathrm{Cov}[y^*] = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*)^T(\mathbf{K} + \beta^{-1}\mathbf{I})^{-1}\mathbf{k}(\mathbf{x}^*)$$

prior variance

reduction in variance due to close training points

- Prediction of new observations

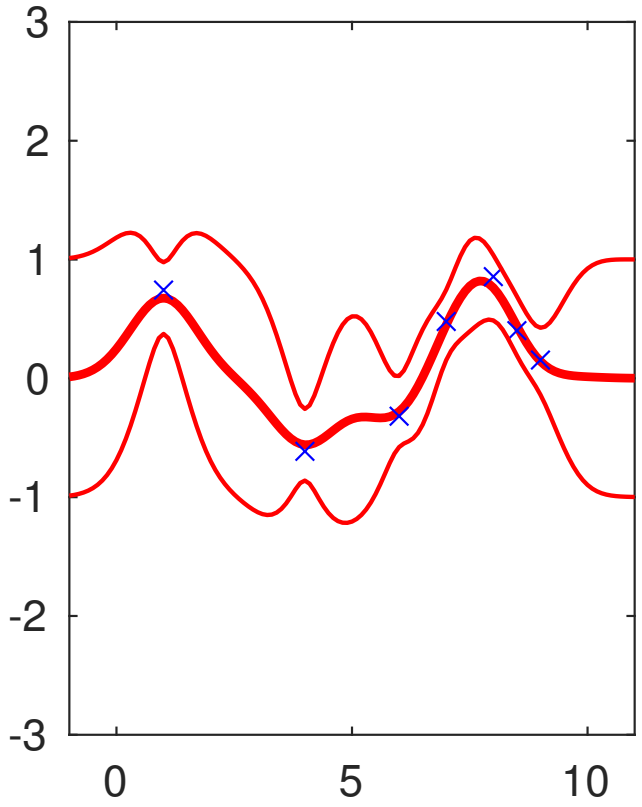
$$\mathrm{Cov}[t^*] = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*)^T(\mathbf{K} + \beta^{-1}\mathbf{I})^{-1}\mathbf{k}(\mathbf{x}^*) + \beta^{-1}$$

- Easy to implement!

add noise term

GP Regression

- Results of GP regression



$$\mathbb{E}[y^*|\mathbf{t}] + \sqrt{\mathrm{Cov}[y^*|\mathbf{t}]}$$

$$\mathbb{E}[y^*|\mathbf{t}]$$

$$\mathbb{E}[y^*|\mathbf{t}] - \sqrt{\mathrm{Cov}[y^*|\mathbf{t}]}$$

Calculated for many possible y^*

GP Regression: hyperparameters

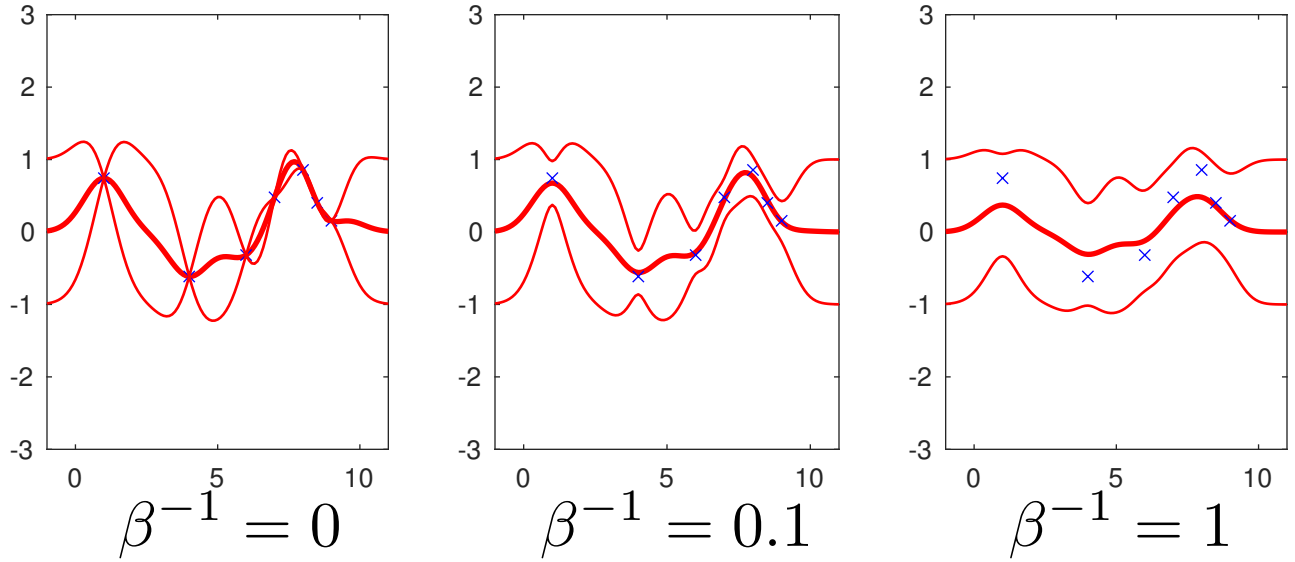
- Hyperparameters
  - Assumed noise (variance of likelihood) $\mathbf{t} \sim \mathcal{N}(\mathbf{y}, \beta^{-1}\mathbf{I})$
  - Any parameters of the kernel
    - Typical kernel:

$$k(\mathbf{x}_i, \mathbf{x}_j) = s^2 \exp -\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}$$

      - s: scale (standard deviation prior to seeing data)
      - σ : bandwidth (which datapoint are considered close)
  - Effective regularisation: $\beta^{-1} s^{-1}$
  - Knowing the 'meaning' of parameters helps tune them

GP Regression: hyperparameters

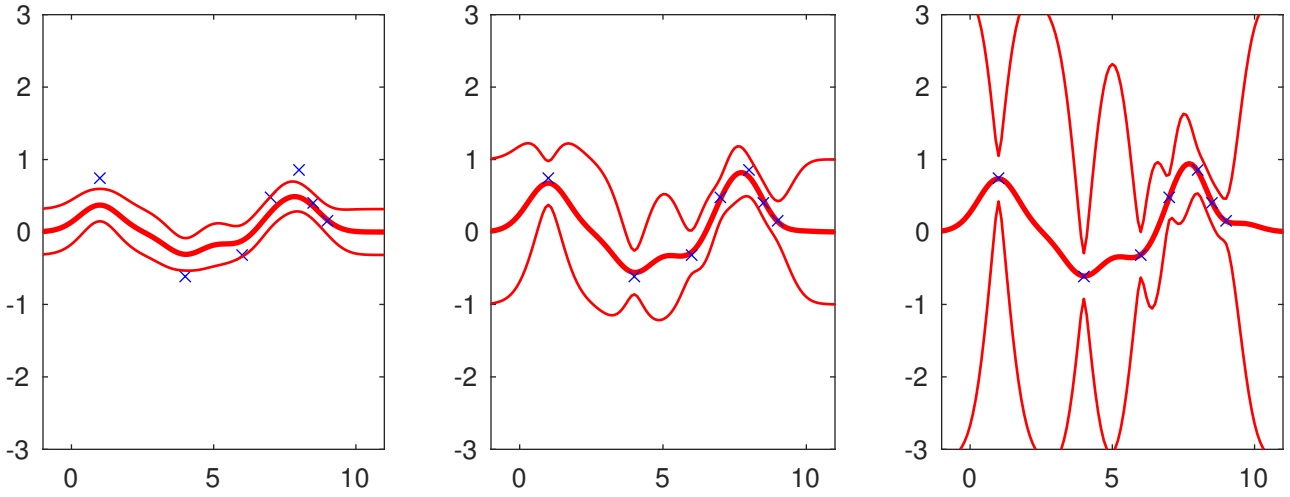
- Assumed noise (variance of likelihood) $\mathbf{t} \sim \mathcal{N}(\mathbf{y}, \beta^{-1}\mathbf{I})$
- Effective regularisation: $\beta^{-1} s^{-1}$



- Mostly changes behaviour close to train points

GP Regression: hyperparameters

- Kernel $k(\mathbf{x}_i, \mathbf{x}_j) = s^2 \exp -\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}$
- Effective regularisation $\beta^{-1} s^{-1}$

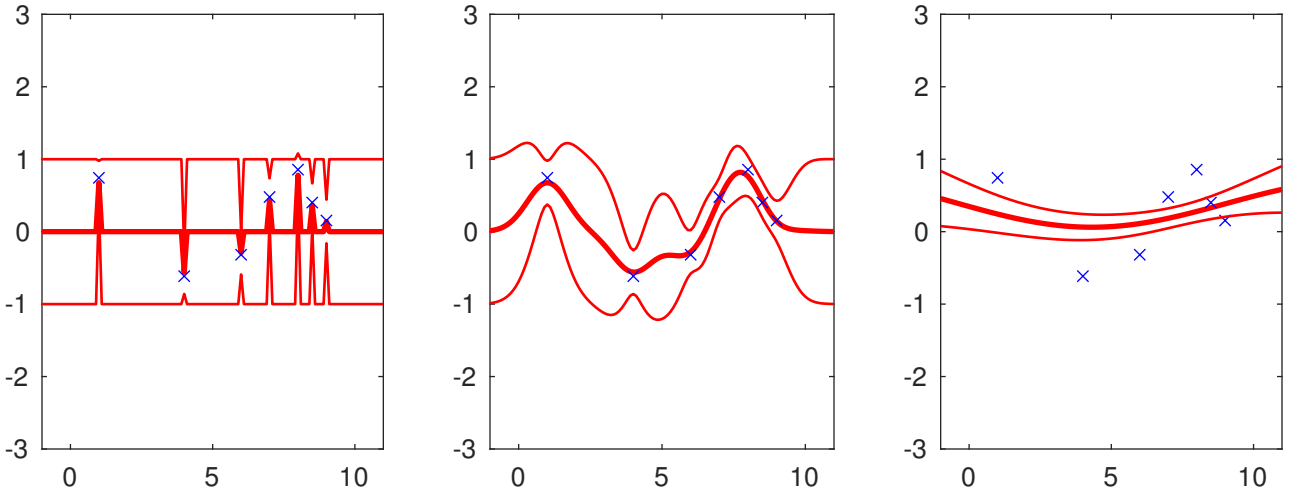


$s = 0.1$ $s = 1$ $s = 10$

- Mostly changes behaviour further away from training points

GP Regression: hyperparameters

- Kernel $k(\mathbf{x}_i, \mathbf{x}_j) = s^2 \exp -\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}$



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

- Changes what is considered ‘close’ or ‘far’

GPs: Practical issues

- Complexity pretty much similar to kernel regression
- Except for calculating predictive variance

$$\mathbb{E}[y^*] = \mathbf{y}^T(\mathbf{K} + \beta^{-1}\mathbf{I})^{-1}\mathbf{k}(\mathbf{x}^*)$$

$$\mathrm{Cov}[y^*] = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*)^T(\mathbf{K} + \beta^{-1}\mathbf{I})^{-1}\mathbf{k}(\mathbf{x}^*)$$

- inverse, product $O(N^3)$
- prediction ~~$O(N)$~~ $O(N^2)$
- memory $O(N)$

GPs: Practical issues

- For small dataset, GPR is a state-of-the-art method!
  - Advantage: uncertainty, flexible yet can control overfitting
  - Computational costs acceptable for small datasets (<10 000)
- For large datasets, uncertainty not so important, expensive, ….
- Good approximations exist

- Specifying the right prior (kernel!) is important!
- This means choosing good hyper parameters
- Choice of hyper parameters investigated in next lecture

Bayesian methods in practice

- Time complexity varies compared to frequentist methods
- Memory complexity can be higher
  - e.g. need to store mean + uncertainty : quadratic, not linear
- Much data everywhere: posterior close to point estimate
  - (might as well use frequentist methods)
- Little data everywhere
  - Prior information helps bias/variance trade-off
- Some areas with little, some areas with much data
  - Uncertainty helps to decide where predictions are reliable

Inference in more complex models

- We saw some examples with closed-form posterior
- In many complex models, no closed-form representation
- **Variational inference** (deterministic)
  - Consider family of distributions we **can** represent (Gaussian)
  - Use optimisation techniques to find best of these
- **Sampling** (stochastic)
  - Try to directly sample from the posterior
  - Expectations can be approximated using the samples
- **Maximum a posterior** (point estimate)

What you should know

- Last lecture:
  - What is the Bayesian view of probability?
  - Why can the Bayesian view be beneficial?
  - Role of the following distributions:
    - Likelihood, prior, posterior, posterior predictive
- This lecture:
  - Key idea of Bayesian regression and its properties
  - Key idea of kernel regression and its properties
  - Key idea of Gaussian process regression and its properties

Why does it work?

- In linear regression, we choose M basis functions, then find the best function of the form

$$y = \sum_{i=1}^{M} w_i \boldsymbol{\phi}(\mathbf{x})$$

- In kernel regression, we find the best function of the form

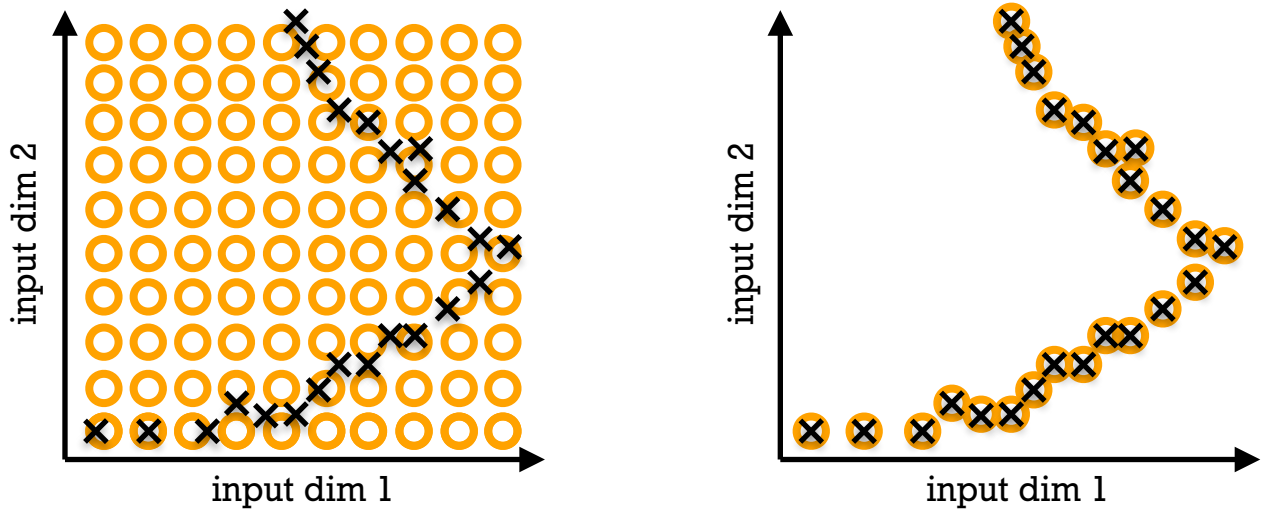
$$y = \sum_{\mathbf{s} \in \mathcal{S}} \alpha_{\mathbf{s}} k(\mathbf{x}, \mathbf{s})$$

  For any set $\mathcal{S}$

- Basis function $k(\cdot, \mathbf{s})$ available at every possible input

Why does it work?

- Basis function $k(\cdot, \mathbf{s})$ available at every possible input
- But only function at the training points are used to minimise training error (representer theorem, Schölkopf et al., 2001)



Why called Gaussian process?

Stochastic process

Collection of random variables indexed by some set

Ie. R.V. y_i for every element i in index set

Here, we will consider the function values y_i as random variables

The index is the function argument, e.g. $\in \mathbb{R}^d$

Furthermore, we assume that any subset of these random variables is jointly Gaussian distributed - thus defining a Gaussian process

This same assumption underlies Bayesian linear regression!