Bayesian Inference: Principles and Practice

1. Introduction to Bayesian Inference

Mike Tipping

Microsoft Research *Cambridge, UK*

Global Overview

1. Introduction to Bayesian Inference

2. Bayesian Inference: Marginalisation Rules!

3. Sparse Bayesian models (the “relevance vector machine”)

4. Further adventures with sparse Bayes

Lecture 1: Overview

- Bayesian inference
  - Basic philosophy
  - Advantages
  - Disadvantages
  - Mathematical framework

- Step-by-step: controlling model complexity in an example regression problem:
  - Via ‘classical’ techniques with penalty functions
  - Via Bayesian inference with priors

Principles of Bayesian Inference

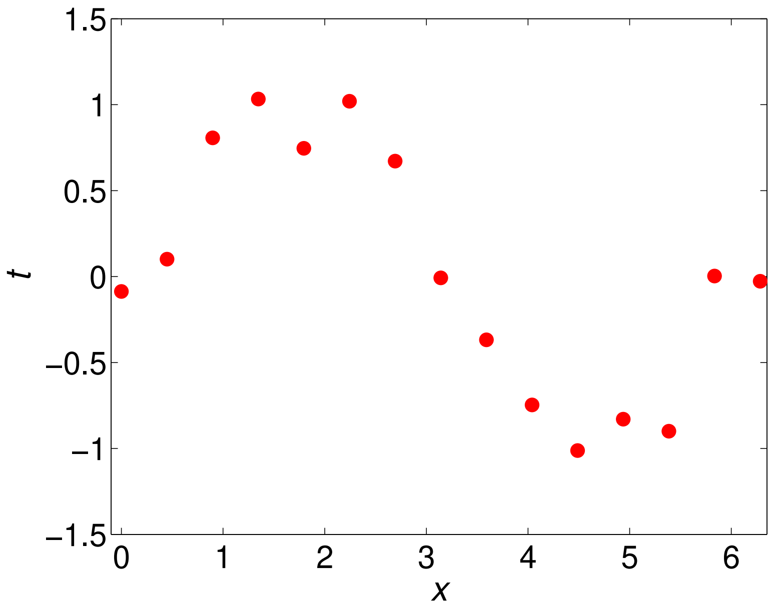
- Basic philosophy:
  - Define probability distributions over *all* quantities within the model
  - Update distributions in light of data using Bayes' rule
  - *Integrate out* variables not directly of interest for making predictions
- Main features:
  - A highly principled way to deal with all sources of uncertainty
  - An explicit framework for encoding prior knowledge
  - Automatic implementation of "Ockham's Razor"
- Limitation:
  - Most desired integral calculations are analytically intractable

Applying Bayesian Inference

- Simply implemented. Required tools are:
  - Product rule of probability: $p(a, b) \equiv p(a|b)\, p(b)$
  - Sum (integral) rule of probability: $p(a) \equiv \int p(a, b)\, db$
  - From the product rule: Bayes's theorem $p(b|a) \equiv p(a|b)\, p(b)/p(a)$
- Helpful tools:
  - Familiarity with computing $\int p(a, b)\, db$ when $p(a|b)$ and $p(b)$ are Gaussian
  - Familiarity with techniques for approximating integrals
- Approximating integrals is the Bayesian "voodoo"

An Example Learning Problem

- We have a data set comprising $N = 15$ samples generated from the function $y = \sin(x)$ with added Gaussian noise of variance 0.2:



- The 'input' variables are denoted x_n, $n = 1 \ldots N$

- For each x_n, there is an associated real-valued 'target' t_n, $n = 1 \ldots N$

Linear (in-the-parameter) Models

- We will model this data with some parameterised function $y(x;\mathbf{w})$, where $\mathbf{w} = (w_1, w_2, \ldots, w_M)$ is the vector of adjustable model parameters

- Here, we consider models which are a linearly-weighted sum of M fixed basis functions $\phi_m(x)$:

$$y(x;\mathbf{w}) = \sum_{m=1}^{M} w_m \phi_m(x)$$

- *e.g.* Gaussian data-centred functions: $\phi_m(x) = \exp\left\{-(x - x_m)^2/r^2\right\}$
  - A "radial basis function" (RBF) type model

“Least-squares” Approximation

- Our mission is to find values for $\mathbf{w}$ such that $y(x; \mathbf{w})$ makes good predictions for new data: *i.e.* it models *the underlying generative function*

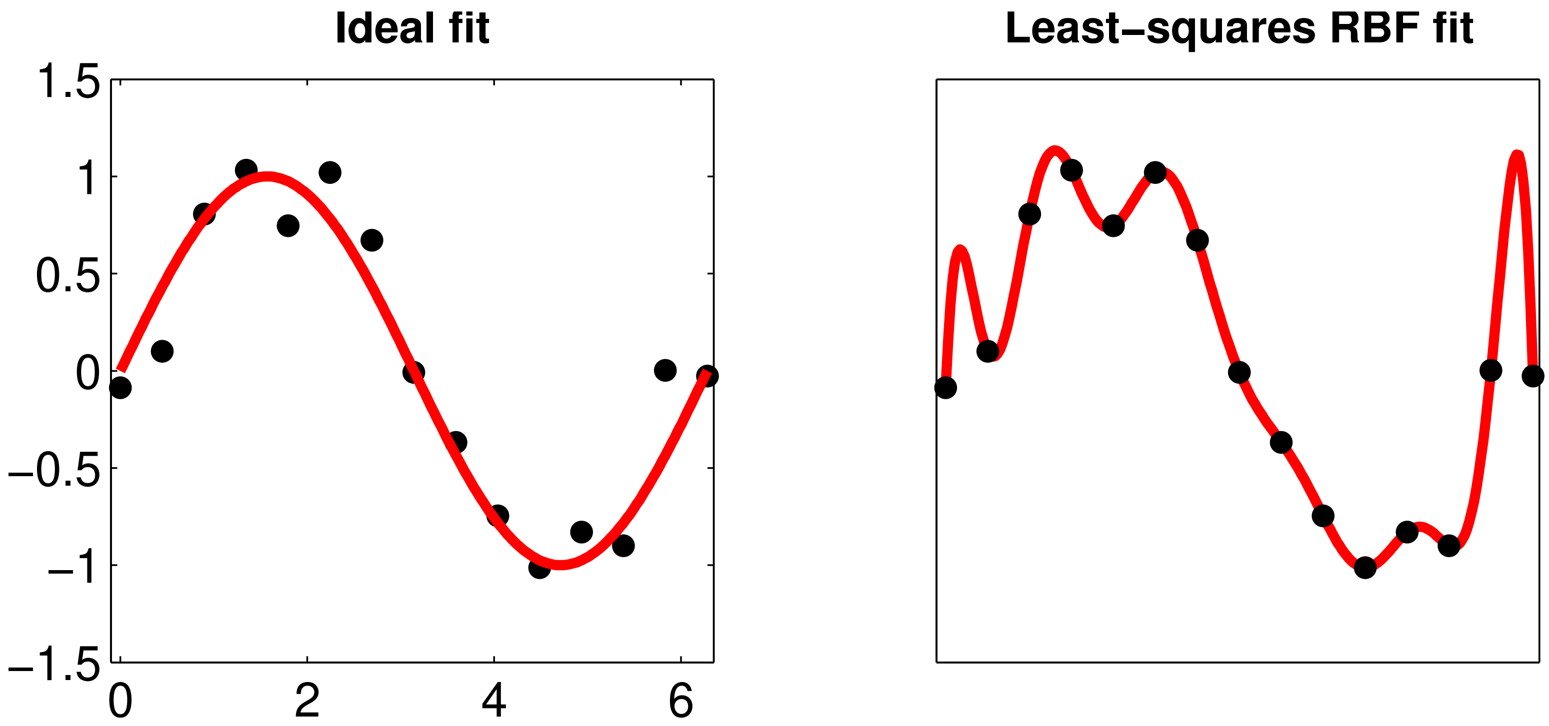
- A classic approach is “least-squares”, minimising the error measure:

$$E_{\mathcal{D}}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \left[t_n - \sum_{m=1}^{M} w_m \phi_m(x_n) \right]^2$$

- If $\mathbf{t} = (t_1, \ldots, t_N)^\mathsf{T}$ and Φ is the ‘design matrix’ such that $\Phi_{nm} = \phi_m(x_n)$, then:

$$\mathbf{w}_{LS} = (\Phi^\mathsf{T}\Phi)^{-1}\Phi^\mathsf{T}\mathbf{t}$$

- With $M = 15$ basis functions and only $N = 15$ examples, minimisation of squared-error leads to "over-fitting":



- Without prior knowledge of the truth, how do we judge which model is better?

Complexity Control: Regularisation

- We typically prefer smoother functions, which typically have smaller weights **w**

- Add a weight penalty term to the error function that we minimise:

$$\widehat{E}(\mathbf{w}) = E_{\mathcal{D}}(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

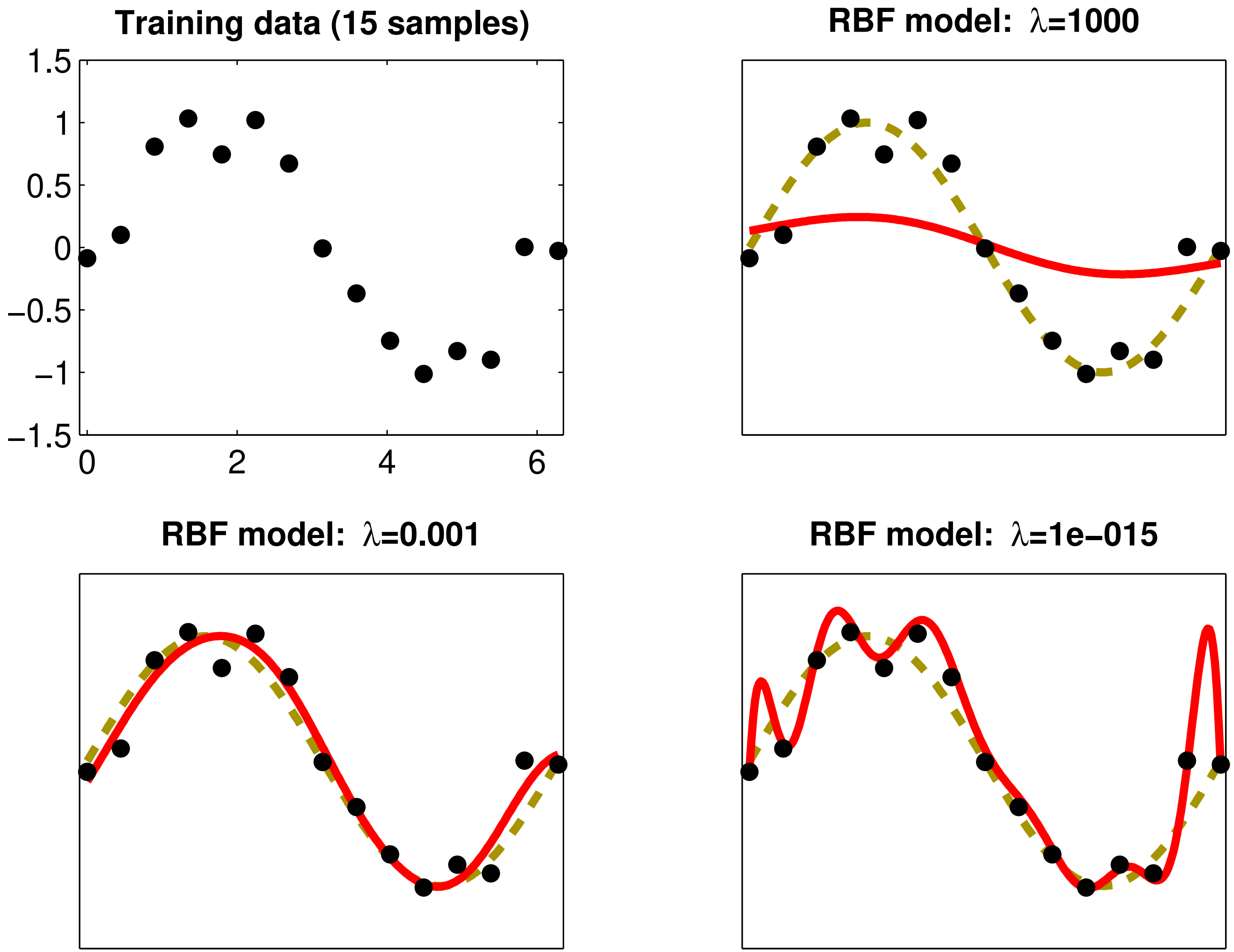
- A standard choice is the squared-weight penalty:

$$E_W(\mathbf{w}) = \frac{1}{2}\sum_{m=1}^{M} w_m^2$$

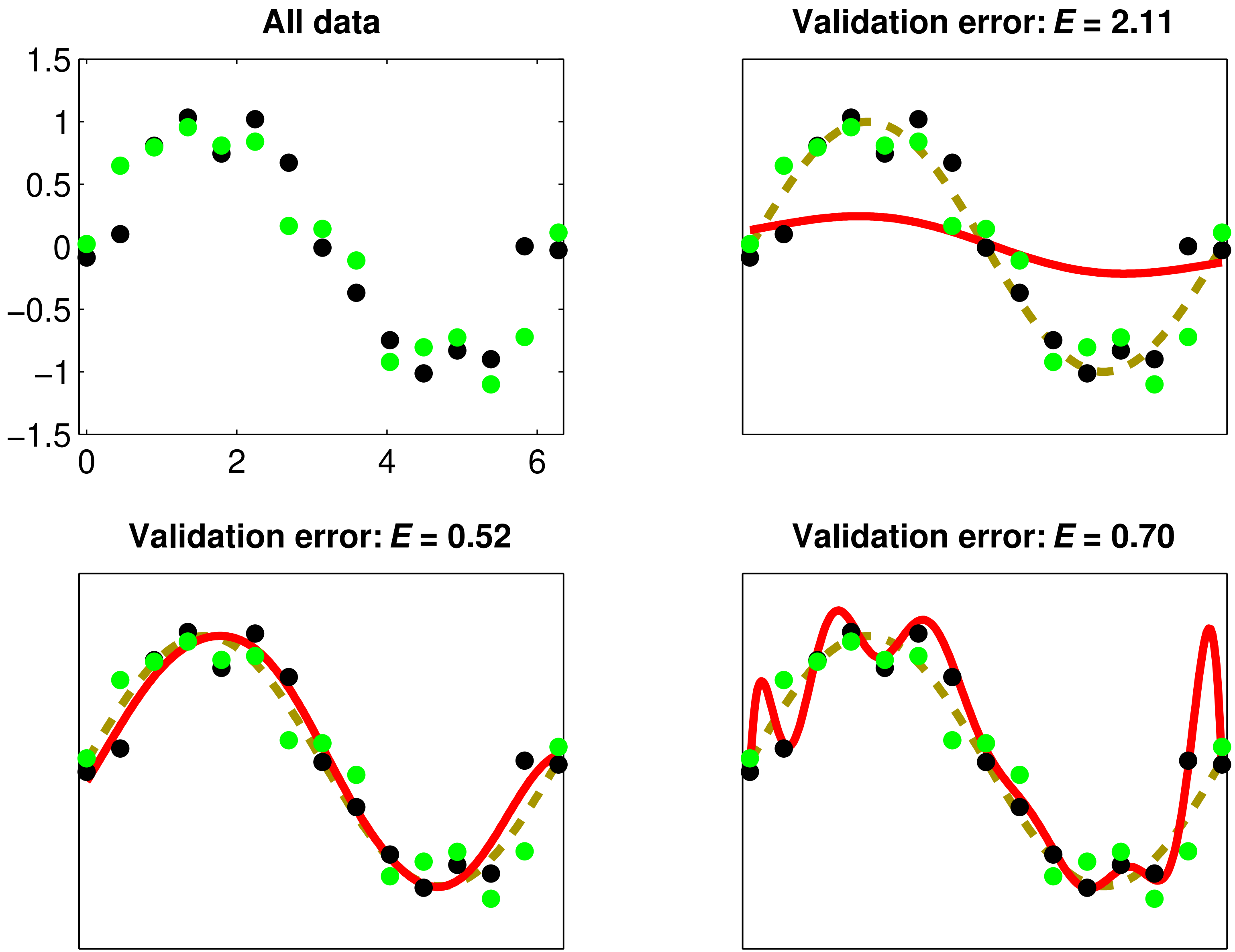
- This conveniently gives the “penalised least-squares” (PLS) estimate:

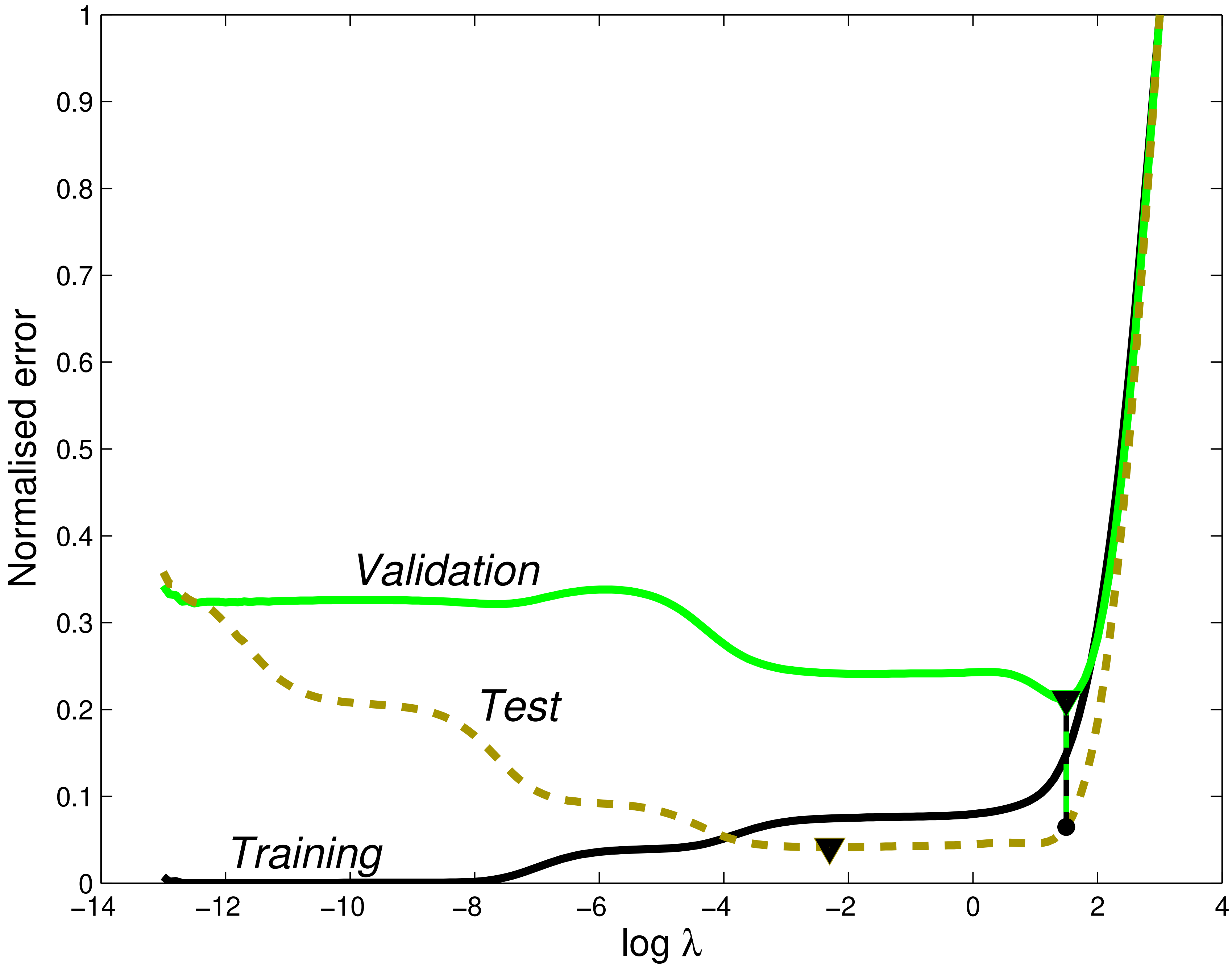
$$\mathbf{w}_{PLS} = (\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Phi} + \lambda\mathbf{I})^{-1}\boldsymbol{\Phi}^{\mathsf{T}}\mathbf{t}$$

- The *hyperparameter* λ balances the trade-off between $E_{\mathcal{D}}(\mathbf{w})$ and $E_W(\mathbf{w})$: between how well the function fits the data and how smooth it is



- Assess possible values of λ according to validation set data error





Normalised error
log λ
Validation
Test
Training
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
−14
−12
−10
−8
−6
−4
−2
0
2
4

A Probabilistic Regression Framework

- Assume that the data is a noisy realisation of an underlying functional model:

  $$t_n = y(x_n; \mathbf{w}) + \epsilon_n$$

  where $y(x_n; \mathbf{w})$ is our earlier model and ϵ_n is the noise component

- We now define an explicit noise model, chosen to be a Gaussian distribution with mean zero and variance σ^2: $p(\epsilon_n|\sigma^2) = N(0, \sigma^2)$

- Since $t_n = y(x_n; \mathbf{w}) + \epsilon_n \quad \Rightarrow \quad p(t_n|x_n, \mathbf{w}, \sigma^2) = N(y(x_n; \mathbf{w}), \sigma^2)$

- Assuming independence, the likelihood of the data set is:

  $$p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \sigma^2) = \prod_{n=1}^{N} p(t_n|x_n, \mathbf{w}, \sigma^2) = \prod_{n=1}^{N} (2\pi\sigma^2)^{-1/2} \exp\left[-\frac{\{t_n - y(x_n; \mathbf{w})\}^2}{2\sigma^2}\right]$$

A Probabilistic Regression Framework

- Assume that the data is a noisy realisation of an underlying functional model:

$$t_n = y(x_n; \mathbf{w}) + \epsilon_n$$

  where $y(x_n; \mathbf{w})$ is our earlier model and ϵ_n is the noise component

- We now define an explicit noise model, chosen to be a Gaussian distribution with mean zero and variance σ^2: $p(\epsilon_n|\sigma^2) = N(0, \sigma^2)$

- Since $t_n = y(x_n; \mathbf{w}) + \epsilon_n \quad \Rightarrow \quad p(t_n|\mathbf{w}, \sigma^2) = N(y(x_n; \mathbf{w}), \sigma^2)$

- Assuming independence, the likelihood of the data set is:

$$p(\mathbf{t}|\mathbf{w}, \sigma^2) = \prod_{n=1}^{N} p(t_n|\mathbf{w}, \sigma^2) = \prod_{n=1}^{N} (2\pi\sigma^2)^{-1/2} \exp\left[-\frac{\{t_n - y(x_n; \mathbf{w})\}^2}{2\sigma^2}\right]$$

Maximum Likelihood and Least-Squares

- The "maximum-likelihood" estimate for **w** is that value which maximises $p(\mathbf{t}|\mathbf{w},\sigma^2)$

- This is identical to the "least-squares" solution

- To see this, note that minimising squared-error is equivalent to minimising the negative logarithm of the likelihood:

$$-\log p(\mathbf{t}|\mathbf{w},\sigma^2) = \frac{N}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\sum_{n=1}^{N}\{t_n - y(x_n;\mathbf{w})\}^2$$

Incorporating Bayesian Priors

- Instead of the earlier regularisation weight penalty $E_W(\mathbf{w})$, we now control the complexity of the model via a *prior* distribution which expresses our 'degree of belief' over values that $\mathbf{w}$ might take:

$$p(\mathbf{w}|\alpha) = \prod_{m=1}^{M} \left(\frac{\alpha}{2\pi}\right)^{1/2} \exp\left\{-\frac{\alpha}{2}w_m^2\right\}$$

- This is a zero-mean Gaussian prior, independent for each weight, with common inverse variance hyperparameter α

- We're expressing a preference for smoother models by declaring smaller weights to be *a priori* more probable

Bayesian Inference

- Given the likelihood and the prior, rather than computing a single *point estimate* $\mathbf{w}_{LS}$ for the weights, we compute the *posterior distribution* via Bayes' rule:

$$p(\mathbf{w}|\mathbf{t}, \alpha, \sigma^2) = \frac{\text{likelihood} \times \text{prior}}{\text{normalising factor}} = \frac{p(\mathbf{t}|\mathbf{w}, \sigma^2)p(\mathbf{w}|\alpha)}{p(\mathbf{t}|\alpha, \sigma^2)}$$

- Here, the posterior is Gaussian: $p(\mathbf{w}|\mathbf{t}, \alpha, \sigma^2) = N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with

$$\boldsymbol{\mu} = (\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Phi} + \sigma^2\alpha\mathbf{I})^{-1}\boldsymbol{\Phi}^{\mathsf{T}}\mathbf{t}$$

$$\boldsymbol{\Sigma} = \sigma^2(\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Phi} + \sigma^2\alpha\mathbf{I})^{-1}$$

MAP Estimation — a 'Bayesian' Short-cut

- The "maximum *a posteriori*" (MAP) estimate for **w** is the single most probable value under the posterior distribution $p(\mathbf{w}|\mathbf{t}, \alpha, \sigma^2)$

- Since the denominator in Bayes' rule earlier is independent of **w**, this is equivalent to minimising $E_{MAP}(\mathbf{w}) = -\log p(\mathbf{t}|\mathbf{w}, \sigma^2) - \log p(\mathbf{w}|\alpha)$

- Retaining only terms dependent on **w**:

$$E_{MAP}(\mathbf{w}) = \frac{1}{2\sigma^2} \sum_{n=1}^{N} \{t_n - y(x_n; \mathbf{w})\}^2 + \frac{\alpha}{2} \sum_{m=1}^{M} w_m^2$$

- The MAP estimate is therefore identical to the PLS estimate with $\lambda = \sigma^2 \alpha$

Demonstration: Bayesian 'Learning'

- Let's look at how the posterior $p(\mathbf{w}|\mathbf{t}, \alpha, \sigma^2)$ evolves as we observe data t_n

- Note that we can compute the posterior incrementally if the data are assumed independent (given $\mathbf{w}$). *e.g.* for $\mathbf{t} = (t_1, t_2, t_3)$:

$$\begin{aligned} p(\mathbf{w}|t_1, t_2, t_3) &\propto p(t_1, t_2, t_3|\mathbf{w})\, p(\mathbf{w}) \\ &\propto p(t_2, t_3|\mathbf{w})\, p(t_1|\mathbf{w})\, p(\mathbf{w}) \\ &\propto \text{Likelihood of } (t_2, t_3) \times \text{posterior having observed } t_1 \end{aligned}$$

- So, more generally, we can treat the posterior having observed $(t_1, \ldots, t_K)$ as the 'prior' for the remaining data $(t_{K+1}, \ldots, t_N)$