Statistical Machine Learning
Linear Regression

Luca Bortolussi

Department of Mathematics and Geosciences
University of Trieste

Office 238, third floor, H2bis
luca@dmi.units.it

Trieste, Spring Semester 2019

Outline

GENERALISED BASIS FUNCTIONS

- Suppose our inputs are real vectors, and outputs are real numbers, and we have observations $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, N$.
- We consider a set of M basis functions $\phi_j : \mathbb{R}^n \to \mathbb{R}$, and write $\boldsymbol{\phi}(\mathbf{x}) = (\phi_0(\mathbf{x}), \ldots, \phi_{M-1}(\mathbf{x}))$. By convention, $\phi_0 \equiv 1$.
- We consider the linear model

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x})$$

- $y(\mathbf{x}, \mathbf{w})$ is linear in the parameters $\mathbf{w}$, but can be non-linear in the input state $\mathbf{x}$.

GENERALISED BASIS FUNCTIONS

Basis functions can, and usually are, non-linear functions of the inputs. Examples are

- Polynomials up to degree d. In 1 dimension, $1, x, x^2, \ldots, x^d$
- Gaussian basis functions: $\phi_j = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$, where μ_j is the location and s is the lengthscale of the Gaussian.
- Sigmoid functions $\phi_j = \sigma\left(\frac{x-\mu_j}{s}\right)$, with $\sigma(a) = \frac{1}{1+\exp(-a)}$

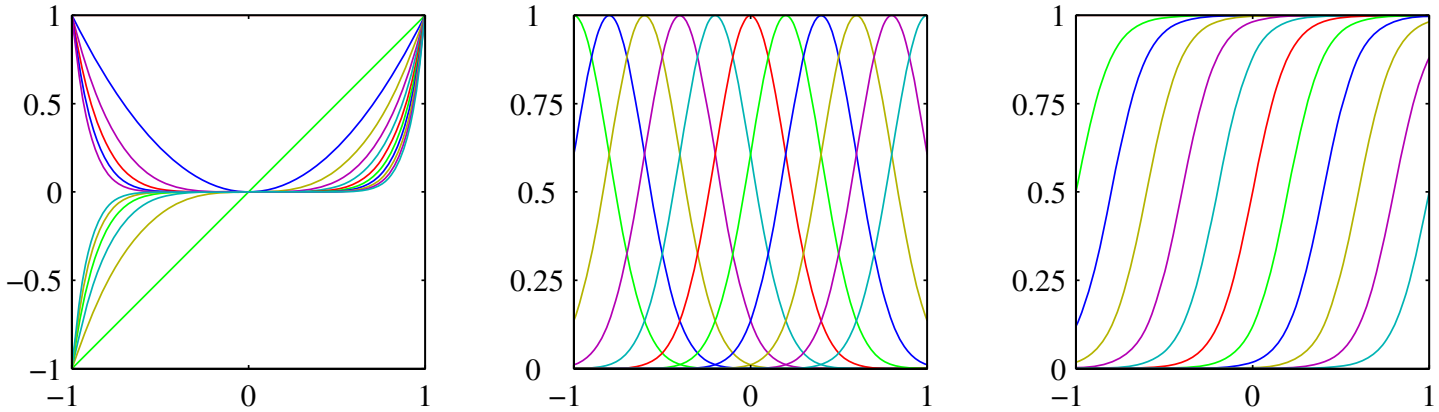


Figure 3.1 Examples of basis functions, showing polynomials on the left, Gaussians of the form (3.4) in the centre, and sigmoidal of the form (3.5) on the right.

MAXIMUM LIKELIHOOD REGRESSION

- Assume Gaussian noise: $t = y(\mathbf{x}, \mathbf{w}) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \beta^{-1})$
- Hence $p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(y(\mathbf{x}, \mathbf{w}), \beta^{-1})$
- Given observations $\mathbf{X}, \mathbf{t}$: $(\mathbf{x_i}, t_i)_{i=1,\ldots,N}$, the likelihood is then

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{i=1}^{N} \mathcal{N}(y_i|\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x_i}), \beta^{-1})$$

giving a log likelihood of

$$\begin{aligned} \ln p(\mathbf{t}|\mathbf{w}, \beta) &= \sum_{n=1}^{N} \ln \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) \\ &= \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta E_D(\mathbf{w}) \end{aligned} \tag{3.11}$$

where the sum-of-squares error function is defined by

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{t_n - \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n)\}^2. \tag{3.12}$$

MAXIMUM LIKELIHOOD REGRESSION

- Compute the gradient w.r.t. **w** of the log-likelihood, set it to zero and solve for **w**.

$$\mathbf{w}_{\mathrm{ML}} = \left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t} \tag{3.15}$$

which are known as the *normal equations* for the least squares problem. Here $\mathbf{\Phi}$ is an $N \times M$ matrix, called the *design matrix*, whose elements are given by $\Phi_{nj} = \phi_j(\mathbf{x}_n)$, so that

$$\mathbf{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}. \tag{3.16}$$

- Looking for the ML solution of the precision β, we get

$$\frac{1}{\beta_{\mathrm{ML}}} = \frac{1}{N}\sum_{n=1}^{N}\{t_n - \mathbf{w}_{\mathrm{ML}}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 \tag{3.21}$$

MAXIMUM LIKELIHOOD REGRESSION: BIAS TERM

- The parameter w_0 is known also as bias term.

At this point, we can gain some insight into the role of the bias parameter w_0. If we make the bias parameter explicit, then the error function (3.12) becomes

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - w_0 - \sum_{j=1}^{M-1} w_j\phi_j(\mathbf{x}_n)\}^2. \tag{3.18}$$

Setting the derivative with respect to w_0 equal to zero, and solving for w_0, we obtain

$$w_0 = \bar{t} - \sum_{j=1}^{M-1} w_j\overline{\phi_j} \tag{3.19}$$

where we have defined

$$\bar{t} = \frac{1}{N}\sum_{n=1}^{N} t_n, \qquad \overline{\phi_j} = \frac{1}{N}\sum_{n=1}^{N}\phi_j(\mathbf{x}_n). \tag{3.20}$$

Thus the bias w_0 compensates for the difference between the averages (over the training set) of the target values and the weighted sum of the averages of the basis function values.

MULTIPLE OUTPUTS

- What if we have a vector of d-outputs rather than a single one, i.e. what if observations $\mathbf{X}, \mathbf{T}$ are $(\mathbf{x_i}, \mathbf{t_i})_{I=1,\ldots,N}$?
- If we use separate weights for each output dimension, $\mathbf{W} = (w_{ij})$, then the model is

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

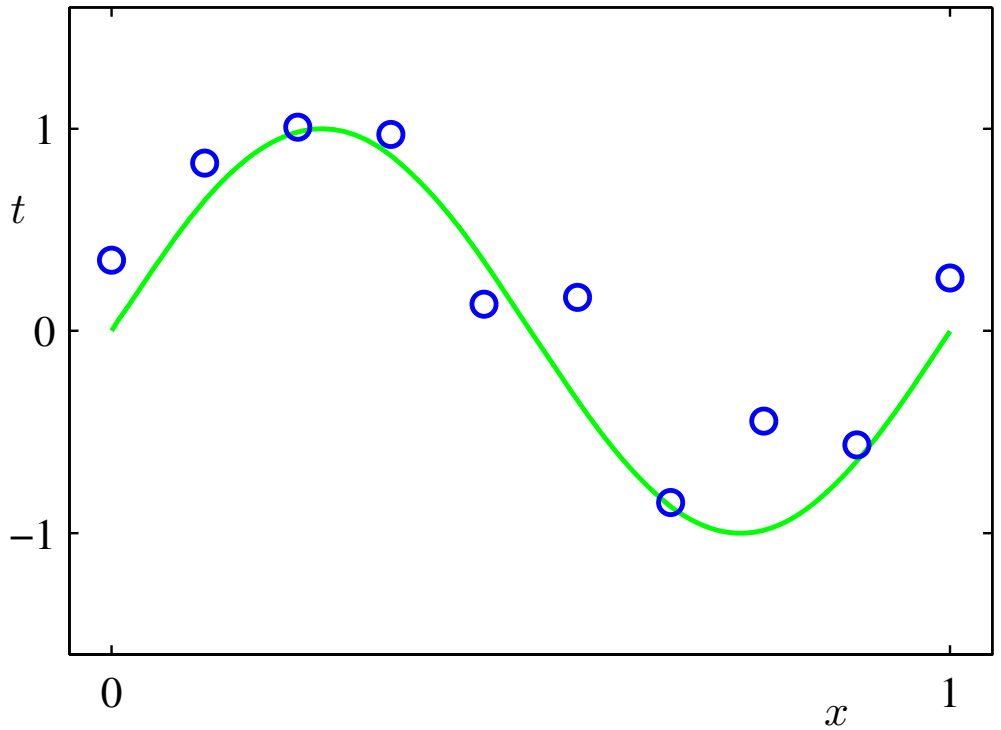
  which is easily seen to factorise in the different outputs, so that we need to solve d independent ML problems, giving

$$\mathbf{W}_{ML} = (\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \mathbf{T}$$

- Generalise to the case in which some coefficients of $\mathbf{W}$ are shared among outputs (i.e., constrained to be equal).

AN EXAMPLE (BISHOP)

- As an example, consider data generated by the model $t = sin(2\pi x) + \epsilon$, from which we generate few observations:

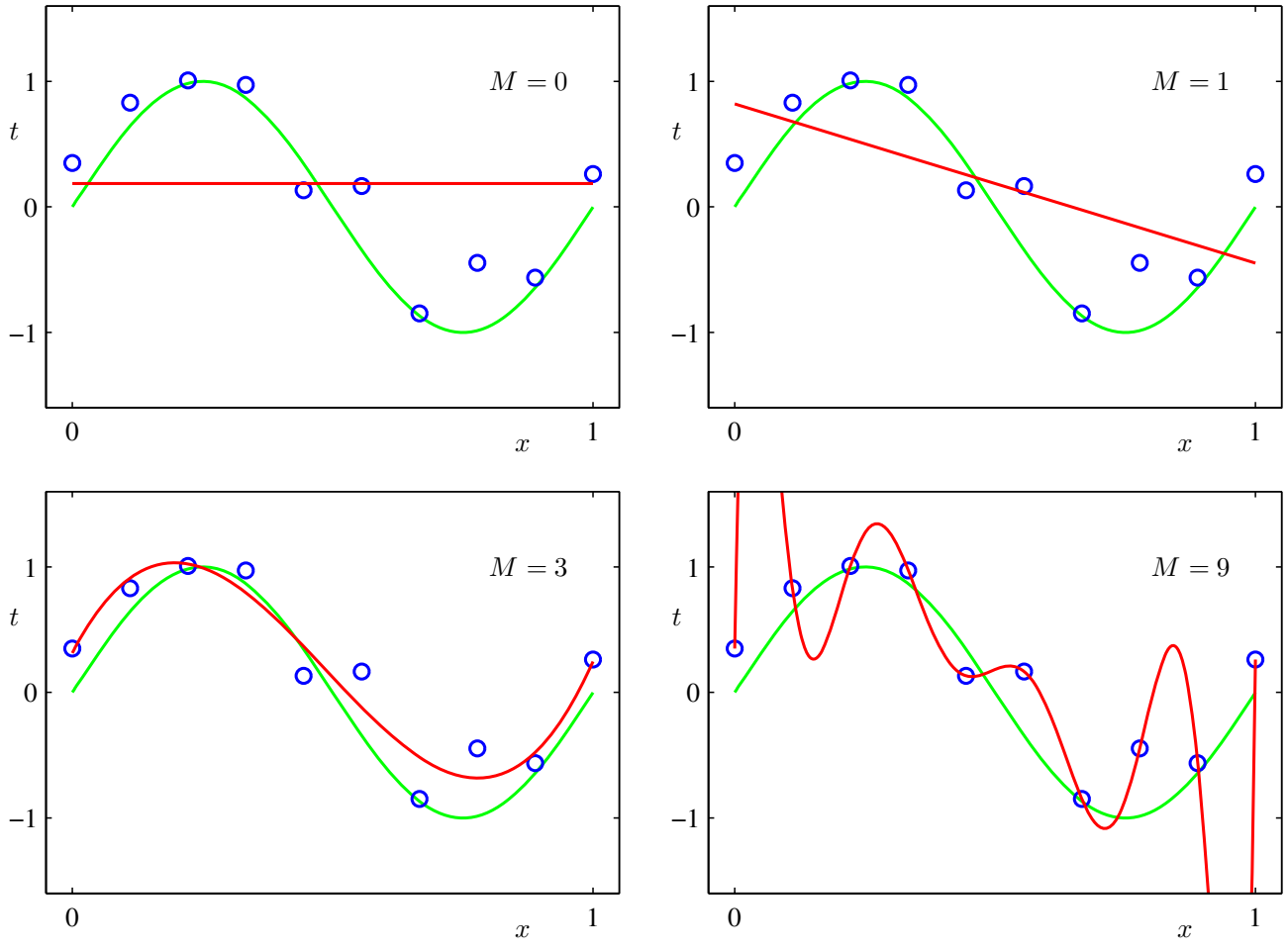


- We want to fit a polynomial model of degree M, where M is to be chosen:

$$y(x, \mathbf{w}) = w_0 x^0 + w_1 x^1 + \ldots + w_M x^M$$

AN EXAMPLE (BISHOP)

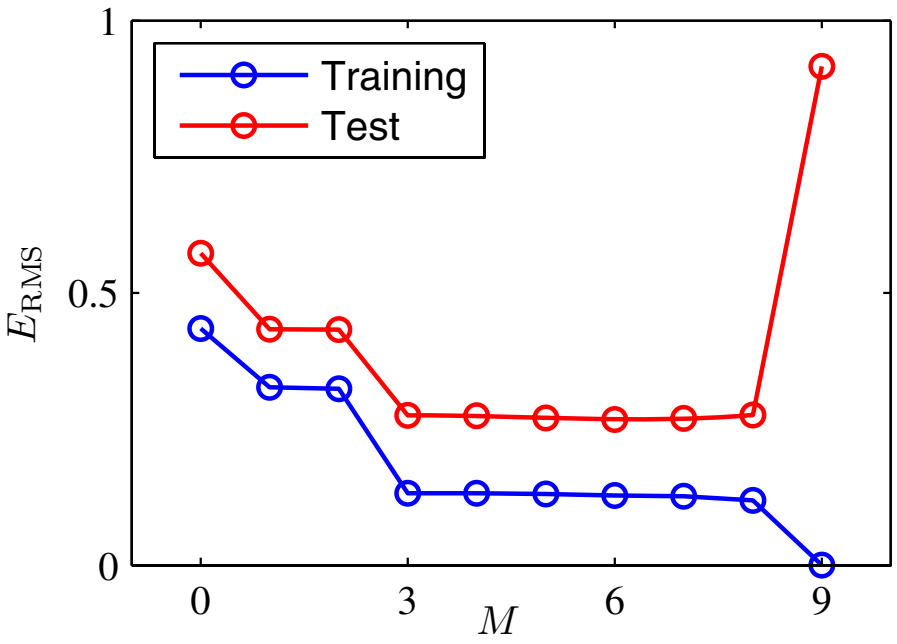
- Max likelihood solution for different M



AN EXAMPLE (BISHOP)

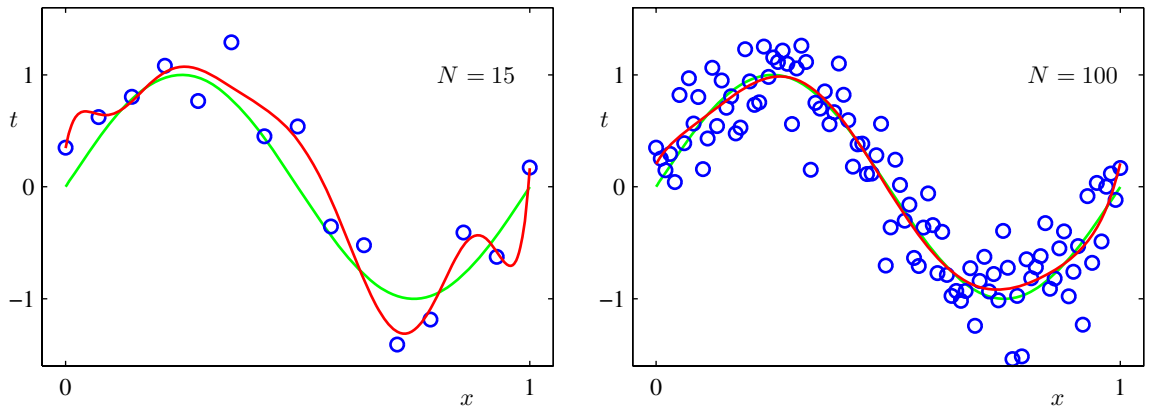
- For $M = 9$ we face the problem of overfitting: the model is too complex - ML explains noise rather than data.
- To validate a model, we need test data, different from the train data. Then we can compute the root mean square error on test (and train) data.

$$E_{RMS} = \sqrt{2E_D(\mathbf{w_{ML}})/N}$$



AN EXAMPLE (BISHOP)

- Overfitting depends also on how many observations: the more observations, the less overfitting:



- The fine-tuning of model to data reflects usually in large coefficients.

	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

REGULARISED MAXIMUM LIKELIHOOD

- One way to avoid overfitting is to penalise solutions with large values of coefficients $\mathbf{w}$.
- This can be enforced by introducing a regularisation term on the error function to be minimised:

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

- $\lambda > 0$ is the regularisation coefficient, and governs how strong is the penalty.
- A common choice is

$$E_W(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} = \frac{1}{2}\sum_j w_j^2$$

known as ridge regression, with solution

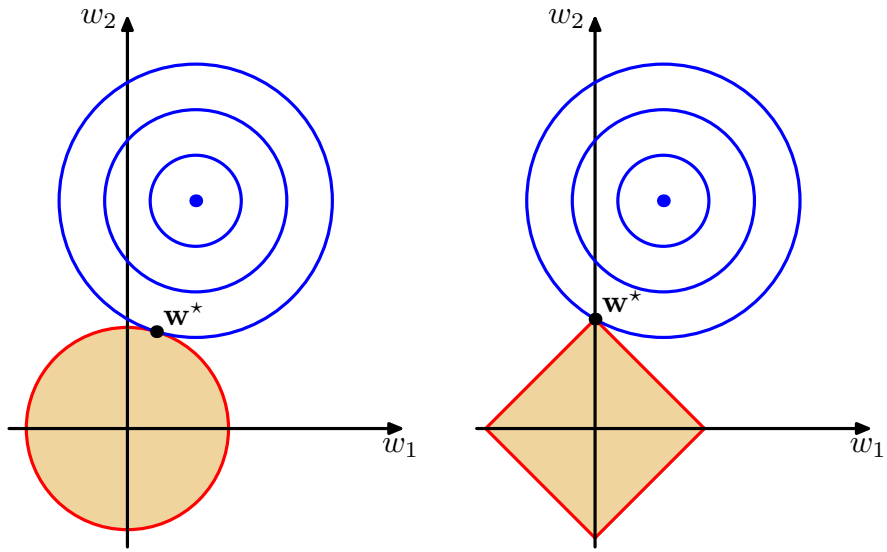
$$\mathbf{w_{RR}} = (\lambda\mathbf{I} + \mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{t}$$

REGULARISED MAXIMUM LIKELIHOOD

- A more general form of the penalty term is

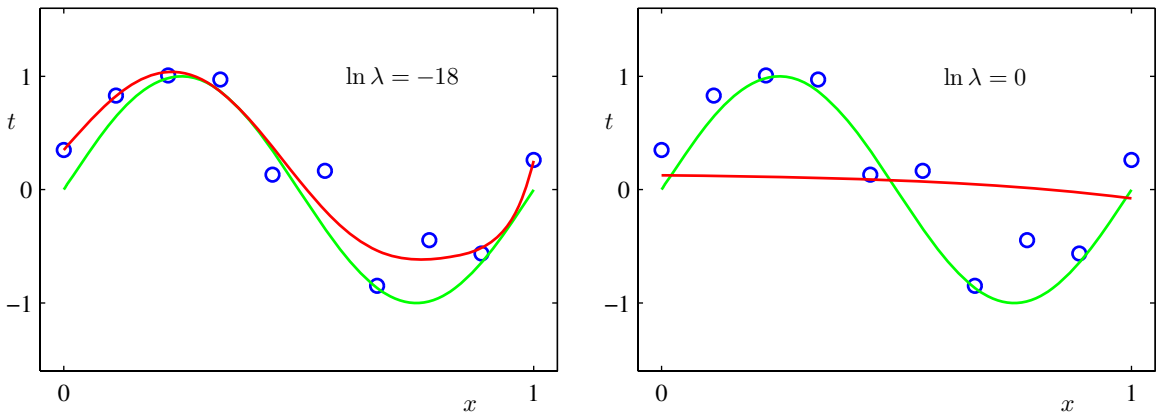
$$E_W(\mathbf{w}) = \frac{1}{2}\sum_j |w_j|^q$$

- $q = 2$ is the ridge regression, while $q = 1$ is the lasso regression.
- Lasso regression has the property that it produces sparse models as some coefficients tend to be set to zero. However, it has no analytic solution.

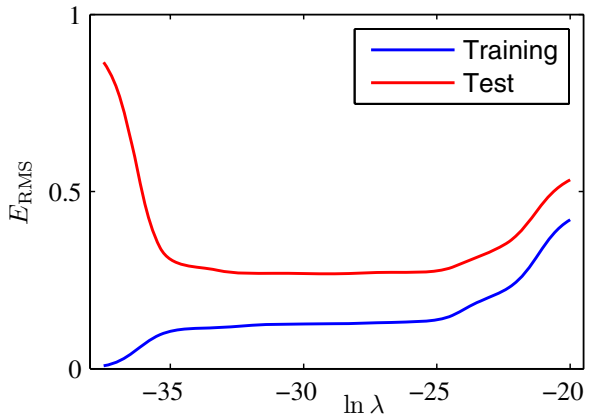


EXAMPLE: REGULARISED ML

- Let's consider the sine example, and fit the model of degree $M = 9$ by ridge regression, for different $\lambda's$.



- If we compute the RMSE on a test set, we can see how the error changes with λ



TRAIN, VALIDATION, AND TEST DATA

- The regularisation coefficient λ is a method parameter. But how can we set it?
- Ideally, we should divide our data in a train set, a test set, and a validation set, which can be used to set method's parameters.
- Often, we do not have all such data, hence we can resort to cross-validation
- *n*-fold cross-validation: split data set in *n* blocks, use in turn each block for validation and the rest for training, average the error on the *n* runs.
- leave one out cross-validation: validate in tuns on a single data point left out from the training set and average.

EXPECTED LOSS

- If we have a model $p(\mathbf{x}, t)$ of input-output, one way to make a prediction (choose t^* given $\mathbf{x}^*$) is by minimising an expected loss functional

$$\mathbb{E}[L] = \iint \{y(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\, \mathrm{d}\mathbf{x}\, \mathrm{d}t. \tag{1.87}$$

- The solution for the square loss functional is the conditional expectation

$$y(\mathbf{x}) = \frac{\int t p(\mathbf{x}, t)\, \mathrm{d}t}{p(\mathbf{x})} = \int t p(t|\mathbf{x})\, \mathrm{d}t = \mathbb{E}_t[t|\mathbf{x}] \tag{1.89}$$

- This can be seen by summing and subtracting $\mathbb{E}[t|\mathbf{x}]$ inside the integral, getting

$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 p(\mathbf{x})\, \mathrm{d}\mathbf{x} + \int \{\mathbb{E}[t|\mathbf{x}] - t\}^2 p(\mathbf{x})\, \mathrm{d}\mathbf{x}. \tag{1.90}$$

BIAS VARIANCE DECOMPOSITION

- If we do not have the full model, but only observe a dataset $\mathcal{D}$, then we can try to find the best approximant to the true conditional expectation, $y(\mathbf{x}, \mathcal{D})$.
- To test a method, we can try to generate many datasets and take the average $\mathbb{E}_{\mathcal{D}}$ w.r.t. the dataset. After some computations, calling $h(\mathbf{x})$ the true conditional expectation:

$$\text{expected loss} = (\text{bias})^2 + \text{variance} + \text{noise} \tag{3.41}$$

where

$$(\text{bias})^2 = \int \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 p(\mathbf{x})\, \mathrm{d}\mathbf{x} \tag{3.42}$$

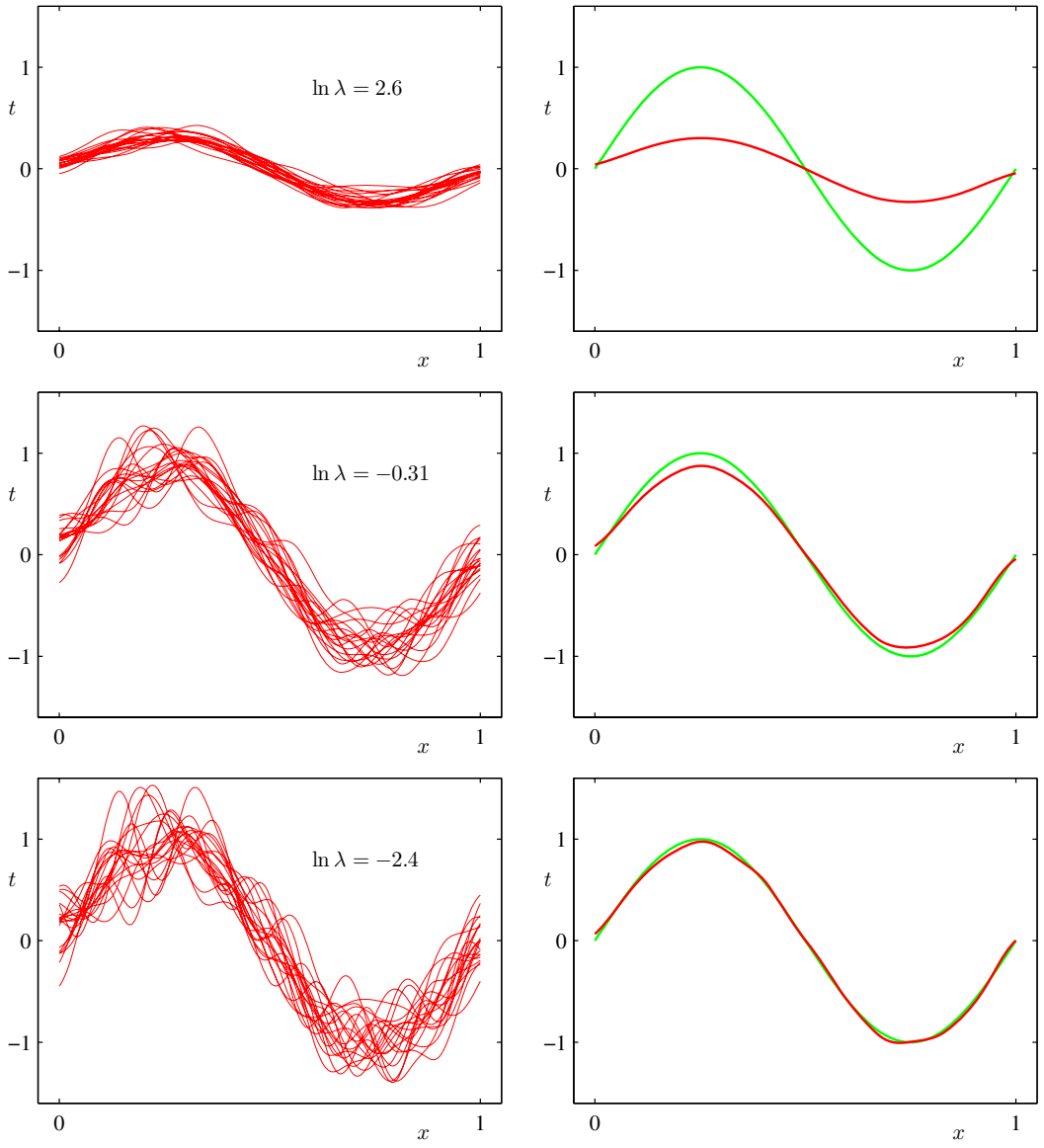
$$\text{variance} = \int \mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2\right] p(\mathbf{x})\, \mathrm{d}\mathbf{x} \tag{3.43}$$

$$\text{noise} = \int \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\, \mathrm{d}\mathbf{x}\, \mathrm{d}t \tag{3.44}$$

EXAMPLE: BIAS VARIANCE DECOMPOSITION

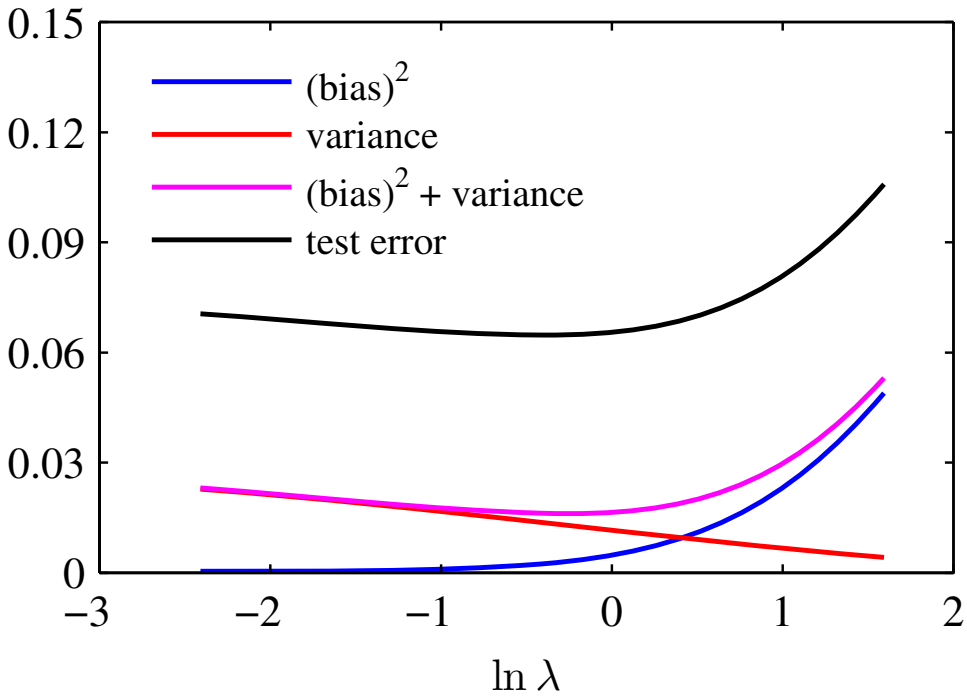
left: solutions for individual datasets

right: averages over datasets



EXAMPLE: BIAS VARIANCE DECOMPOSITION

- For the sine example, we can compute bias and variance as a function of the regularisation coefficient. The trade off is evident.



Outline

THE BAYESIAN APPROACH

- Regularisation works by biasing
- One way to bias estimators is to have prior beliefs and being Bayesian
- Let's assume the regression weights have a Gaussian prior $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \alpha\mathbf{I})$ and that the bias is zero
- The posterior is given by Bayes theorem:

$$p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \alpha, \beta) = \frac{p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \alpha, \beta)p(\mathbf{w}|\alpha)}{p(\mathbf{t}|\mathbf{X}, \alpha, \beta)}$$

THE POSTERIOR DISTRIBUTION

- Hence, the log posterior is

$$\log p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \alpha, \beta) = -\frac{\beta}{2}\sum_{j=1}^{N}[t_j - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x_j})]^2 - \alpha\mathbf{w}^T\mathbf{w} + const$$

- As it is a quadratic function in $\mathbf{w}$, it is the log of a Gaussian:

$$p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \alpha, \beta) = \mathcal{N}(\mathbf{w}|\mathbf{m_N}, \mathbf{S_N})$$

  with mean and variance

$$\mathbf{m_N} = \beta\mathbf{S_N}\mathbf{\Phi}^T\mathbf{t}$$

$$\mathbf{S_N}^{-1} = \alpha\mathbf{I} + \beta\mathbf{\Phi}^T\mathbf{\Phi}$$

- Alternatively: use the formula for the product of two gaussians.

THE POSTERIOR DISTRIBUTION

- In general, we can take a general Gaussian prior

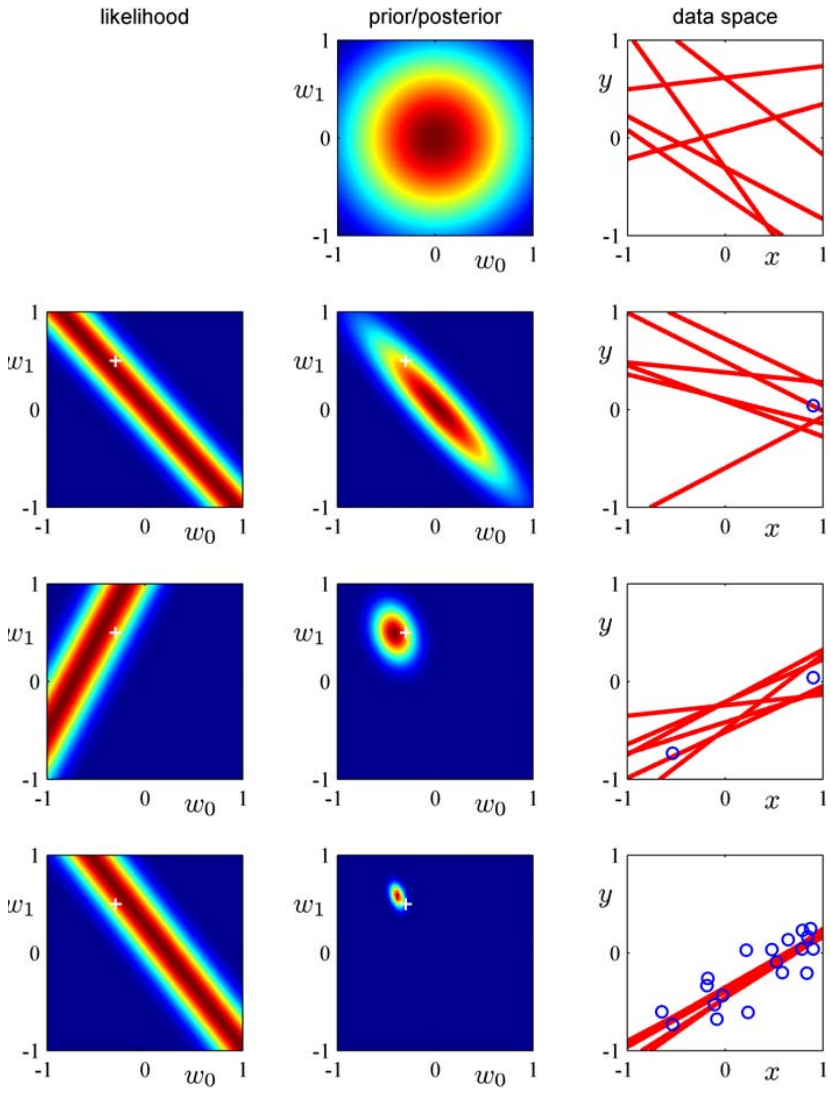
$$p(\mathbf{w}|\mathbf{m_0}, \mathbf{S_0}) = \mathcal{N}(\mathbf{w}|\mathbf{m_0}, \mathbf{S_0})$$

- This will result in a Gaussian posterior $p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \alpha, \beta) = \mathcal{N}(\mathbf{w}|\mathbf{m_N}, \mathbf{S_N})$ with

$$\mathbf{m_N} = \mathbf{S_N}[\mathbf{S_0}^{-1}\mathbf{m_0} + \beta\mathbf{\Phi}^T\mathbf{t}]$$

$$\mathbf{S_N}^{-1} = \mathbf{S_0}^{-1} + \beta\mathbf{\Phi}^T\mathbf{\Phi}$$

POSTERIOR UPDATE



THE PREDICTIVE DISTRIBUTION

- Given the posterior, one can find the MAP estimate. However, in a fully Bayesian treatment, one makes predictions by integrating out the parameters via their posterior distribution.

$$p(t|\mathbf{t},\alpha,\beta) = \int p(t|\mathbf{t},\mathbf{w},\alpha,\beta)p(\mathbf{w}|\mathbf{t},\alpha,\beta)d\mathbf{w}$$

- The predictive distribution is still a Gaussian

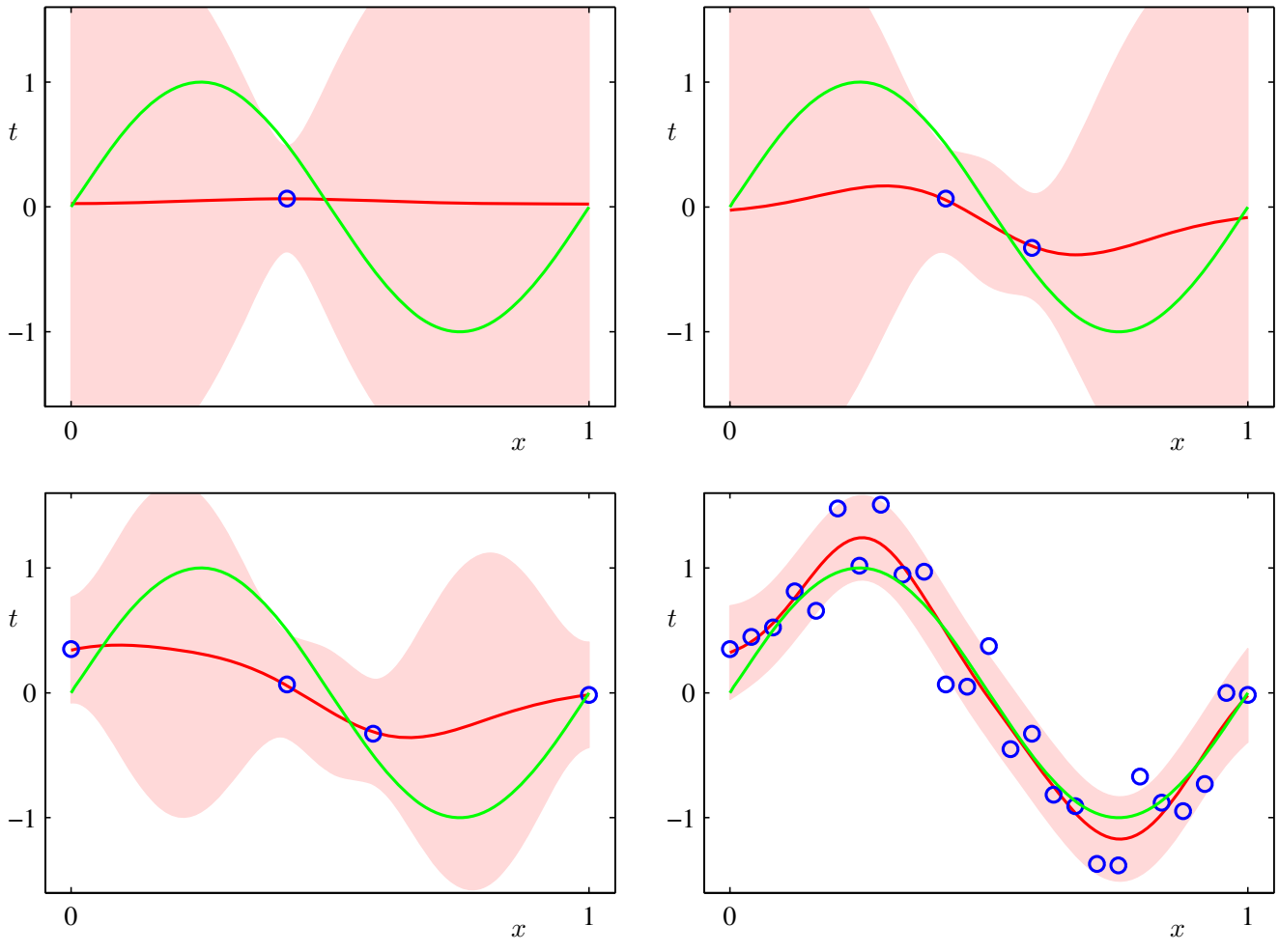
$$p(t|\mathbf{t},\alpha,\beta) = \mathcal{N}(t|\mathbf{m_N}^T\boldsymbol{\phi}(\mathbf{x}),\sigma_N^2(\mathbf{x}))$$

with mean $\mathbf{m_N}^T\boldsymbol{\phi}(\mathbf{x})$ and variance

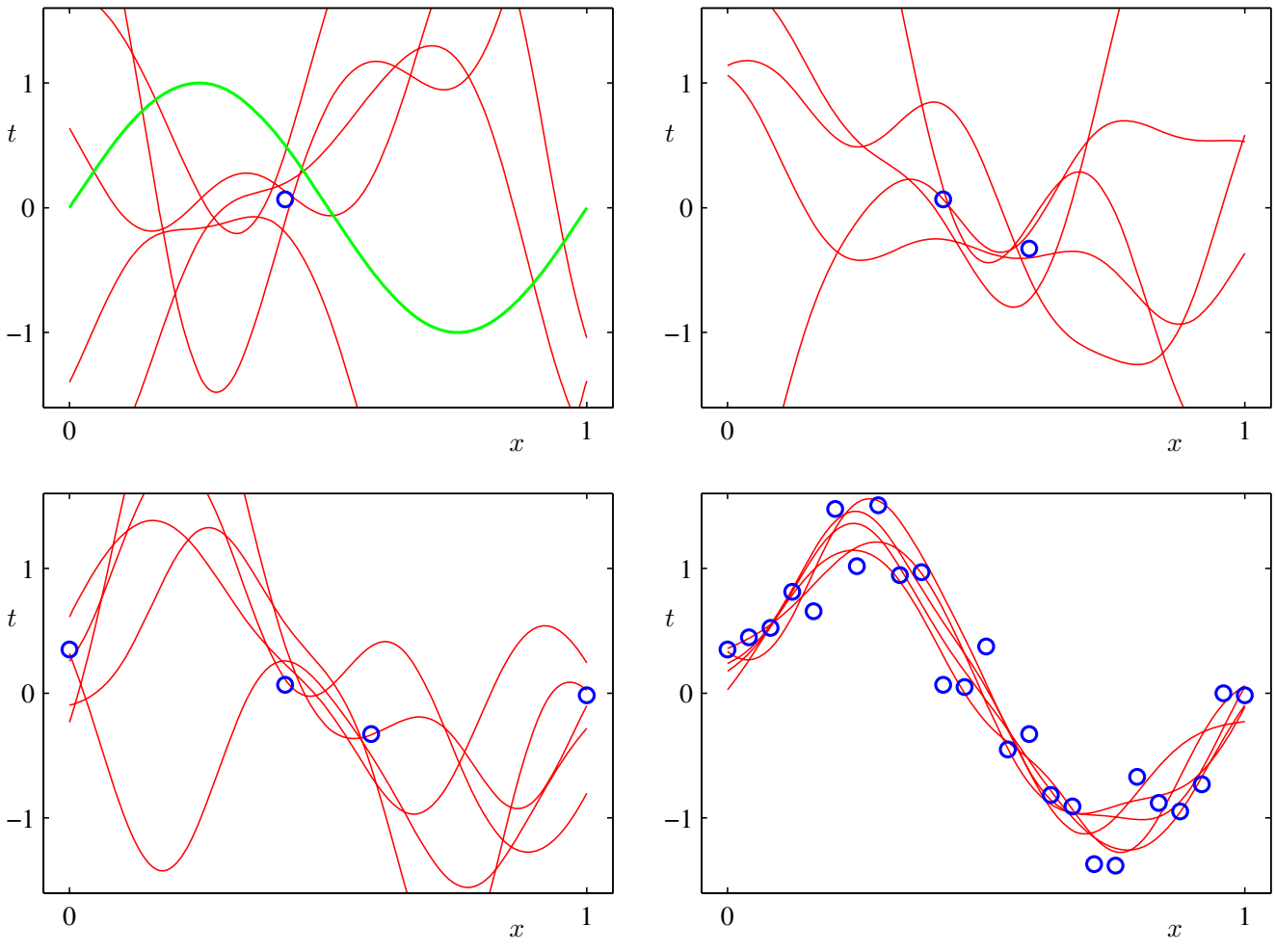
$$\sigma_N^2(\mathbf{x}) = \frac{1}{\beta} + \boldsymbol{\phi}(\mathbf{x})^T\mathbf{S_N}\boldsymbol{\phi}(\mathbf{x})$$

- It can be shown that $\sigma_{N+1}^2(\mathbf{x}) \leq \sigma_N^2(\mathbf{x})$ and $\sigma_N^2(\mathbf{x}) \to 1/\beta$

EXAMPLE



EXAMPLE



MARGINAL LIKELIHOOD

- The marginal likelihood $p(\mathbf{t}|\alpha,\beta)$, appearing at the denominator in Bayes theorem, can be used to identify good α and β, known as hyperparameters.
- Intuitively, we can place a prior distribution over α and β, compute their posterior, and use this in a fully Bayesian treatment of the regression:

$$p(\alpha,\beta|\mathbf{t}) \propto p(\mathbf{t}|\alpha,\beta)p(\alpha,\beta)$$

- If we assume the posterior is peaked around the mode, then we can take the MAP as an approximation of the full posterior for α and β. If the prior is flat, this will boil down to the ML solution.

MARGINAL LIKELIHOOD

- Hence we need to optimise the marginal likelihood, which can be computed as:

$$\log p(\mathbf{t}|\alpha,\beta) = \frac{M}{2}\log\alpha + \frac{N}{2}\log\beta - E(\mathbf{m_N}) - \frac{1}{2}\log|\mathbf{S_N}^{-1}| - \frac{N}{2}\log 2\pi$$

with

$$E(\mathbf{m_N}) = \frac{\beta}{2}\|\mathbf{t} - \mathbf{\Phi m_N}\|^2 + \frac{\alpha}{2}\mathbf{m_N}^T\mathbf{m_N}$$

- This optimisation problem can be solved with any optimisation routine, or with specialised methods.

OPTIMISING THE MARGINAL LIKELIHOOD

- We will present a fix-point algorithm: we will write the gradient equations equal to zero as fix-point equations and iterate until convergence.
- In taking the derivative w.r.t α or β, the most challenging term is the log of the determinant of $\mathbf{S_N}^{-1} = \alpha\mathbf{I} + \beta\mathbf{\Phi}^T\mathbf{\Phi}$.
- To deal with it, let λ_i be the eigenvalues of $\beta\mathbf{\Phi}^T\mathbf{\Phi}$, so that $|\mathbf{S_N}^{-1}| = \prod_{i=0}^{M-1}(\alpha + \lambda_i)$.
- We then have that

$$\partial \log |\mathbf{S_N}^{-1}|/\partial\alpha = \sum_i \frac{1}{\alpha + \lambda_i}$$

- Moreover, λ_i are proportional to β, so that $\partial\lambda_i/\partial\beta = \lambda_i/\beta$

OPTIMISING THE MARGINAL LIKELIHOOD

- Now, define

$$\gamma = \sum_i \frac{\lambda_i}{\alpha + \lambda_i}$$

  (which measures the number of well determined parameters)
- By deriving the log-marginal w.r.t. α and setting derivative to zero, we obtain:

$$\alpha = \frac{\gamma}{\mathbf{m_N}^T \mathbf{m_N}} = g_\alpha(\alpha, \beta)$$

- By deriving the log-marginal w.r.t. β and setting derivative to zero, we obtain:

$$\frac{1}{\beta} = \frac{1}{N - \gamma} \sum_{n=1} N[t_n - \mathbf{m_N}^T \boldsymbol{\phi}(\mathbf{x_n})]^2 = \frac{1}{g_\beta(\alpha, \beta)}$$

- We start from an initial value α_0 and β_0 and iterate $\alpha_{n+1} = g_\alpha(\alpha_n, \beta_n)$, $\beta_{n+1} = g_\beta(\alpha_n, \beta_n)$ until convergence.

BAYESIAN MODEL COMPARISON

- Consider $\mathcal{M}_1$ and $\mathcal{M}_2$ two different models, which one is the best to explain the data $\mathcal{D}$?
- In a Bayesian setting, we may place a prior $p(\mathcal{M}_j)$ on the models, and compute the posterior $p(\mathcal{M}_j|\mathcal{D}) = \frac{p(\mathcal{D}|\mathcal{M}_j)p(\mathcal{M}_j)}{\sum_j p(\mathcal{D}|\mathcal{M}_j)p(\mathcal{M}_j)}$.
- As we typically have additional parameters $\mathbf{w}$, the term $p(\mathcal{D}|\mathcal{M}_j)$ is the model evidence/ marginal likelihood.
- The ratio $p(\mathcal{D}|\mathcal{M}_1)/p(\mathcal{D}|\mathcal{M}_2)$ is known as Bayes Factor.

BAYESIAN MODEL COMPARISON

- In Bayesian model comparison, we can take two approaches.
- We can compute the predictive distribution for each model and average it by the posterior model probability

$$p(\mathbf{t}|\mathcal{D}) = \sum_j p(\mathbf{t}|\mathcal{M}_j, \mathcal{D}) p(\mathcal{M}_j|\mathcal{D})$$

- Alternatively, we can choose the model with larger Bayes Factor. This will pick the correct model on average. In fact, the average log Bayes factor (assuming $\mathcal{M}_1$ is the true model) is

$$\int p(\mathcal{D}|\mathcal{M}_1) \log \frac{p(\mathcal{D}|\mathcal{M}_1)}{p(\mathcal{D}|\mathcal{M}_2)} > 0$$

Outline

DUAL REPRESENTATION

- Consider a regression problem with data $(\mathbf{x_i}, y_i)$, and a linear model $\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x})$.
- We can restrict the choice of $\mathbf{w}$ to the linear subspace spanned by $\boldsymbol{\phi}(\mathbf{x_1}), \ldots, \boldsymbol{\phi}(\mathbf{x_N})$, as any $\mathbf{w}_\perp$ othogonal to this subspace will give a contribution ${\mathbf{w}_\perp}^T\boldsymbol{\phi}(\mathbf{x_i}) = 0$ on input points:

$$\mathbf{w} = \sum_{j=1}^{N} a_j \boldsymbol{\phi}(\mathbf{x_j})$$

- $\mathbf{a}$ are known as the dual variables
- By defining the kernel $k(\mathbf{x_i}, \mathbf{x_j}) := \boldsymbol{\phi}(\mathbf{x_i})^T\boldsymbol{\phi}(\mathbf{x_j})$, we can write

$$\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x_i}) = \mathbf{a}^T\mathbf{K^i}$$

Where $\mathbf{K^i}$ is the ith column of the Gram matrix $\mathbf{K}$, $K_{ij} = k(\mathbf{x_i}, \mathbf{x_j})$.

DUAL REGRESSION PROBLEM

- In the dual variables, we have to optimise the following regression equation

$$E_d(\mathbf{a}) + \lambda E_W(\mathbf{a}) = \sum_{i=1}^{N} (t_i - \mathbf{a}^T \mathbf{K}^{\mathbf{i}})^2 + \lambda \mathbf{a}^T \mathbf{K} \mathbf{a}$$

- By deriving w.r.t $\mathbf{a}$ and setting the gradient to zero, we obtain the solution

$$\hat{\mathbf{a}} = (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{t}$$

- At a new input $\mathbf{x}^*$, the prediction will then be

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

with $\mathbf{k}_*{}^T = (k(\mathbf{x}^*, \mathbf{x_1}), \ldots, k(\mathbf{x}^*, \mathbf{x_N}))$

THE KERNEL TRICK

- The dual objective function depends only on the scalar product of input vectors
- We can replace the Euclidean scalar product with *any* (non-linear) scalar product
- This is usually obtained by giving directly a non-linear *kernel* function $k(\mathbf{x}_i, \mathbf{x}_j)$ (*kernel trick*)
- This enables us to work with more general set of basis functions, even countable. See Gaussian processes.
- The same dual procedure applies to other algorithms, notably linear classification and SVMs
- The computational cost to solve the primal problem is $O(M^3)$, while the dual costs $O(N^3)$. They can be both prohibitive is N and M are large. In this case, one can optimise the log likelihood directly, using gradient based methods.