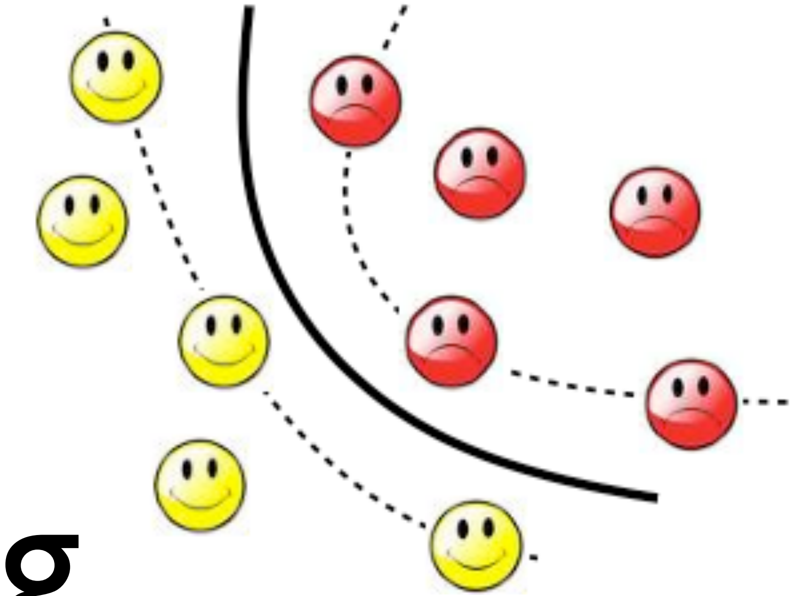
THE UNIVERSITY OF SYDNEY

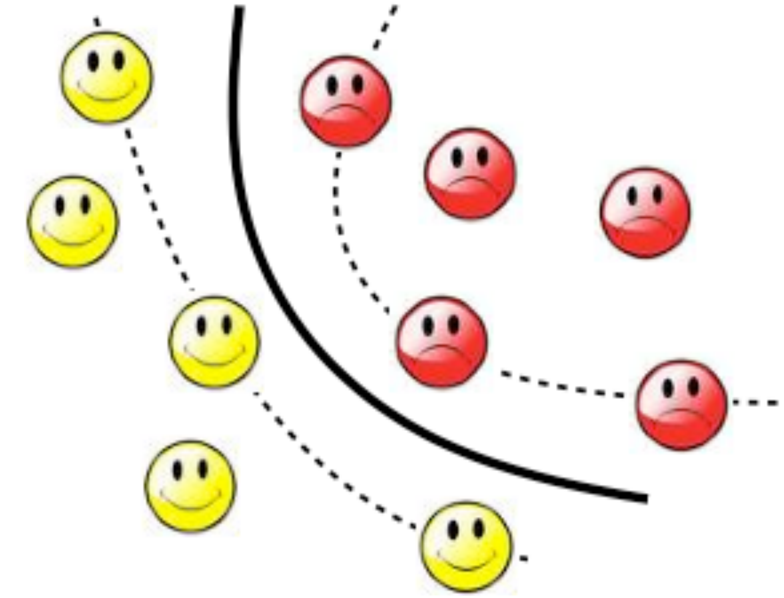


Machine Learning and Data Mining

(COMP 5318)

Linear Regression

Dr Tongliang Liu

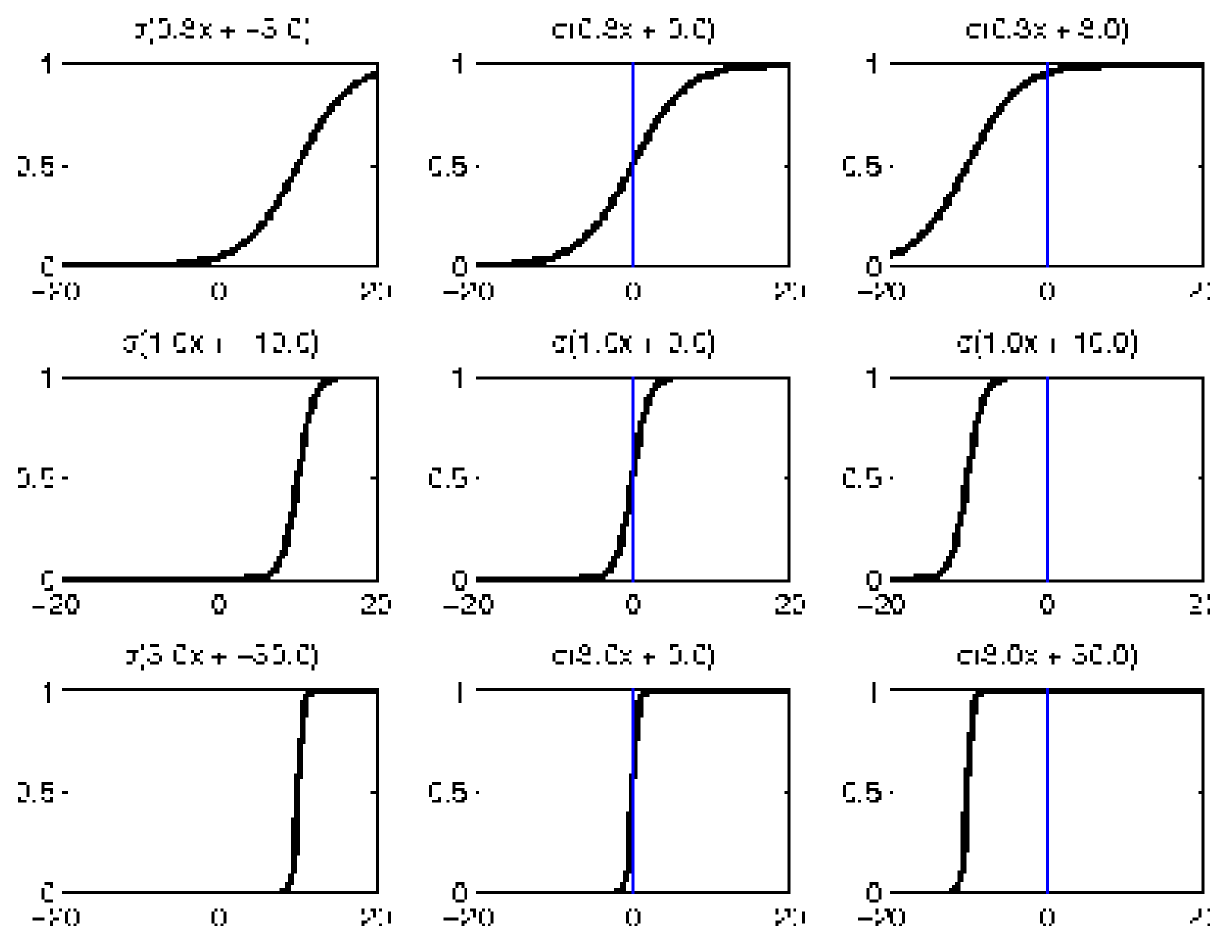


Review

Logistic Regression

- Discriminative model for binary classification

$$
\begin{aligned}
p(y|\mathbf{x}, \mathbf{w}) &= \mathrm{Ber}(y|\sigma(\eta)) = \sigma(\eta)^y(1-\sigma(\eta))^{1-y} \\
\eta &= \mathbf{w}^T\mathbf{x} \\
\sigma(\eta) &\stackrel{\text{def}}{=} \frac{1}{1+\exp(-\eta)} = \frac{e^\eta}{e^\eta + 1}
\end{aligned}
$$



Sigmoid or Logistic function

Logistic Regression

- Assumes a parametric form for directly estimating P(Y | X). For binary concepts, this is:

$$P(Y=0|X) = \frac{1}{1+\exp(w_0 + \sum_{i=1}^{n} w_i X_i)}$$

$$P(Y=1|X) = 1 - P(Y=0|X)$$

$$= \frac{\exp(w_0 + \sum_{i=1}^{n} w_i X_i)}{1+\exp(w_0 + \sum_{i=1}^{n} w_i X_i)}$$

- Equivalent to a one-layer backpropagation neural net.
- Logistic regression is the source of the sigmoid function used in backpropagation.
- Objective function for training is somewhat different.

Logistic Regression Objective

- Weights are set during training to maximise the **conditional data likelihood** :

$$W \leftarrow \arg\max_{W} \prod_{d \in D} P(Y^d \mid X^d, W)$$

where D is the set of training examples and Y^d and X^d denote, respectively, the values of Y and X for example d.

- Equivalently viewed as maximising the **conditional log likelihood** (CLL)

$$W \leftarrow \arg\max_{W} \sum_{d \in D} \ln P(Y^d \mid X^d, W)$$

Logistic Regression Objective

- Equivalently viewed as maximising the **conditional log likelihood** (CLL)

$$W \leftarrow \underset{W}{\operatorname{argmax}} \sum_{d \in D} \ln P(Y^d \mid X^d, W)$$

- The objective function

$$\min_W -\frac{1}{|D|} \sum_{d \in D} \left(Y^d \ln \left(\frac{\exp(W^\top X^d)}{1 + \exp(W^\top X^d)} \right) + (1 - Y^d) \ln \left(\frac{1}{1 + \exp(W^\top X^d)} \right) \right)$$

$$= \min_W \frac{1}{|D|} \sum_{d \in D} \ln\left(1 + \exp(W^\top X^d)\right) - Y^d W^\top X^d$$

Maximum margin classifiers (1)

Given a training set

Inputs $\mathbf{x}_1, \ldots, \mathbf{x}_N$

Targets

$t_1, \ldots, t_N$ where $t_n \in \{-1, 1\}$

Linear classifier

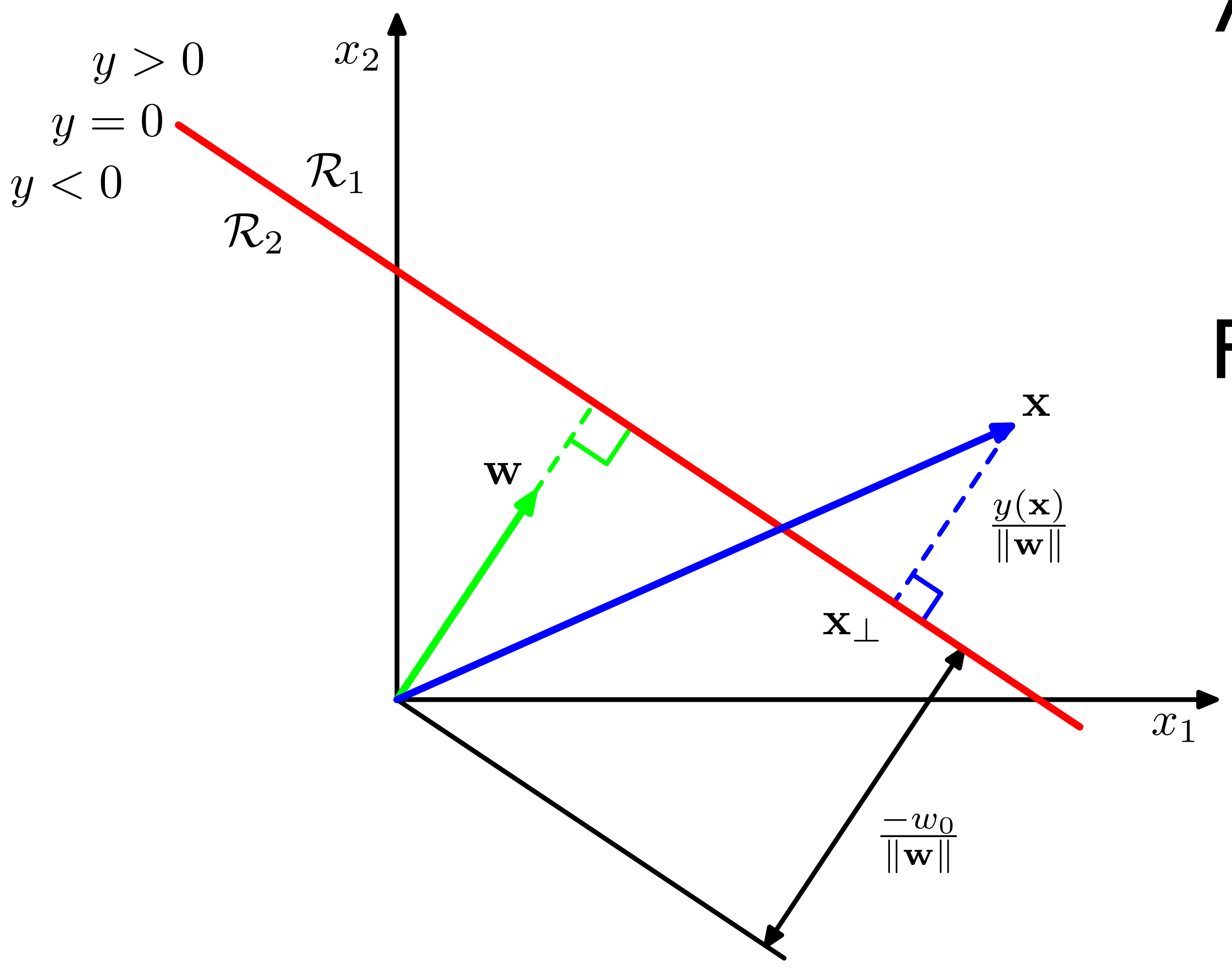
$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + b$$

Output sign of $y(\mathbf{x})$

so that $t_n y(\mathbf{x}_n) > 0$

for all points in the training set

Maximum margin classifiers(2)



Assuming $\phi(\mathbf{x}) = \mathbf{x}$

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$$

For points in the boundary

$y(\mathbf{x}) = 0$ and

$$\frac{\mathbf{w}^{\mathrm{T}}\mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|}$$

For arbitrary $\mathbf{x}$

$$\mathbf{x} = \mathbf{x}_\perp + r\frac{\mathbf{w}}{\|\mathbf{w}\|} \quad \text{and} \quad r = \frac{y(\mathbf{x})}{\|\mathbf{w}\|}$$

Maximum margin classifiers(3)

For all points correctly classified $t_n y(\mathbf{x}_n) > 0$, $b = \omega_0$

and $$\frac{t_n y(\mathbf{x}_n)}{\|\mathbf{w}\|} = \frac{t_n(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) + b)}{\|\mathbf{w}\|}$$

Maximum margin is found by

$$\underset{\mathbf{w},b}{\arg\max} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n \left[t_n \left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) + b \right) \right] \right\}$$

However, this is too difficult to optimise !

Maximum margin classifiers(4)

Note that if $\mathbf{w} \rightarrow \kappa \mathbf{w}$ $\quad t_n y(\mathbf{x}_n)/\|\mathbf{w}\|$ is unchanged

$b \rightarrow \kappa b$

Setting $t_n \left(\mathbf{w}^{\mathrm{T}} \phi(\mathbf{x}_n) + b\right) = 1$ for the support vectors:

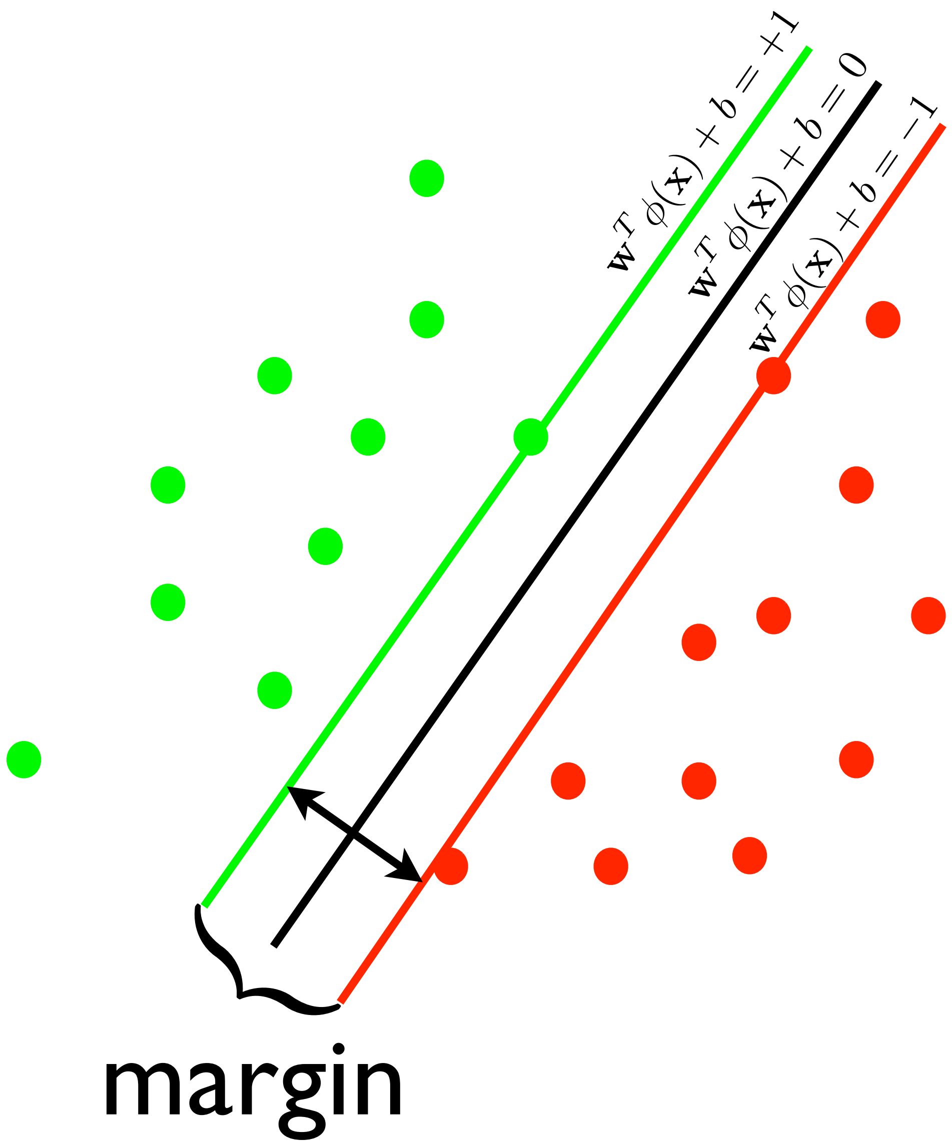
Quadratic programming

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

s.t.

$$t_n \left(\mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n) + b\right) \geqslant 1, \quad n = 1, \ldots, N$$

Support Vector Machines



Quadratic programming

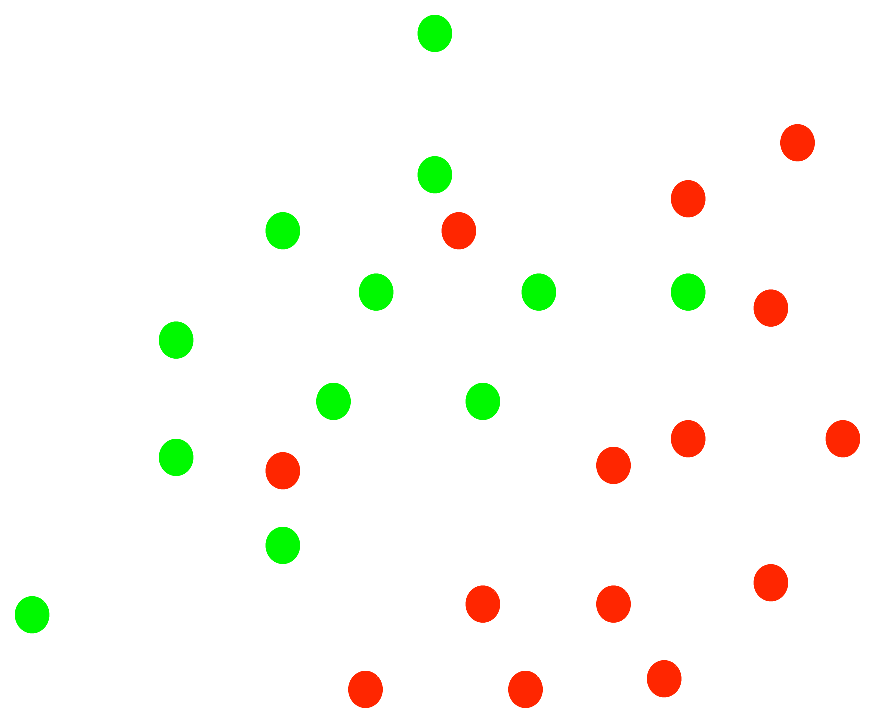
$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

s.t.

$$t_n\left(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)+b\right) \geqslant 1, \quad n=1,\ldots,N$$

- Solve efficiently by quadratic programming (QP)
- Hyperplane defined by support vectors

What happens if the data is not linearly separable?



Quadratic programming

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

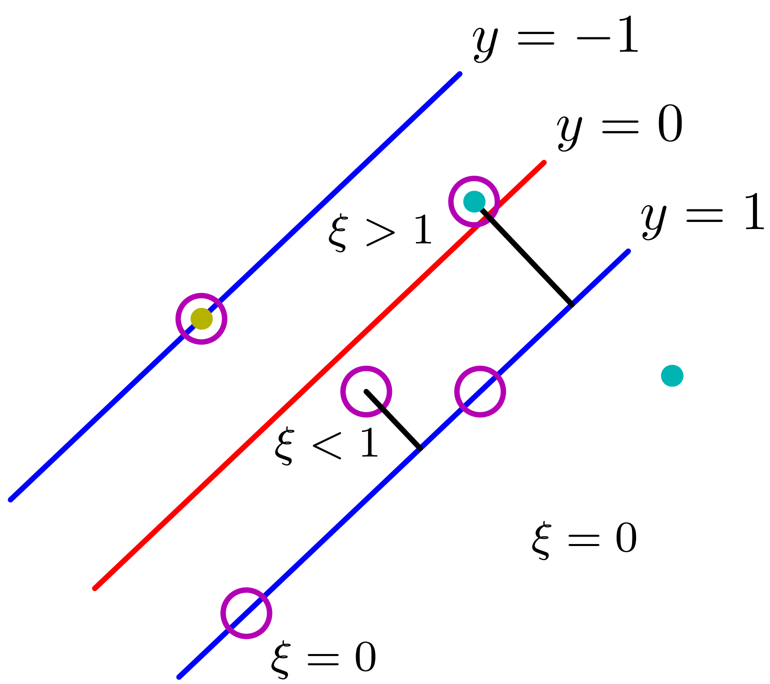
s.t.

$$t_n \left(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + b\right) \geqslant 1, \quad n = 1, \dots, N$$

If margin ≥ 1, don't care

If margin < 1, pay linear penalty

Soft Margin Classification



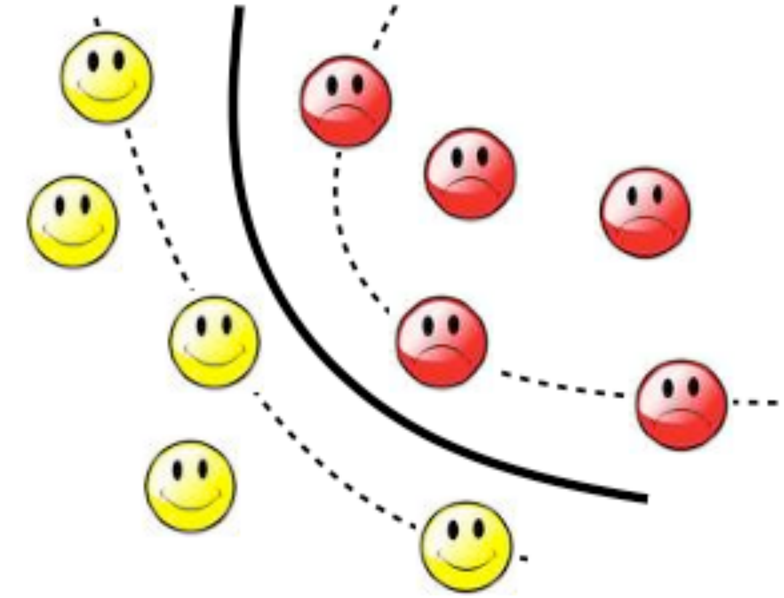
Quadratic programming

$$\underset{\mathbf{w},b}{\arg\min}\; C\sum_{n=1}^{N}\xi_n + \frac{1}{2}\|\mathbf{w}\|^2$$

s.t.

$$t_n y(\mathbf{x}_n) \geqslant 1 - \xi_n \qquad n = 1,\ldots,N$$
$$\xi_n \geqslant 0$$

THE UNIVERSITY OF SYDNEY



Linear Regression

C. Bishop, *Pattern Recognition and Machine Learning,* Chapter 3: Linear Models for Regression Springer New York, 2006

Linear Regression

Goal: Predict the value of one or more continuous target variables t given the value of a D-dimensional vector x of input variables.

Given training data of size N: $\{(\mathbf{x}_n, t_n)\}_{n=1,\ldots,N}$

↓

Deterministic: Construct function $y(\mathbf{x})$ to predict t.

Probabilistic: Find predictive distribution $p(t|\mathbf{x})$

Areas for Applying Regression

Academia

Robotics: Autonomous Navigation, Agriculture,

Environmental Monitoring.

Biology: Cancer Research, Metabolic inference,

Brain and Mind Centre, Milk Production.

Astronomy: Light Curve Modelling.

Social Sciences: Criminology, Subnational/International Conflict,

Linguistics, Aged Care Facilities.

Industry

Retail: Amazon, Facebook, Google, etc.

Consultancy: Mining, Energy Generation and Distribution.

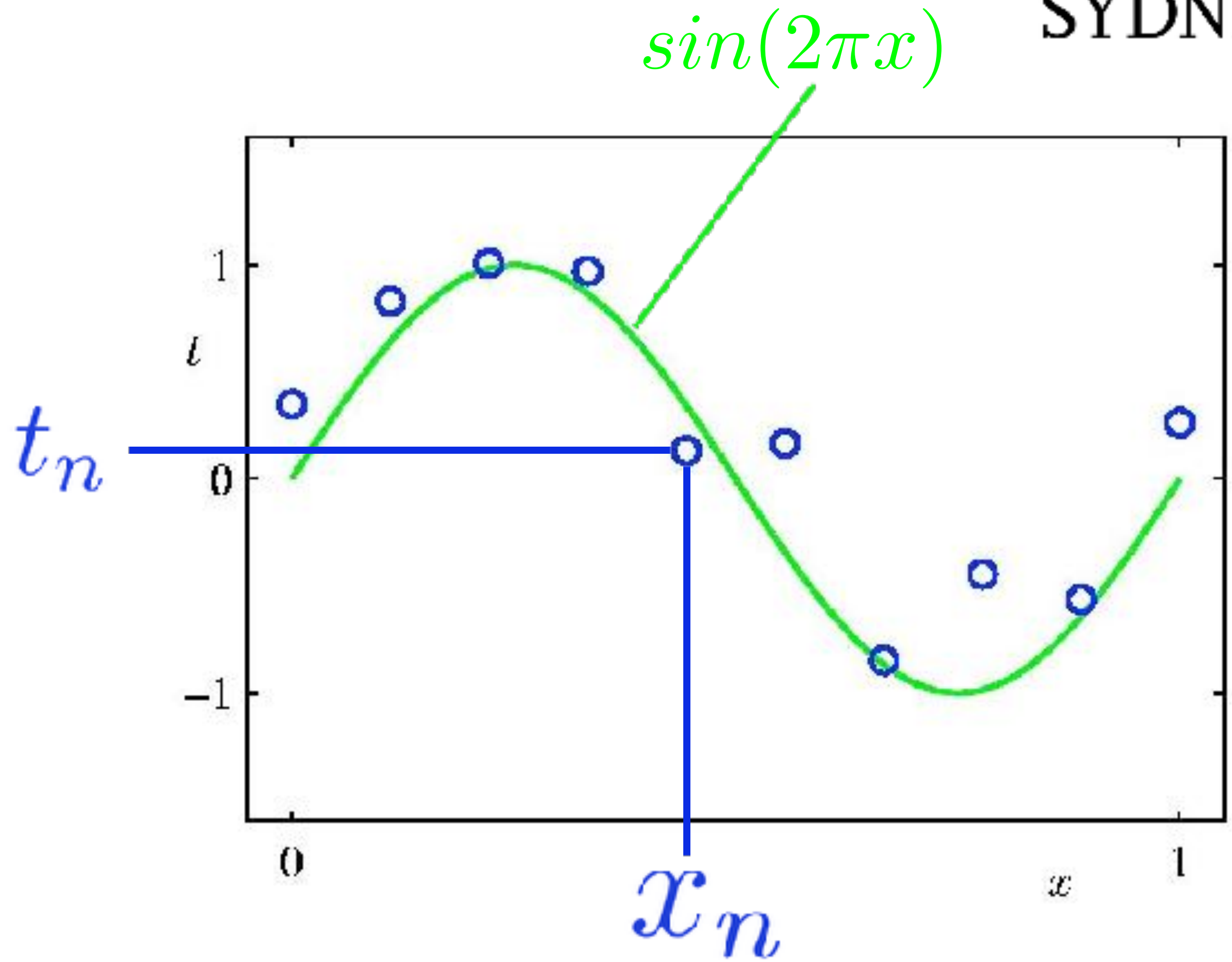
Banks: Loan estimation, Risk assessment.

Example: Polynomial Curve Fitting

$$t = f(x) = \sin(2\pi x)$$

$$N = 10$$

$$\mathcal{D} = \{(x_n, t_n)\}_{n=1,\dots,10}$$



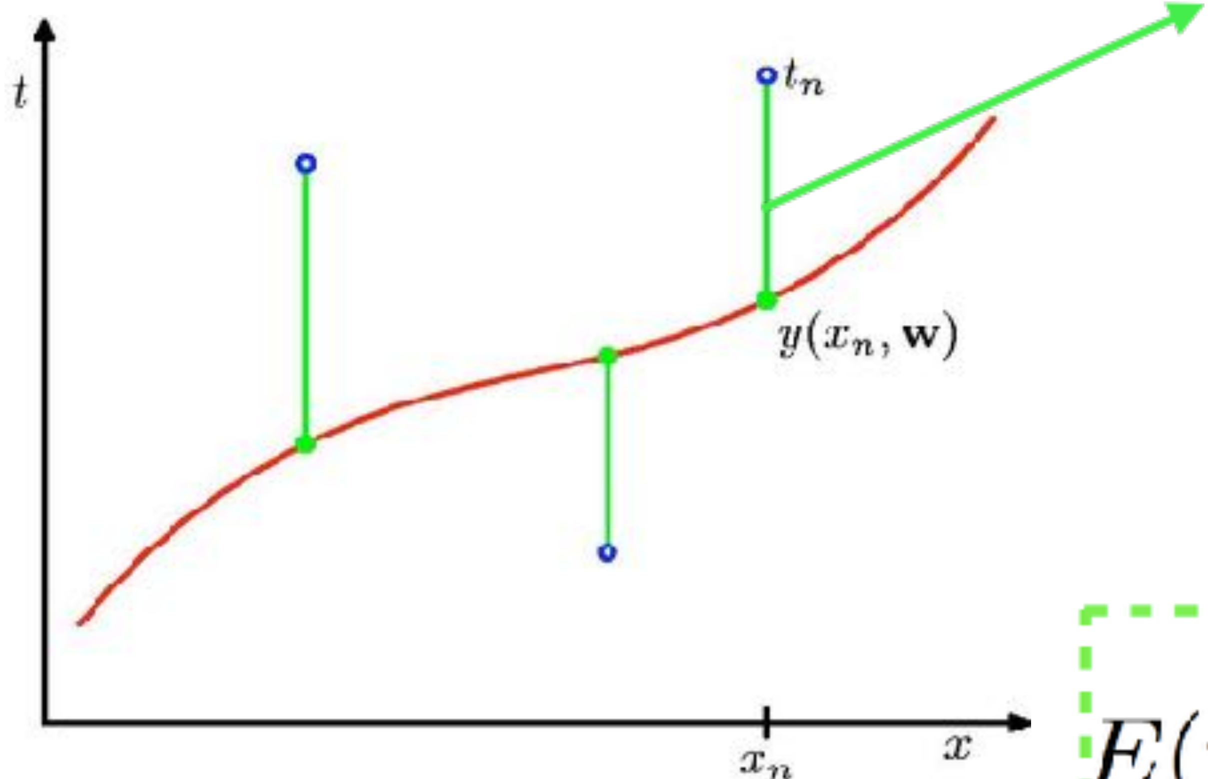
Polynomial fit:

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

Example: Polynomial Curve Fitting

Sum-of-Squares Error Function $E(\mathbf{w})$

Polynomial Fit: $y(x, \mathbf{w})$



$$E_n = y(x_n, \mathbf{w}) - t_n$$

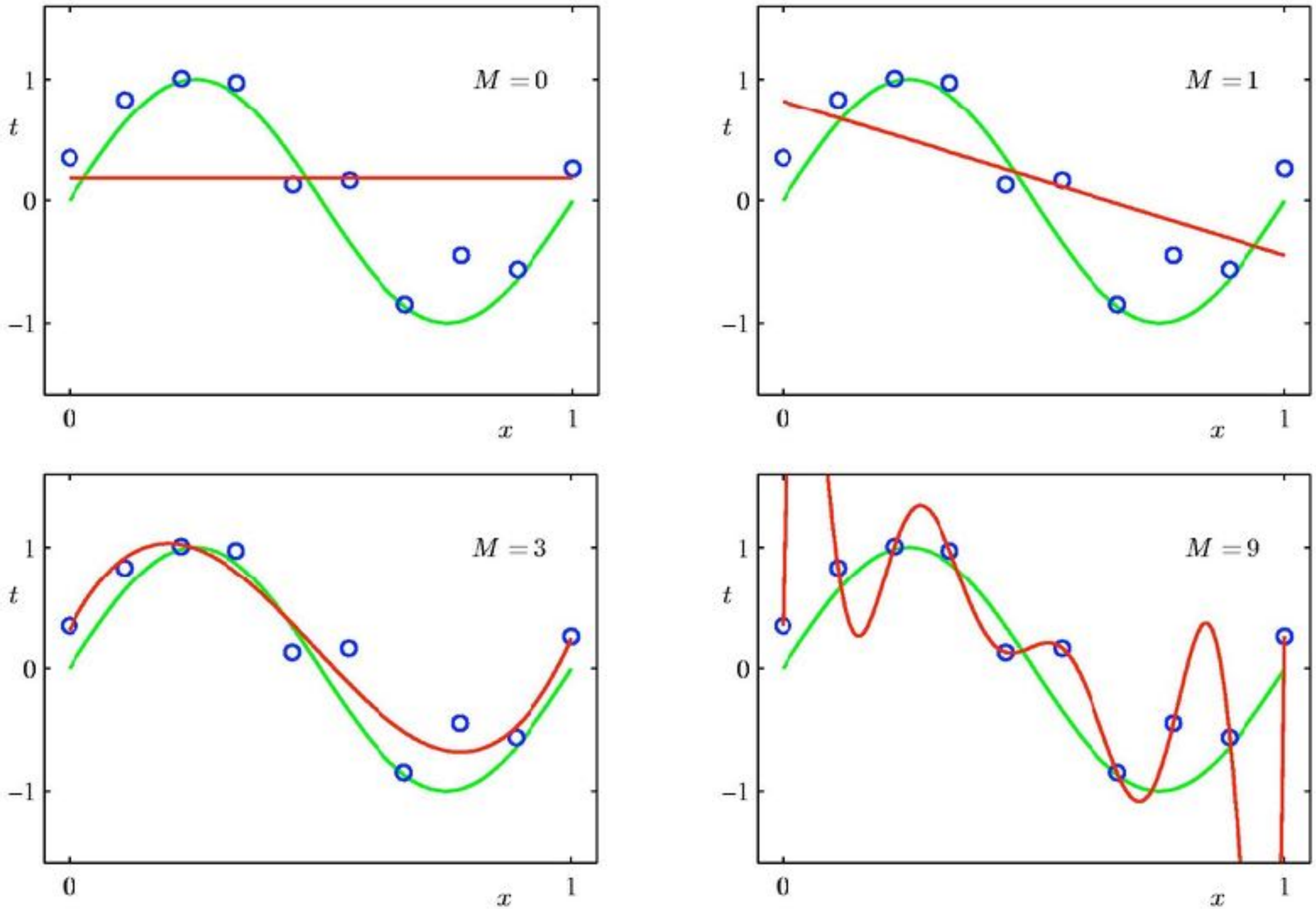
$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} E_n^2$$

$$\boxed{E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2}$$

Best fit: $\mathbf{w}^\star = \underset{\mathbf{w}}{\operatorname{argmin}}\, E(\mathbf{w})$

Example: Polynomial Curve Fitting

Degree of Polynomial: M

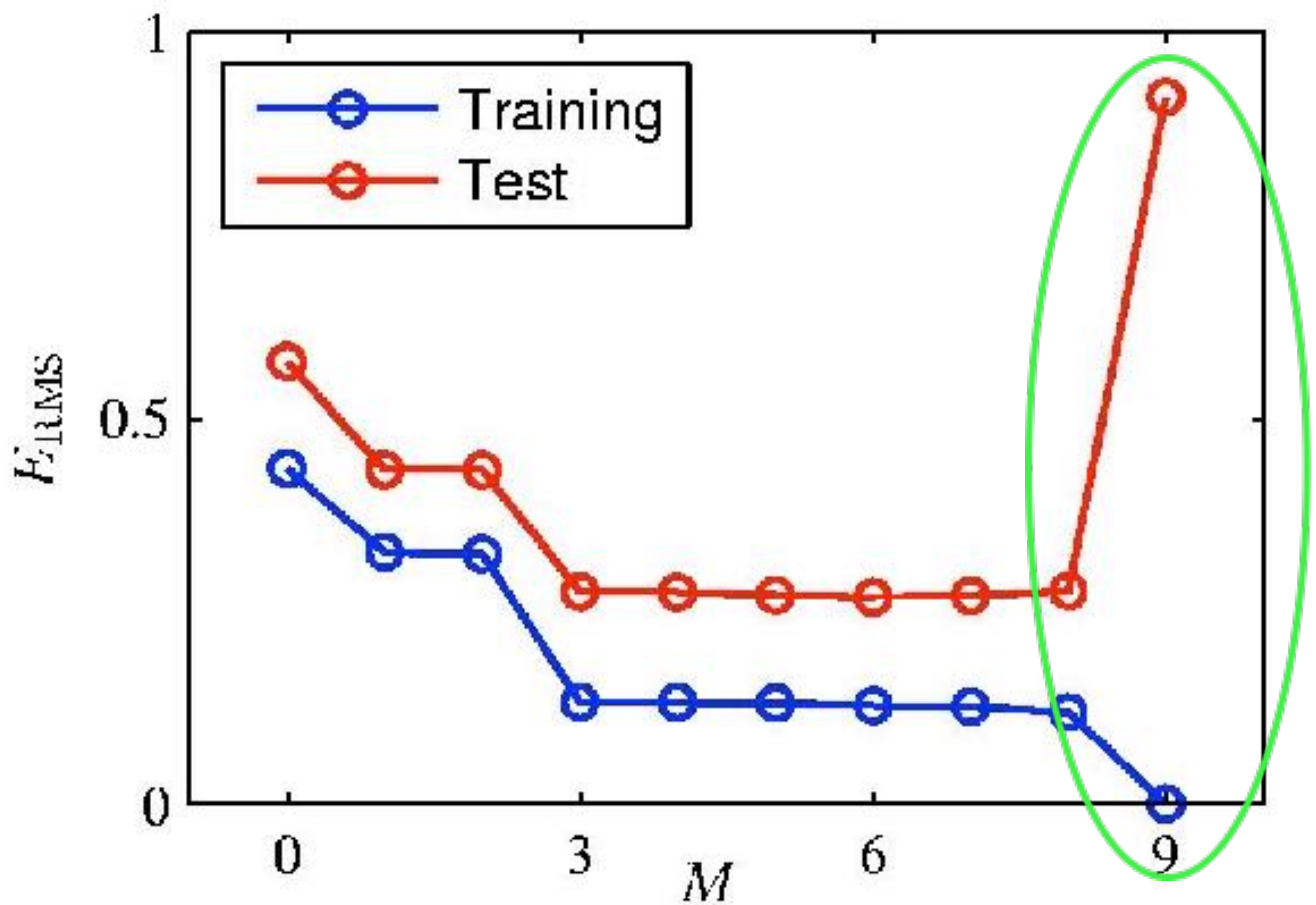


Example: Polynomial Curve Fitting

Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^{\star})/N}$

Testset: N=10

Overfitting



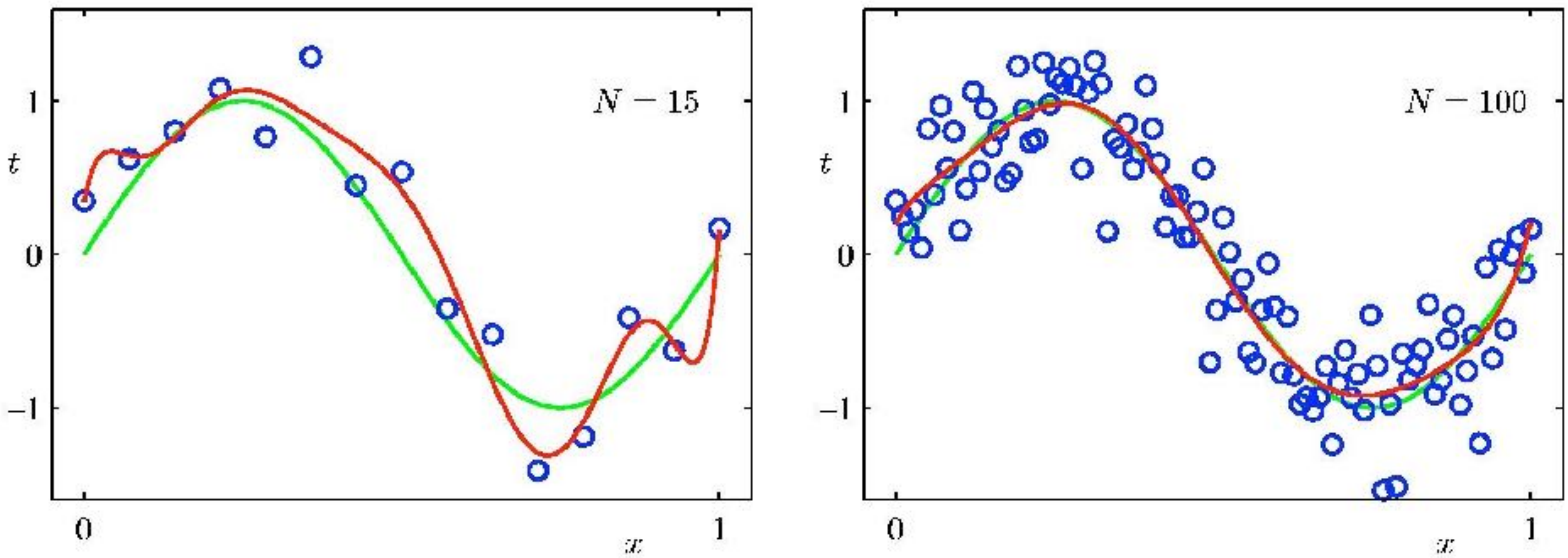
Example: Polynomial Curve Fitting

Polynomial Coefficients

	$M = 0$	$M = 1$	$M = 3$	$M = 9$
$w_0^\star$	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

Example: Polynomial Curve Fitting

Behaviour with dataset size $(M = 9)$:



Example: Polynomial Curve Fitting

Regularisation:

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

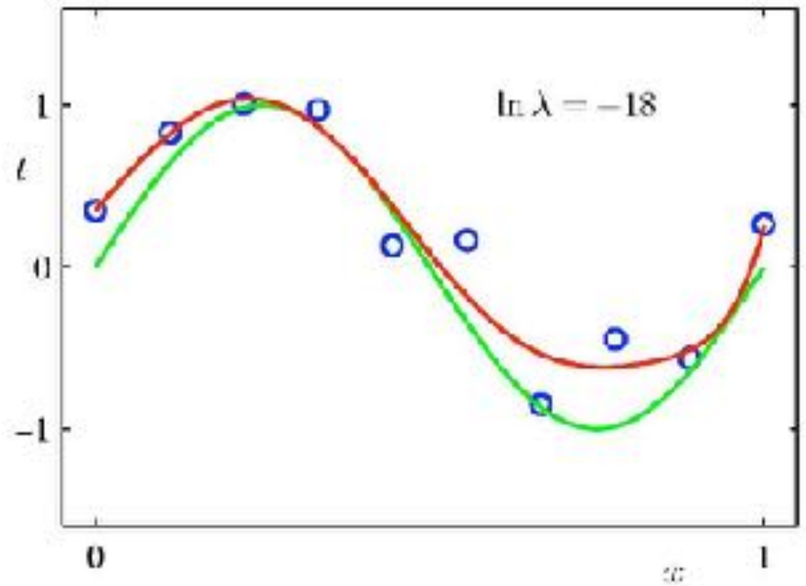
Penalise large coefficient values

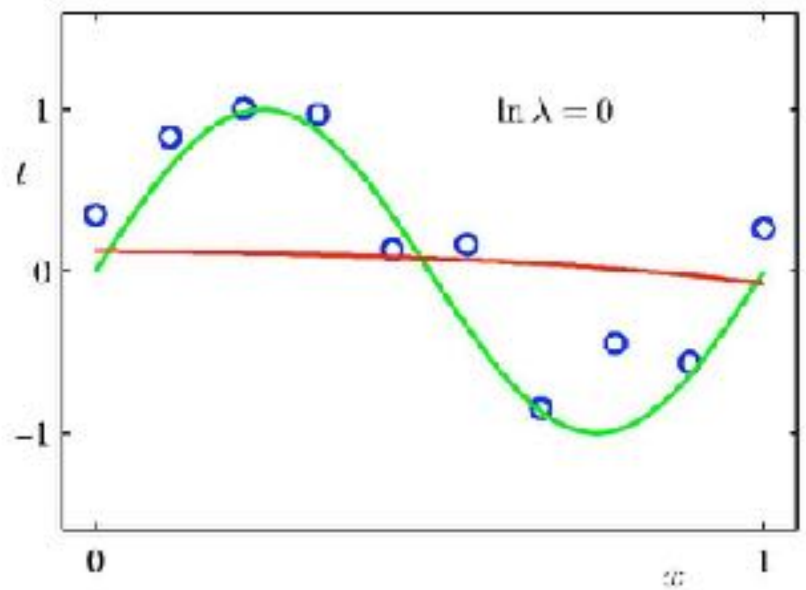
$$\|\mathbf{w}\|^2 \equiv \mathbf{w}^{\mathrm{T}}\mathbf{w} = w_0^2 + w_1^2 + \ldots + w_M^2$$

λ encodes the relative importance of the regularisation.

Example: Polynomial Curve Fitting

Regularisation:

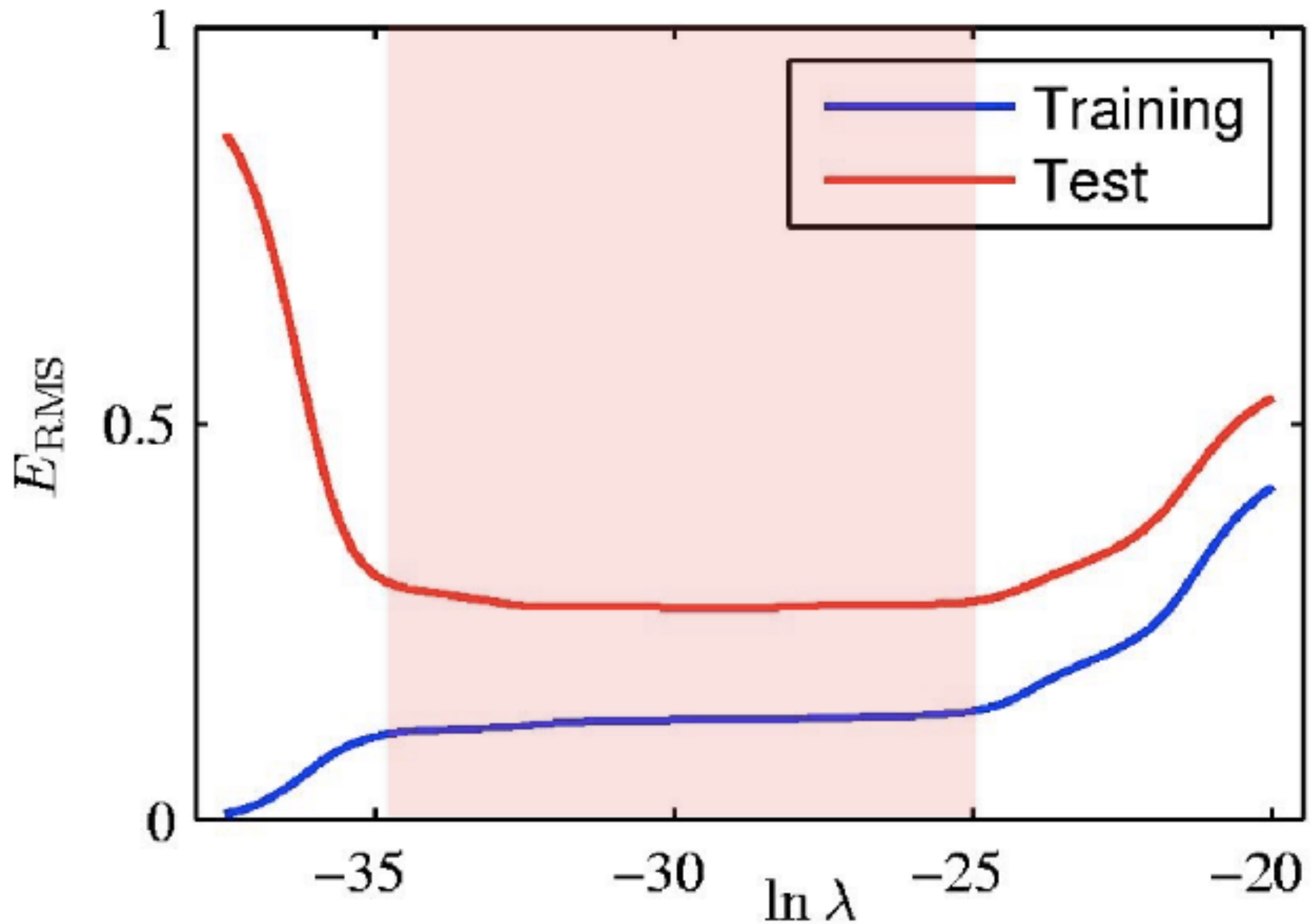




	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

Example: Polynomial Curve Fitting

Regularisation: $(M = 9)$



Linear Basis Function Models

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

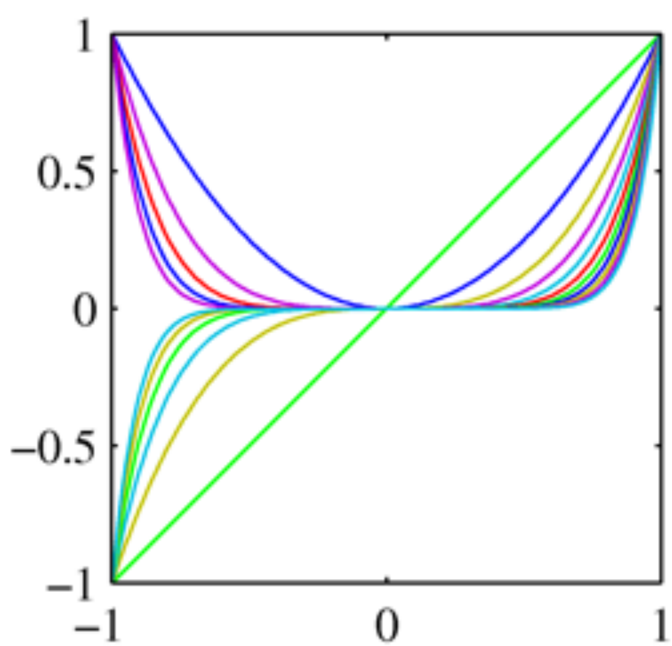
$$\mathbf{w} = (w_0, \ldots, w_{M-1})^{\mathrm{T}} \quad \boldsymbol{\phi} = (\phi_0, \ldots, \phi_{M-1})^{\mathrm{T}}$$

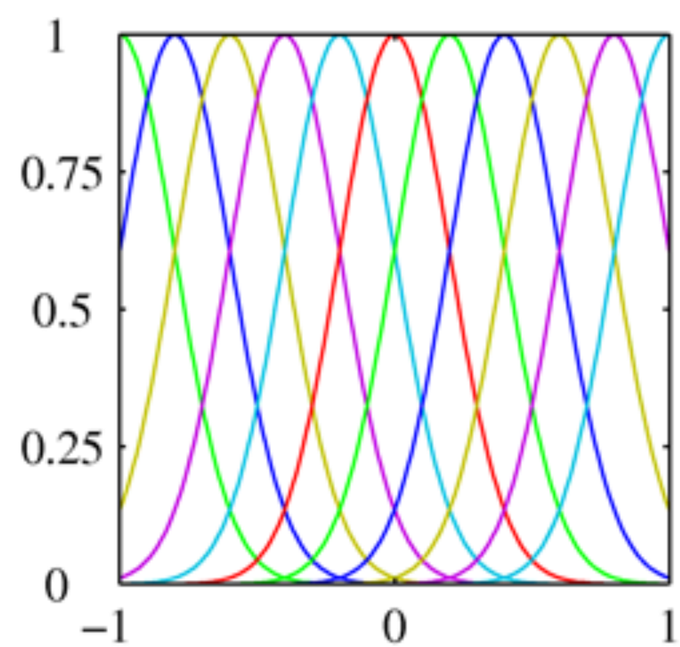
$\phi_j(\mathbf{x})$ are the *basis functions.* $\phi_0(\mathbf{x}) = 1$

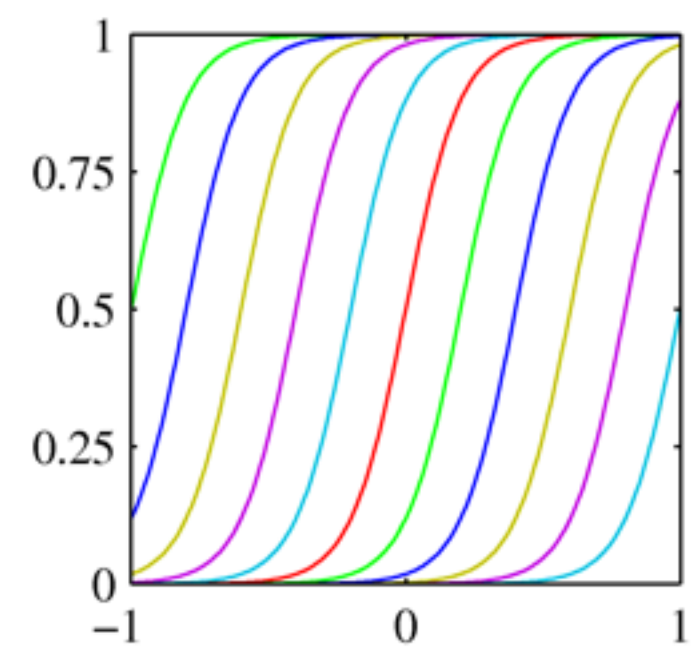
Linear Basis Function Models

Examples of Basis Functions:

Polynomial Basis Functions	Gaussian Basis Functions	Sigmoidal Basis Functions
$\phi_j(x) = x^j$	$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$	$\phi_j(x) = \frac{1}{1+\exp\left\{-\frac{x-\mu_j}{s}\right\}}$







Modelling Noisy Observations

Lets assume observations from a deterministic function with added Gaussian noise.

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon \qquad \text{with} \qquad \epsilon \sim \mathcal{N}(0, \beta^{-1})$$

equivalently,

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

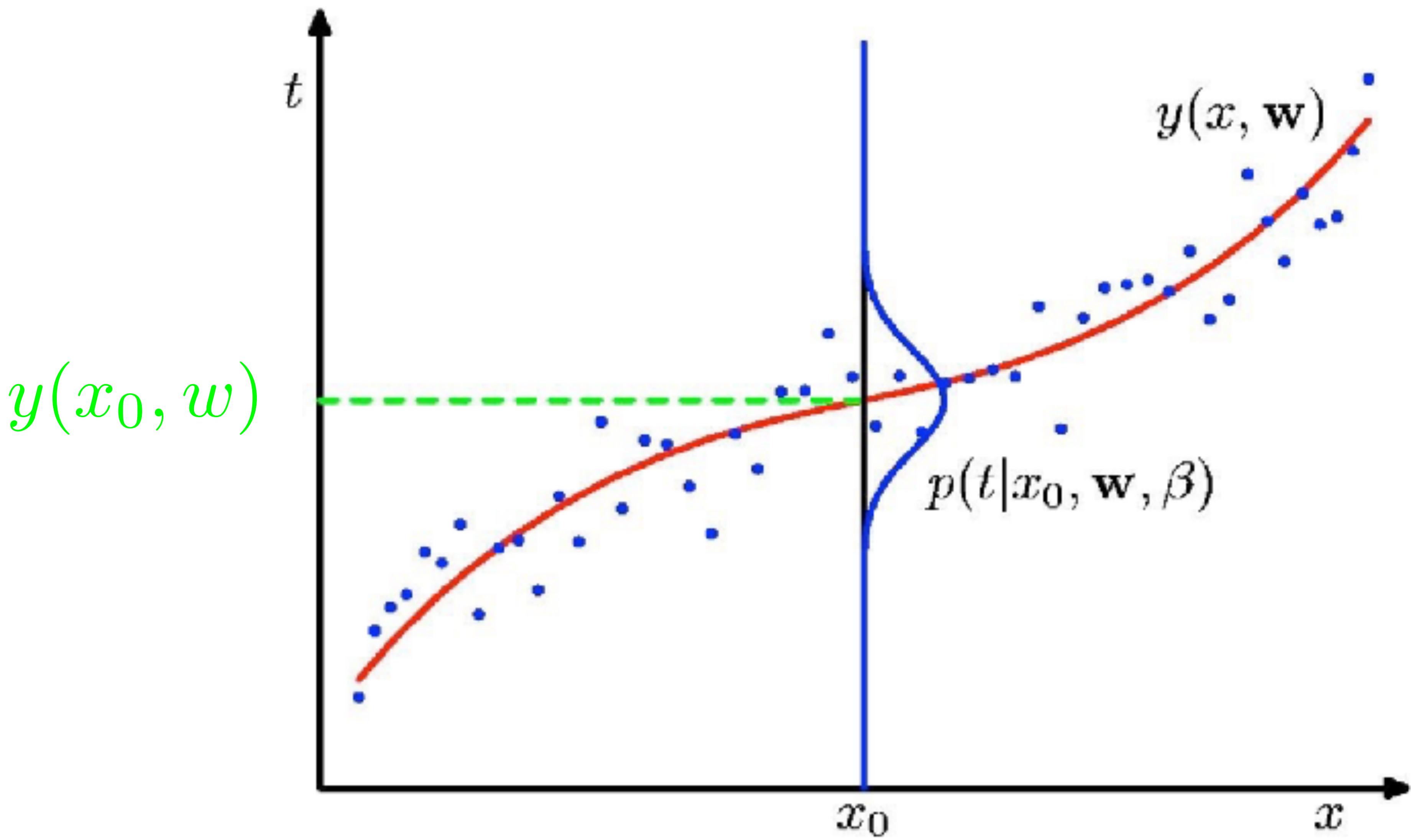
Given the training data: $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ $\boldsymbol{\tau} = \{t_1, ..., t_N\}$

The expression of the likelihood of the iid data given the model is:

$$p(\boldsymbol{\tau}|\mathbf{X}, \mathbf{w}, \beta^{-1}) = \prod_{n=1}^{N} \mathcal{N}\left(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}\right)$$

Modelling Noisy Observations

Lets assume observations from a deterministic function with added Gaussian noise.



Maximum Likelihood

$$p(\boldsymbol{\tau}|\mathbf{X}, \mathbf{w}, \beta^{-1}) = \prod_{n=1}^{N} \mathcal{N}\left(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}\right)$$

$$= \prod_{n=1}^{N} \sqrt{\frac{\beta}{2\pi}} \exp - \frac{\beta(t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n))^2}{2}$$

$$= \left(\frac{\beta}{2\pi}\right)^{N/2} \prod_{n=1}^{N} \exp - \frac{\beta(t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n))^2}{2}$$

$$\ln(\cdot) = \ln(\cdot)$$

$$\ln p(\boldsymbol{\tau}|\mathbf{X}, \mathbf{w}, \beta^{-1}) = \frac{N}{2}\ln\beta - \frac{N}{2}\ln 2\pi - \beta \sum_{n=1}^{N} \frac{\left(t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\right)^2}{2}$$

$$= \frac{N}{2}\ln\beta - \frac{N}{2}\ln 2\pi - \beta E(\mathbf{w})$$

Maximum Likelihood

$$\mathbf{w}_{\mathrm{ML}} = \underset{\mathbf{w}}{\operatorname{argmax}} \; \ln p(\boldsymbol{\tau}|\mathbf{X}, \mathbf{w}, \beta^{-1})$$

$$\Rightarrow \qquad \left. \frac{\partial \ln p(\boldsymbol{\tau}|\mathbf{w}, \beta^{-1})}{\partial \mathbf{w}} \right|_{\mathbf{w}_{\mathrm{ML}}} = 0$$

$$\sum_{n=1}^{N} \left(t_n - \mathbf{w}_{\mathrm{ML}}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n) \right) \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}} = 0$$

Solving for $\mathbf{w}_{\mathrm{ML}}$

$$\boxed{\mathbf{w}_{\mathrm{ML}} = \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\tau}}$$

where the design matrix is:

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

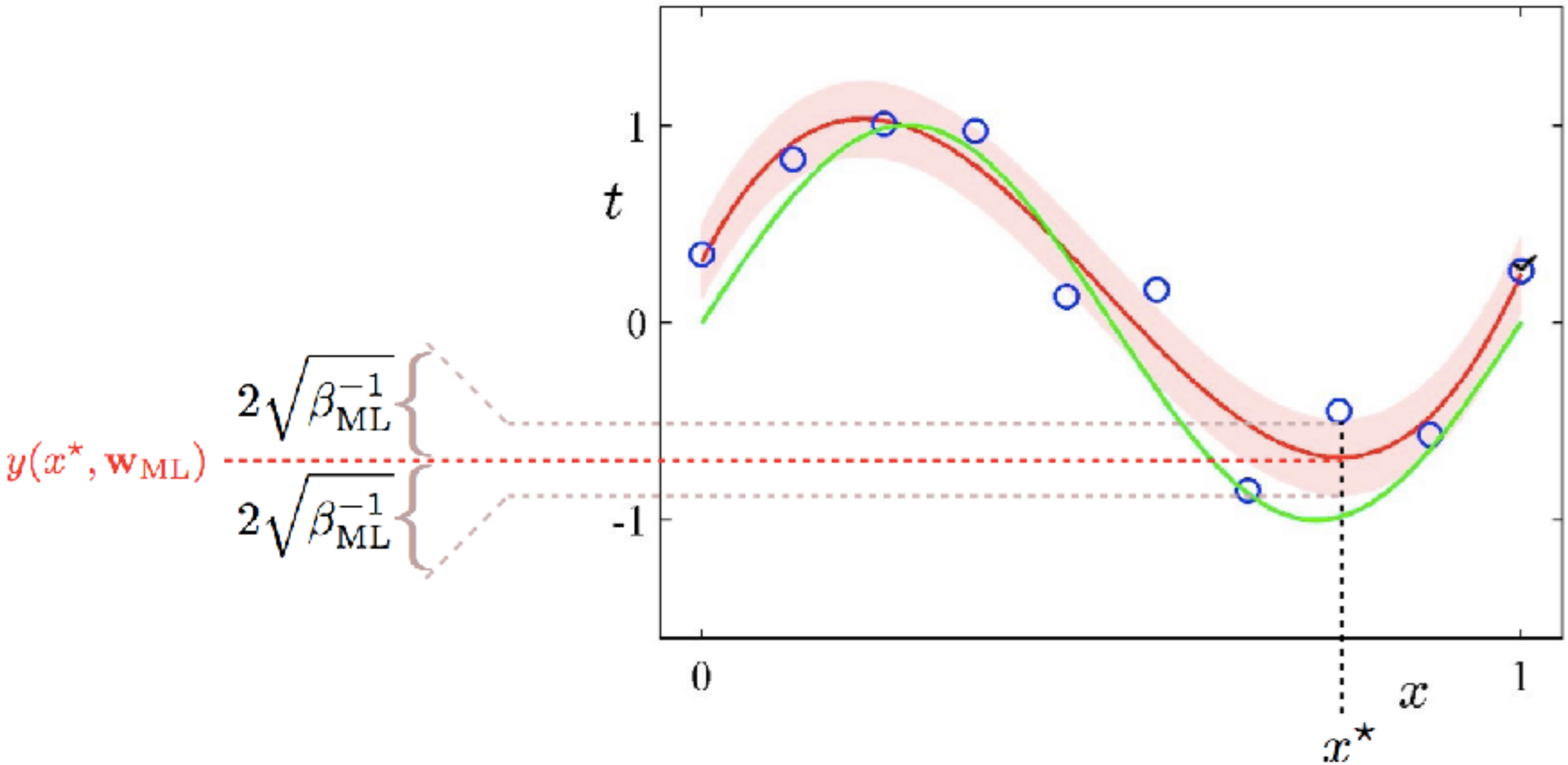
Maximum Likelihood

We can also maximise the log likelihood with respect to the noise β .

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

Predictive Distribution

$$p(t|x, \mathbf{w}_{\mathrm{ML}}, \beta_{\mathrm{ML}}) = \mathcal{N}(t|y(x, \mathbf{w}_{\mathrm{ML}}), \beta_{\mathrm{ML}}^{-1})$$



Sequential Learning

Deal with Large Datasets

Two options: Incremental or Stochastic selection of data-points.

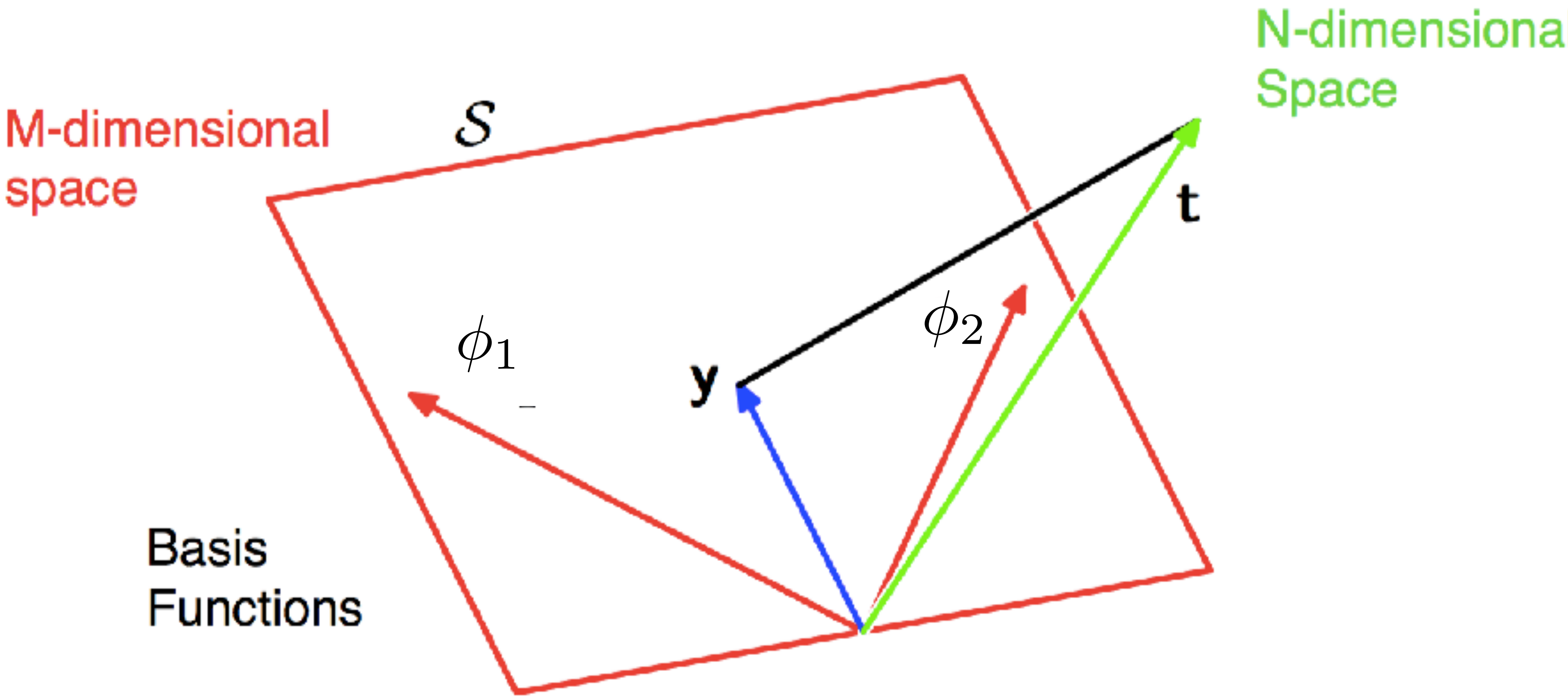
Stochastic Gradient Descent (for iteration i)

$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} - \eta \nabla E_n$$

$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} + \eta \left(t_n - \mathbf{w}^{(i)\mathrm{T}} \boldsymbol{\phi}_n \right) \boldsymbol{\phi}_n$$

Least Mean Square algorithm (LMS)

Geometry of Least Squares



Regularised Least Squares

Are we dealing with overfitting? NO

The expression of the log likelihood:

$$\ln p(\boldsymbol{\tau}|\mathbf{w}, \beta^{-1}) = \frac{N}{2}\ln\beta - \frac{N}{2}\ln 2\pi - \beta E(\mathbf{w})$$

Introduce regulariser: $E(\mathbf{w}) = \underbrace{E_D(\mathbf{w})}_{\text{Data Term}} + \underbrace{\lambda E_W(\mathbf{w})}_{\text{Regularisation Term}}$

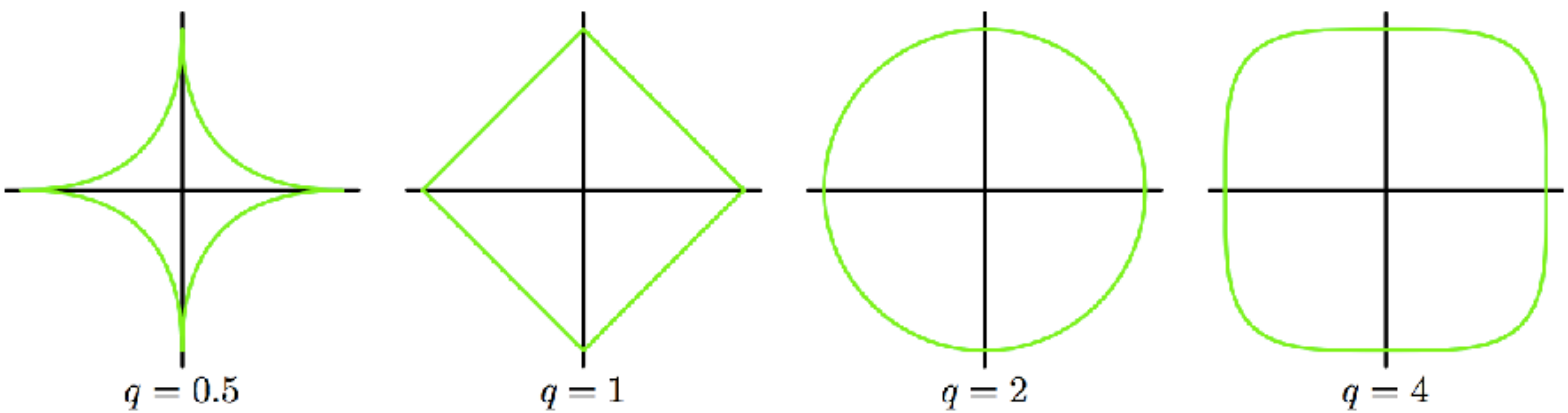
$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(t_n - \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)\right)^2 + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

Following previous maximisation procedure:

$$\mathbf{w} = \left(\lambda\mathbf{I} + \mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}\boldsymbol{\tau}$$

Generalised Regulariser

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$



Multiple Outputs

K > 1 target variables.

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

$\mathbf{W}$ is a weight matrix $M \times K$

$\mathbf{T}$ is a target matrix $N \times K$

$$\begin{aligned} \ln p(\mathbf{T}|\mathbf{X}, \mathbf{W}, \beta) &= \sum_{n=1}^{N} \ln \mathcal{N}(\mathbf{t}_n | \mathbf{W}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}\mathbf{I}) \\ &= \frac{NK}{2} \ln\left(\frac{\beta}{2\pi}\right) - \frac{\beta}{2} \sum_{n=1}^{N} \left\| \mathbf{t}_n - \mathbf{W}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n) \right\|^2 \end{aligned}$$

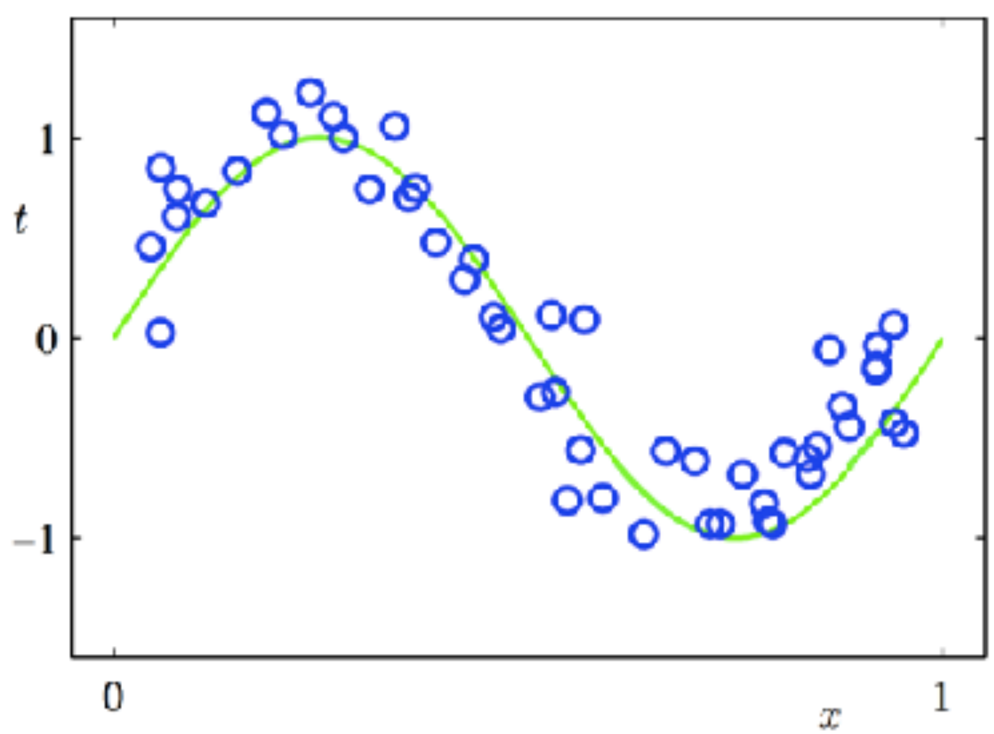
$$\mathbf{W}_{\mathrm{ML}} = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}}\mathbf{T}.$$

Bias-Variance Trade Off

Minimising both the weight vector and regularisation coefficient is not the right approach.

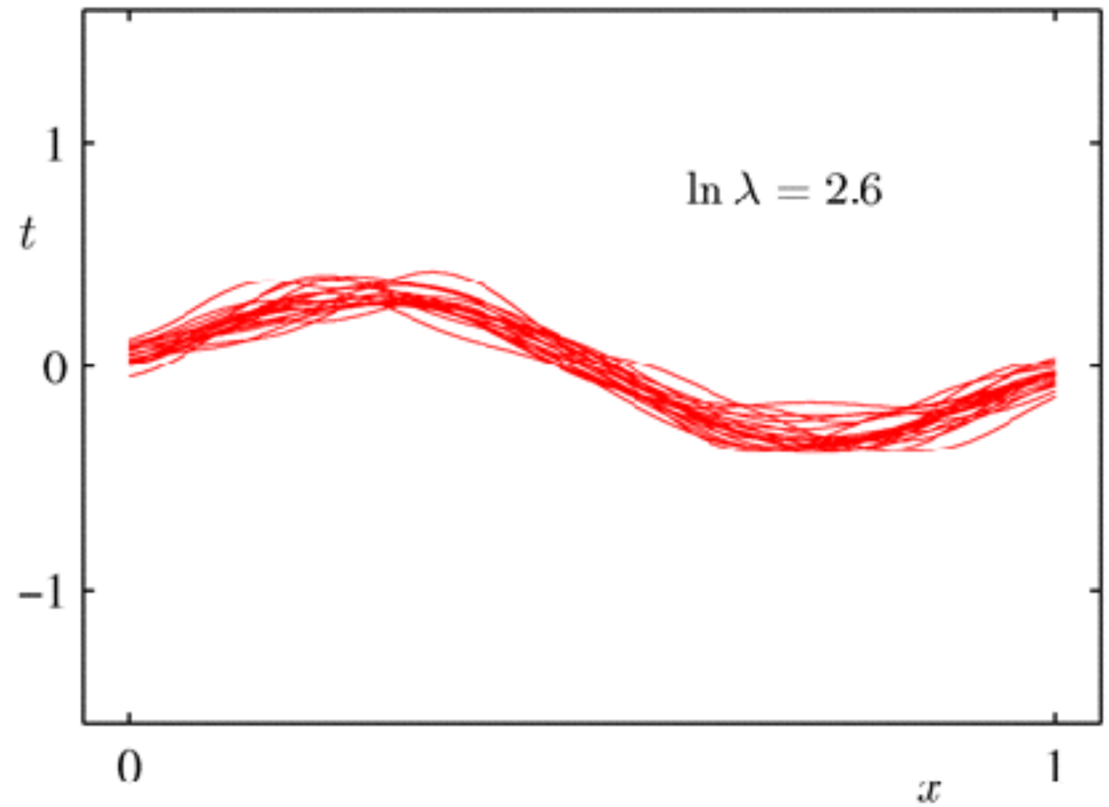
Over-fitting in the maximum-likelihood setting does not happen when marginalising out parameters as in the Bayesian setting (next lecture).

Let us look at various fits for subsets of the training data.

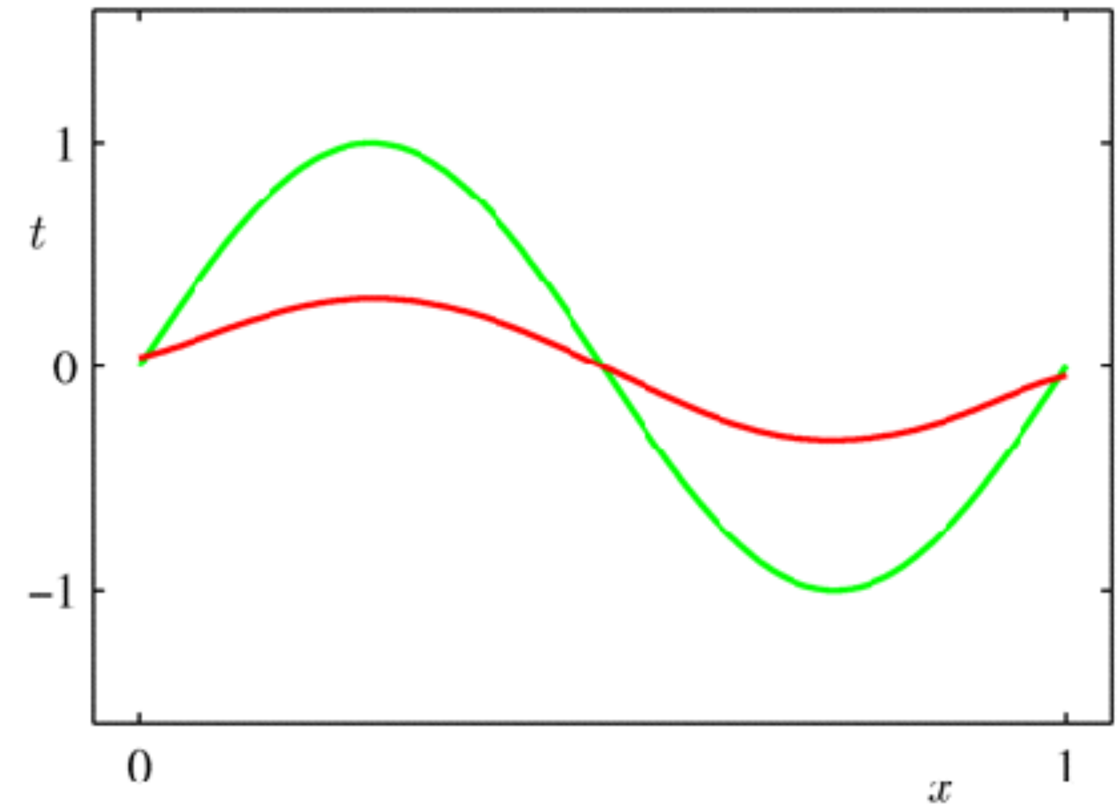


Bias-Variance Visualisation

20 datasets with varying regularisation parameter.



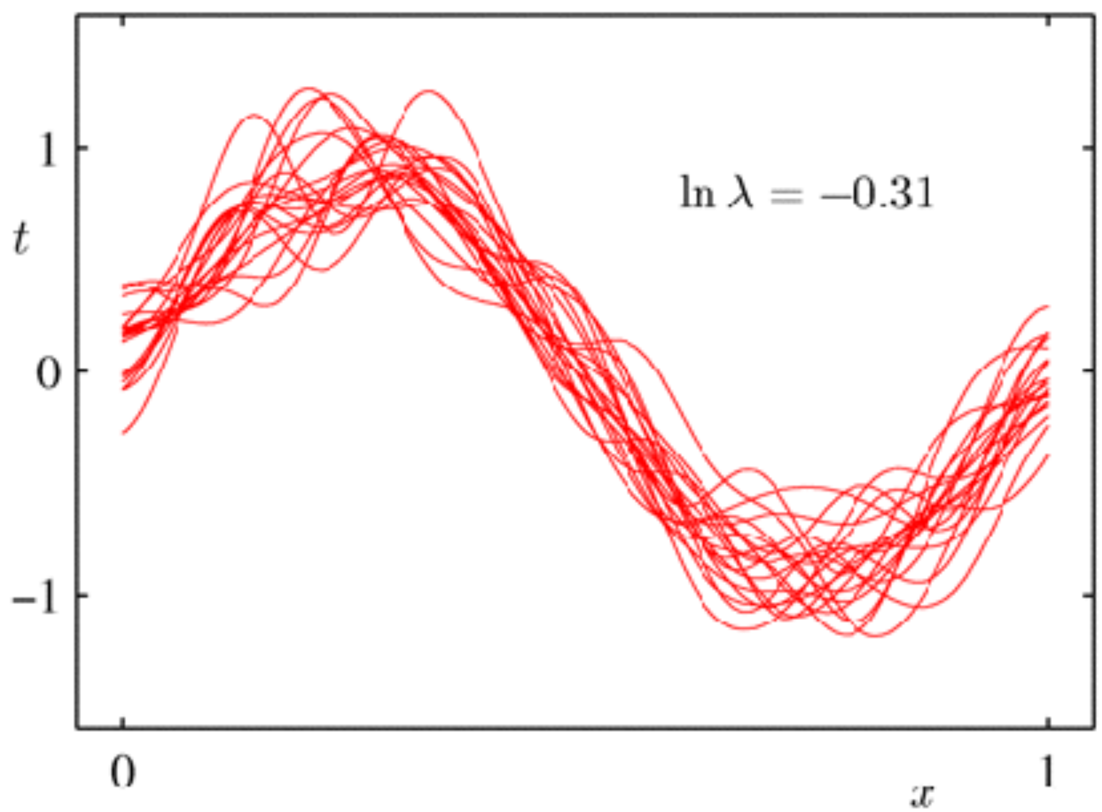
Result of fitting the model to each dataset.



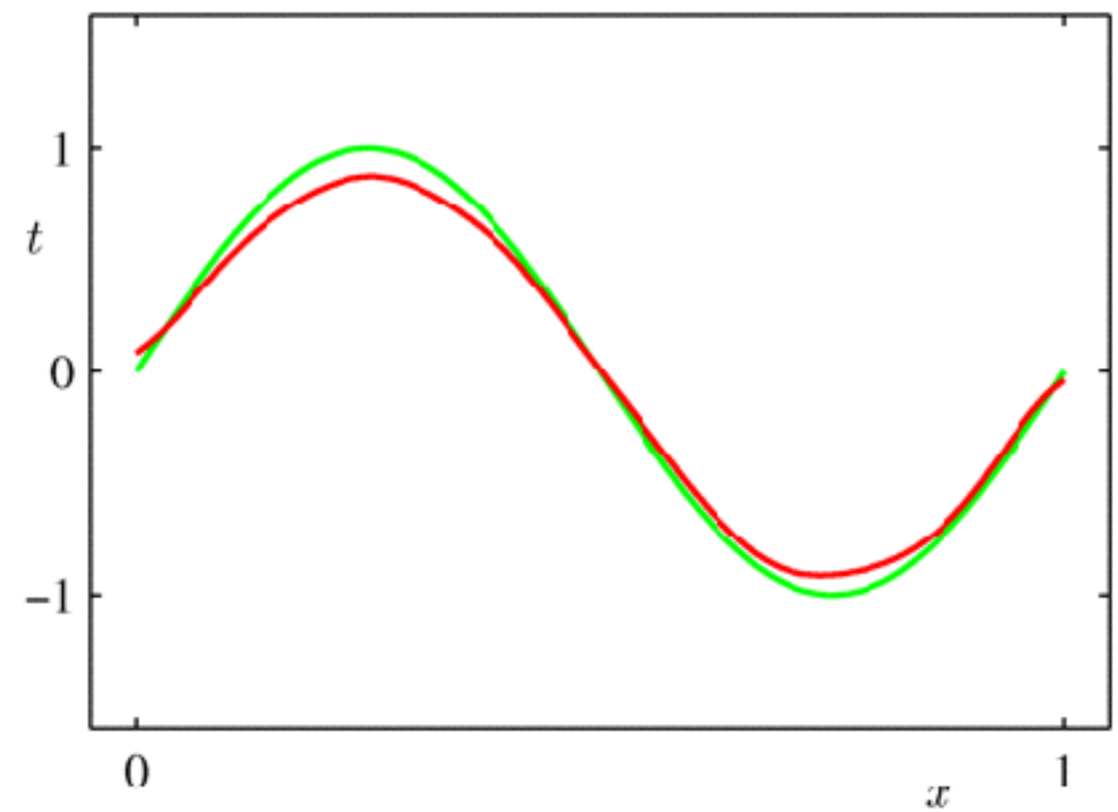
Average of the fits.

Bias-Variance Visualisation

20 datasets with varying regularisation parameter.



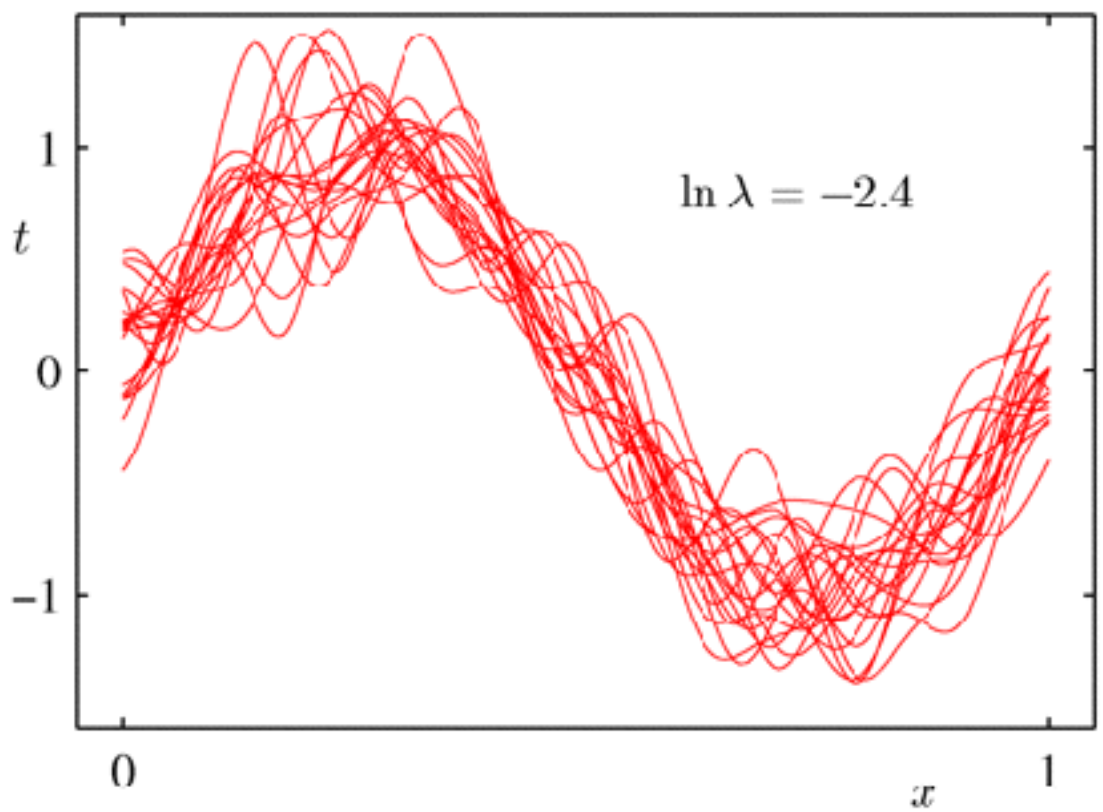
Result of fitting the model to each dataset.



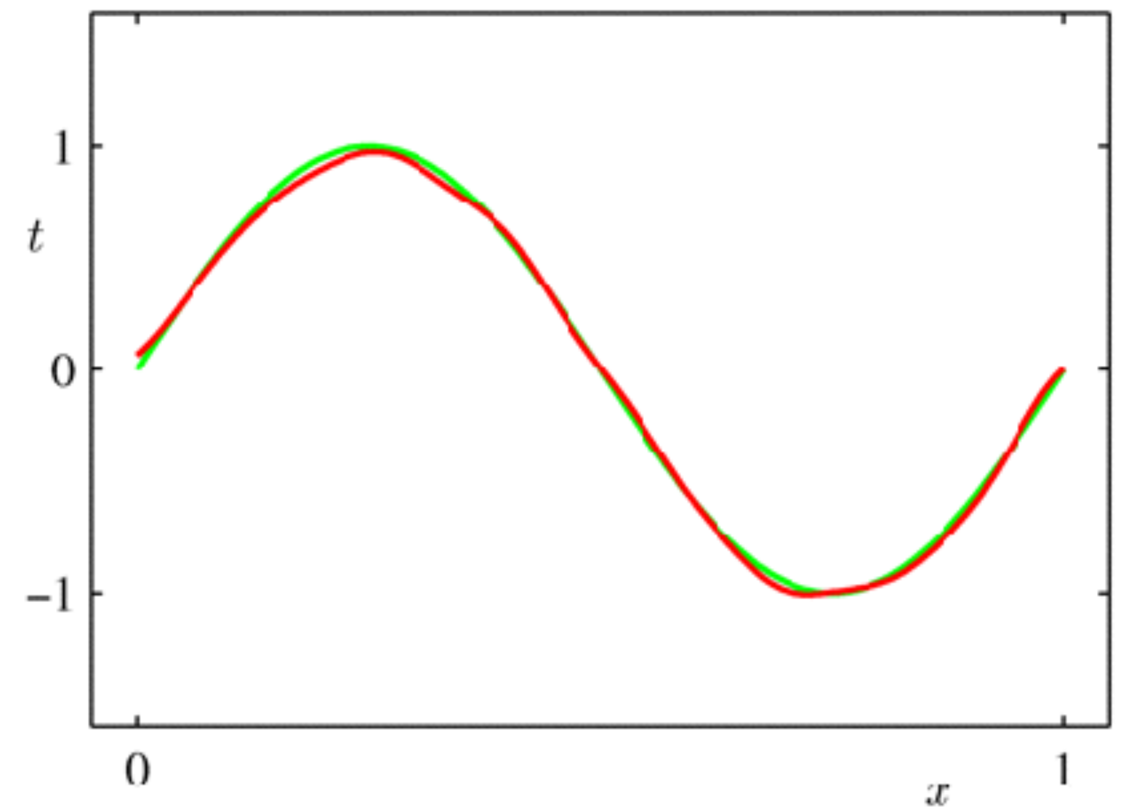
Average of the fits.

Bias-Variance Visualisation

20 datasets with varying regularisation parameter.



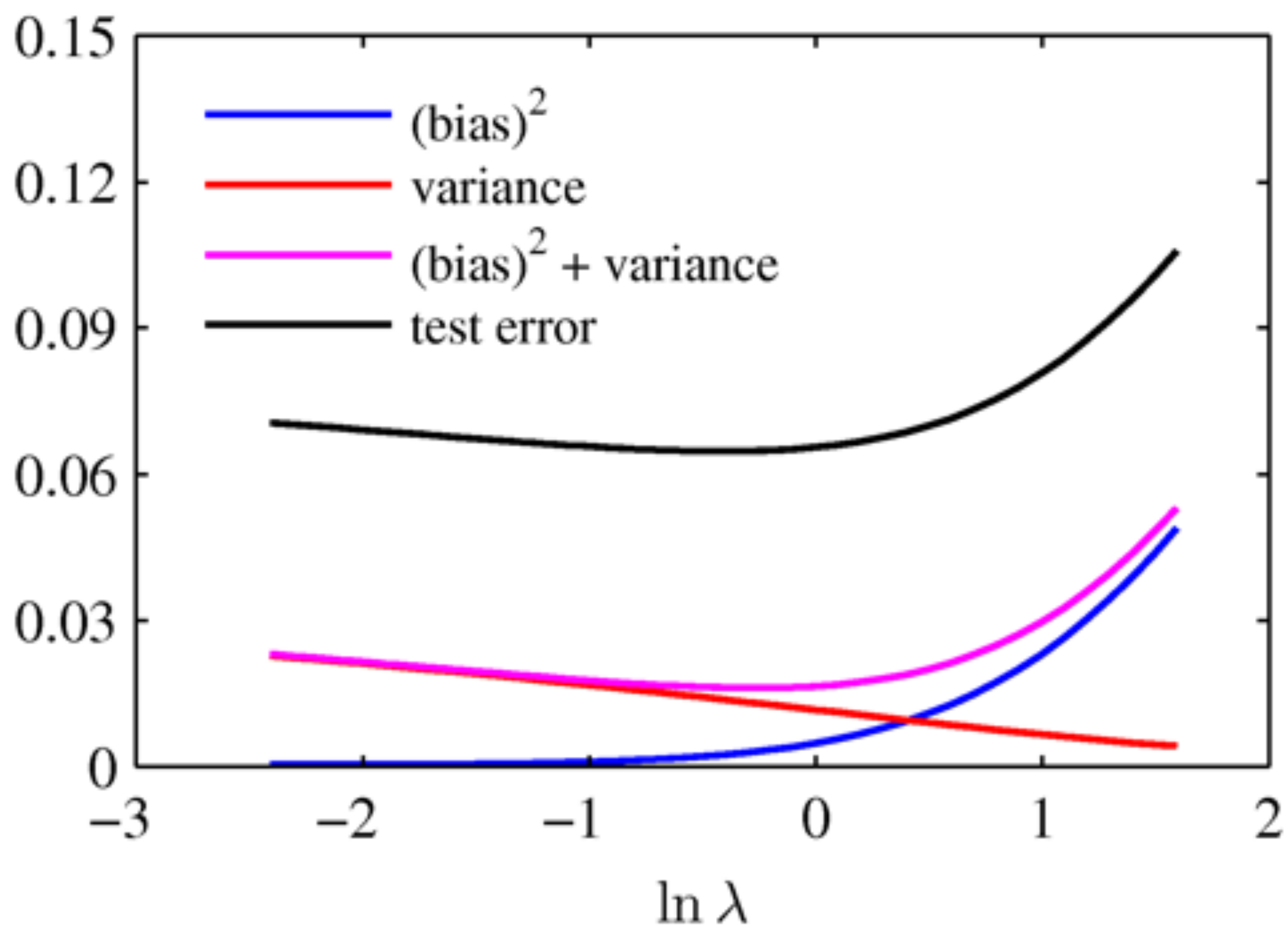
Result of fitting the model to each dataset.



Average of the fits.

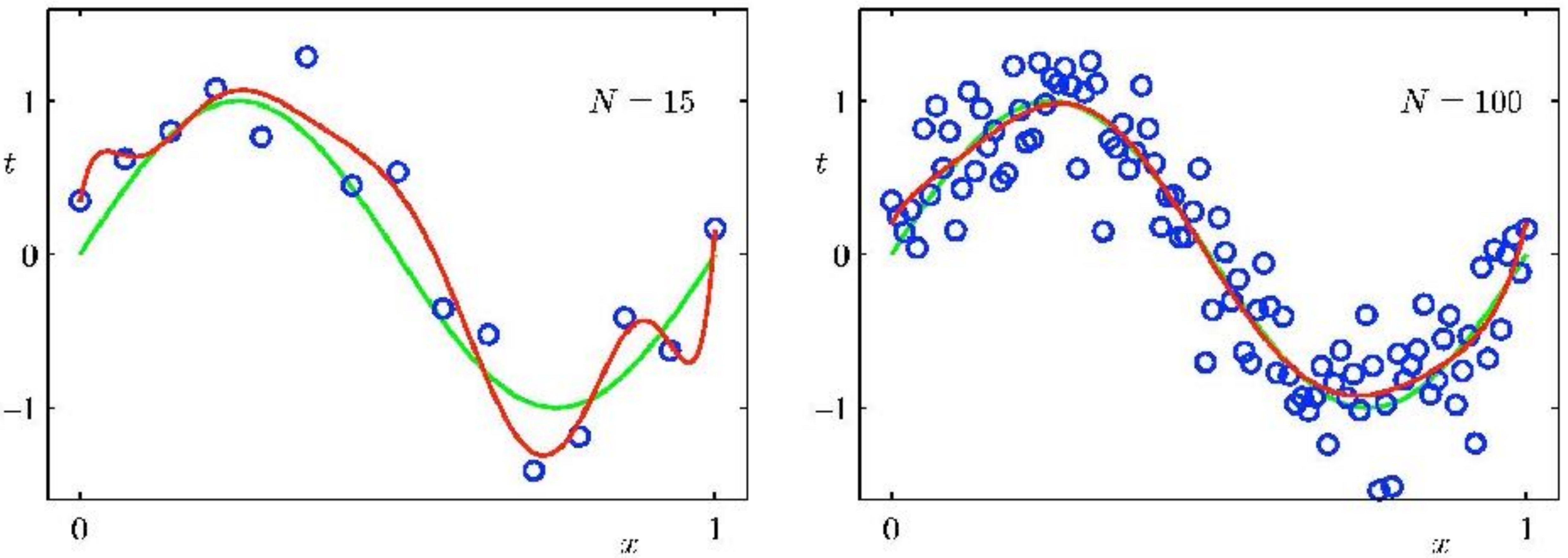
The Bias-Variance Trade Off

From these plots, we note that an over-regularised model (large λ) will have a high bias, while an under-regularised model (small λ) will have a high variance.



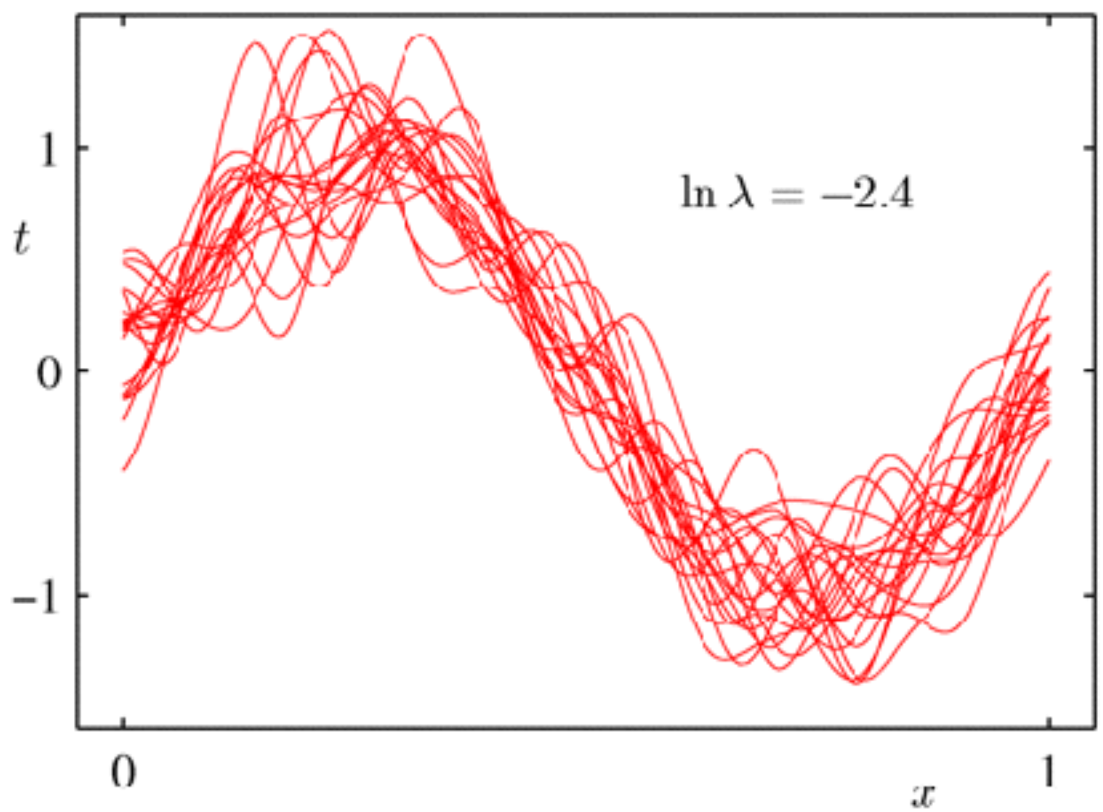
How to prevent overfitting?

Behaviour with dataset size :

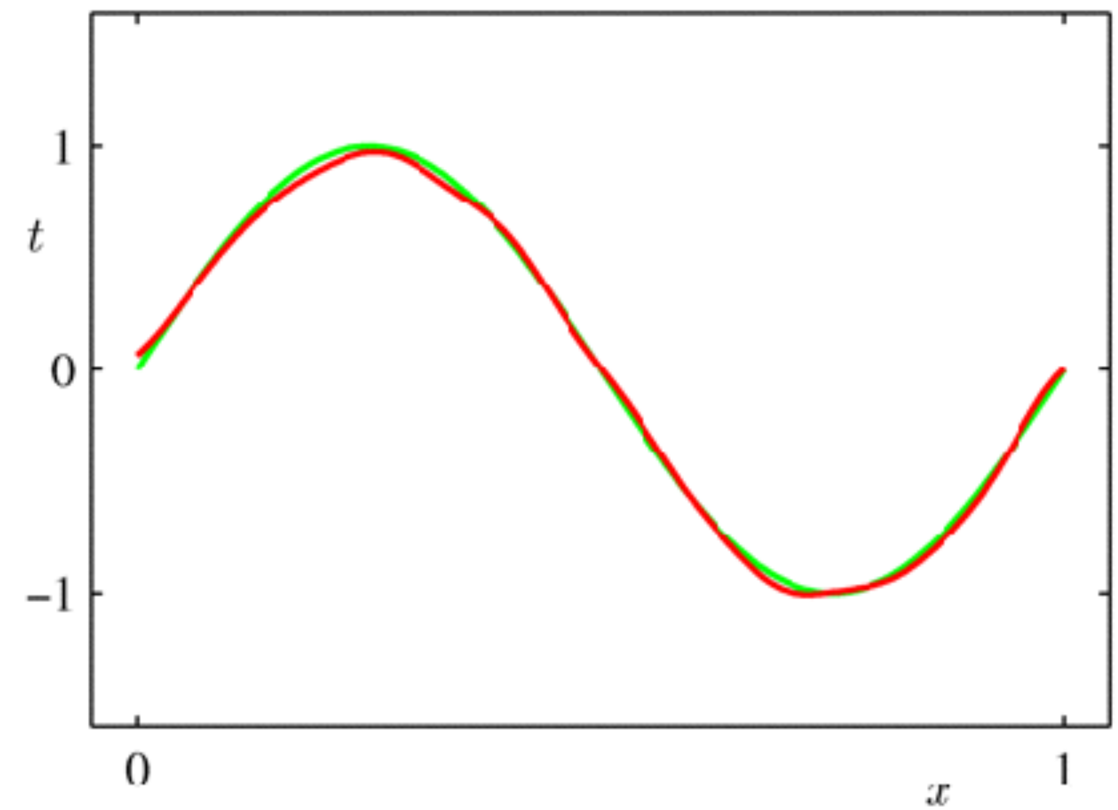


How to prevent overfitting?

20 datasets with varying regularisation parameter.



Result of fitting the model to each dataset.



Average of the fits.

How to prevent overfitting?

- Increasing training sample size.
- Controlling the complexity of hypothesis, e.g., by employing regularisation.

Estimation vs Inference

THE UNIVERSITY OF SYDNEY

- Learning as optimisation (frequentist): Given D, choose $\hat{f}$ to approximate f as closely as possible, so as to minimise (future) expected risk

- Usually compute parameter estimate $\hat{\theta}$

- Learning as inference (Bayesian): Given D, compute posterior over functions $p(f|D)$

- Or posterior over parameters

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$$

- Decision theory demonstrates that one of the best ways to minimise frequentist risk is to be Bayesian.