Machine Learning 1

- Lecture 4 -

Bayesian Linear Regression

- Patrick Forré -

Slides created by:
Rianne van den Berg

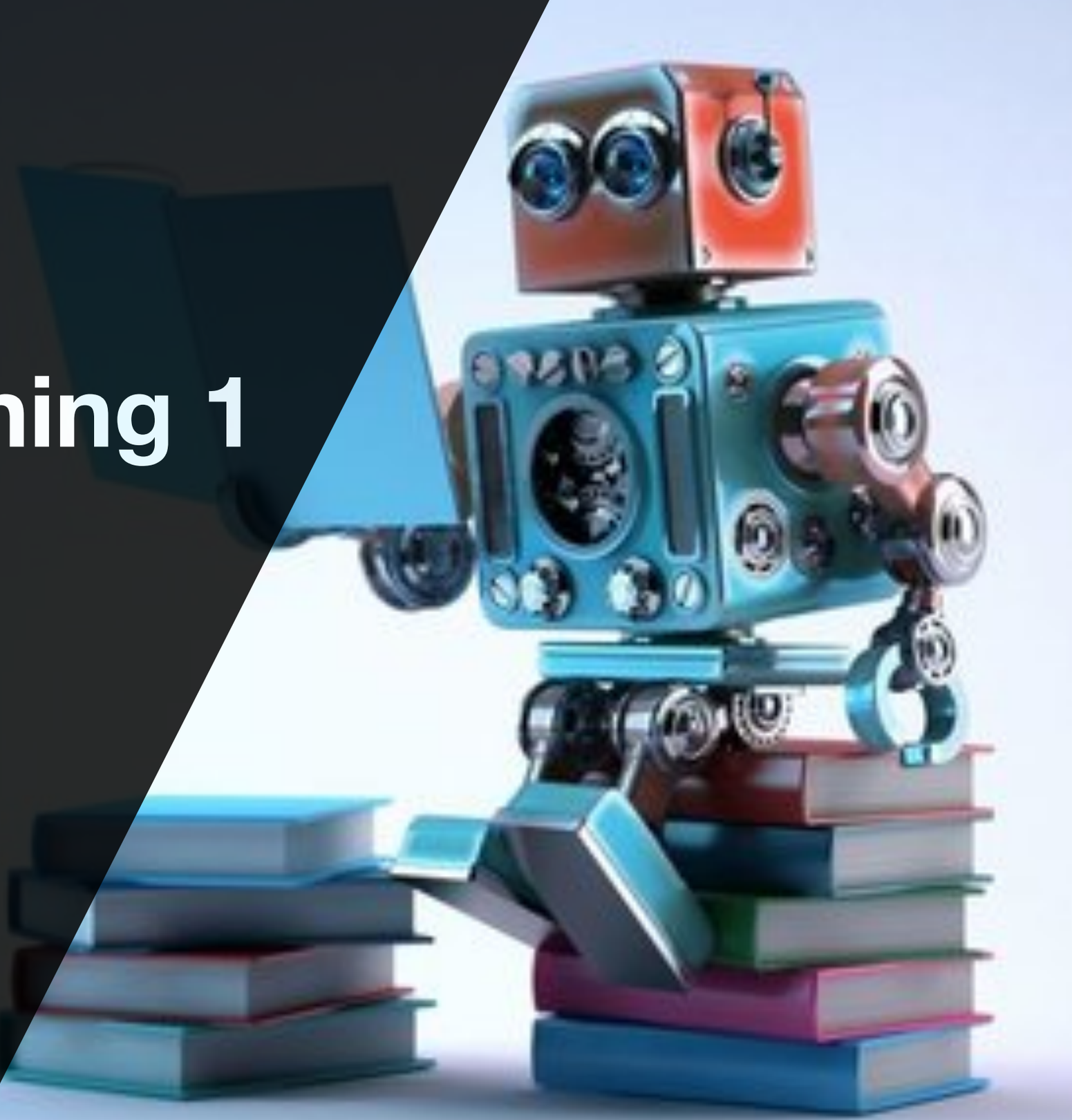


Image credit: Kirillm | Getty Images

Overview

1. **Recap: Linear Basis Model, Maximum Likelihood**

2. **Evaluating models**

3. **Regularized Least Squares, Maximum A Posteriori**

4. **Bayesian Linear Regression**

Overview

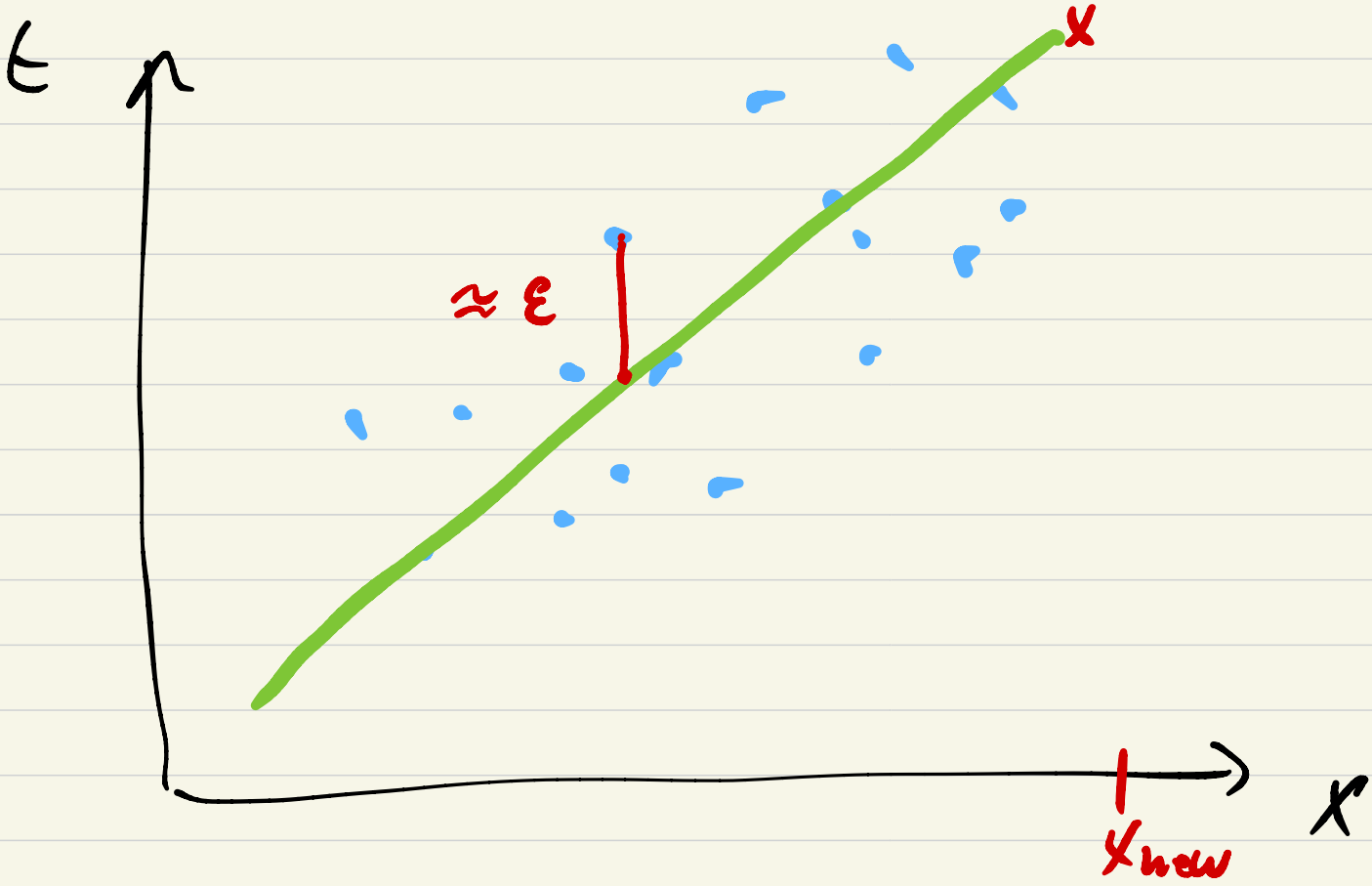
1. **Recap: Linear Basis Model, Maximum Likelihood**
2. Evaluating models
3. Regularized Least Squares, Maximum A Posteriori
4. Bayesian Linear Regression

Linear Basis Models

- Data set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with targets. $\mathbf{t} = (t_1, \ldots, t_N)^\top$
- Linear basis model: $t = y(\mathbf{x}, \mathbf{w}) + \varepsilon$
- $\mathbf{x}_n \in \mathbb{R}^D$ $t_n \in \mathbb{R}$
- $y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x})$
- $\mathbf{w} = \begin{pmatrix} w_0 \\ \vdots \\ w_{M-1} \end{pmatrix} \in \mathbb{R}^M$ $w_0 \to$ bias
- $\boldsymbol{\phi}(\mathbf{x}) = \begin{pmatrix} \phi_0 \\ \vdots \\ \phi_{M-1} \end{pmatrix}$ $\phi_0 = 1$
- $\varepsilon \sim \mathcal{N}(0, \beta^{-1})$

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

$N \times M$


ε
≈ ε
y
x
x_new

Maximum Likelihood Estimate

- Likelihood: $$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{i=1}^{N} \mathcal{N}(t_i|\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i), \beta^{-1})$$ (iid)
- Log-likelihood: $\log p(t|X, w, \beta) = \text{const} - \frac{\beta}{2} E_D(w)$
- $E_D(\mathbf{w}) = \sum_{n=1}^{N} (t_n - w^T \phi(x_n))^2$ sum-of-squares

- Maximum likelihood solution: $\mathbf{w}_{\mathrm{ML}} = \boldsymbol{\Phi}^{\dagger}\mathbf{t}$
- Moore-Penrose pseudo inverse: $\boldsymbol{\Phi}^{\dagger} = (\Phi^T\Phi)^{-1}\Phi^T$ if inverse exists.
- Predictions: $\mathbb{E}[t'|\mathbf{x}', \mathbf{w}_{\mathrm{ML}}] = w_{ML}^T \phi(x')$

Stochastic gradient descent

- for N >> 1 $\mathbf{w}_{\mathrm{ML}} = \left(\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\mathbf{t}$ is very costly to compute!

  $M \times M$

  - Needs to process all data $(\mathbf{x}_1, ..., \mathbf{x}_N)$ at once.

  - Matrix inversion of M x M matrix: $O(M^3)$

- Loss is a sum of error terms for each datapoint:

$$E_D(\mathbf{w}) = \sum_{i=1}^{N} E(\mathbf{x}_i, t_i, \mathbf{w})$$

$$E(\mathbf{x}_i, t_i, \mathbf{w}) = \frac{1}{2}\left(t_i - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i)\right)^2$$

- Approach for large dataset: stochastic gradient descent

Recap: Facts about the Gradient

- The gradient encodes all directional derivative via scalar product.

$$\nabla f \approx \left(\frac{\partial f}{\partial x_1} \; \dots \; \frac{\partial f}{\partial x_D} \right) \quad \hookrightarrow \boxed{\nabla f \cdot v}$$

$$v \in \mathbb{R}^D$$

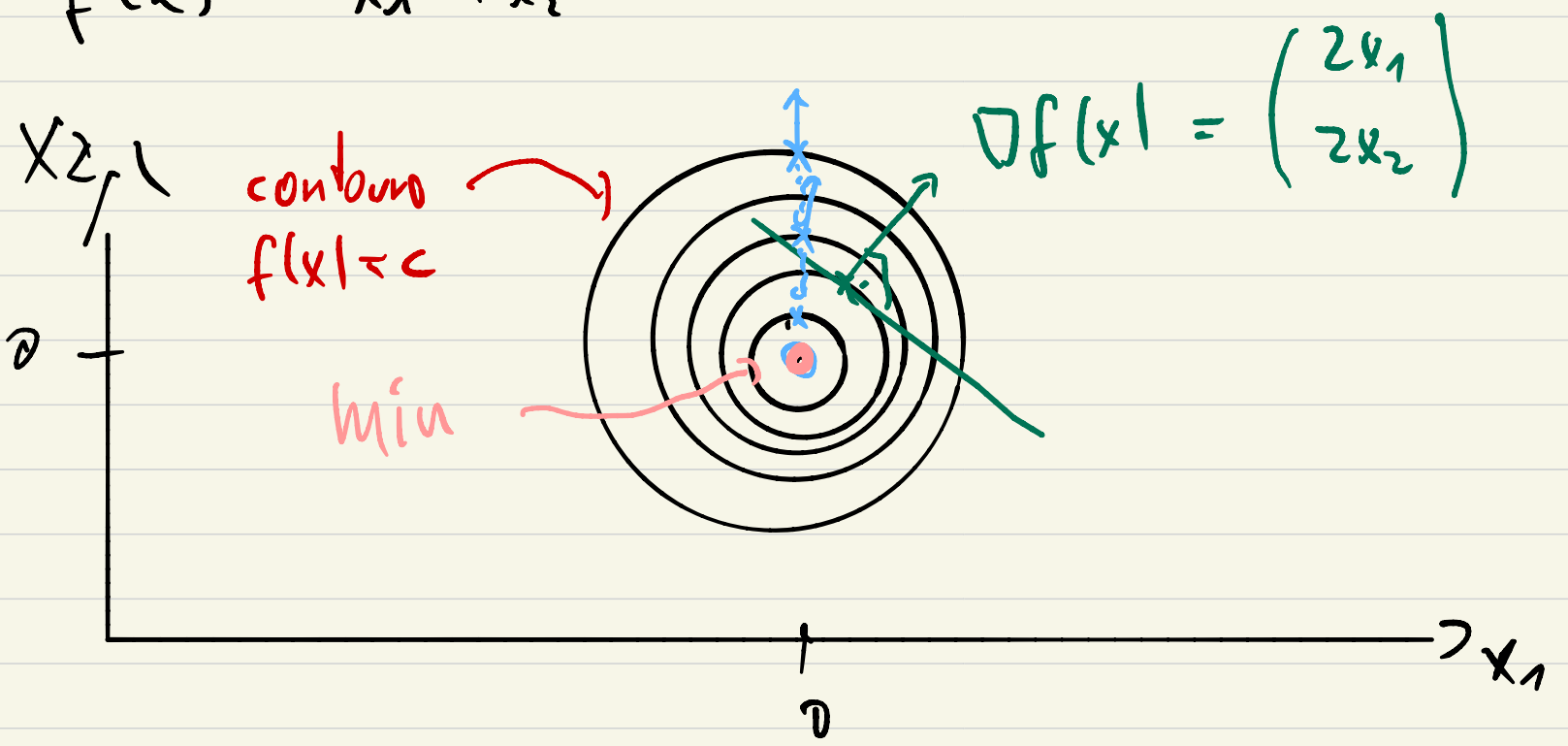
- The gradient is always perpendicular to the contours of a function.

$$x \in f^{-1}(c) = \{ x \in \mathbb{R}^D \mid f(x) = c \} \qquad c \in \mathbb{R}$$

$$\nabla f(x) \perp f^{-1}(c)$$

- The gradient always point in the direction of steepest ascent.

$$f(x) = x_1^2 + x_2^2$$


$\nabla f(x) = \begin{pmatrix} 2x_1 \\ 2x_2 \end{pmatrix}$
x_2
contours
$f(x) = c$
min
0
x_1
0

Stochastic gradient descent

$$E_D(\mathbf{w}) = \sum_{i=1}^{N} E(\mathbf{x}_i, t_i, \mathbf{w}) \qquad E(\mathbf{x}_i, t_i, \mathbf{w}) = \frac{1}{2}\left(t_i - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i)\right)^2$$

- Stochastic gradient descent:
  - Initialize $\mathbf{w}^{(0)}$, choose learning rate η .
    - Iterate over data points, and update

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta(\nabla_{\mathbf{w}} E(\mathbf{x}_i, t_i, \mathbf{w}^{(\tau)}))^T$$

$$= w^{(\tau)} - \eta \cdot (w^{(\tau)T}\phi(x_i) - t_i) \cdot \phi(x_i)$$

- If $E_D(\mathbf{w})$ is convex in $\mathbf{w}$ and η is small enough: convergence

↳ global minimum

Overview

1. Recap: Linear Basis Model, Maximum Likelihood

2. **Evaluating models**

3. Regularized Least Squares, Maximum A Posteriori

4. Bayesian Linear Regression

Supervised Learning: Evaluating Errors

Question 1:

How can we estimate the model performance properly for unknown data?

Question 2:

How can we choose the optimal hyperparameters?

Supervised Learning: Evaluating Errors

Gold Standard

Solution

Divide data $D = \{(\mathbf{x}_1, t_1), ..., (\mathbf{x}_N, t_N)\}$ in 3 groups:

- Training set D_{train} (± 60 % of D):
  - Minimize the error $E(y(\mathbf{x}_{\text{train}}, \mathbf{w}^*), t_{\text{train}})$ for $(\mathbf{x}, t) \in D_{\text{train}}$
- Validation set D_{val} (± 20 % of D):
  - Used to estimate test error $E(y(\mathbf{x}_{\text{val}}, \mathbf{w}^*), t_{\text{val}})$
- Test set D_{test} (± 20 % of D):
  - final test/generalization error estimate $E(y(\mathbf{x}_{\text{test}}, \mathbf{w}^*), t_{\text{test}})$

↳ w*

feedback

don't touch

↳ Reporting

Supervised Learning: Small Datasets

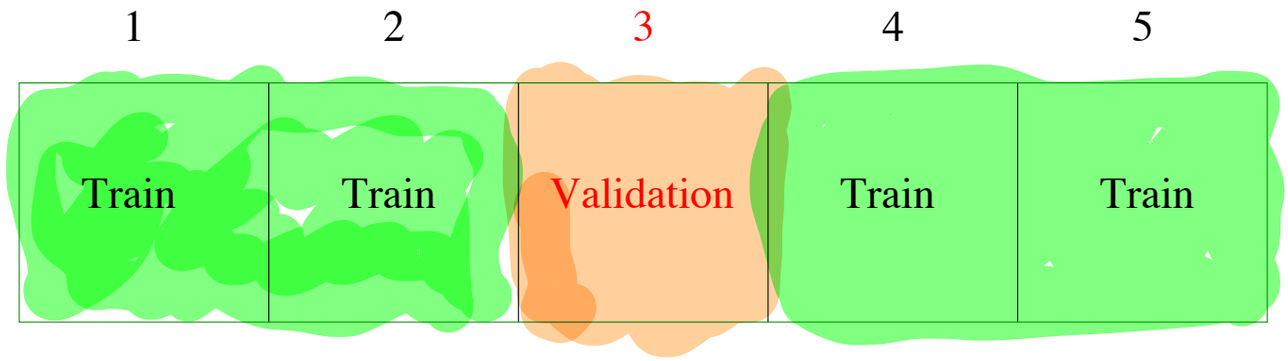
- Small dataset → small validation and test set
- Approximate validation step!

Cross-validation

- Split data: $D = \{(x_1, t_1), ..., (x_N, t_N)\}$

K bins

- Train *y* on *K-1* folds $\hat{y}^{-k}(x)$



K = S

Figure: K-fold splitting of dataset (ESL 7.10)

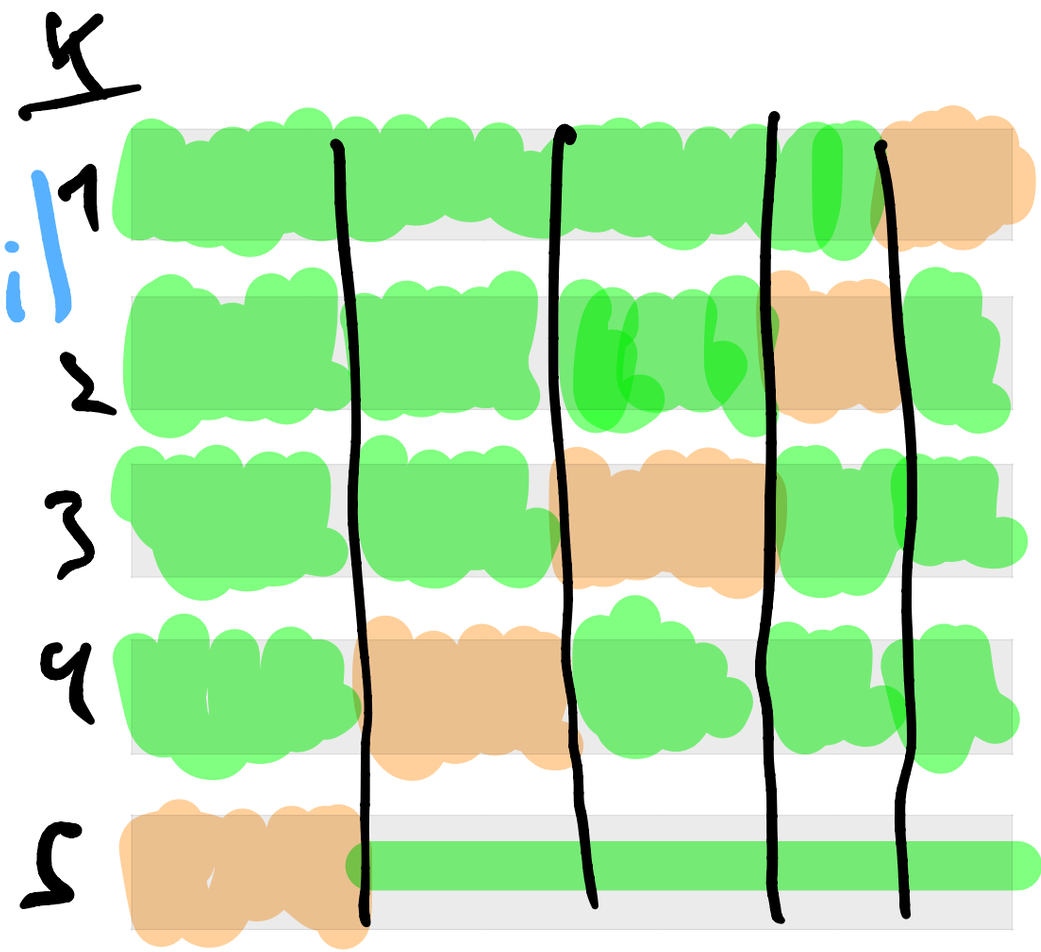
- $K = N$: leave-one-out cross validation

Cross-Validation

leave out bin k

- *K* trained functions $\hat{y}^{-k}(x)$
- Indexing function $\kappa : \{1, ..., N\} \mapsto$ $\{1, \ldots, K\}$
- Estimate of prediction error

$$CV(\hat{y}) = \frac{1}{K} \sum_{i=1}^{N} E\left(y^{-\kappa(i)}(x_i), t_i\right)$$



2 tasks

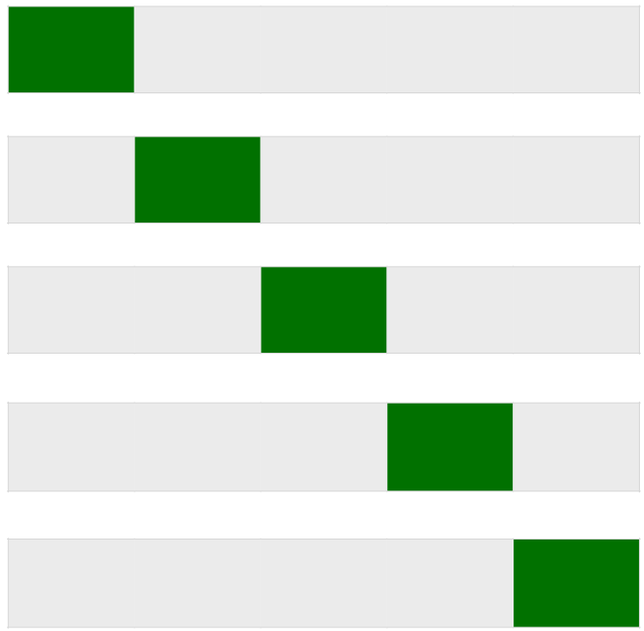
1. Estimation of generalization error
2. Model selection

Cross-Validation: Model Selection

- Hyperparameter selection $\alpha_1, \alpha_2 \dots$
- $CV(\hat{y}_\alpha) = \frac{1}{N} \sum_{n=1}^{N} E(\hat{y}_\alpha^{-\kappa(i)}(x_i), t_i)$
- Optimal $\alpha^* = \arg\min_\alpha CV(\hat{y}_\alpha)$

$\hat{y}_{\alpha,\beta}$

- Multiple hyperparameters: $\{\alpha_1, \alpha_2\} \quad \{\beta_1, \beta_2, \beta_3\}$



  - How many times should CV be performed?

    $2 \times 3 \quad (\alpha_i, \beta_j) \quad i = 1, 2, \quad j = 1, 2, 3$

  - Total number of training runs?

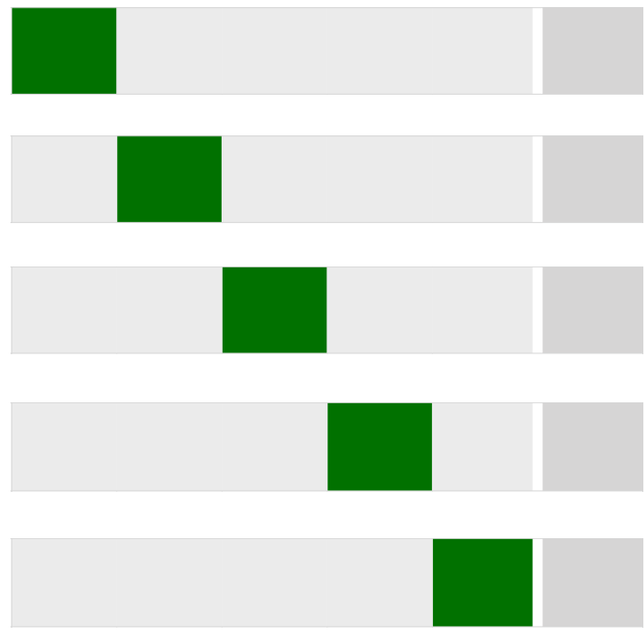
    $2 \times 3 \times K$

Cross-Validation: Test Error Estimation

- After Model selection

$$\alpha^*, \beta^*$$

- Retrain *f* on all K folds with α^*, β^*



- Evaluate model on held-out test set

- Nested cross validation!

Nested Cross-Validation

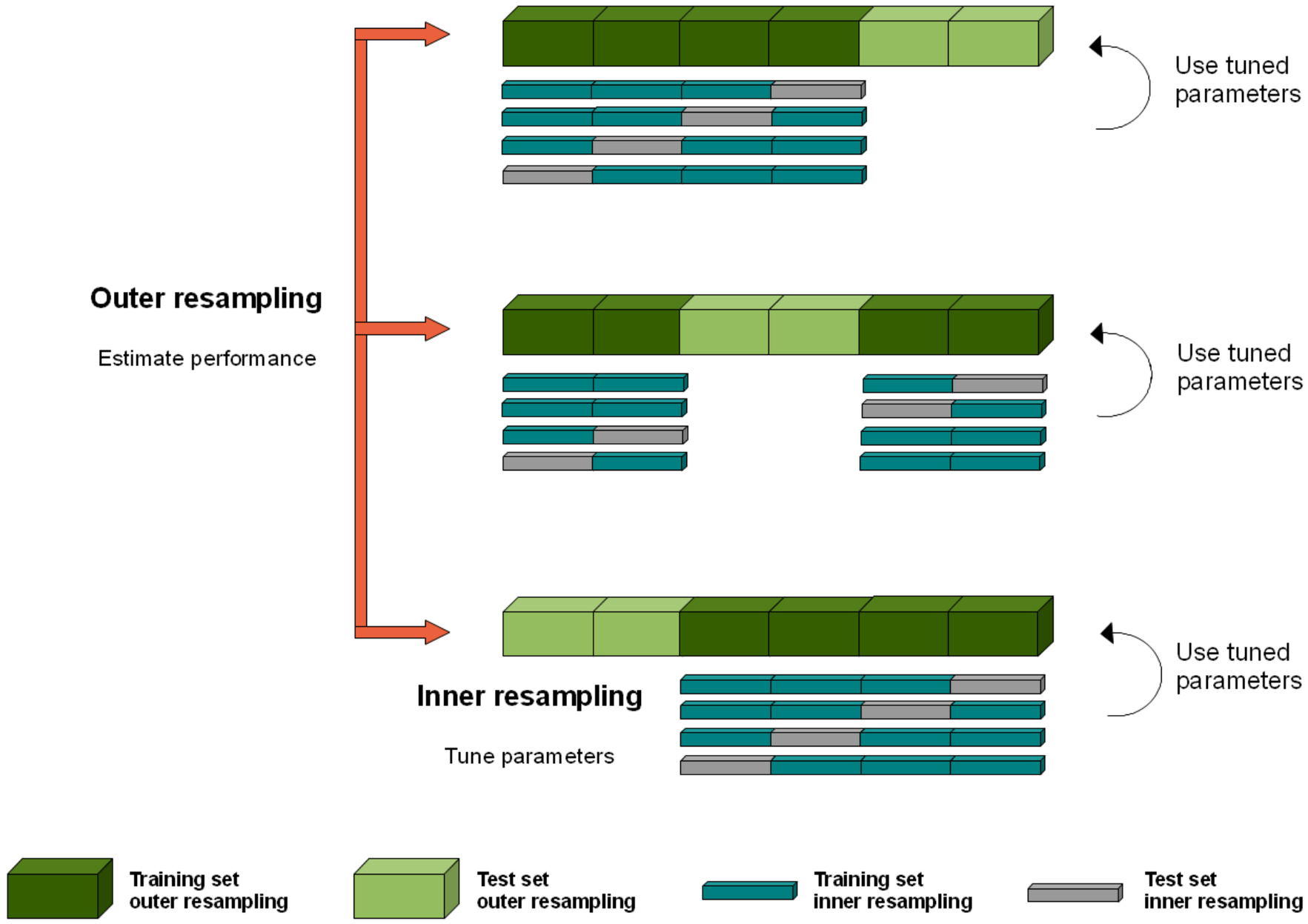


Figure: Nested cross-validation
https://mlr-org.github.io/mlr-tutorial/devel/html/nested_resampling/index.html

How to choose K in K-fold CV?

1.) Old books recommend

$K \simeq 5 - 10$ (data independent)

2.) New analysis (P. Orbanz):

$K \sim \sqrt[4]{N}$

↑ # data points

Overview

1. Recap: Linear Basis Model, Maximum Likelihood

2. Evaluating models

3. **Regularized Least Squares, Maximum A Posteriori**

4. Bayesian Linear Regression

Example: Overfitting and Underfitting

$$t = \sin(2\pi x) + \varepsilon \qquad \varepsilon \sim \mathcal{N}(0, \beta^{-1})$$

$$y(x, \mathbf{w}) = w_0 + \sum_{i=1}^{M} w_i x^i$$

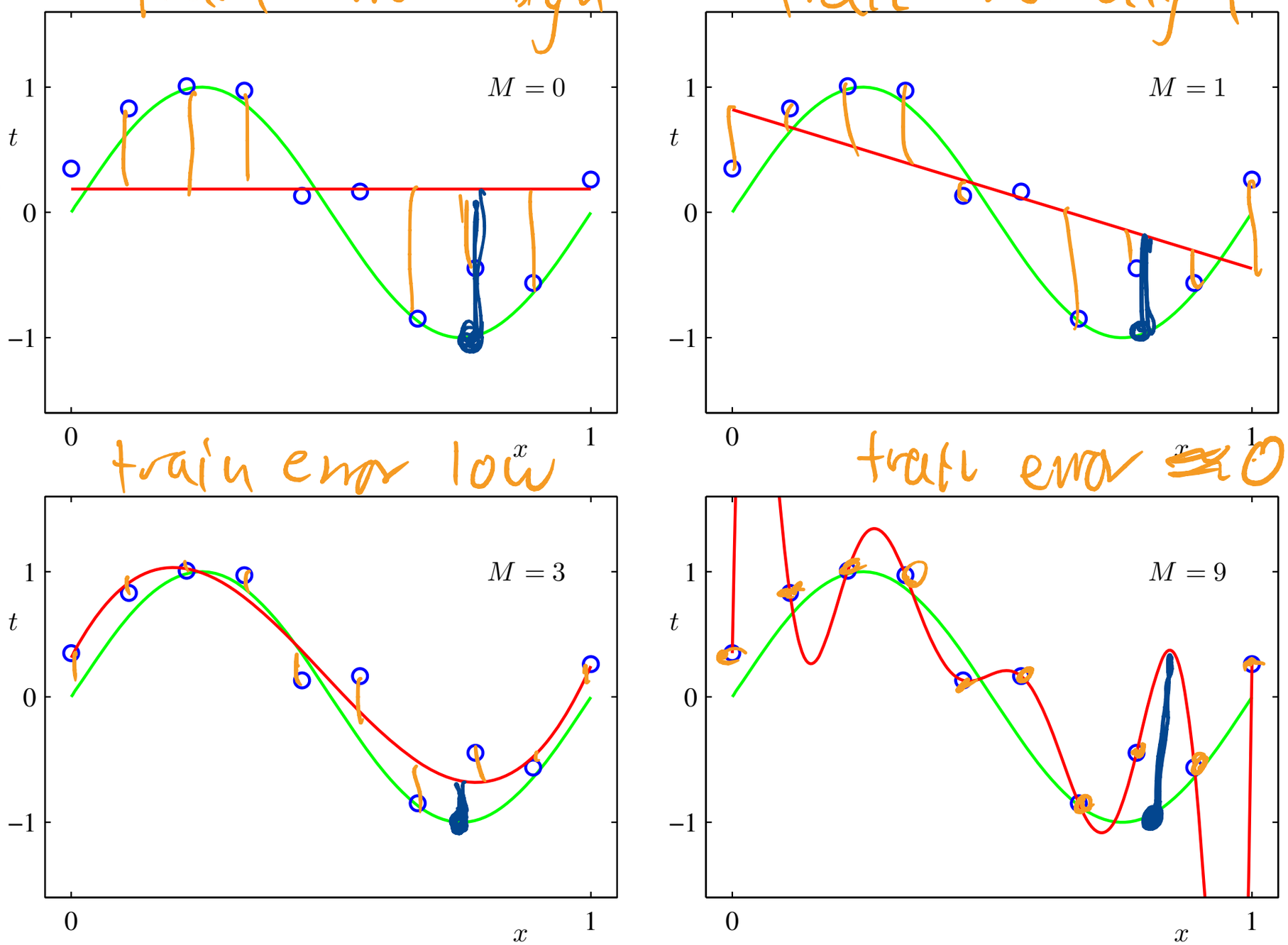
train error high

test error high

train error

test/val error

under-fitting



train error low

train error ≈ 0

over-fitting

Figure: Fits of different polynomials (Bishop 1.4)

Example: Overfitting and Underfitting

big 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

overfitting

Table: Polynomial coefficients (Bishop 1.1)

In linear model parameter $w = \begin{pmatrix} w_0 \\ \vdots \\ w_{n-1} \end{pmatrix}$ plays two roles:

1.) → parameters

2.) $\nabla_x y(x, w) = w^T$ ~) gradient wrt input x

$y(x, w) = w^T x$

Example: Overfitting and Underfitting

$$E_{\mathrm{RMS}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\{t_i - y(x, \mathbf{w})\}^2}$$

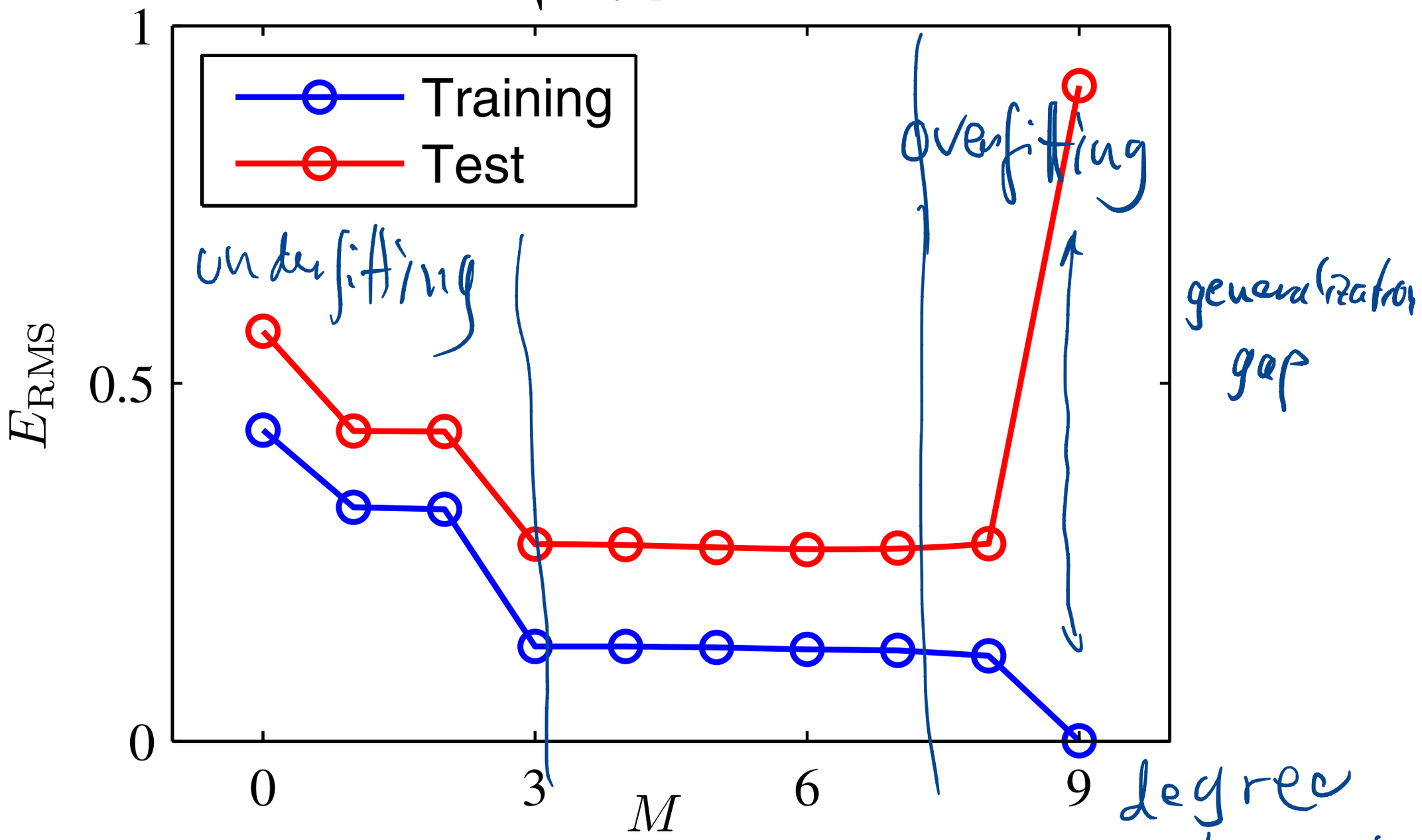
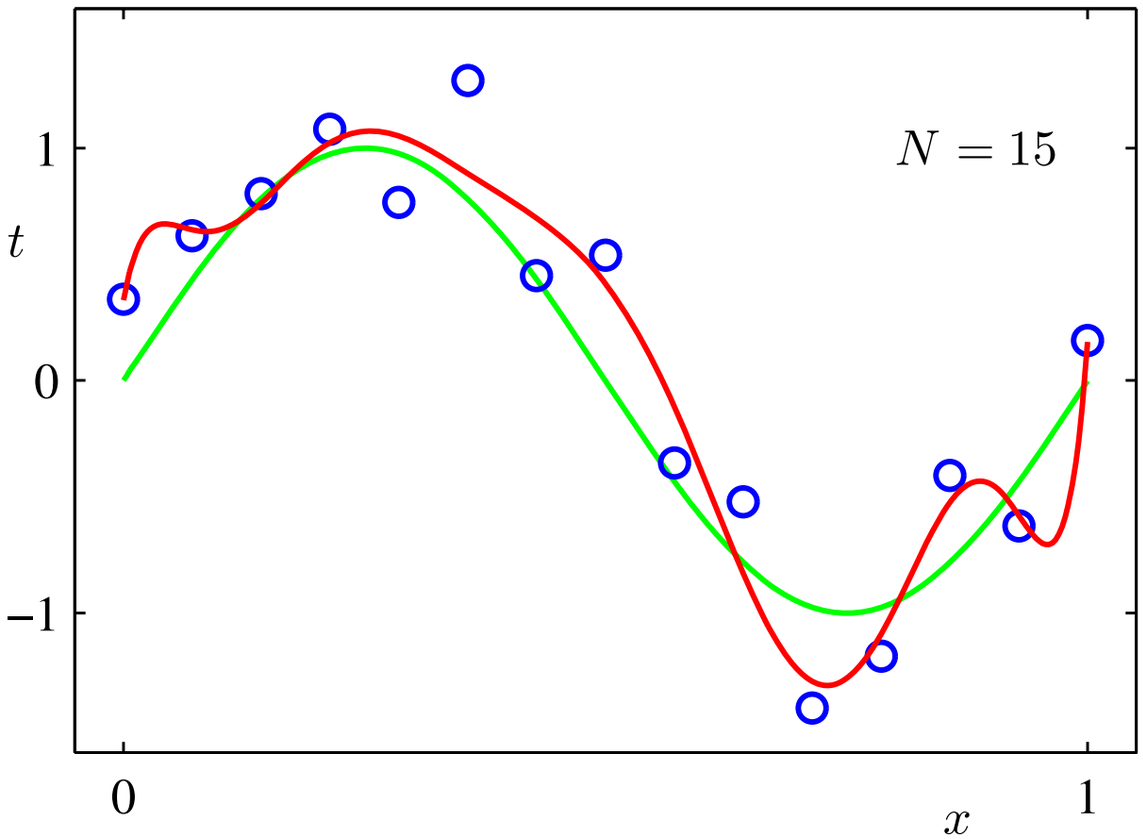
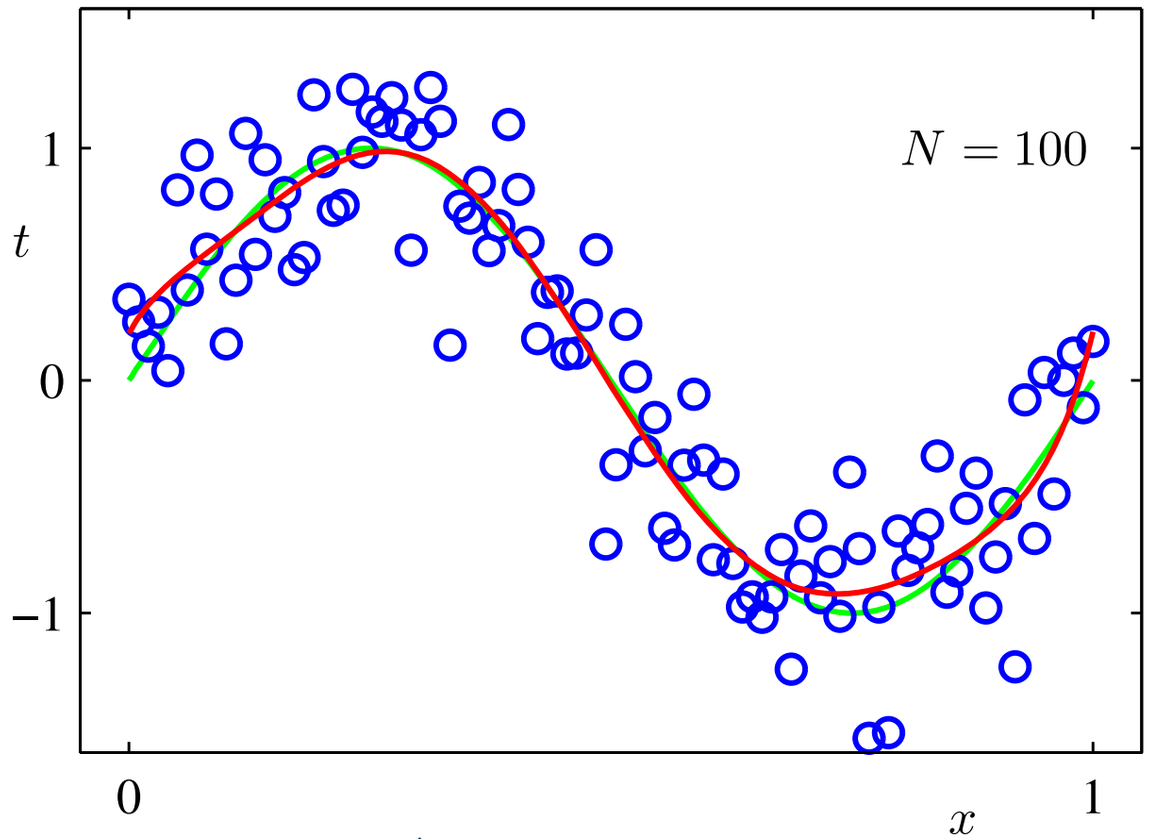


Figure: training and test RMSE (Bishop 1.5)

Example: Overfitting

more datapoints less overfitting





N >> M

Figure: *M*=9 Polynomial fit with increased datapoints N. (Bishop 1.6)

Regularized Least Squares

- Instead of manually constraining the number of parameters for small datasets, add penalty term for large parameter values:

$$\tilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\{t_i - y(\mathbf{x}_i, \mathbf{w})\}^2 + \frac{\lambda}{2}\sum_{m=1}^{M-1} |w_m|^2$$

$m = 0, 1$

- The bias term w_0 is not always included in regularization

$$\arg\min_{w} \tilde{E}(w)$$

Maximum A Posteriori (MAP)

$$\tilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\{t_i - y(\mathbf{x}_i, \mathbf{w})\}^2 + \lambda \mathbf{w}^T\mathbf{w}$$

- Note: equivalent to Maximum A Posteriori (MAP) approach for estimating **w** with Gaussian prior:

likelihood

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

prior

posterior

evidence

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

$$\mathbf{w}_{\mathrm{MAP}} = \arg\min_{\mathbf{w}} -\log p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \alpha) = \arg\min_{\mathbf{w}} -\log p(\mathbf{t}|\mathbf{X}, \mathbf{w}) - \log p(\mathbf{w}|\alpha)$$

$$= \arg\min_{w} \frac{\beta}{2}\sum_{n=1}^{N}(t_i - y(x_i; w))^2 + \frac{\alpha}{2} w^T w + \text{const.}$$

$$= \arg\min_{w} \tilde{E}(w) \quad \text{with} \quad \lambda := \frac{\alpha}{\beta}$$

$$w_{MAP} = \underset{w}{\operatorname{argmin}} \underbrace{\frac{1}{2}\|t - \Phi w\|_2^2 + \frac{\lambda}{2}\|w\|_2^2}_{\tilde{E}_D(w)}$$

$$\frac{\partial}{\partial w}\tilde{E}_D(w) \overset{!}{=} 0$$

$$\frac{\partial}{\partial w}\tilde{E}_D(w) = -\Phi^T(t - \Phi w) + \lambda \cdot w$$

$$\Rightarrow w_{MAP} = \left(\Phi^T\Phi + \lambda I_M\right)^{-1}\Phi^T t$$

deviation from w_{ML}

Example: Regularized Polynomial Regression

overfitting ↓ under-regularization

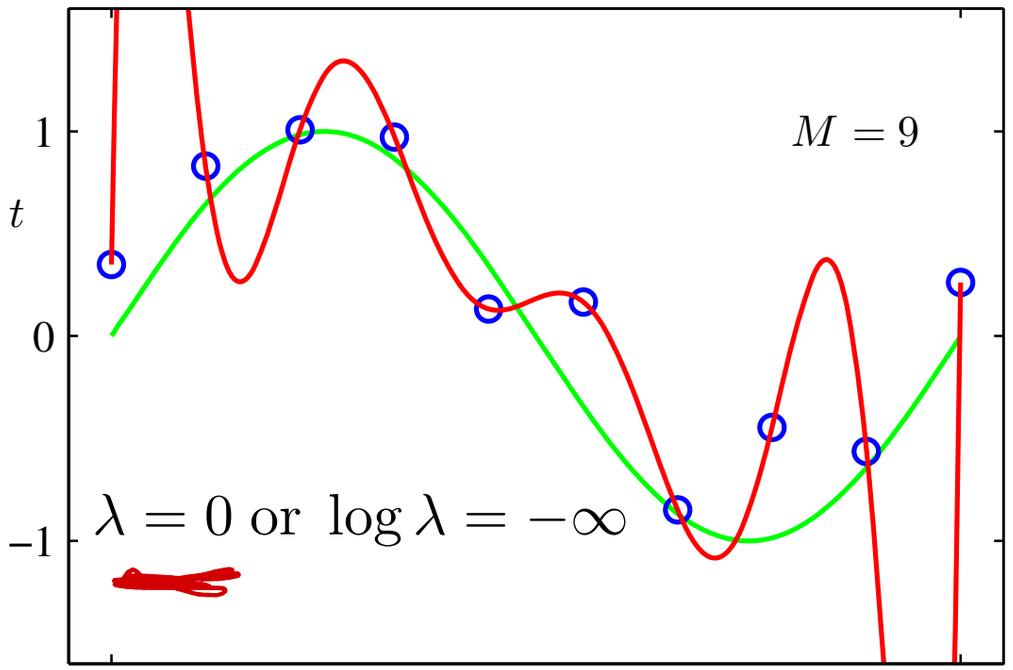


Figure: polynomial regression (Bishop 1.4)

good fit

underfitting over-regularized ↑

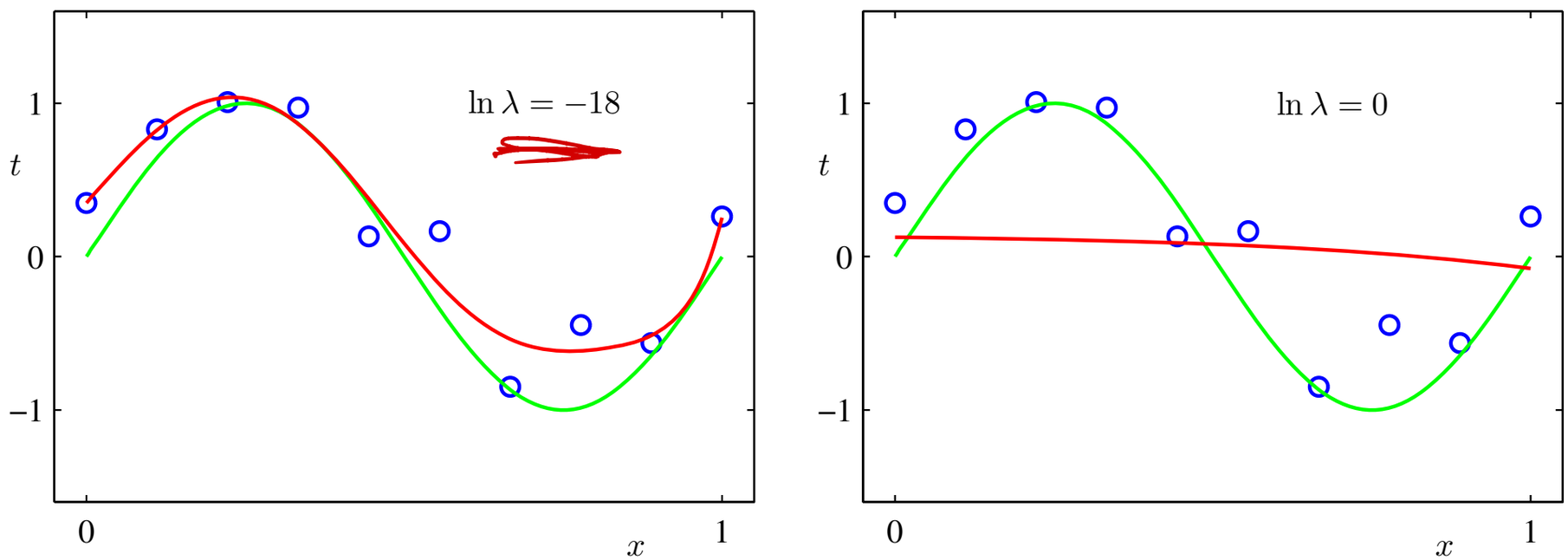


Figure: Regularized polynomial regression (Bishop 1.7)

Example: Regularized Polynomial Regression

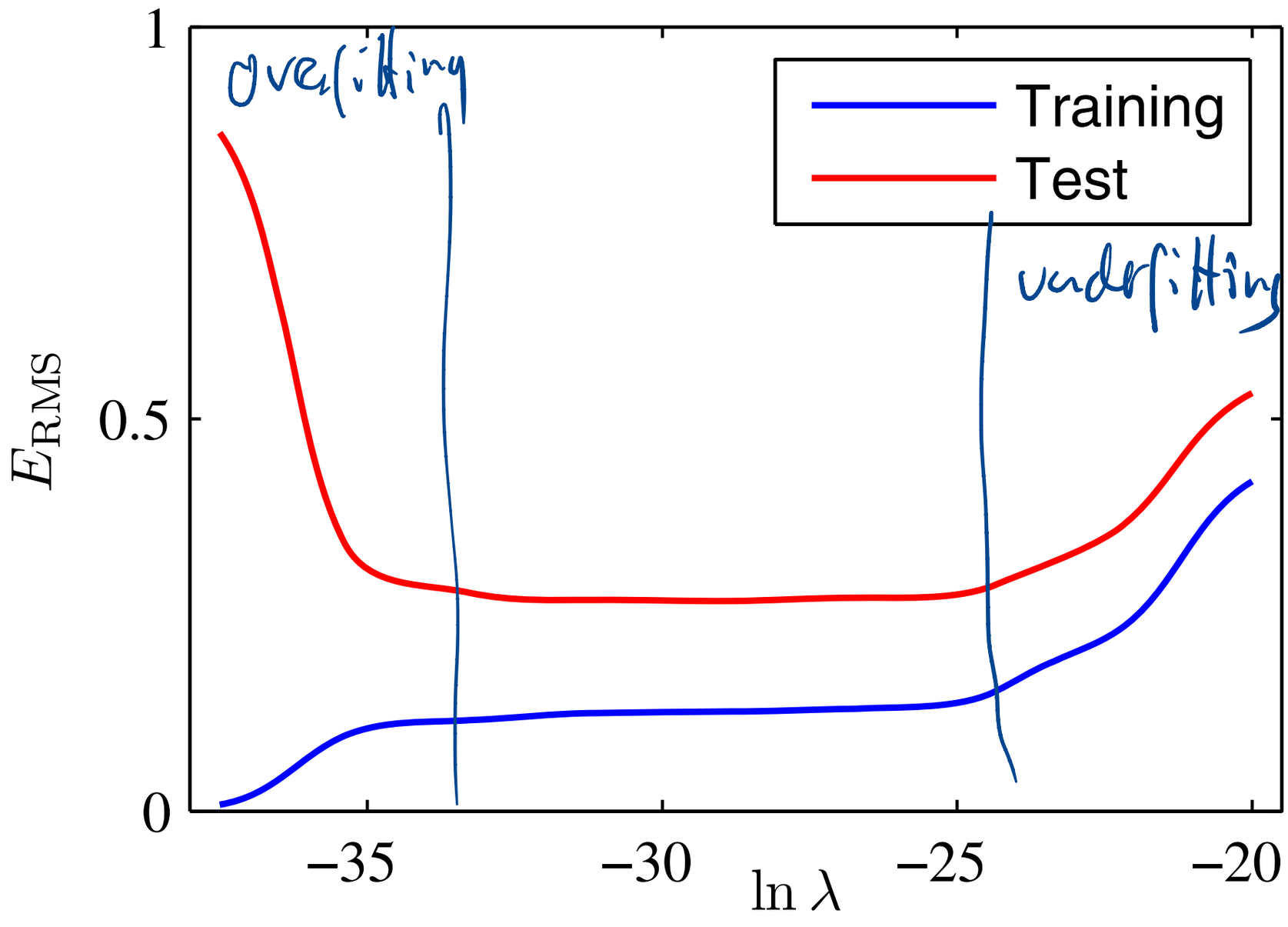


Figure: train and test errors for regularized M=9 polynomial regression (Bishop 1.8)

Regularized Least Squares (II)

- Weight decay : $\tilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\{t_i - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i)\}^2 + \frac{\lambda}{2}\sum_{i=1}^{M}|w_i|^2$

- More general :

$$\tilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\{t_i - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i)\}^2 + \frac{\lambda}{2}\sum_{i=1}^{M}|w_i|^q$$

- $q = 1$: Lasso
  - → sparse model: many $w_i = 0$
- Equivalent to minimizing

$$\frac{1}{2}\sum_{i=1}^{N}\{t_i - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i)\}^2 \quad \text{with} \quad \sum_{j=1}^{M}|w_j|^q \leq \eta$$

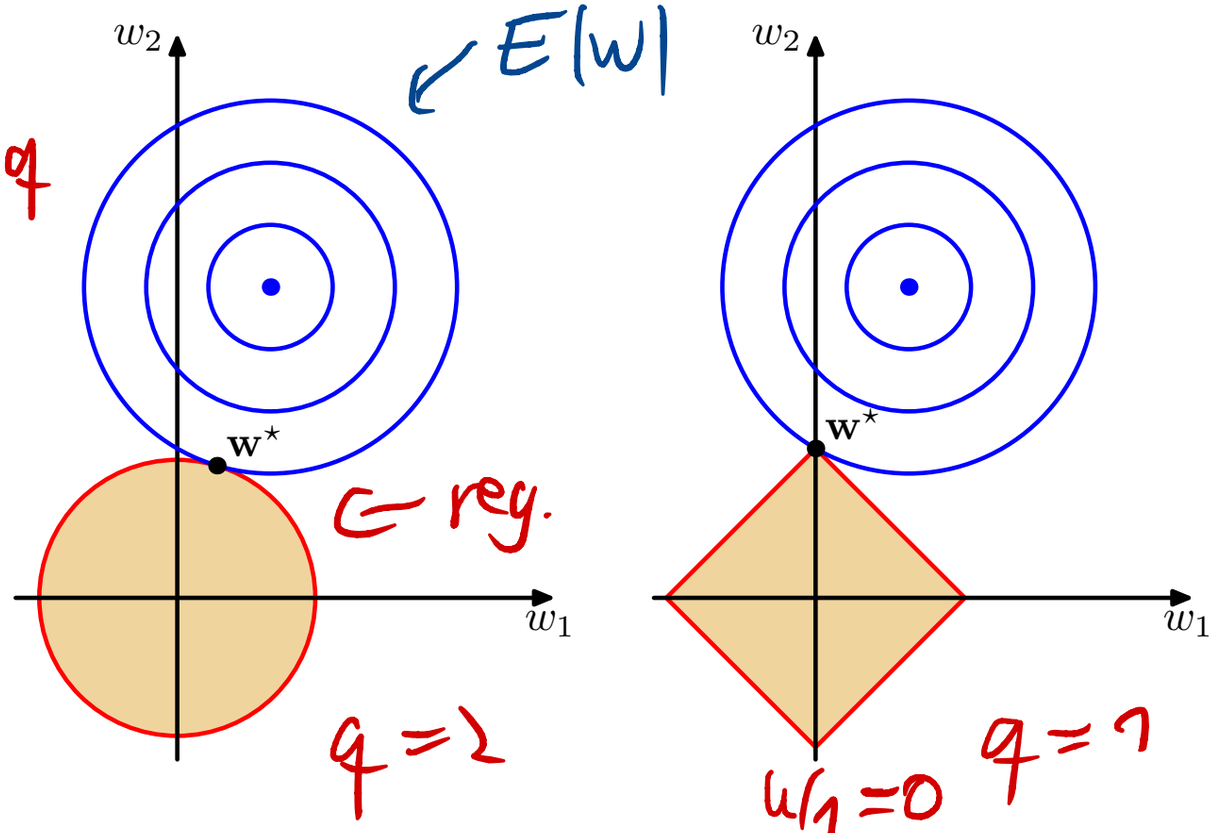


Figure: regularization as constrained optimization (Bishop 3.4)

Regularized Least Squares: sparse weights

$$\frac{1}{2}\sum_{i=1}^{N}\{t_i - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i)\}^2 \quad \text{with} \quad \sum_{j=1}^{M}|w_j|^q \leq \eta$$

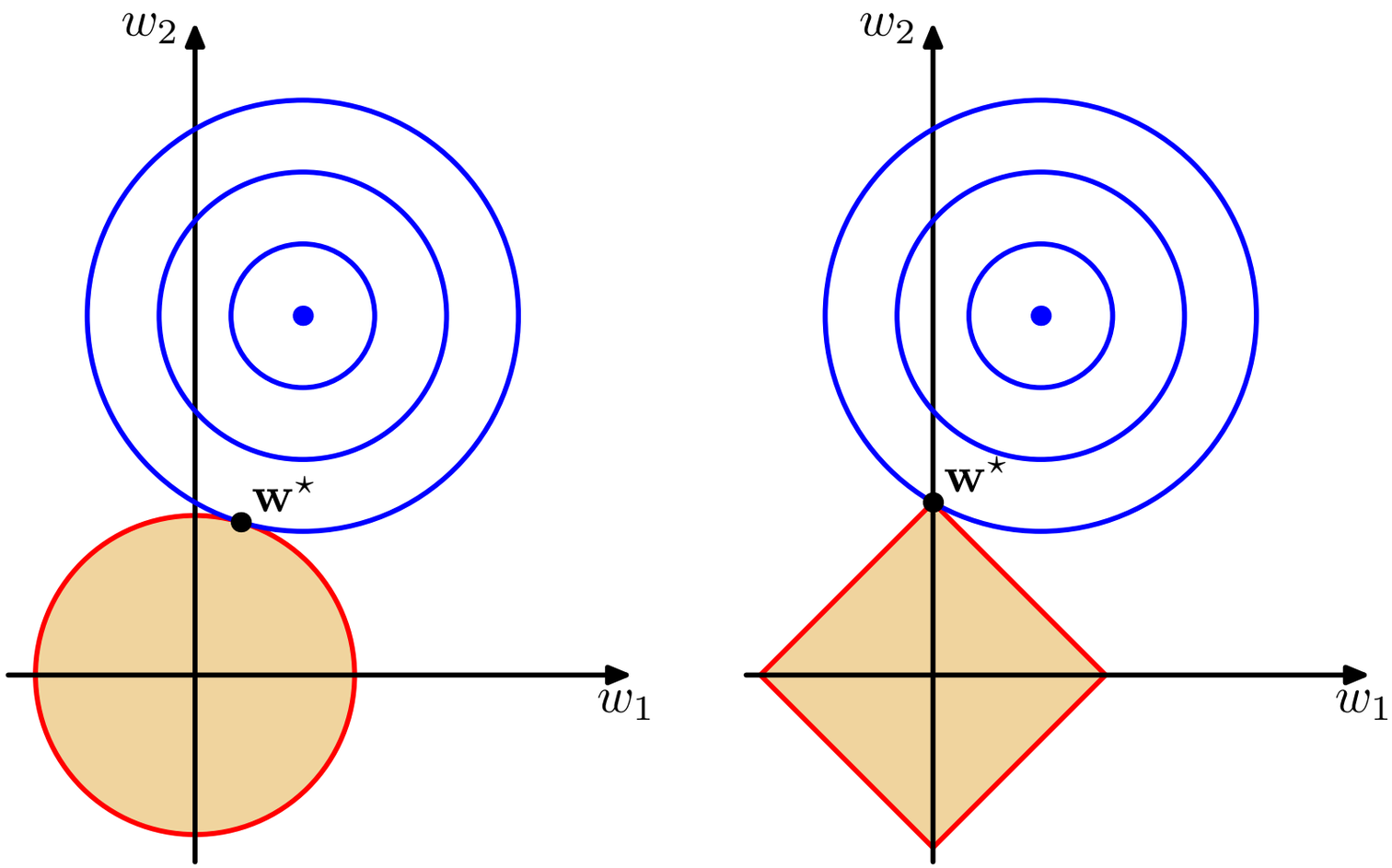


Figure: regularization as constrained optimization (Bishop 3.4)