Grundlegende Begriffe am Bsp. Regression

Lernziele:

- Grundbegriffe überwachtes Lernen (Eingabe, Ausgabe, Modell, quadratischer Fehler)
- Grundbegriffe: Modellselektion vs. Parameteroptimierung, Overfitting, Regularisierung

(Mathem.) Voraussetzung:

- Polynome

Vorgehen:

- “triviales” Funktionsbeispiel: sin( )
- Ansatz: Summe von Polynomen
- nach: Bishop, Kap 1.

Notation

Trainigsdaten
Inputdaten
Beispiele
Inputs
Beobachtungen
Observations

$$X = \{\vec{x}_1, ..., \vec{x}_N\}$$

Ausgaben
Sollwerte
Targets
Outputs

$$Y = \{\vec{y}_1, ..., \vec{y}_N\}$$

oder

$$T = \{\vec{t}_1, ..., \vec{t}_N\}$$

N: Anzahl Trainingsdatenpunkte
(jeder Punkt ist ein Vektor der Dimension D)

Notation

Interpretation teilweise kontextabhängig:

$$x_i = i-th \text{ component}$$

$$\vec{x}_i = i-th \text{ vector}$$

$$t_n = n-th \text{ component}$$

$$\textbf{but if } dim = 1 \text{ then}$$

$$t_n = n-th \text{ vector}$$

(sollte aus dem Kontext klar sein)

Überwachtes Lernen *(supervised learning)*

Form des induktiven Lernens:

- gegeben eine Anzahl Trainingsdaten (Eingaben und Sollausgaben)
  - Zwischenziel: finde Modell für die Daten
  - Ziel: Generalisierung
    - d.h.: wende das Modell auf Daten an,
      die nicht in den Trainingsdaten enthalten sind
      - wird auch “Exploitation” genannt

- verschiedene Möglichkeiten für “Exploitation”
  - deterministisch: berechne neue Ausgabe für neue Eingabe
  - probabilistisch: erzeuge zufällige Ausgabe für neue Eingabe
    mit gelernter Wahrscheinlichkeitsverteilung (predictive distribution)

Überwachtes Lernen *(supervised learning)*

Inductive Learning Hypothesis:

- Daten sind charakteristisch für generellere Regel/Funktion
- ist immer eine implizite Annahme

Inductive Bias:*

- bezeichnet die implizite Annahmen über die Art der Generalisierung
- ist immer notwendig

* wesentliches Lernziel der Vorlesung: Verstehen, was dieser Bias bedeutet und wo welche Annahmen gemacht werden

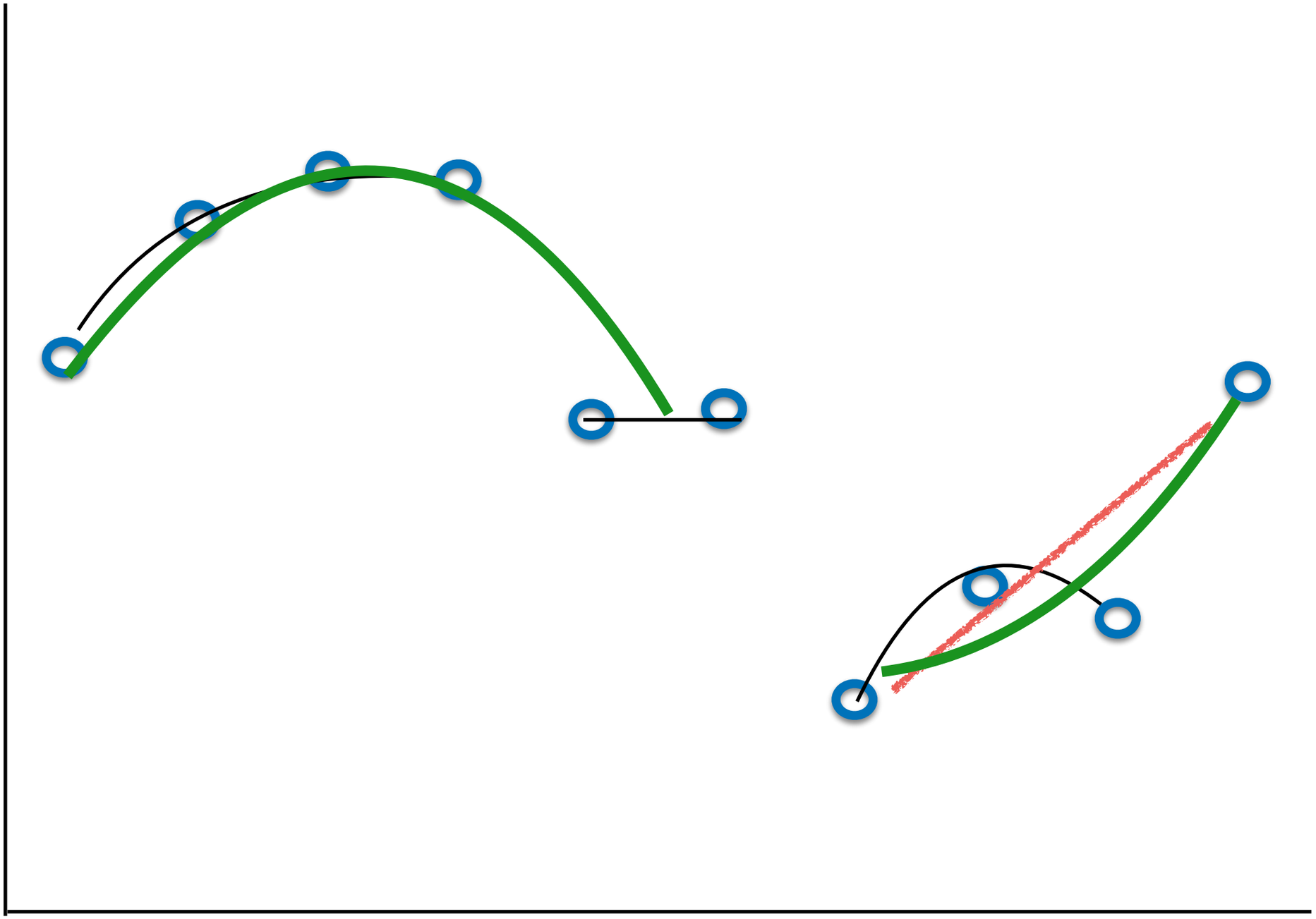
Modellselektion vs Parameteroptimierung

gegeben Trainingsdaten, zwei Aufgaben

- **wähle eine parametrisierte Form des (Daten-)Modells (= *model selection*)**
- **optimiere (lerne) die Parameter (= learning, parameter optimisation)**
- **Modell wird hier ganz generell verwendet im Sinne eines beliebigen parametrisierten Ansatzes***

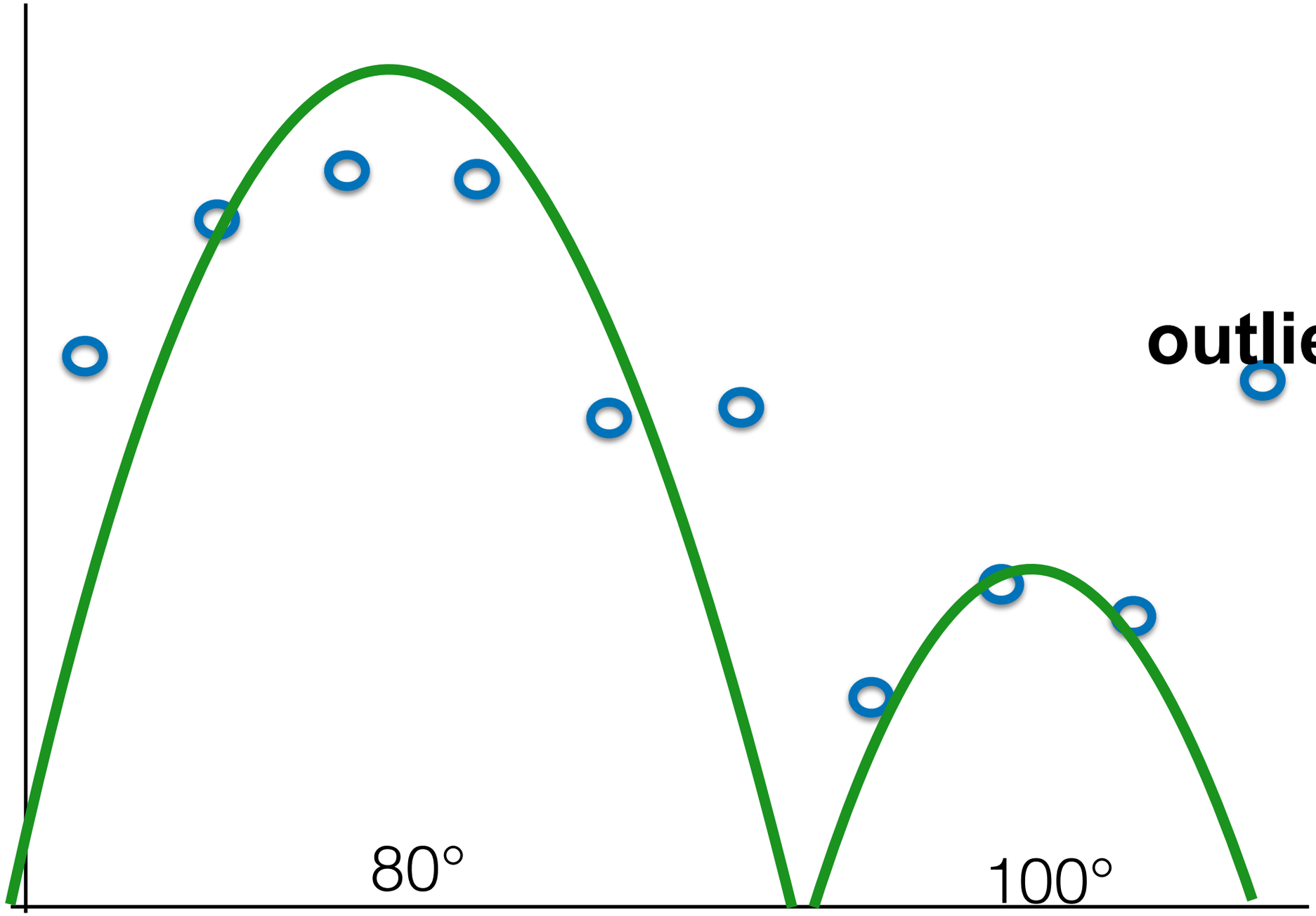
* Bemerkung: der Begriff “Modell” ist oft mißverständlich. Häufig werden Lernverfahren auch als “model-free” bezeichnet. Dies ist dann gemeint im Gegensatz zu einem analytischen oder physikalischem Modell eines bekannten Prozesses (z.B. Bewegungsgleichungen für einen Roboter), während “model-free” meint: kein physikalisches Modell. Trotzdem wird dann immer noch ein Datenmodell verwendet, welches aber keine Information über den zugrundeliegenden Prozess trägt.

Überwachtes Lernen: Beispiel Lernbias



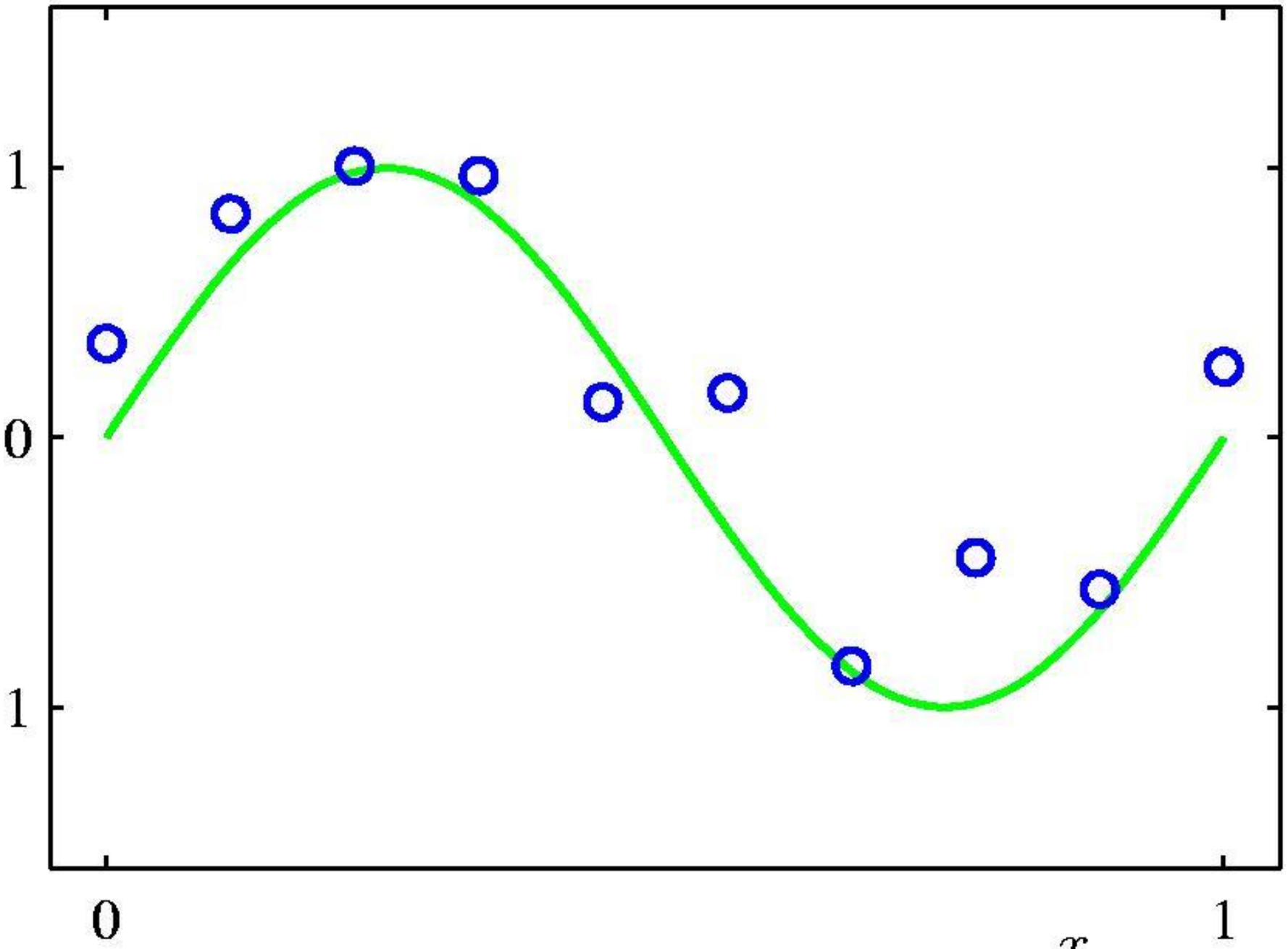
Welcher Bias ? Typisch: “Glattheit” und lokale Nähe (beim Mensch: die Gestaltgesetze)

Überwachtes Lernen: Beispiel Lernbias



Anderer Bias: Bivariate Gaussverteilungen (z.B. da Messwerte um zwei Meßpunkte)

Überwachtes Lernen: Beispiel Lernbias



“true function”: $t_n = sin(2\pi x_n) + \nu$

“sinus + noise” $\nu \sim N(0, \sigma^2)$ Gaussian distributed

Überwachtes Lernen: Beispiel Modellselektion

wähle als Modell Polynom vom Grad M:

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

formula from Bishop, machine learning

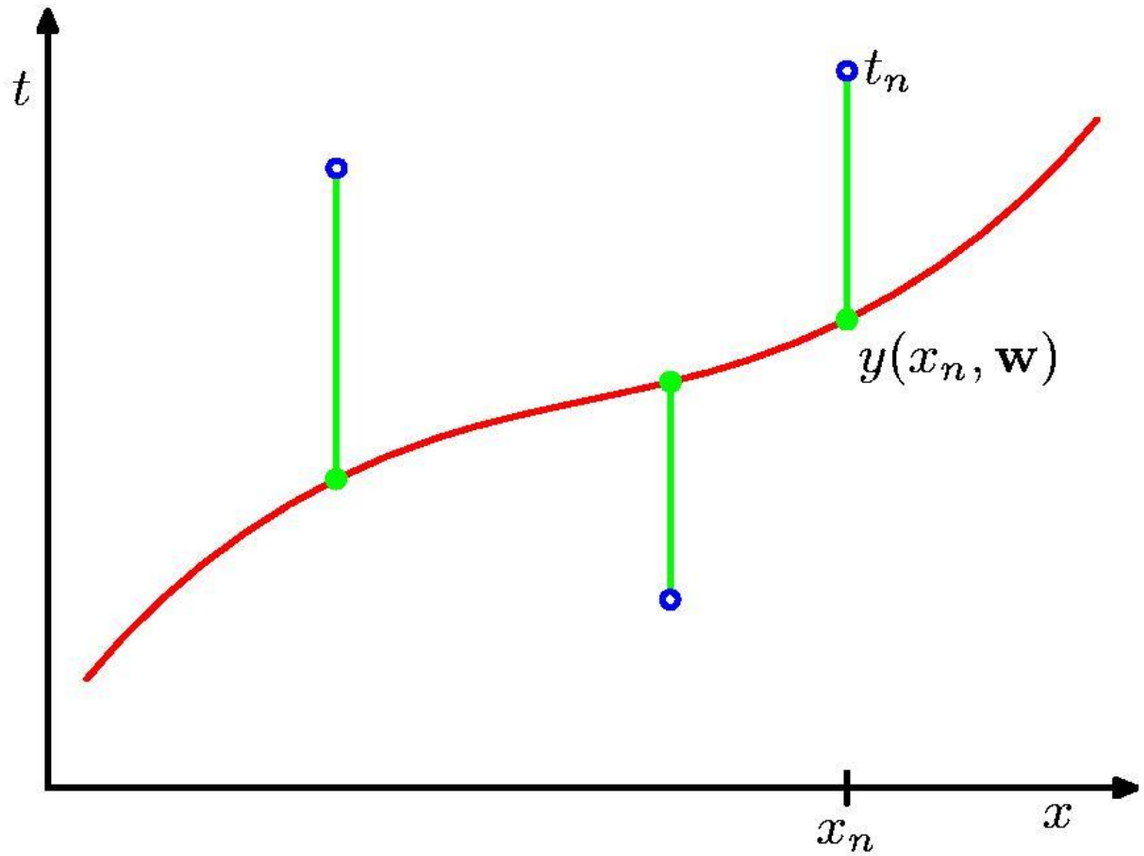
die Wahl hat 2 wesentliche Annahmen

- Form des Modells: Polynom
- Grad des Polynoms: bestimmt die Komplexität des Modells
- Grad des Polynoms: bestimmt die Anzahl der Parameter (=M)

- beide Auswahlen zusammen bestimmen die Modellkomplexität

Beispiel: Parameteroptimierung

Summe der quadratischen Fehler als Fehlerfunktion



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

Slide from Bishop, machine learning

Beispiel: Parameteroptimierung

die Wahl hat 2 wesentliche Annahmen

- quadratische Fehlerfunktion

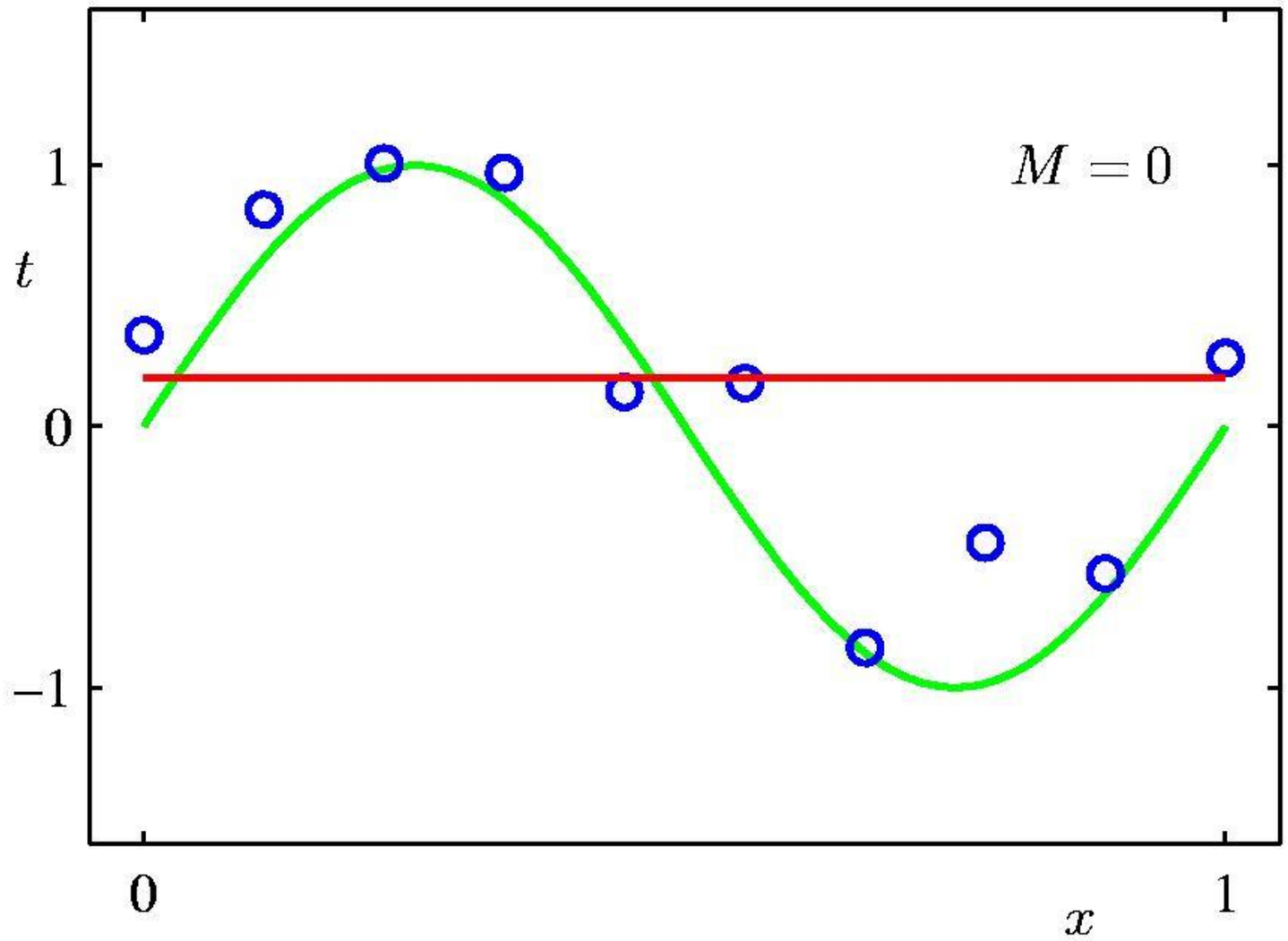
$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2$$

- Ansatz “gute” Parameter für das Modell zu finden:
  - Betrachte E(w) als Funktion der Parameter w
  - Minimiere den quadratischen Fehler bzgl. w
  - wir erhalten durch Einsetzen der Trainingsdaten den *Trainingsfehler*

- danach: teste mit Daten, die nicht zum Lernen verwendet wurden
  - wir erhalten den *Testfehler*
- Ziel: Minimiere den *Testfehler*

Illustration Modellselektion (bei optimalen Parametern für 10 Datenpunkte)

0^{th} Order Polynomial

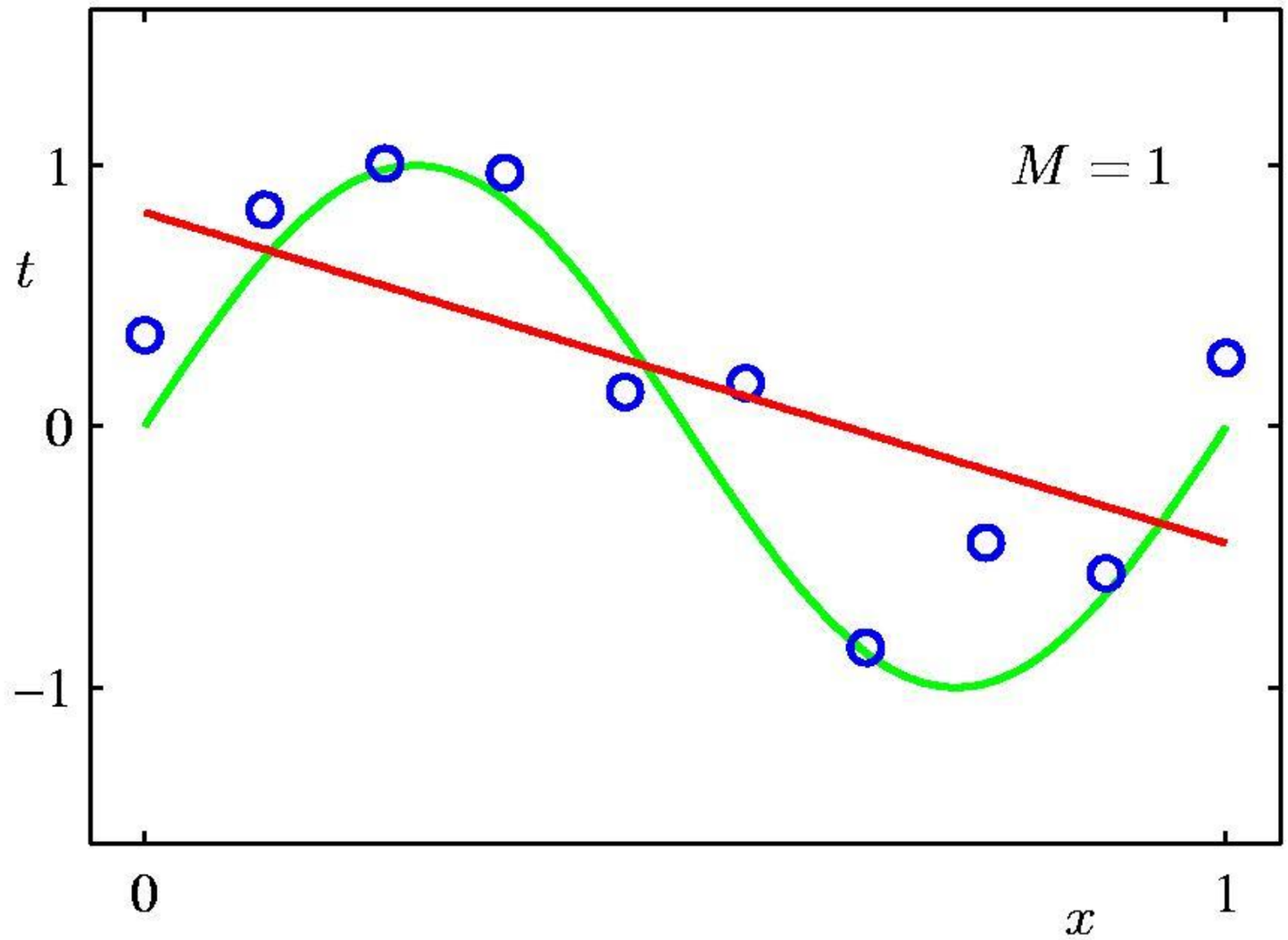


- **schlechtes Ergebnis: Modell zu einfach**

Slide from Bishop, machine learning

Illustration Modellselektion (bei optimalen Parametern für 10 Datenpunkte)

1st Order Polynomial

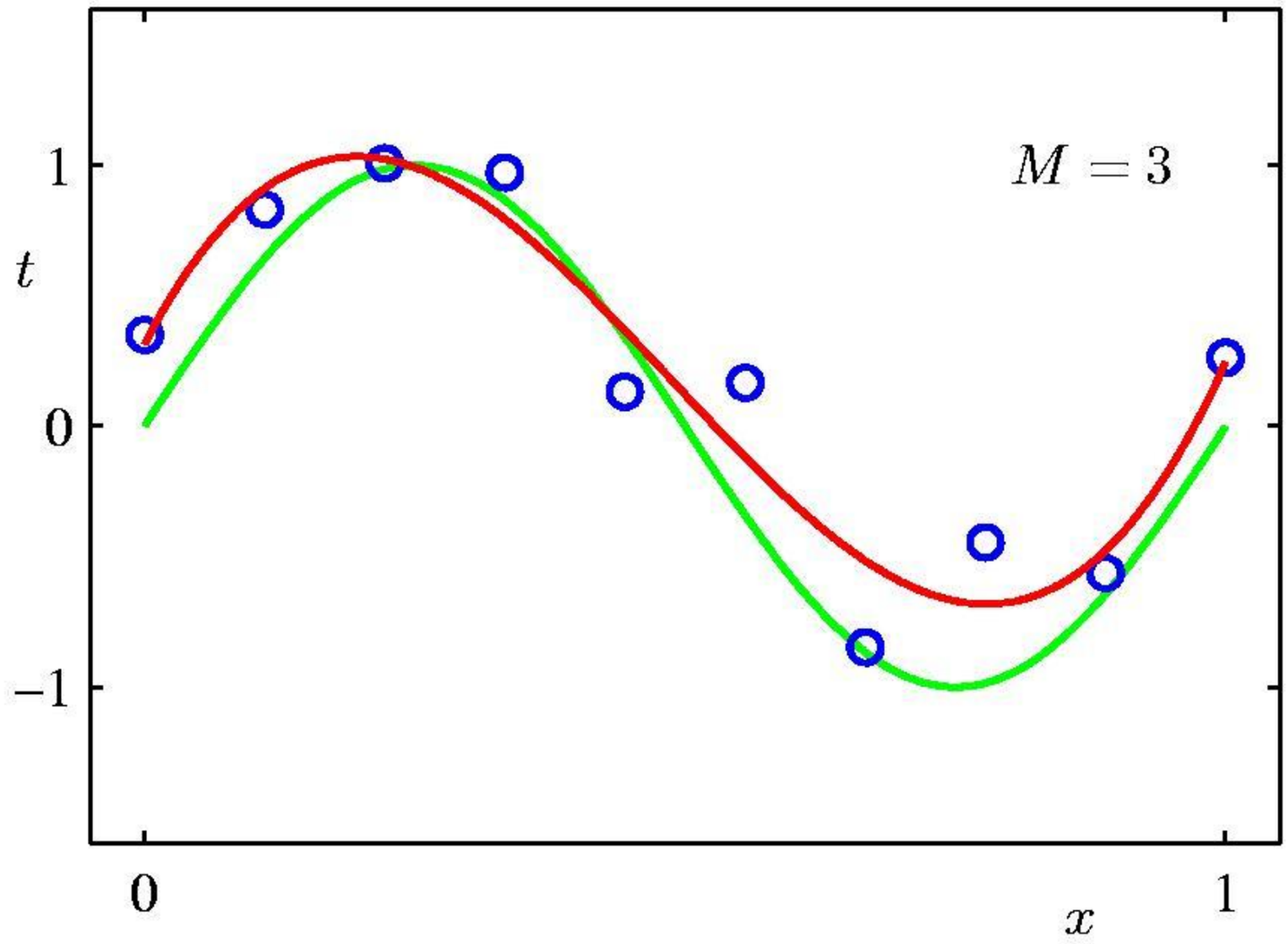


Slide from Bishop, machine learning

- schlechtes Ergebnis: Modell (rote Linie) zu einfach

Illustration Modellselektion (bei optimalen Parametern für 10 Datenpunkte)

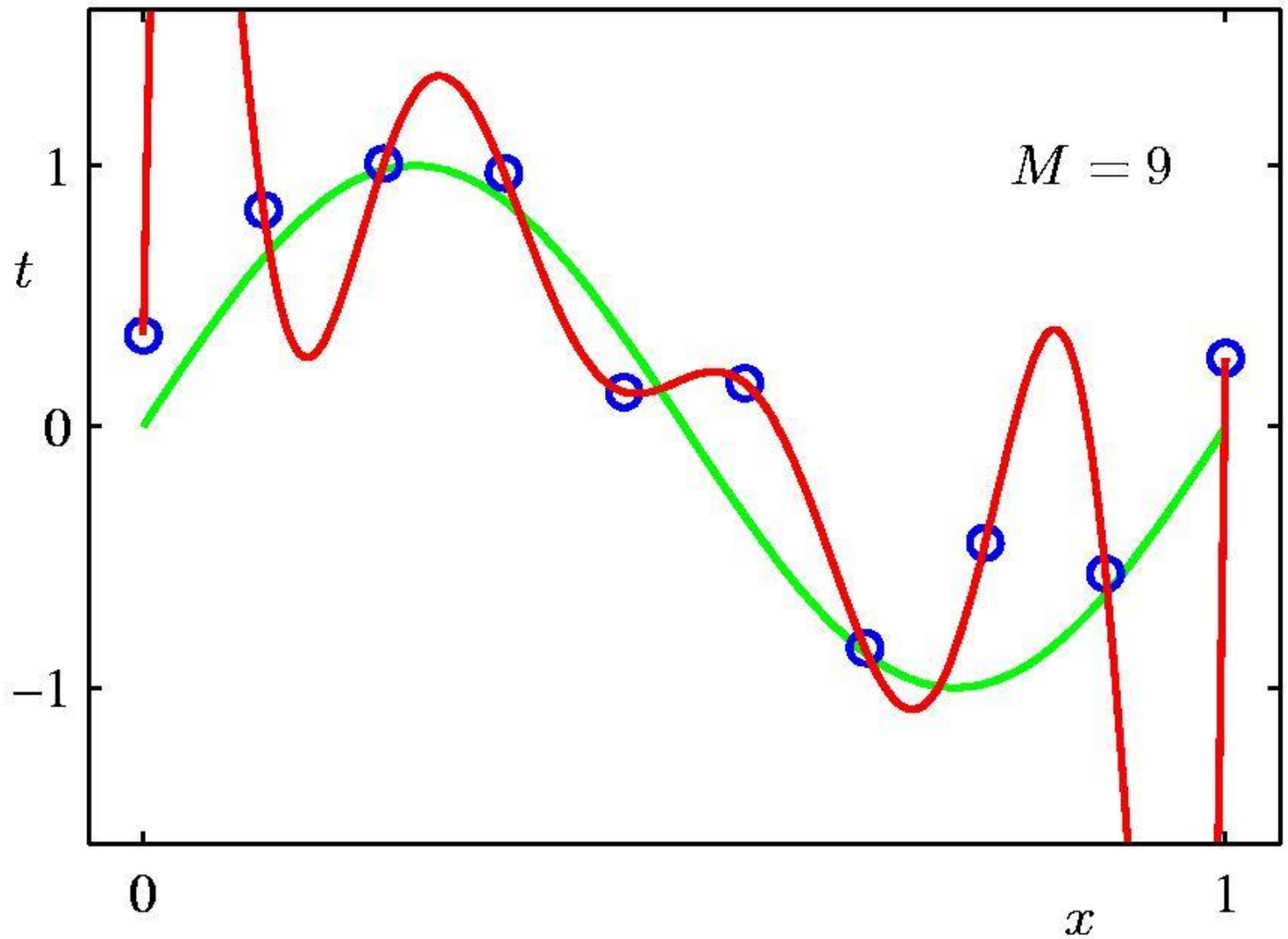
3rd Order Polynomial



Slide from Bishop, machine learning

- **gutes Ergebnis: Modell passend**

9th Order Polynomial



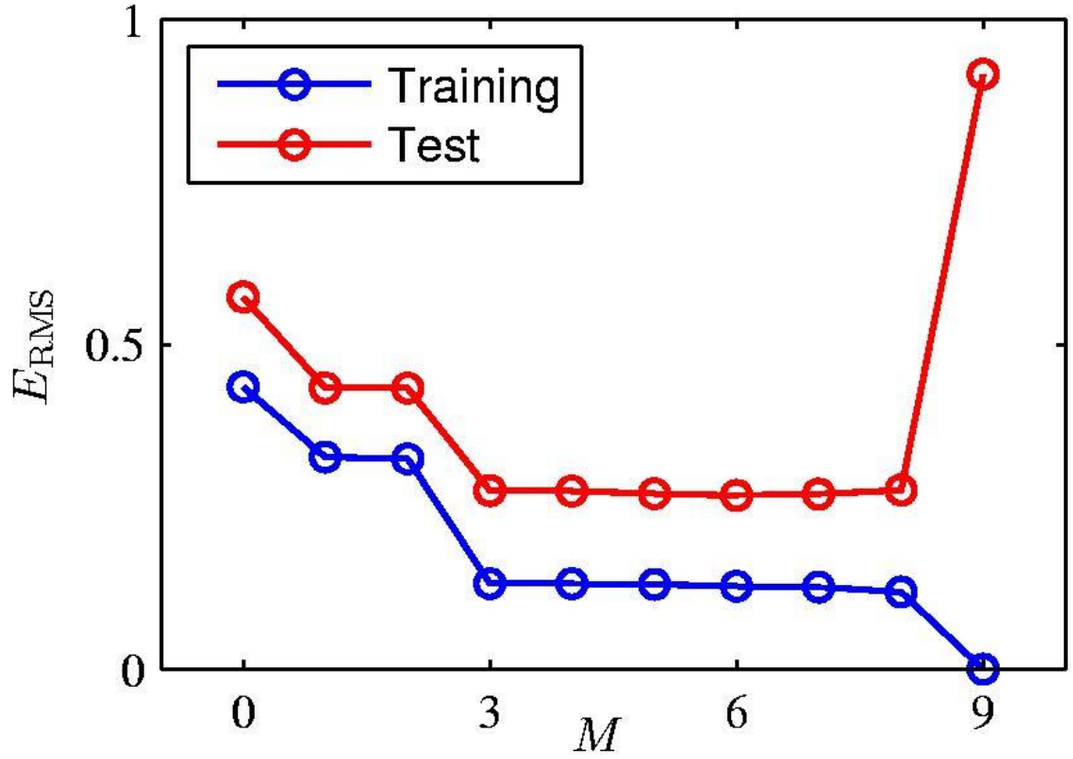
- **schlechtes Ergebnis: Modell zu kompliziert**

Slide from Bishop, machine learning

Problem: finde die richtige Modellkomplexität

Fehlerminimierung allein funktioniert nicht

- wenn die Daten verrauscht sind, dann ist E(w)=0 unerwünscht
- wenn das Modell zu komplex ist (zu viele freie Parameter), dann kann es das Rauschen lernen und E(w) << 1
- wenn Rauschen gelernt, dann ist die Generalisierung schlecht => Overfitting



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

Slide from Bishop, machine learning

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

Slide from Bishop, machine learning

Maßnahmen gegen Overfitting

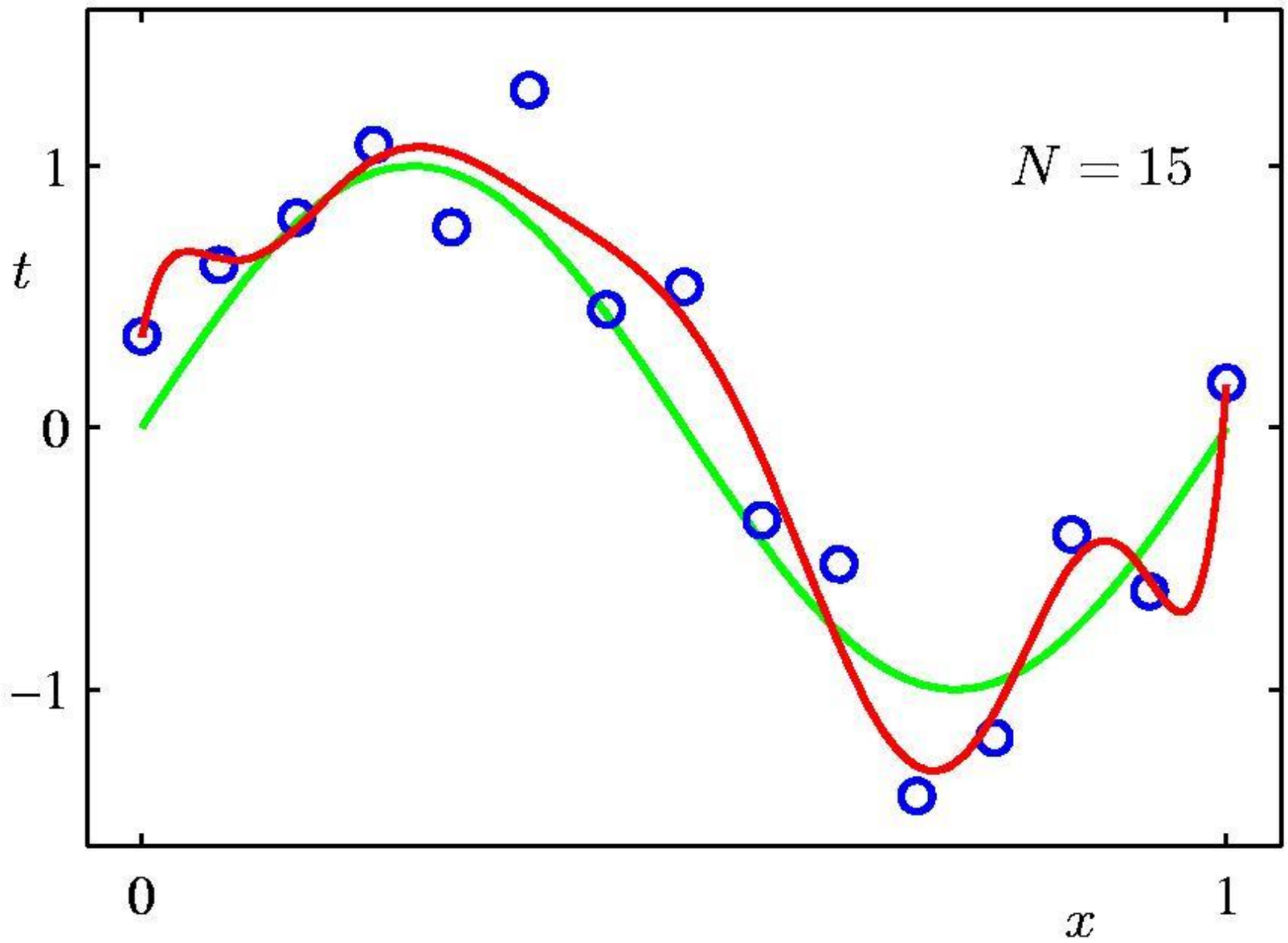
3 Ansätze

- Modellselektion
- verändere #Daten relativ zu #Parameter
- stelle zusätzliche Bedingungen an die Parameter => Regularisierung

in der Praxis ist eine Kombination oft nützlich

Data Set Size: $N = 15$

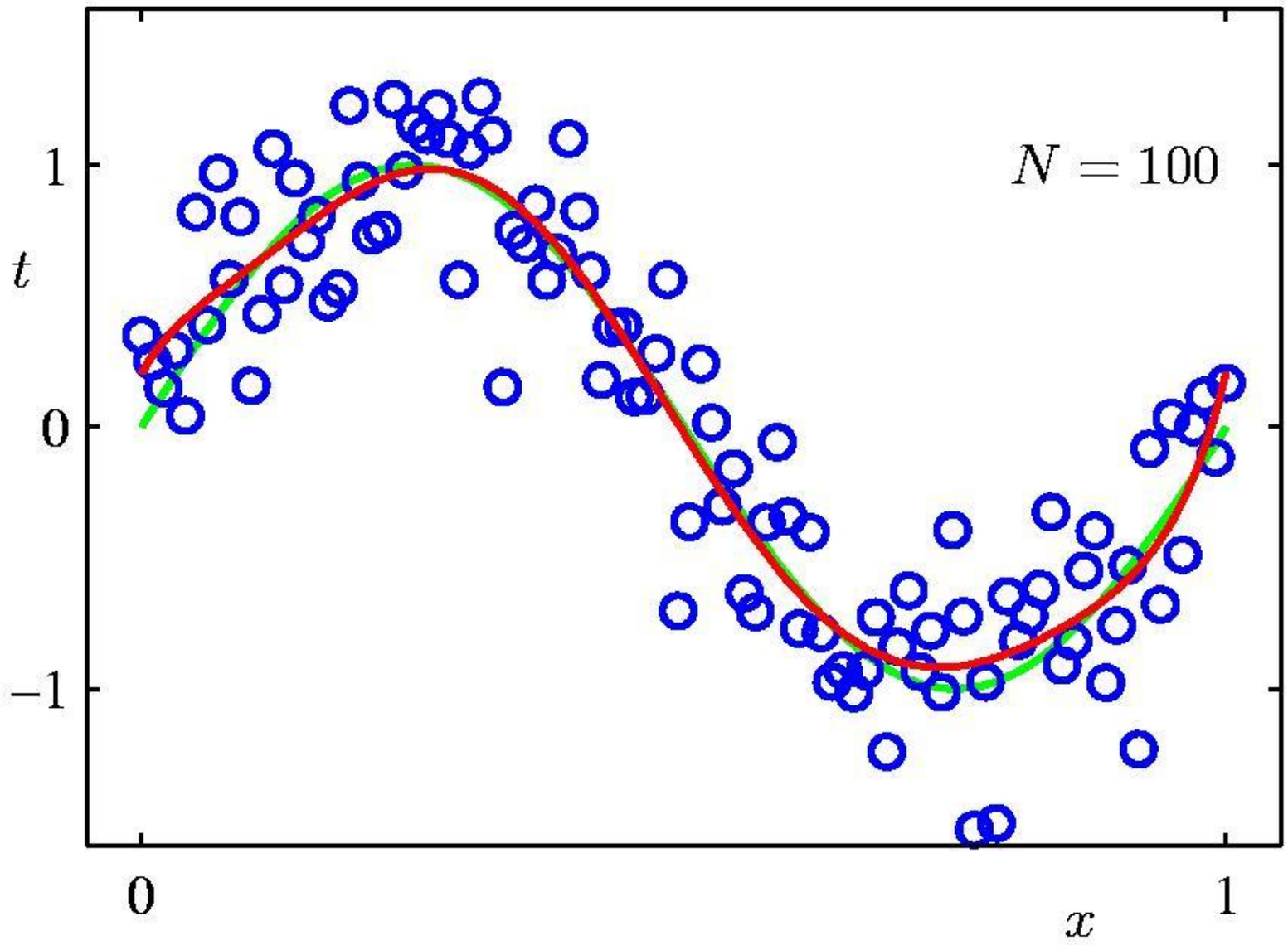
9th Order Polynomial



Slide from Bishop, machine learning

Data Set Size: $N = 100$

9th Order Polynomial



Regularisierung

hier:

- **“verbiete/verhindere” große Koeffizientenwerte**
- **beugt Oszillationen vor (! Bias: Oszillationen unerwünscht)**
- **macht die gelernt Funktion glatter (! Bias: glatte Funktion erwünscht)**

Slide from Bishop, machine learning

Penalize large coefficient values

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

- **zusätzlicher Term in der Fehlerfunktion “bestraft” große w-Werte**
- **Lambda gewichtet Fehler gegen Regularisierungsterm**

das ist nicht nur theoretisch sondern …

… aus einem aktuellen Paper*:

$$W^{\mathbf{out}} = \underset{W}{\operatorname{argmin}}(\|W \cdot H(X) - T\|^2 + \alpha\|W\|^2), \qquad (2)$$

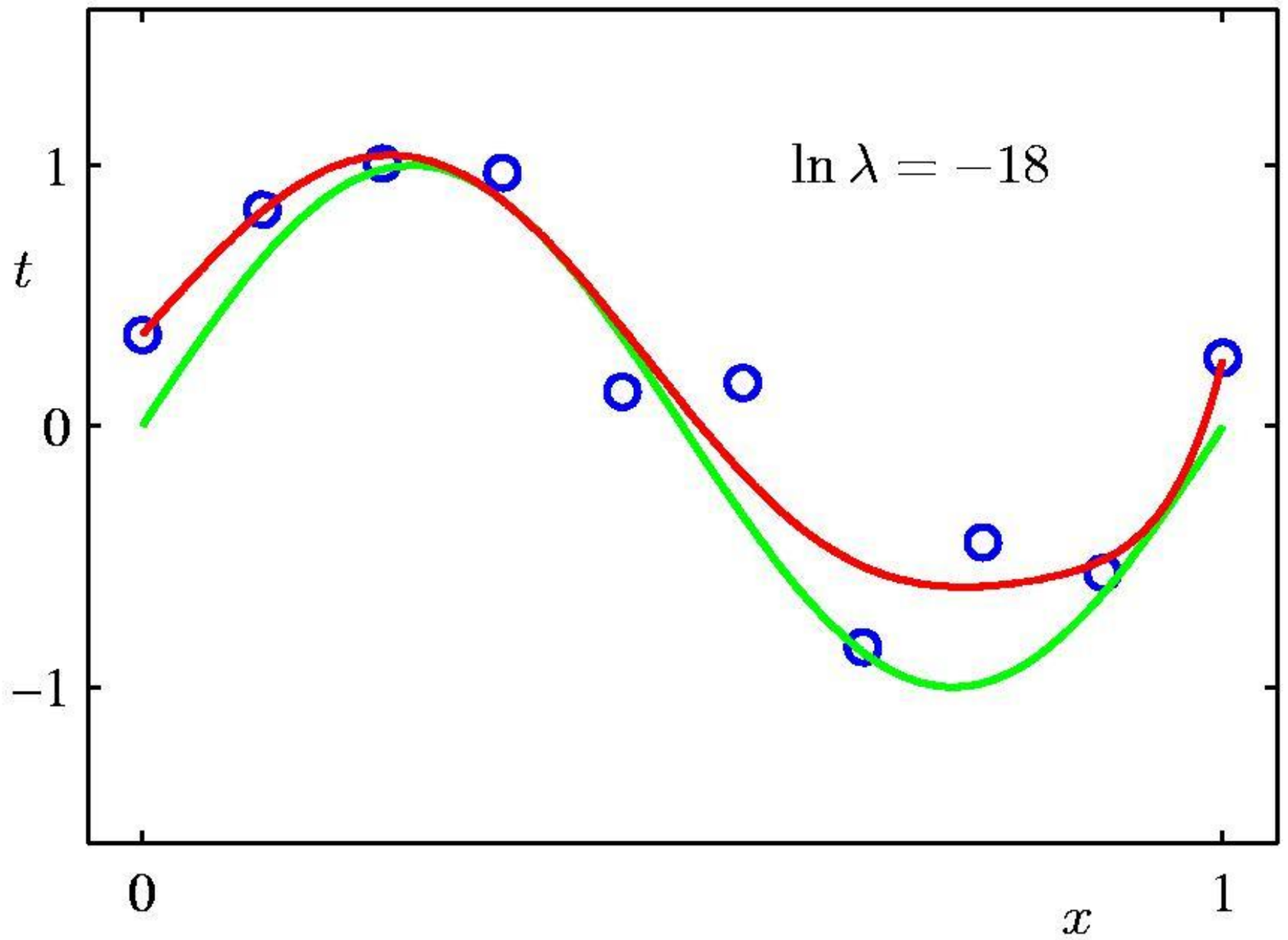
where α modulates the trade-off between weight decay and training error.

- **Minimiere quadratischen Fehler**
- **verwende ein multidim. lineares Modell W*H(X)**
- **viele Parameterdimensionen, W, H(X) sind Matrizen**
- **H(X) nichtlinear**

Regularisierung durch “Bestrafung” großer Parameter

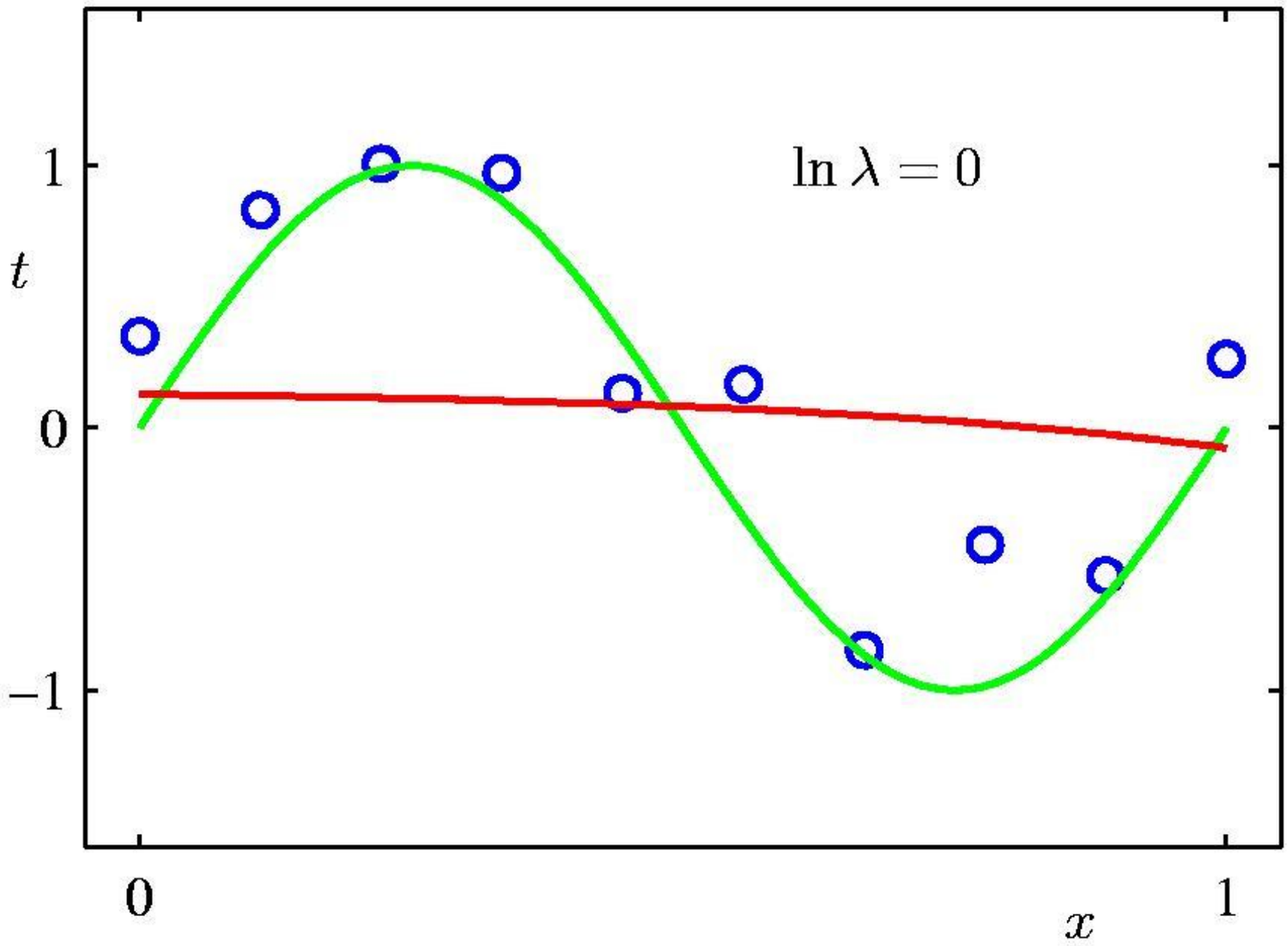
*** Neumann et al. , Reliable integration of continuous constraints into extrem learning machines, J. of Uncertainty, Fuzziness and Knowledge-based Systems, 2013**

Regularization: $\ln \lambda = -18$



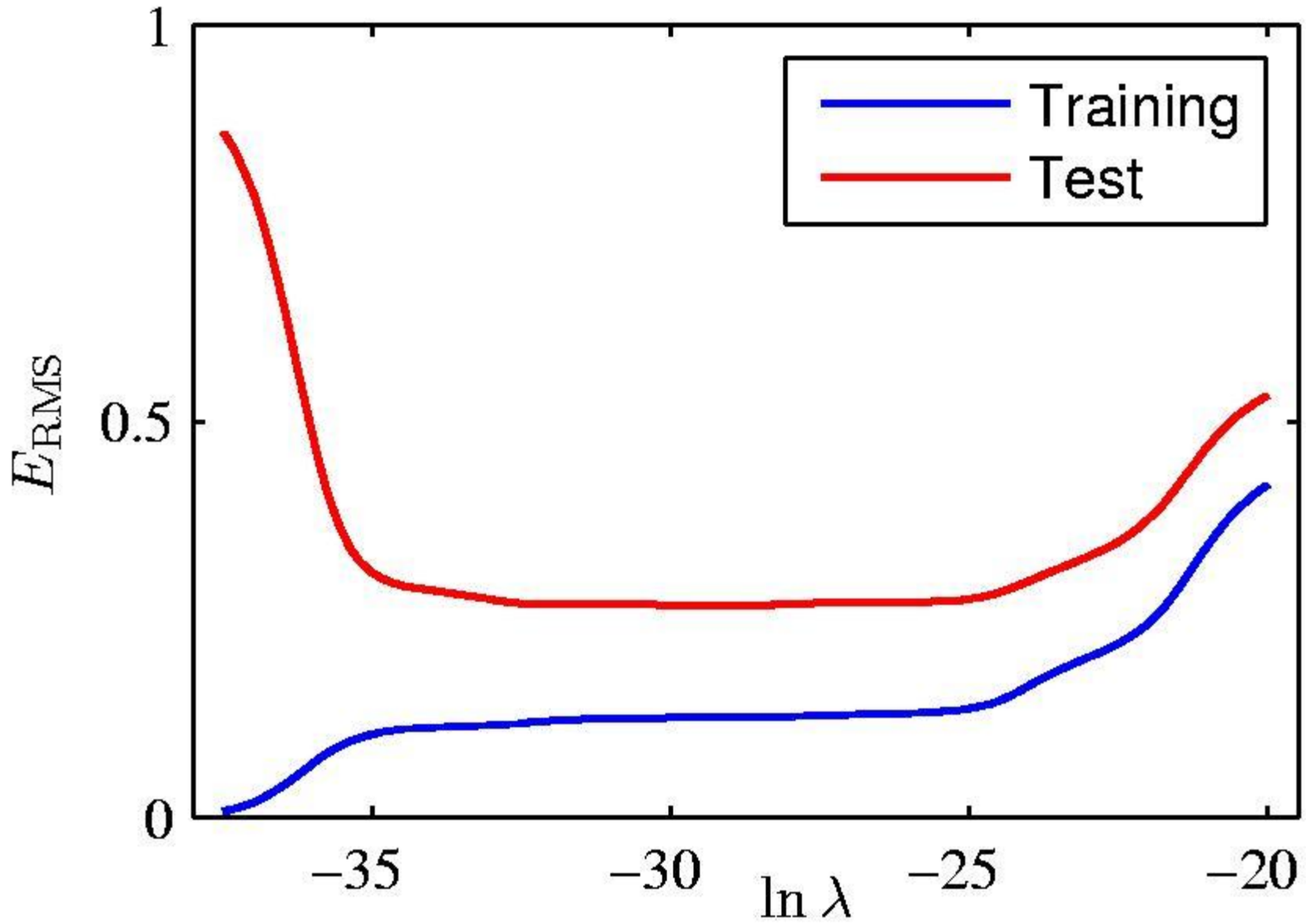
Slide from Bishop, machine learning

Regularization: $\ln \lambda = 0$



Slide from Bishop, machine learning

Regularization: E_{RMS} vs. $\ln \lambda$



Slide from Bishop, machine learning

Take-Home

- **Daten ohne Annahmen liefern nichts**
- **es kann schwierig sein, die “richtigen Daten” zu bekommen**
- **“Ausreißer” könnten auch schlicht Daten sein**

- **Modellselektion hat verschiedene Aspekte (Form, Komplexität)**
- **man *kann* den Trainingsfehler reduzieren, aber man *will* den Testfehler reduzieren**
- **wenn der Trainingsfehler = 0 ist, gibt es sehr wahrscheinlich ein Problem, (ganz sicher für verrauscht Realweltdaten)**
- **wenn Modellparameter groß sind, gibt es sehr wahrscheinlich ein Problem (ganz sicher für Polynommodell => Oszillationen)**

Take-Home

Parameteroptimierung:

- “kleine Parameter” ist eine gängige Annahme (Bias)
- Regularisierung => neuer Parameter der Fehlerminimierung gegen “Glattheit” balanciert
- häufig: systematische Suche nach “regularization parameter”