Exkurs: Funktionsapproximation mittels Polynomen

SS 19 – Machine Learning and Computer Vision

Multimedia Computing & Computer Vision, Universität Augsburg
Rainer.Lienhart@informatik.uni-augsburg.de
www.multimedia-computing.{de,org}

Literatur

Christopher Bishop. **Pattern Recognition and Machine Learning** . Springer Verlag.

Kapitel **Introduction**

(Kapitel 1 in der ersten Auflage)

Ein Großteil der Bilder wurde diesem Buch entnommen.

Beispiel

- Mit synthetischen Daten
- Gegeben sei der Prozess:

$$y(x) = \sin(2\pi x) \text{ mit } x \in [0,1],$$

der aber nur verrauscht (mit additivem Gaußrauschen $N(0, \frac{3}{10})$ gemessen werden kann:

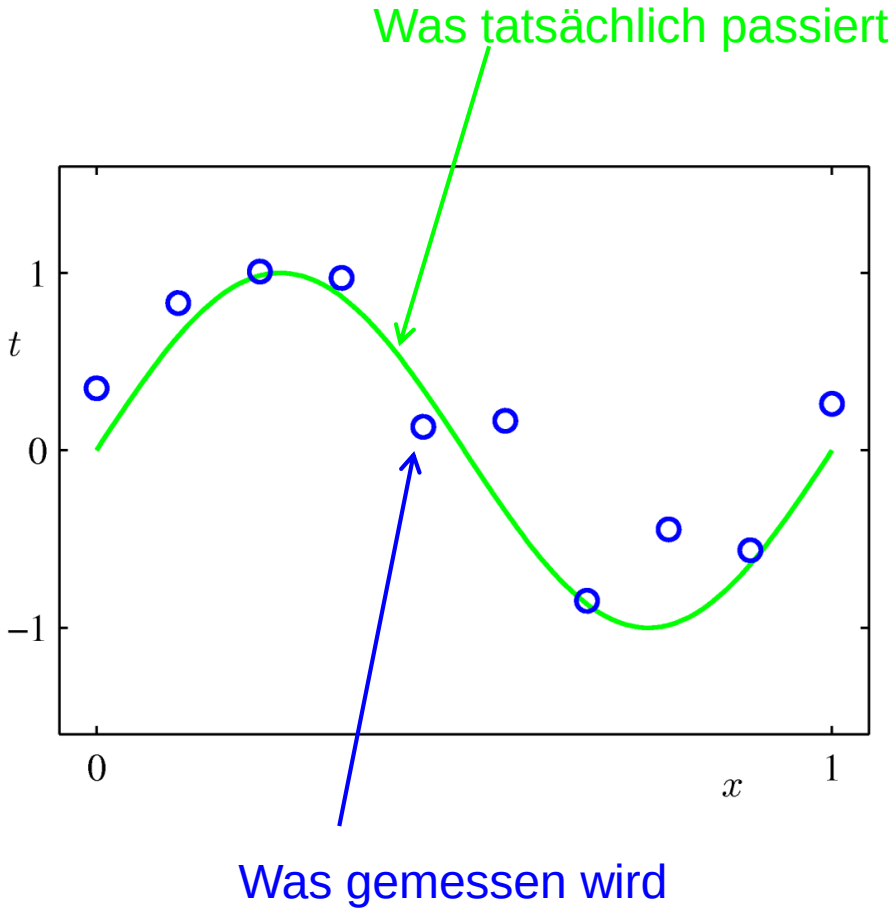
$$y(x) = \sin(2\pi x) + \varepsilon$$

$$\text{mit } x \in [0,1], \varepsilon \sim N(0, \frac{3}{10})$$

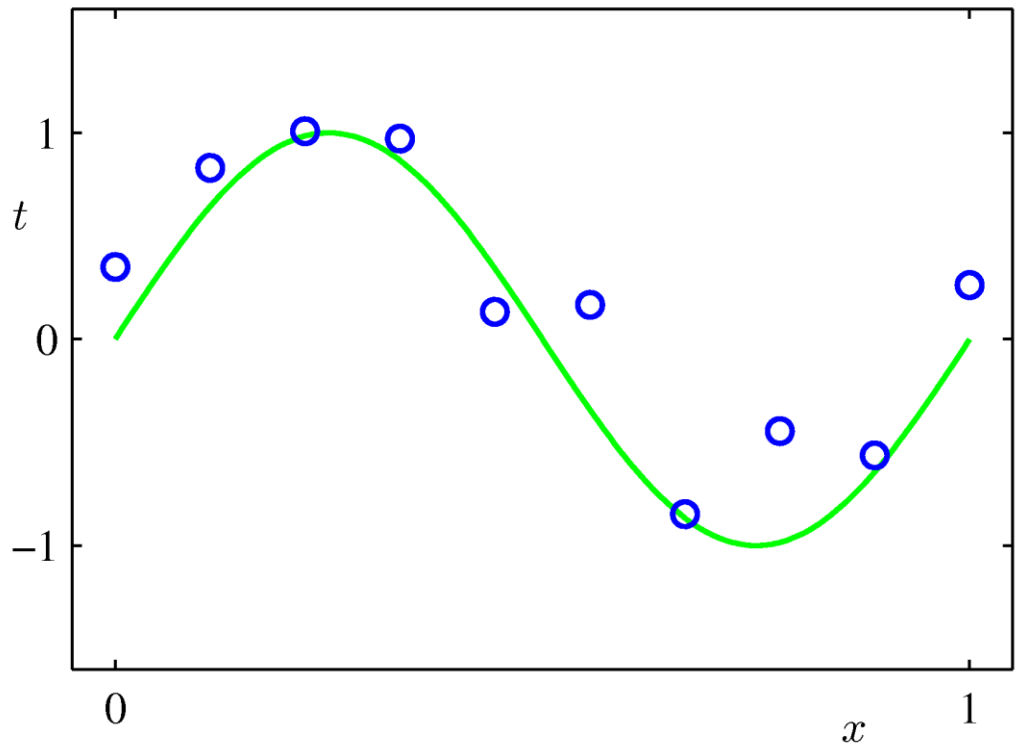
10 Messwerte

in gleichmäßigem Abstand:

x	t
0.000000	0.349486
0.111111	0.830839
0.222222	1.007332
0.333333	0.971507
0.444444	0.133066
0.555556	0.166823
0.666667	-0.848307
0.777778	-0.445686
0.888889	-0.563567
1.000000	0.261502



Funktionsapproximation mittels Polynomen



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

Definition: Linearität

Gegeben $V \in \mathbb{R}^n/\mathbb{C}^n$, $W \in \mathbb{R}^m/\mathbb{C}^m$ mit $n, m \in \mathbb{N}$.

Ein Abbildung $f: V \to W$ heißt lineare Abbildung, wenn

1. $f(ax) = af(x)$ $\quad \forall x \in V, \forall a \in \mathbb{R} / \mathbb{C}$
2. $f(x + y) = f(x) + f(y)$ $\quad \forall x, y \in V$

Man kann beide Bedingungen zu einer zusammenfassen:

$f(ax + y) = af(x) + f(y)$ $\quad \forall x, y \in V, \forall a \in \mathbb{R}/ \mathbb{C}$

Funktionsapproximation mittels Polynomen (2)

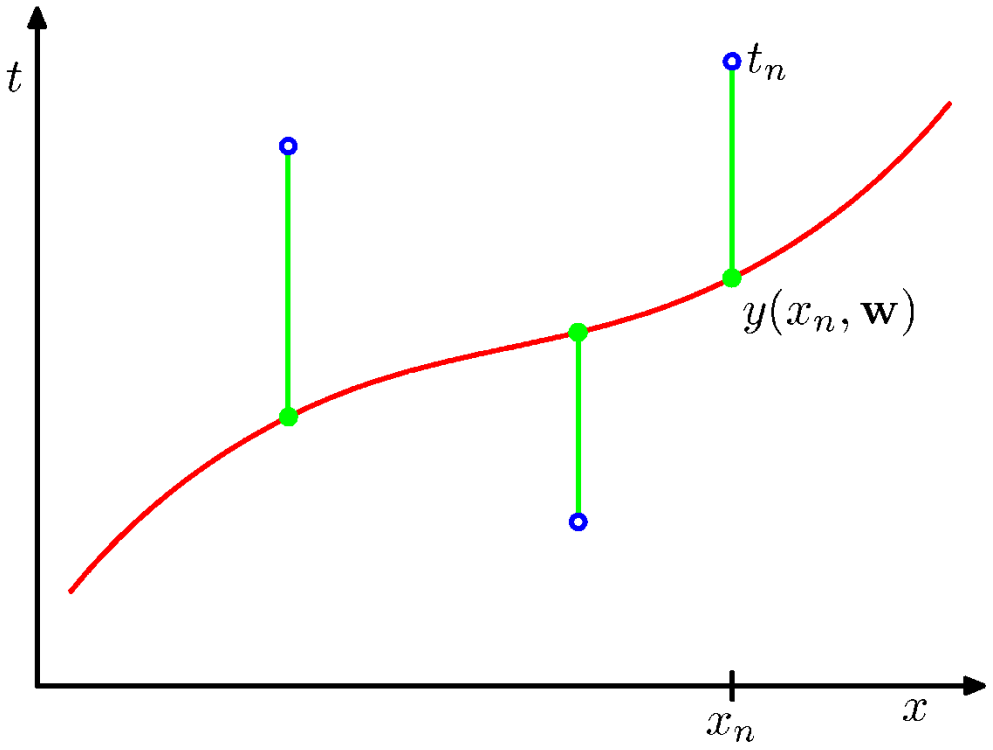
$$y(x,\mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

- Polynom *M*-ten Grades
- y(x,**w**) nichtlinear in x
- y(x,**w**) linear in **w**, den unbekannten Parametern

- Bestimme **w** so, dass für die *N*=10 Messwerte $\{(x_n,t_n)\}$ der Vorhersagefehler minimal wird

$$Error(\mathbf{w}) = \frac{1}{2}\sum_{n=0}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 = \frac{1}{2}\sum_{n=0}^{N}\{\sum_{j=0}^{M} w_j x_n^j - t_n\}^2$$

Summe der Fehlerquadrate



$$Error(\mathbf{w}) = \frac{1}{2}\sum_{n=0}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 = \frac{1}{2}\sum_{n=0}^{N}\{\sum_{j=0}^{M} w_j x_n^j - t_n\}^2$$

Ansatz

$$Error(\mathbf{w}) = \frac{1}{2}\sum_{n=0}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 = \frac{1}{2}\sum_{n=0}^{N}\{\sum_{j=0}^{M} w_j x_n^j - t_n\}^2$$

- Minimiere $Error(\boldsymbol{w})$ bezüglich $\boldsymbol{w}$

➔Suche lokales Minimum durch Bestimmung der 1.ten Ableitung nach $\boldsymbol{w}$, d.h. nach jedem Parameter w_i

$$\frac{\partial}{\partial w_i} Error(\mathbf{w}) = \frac{\partial}{\partial w_i}\left(\frac{1}{2}\sum_{n=0}^{N}\{\sum_{j=0}^{M} w_j x_n^j - t_n\}^2\right)$$

Ansatz (2)

$$\frac{\partial}{\partial w_i} Error(\mathbf{w}) = \frac{\partial}{\partial w_i}\left(\frac{1}{2}\sum_{n=0}^{N-1}[\sum_{j=0}^{M} w_j x_n^j - t_n]^2\right)$$

$$= \frac{1}{2}\frac{\partial}{\partial w_i}\left(\sum_{n=0}^{N-1}[\sum_{j=0}^{M} w_j x_n^j - t_n]^2\right)$$

$$= \frac{1}{2}\sum_{n=0}^{N-1}\frac{\partial}{\partial w_i}\left(\sum_{j=0}^{M} w_j x_n^j - t_n\right)^2$$

$$= \sum_{n=0}^{N-1}\left[\left(\sum_{j=0}^{M} w_j x_n^j - t_n\right)\frac{\partial}{\partial w_i}\left(\sum_{j=0}^{M} w_j x_n^j - t_n\right)\right]$$

Ansatz (2)

$$\frac{\partial}{\partial w_i} Error(\mathbf{w}) = \sum_{n=0}^{N-1}\left[\left(\sum_{j=0}^{M} w_j x_n^j - t_n\right)\frac{\partial}{\partial w_i}\left(\sum_{j=0}^{M} w_j x_n^j - t_n\right)\right]$$

$$= \sum_{n=0}^{N-1}\left[\left(\sum_{j=0}^{M} w_j x_n^j - t_n\right)\left(\sum_{j=0}^{M}\frac{\partial}{\partial w_i}\left[w_j x_n^j\right] - \frac{\partial}{\partial w_i} t_n\right)\right]$$

$$= \sum_{n=0}^{N-1}\left[\left(\sum_{j=0}^{M} w_j x_n^j - t_n\right)\left(\sum_{j=0}^{M}\frac{\partial}{\partial w_i}\left[w_j x_n^j\right]\right)\right]$$

$$= \sum_{n=0}^{N-1}\left[\left(\sum_{j=0}^{M} w_j x_n^j - t_n\right)\frac{\partial}{\partial w_i}\left(w_i x_n^i\right)\right]$$

$$= \sum_{n=0}^{N-1}\left[\left(\sum_{j=0}^{M} w_j x_n^j - t_n\right)x_n^i\right]$$

Ansatz (3)

- Diese Ableitungen müssen im Minimum gleich 0 sein

  ➔ (*m+1)* Gleichungen für die (*m+1)* Parameter in ***w***

  ➔ Gleichungssystem kann geschlossen gelöst werden (z.B. Gaußeliminationsverfahren)

Eleganterer Ansatz

- Grundansatz:

Lineares Model:

$$t = w_0 \cdot 1 + w_1 \cdot x^1 + \cdots + w_M \cdot x^M$$

$$= \begin{bmatrix} 1 & x^1 & \cdots & x^M \end{bmatrix} \cdot \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_M \end{bmatrix}$$

Ein Daten-beispiel

Modell-parameter

N>M lineare Gleichungen:

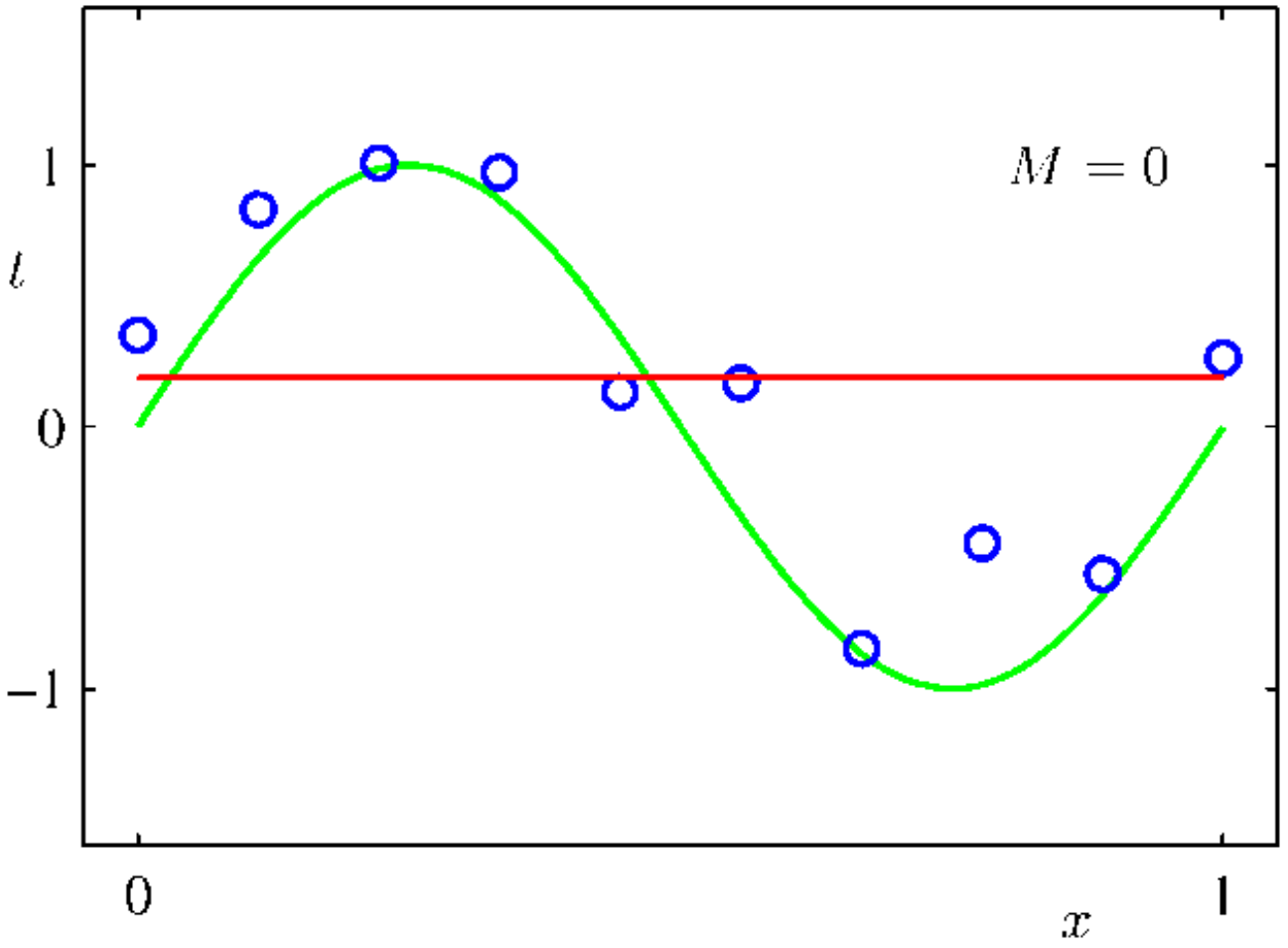
$$\begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix} = \begin{bmatrix} 1 & x_1^1 & \cdots & x_1^M \\ 1 & x_2^1 & \cdots & x_2^M \\ 1 & \vdots & \ddots & \vdots \\ 1 & x_N^1 & \cdots & x_N^M \end{bmatrix} \cdot \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_M \end{bmatrix}$$

$$\mathbf{t} = \mathbf{G}\mathbf{w}$$

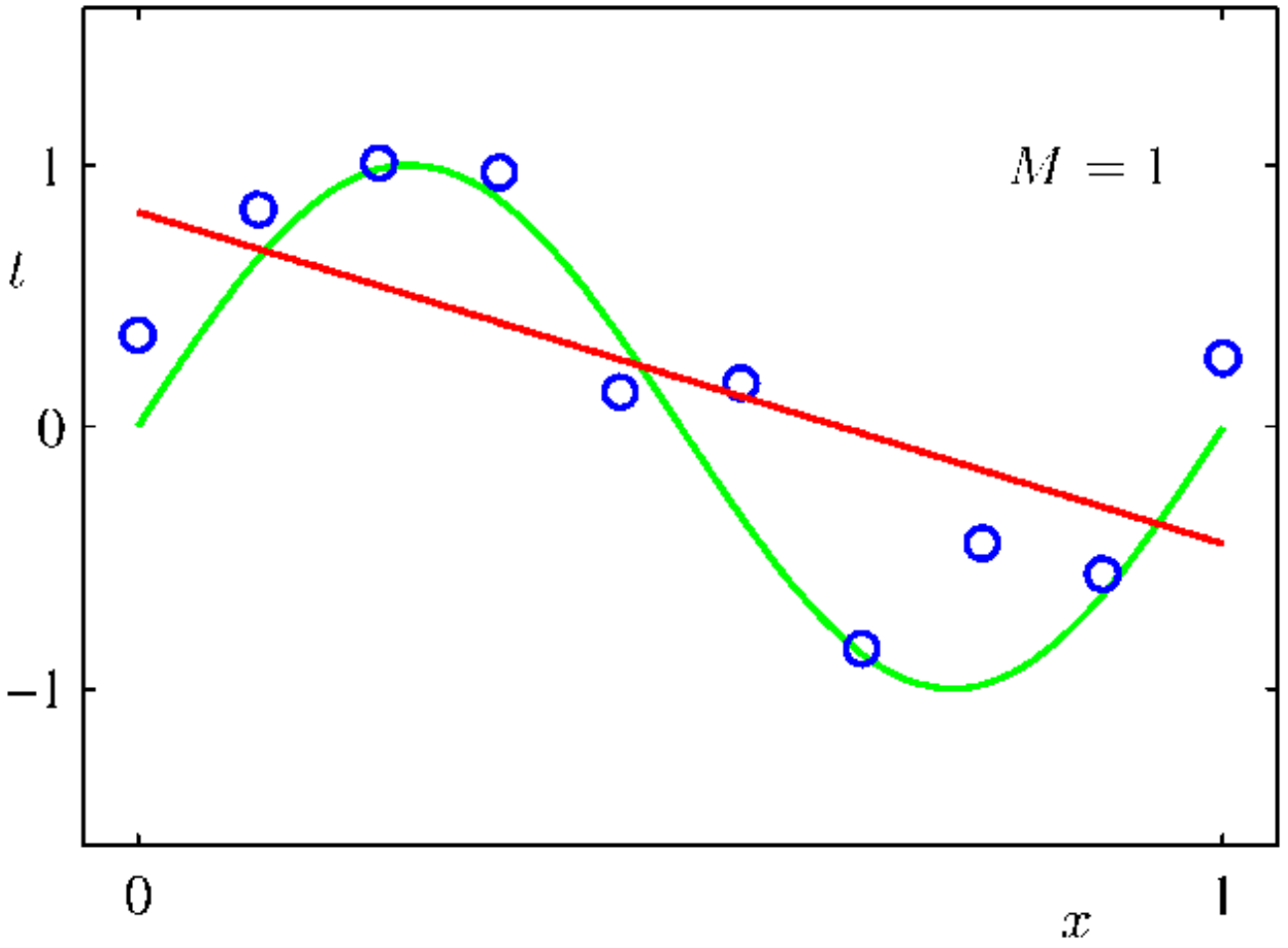
- Lösung: Falls $\mathbf{G}^T\mathbf{G}$ invertierbar, dann

$$w_{opt} = \left(\mathbf{G}^T\mathbf{G}\right)^{-1}\mathbf{G}^T\mathbf{t}$$

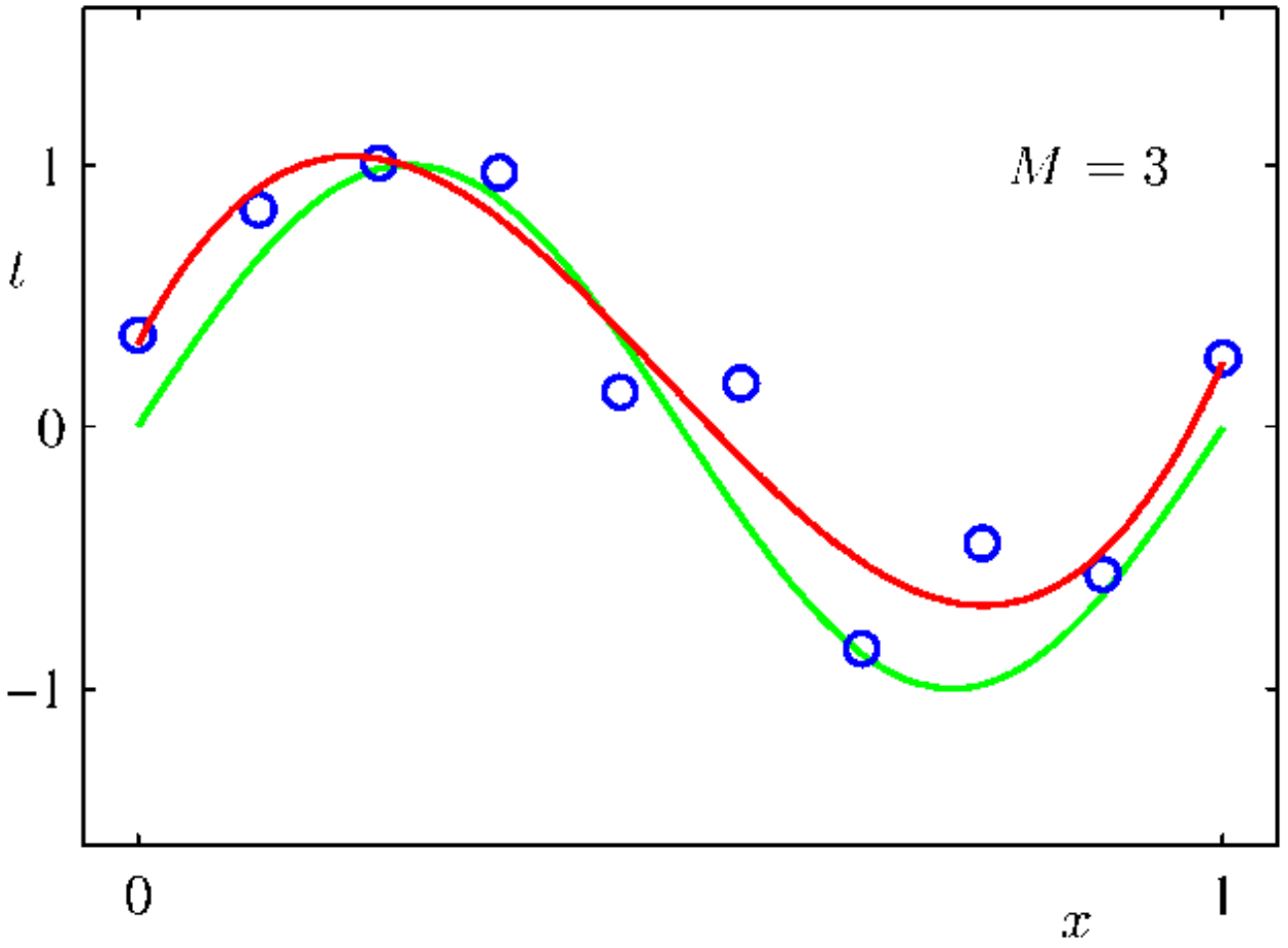
Polynom 0-ter Ordnung



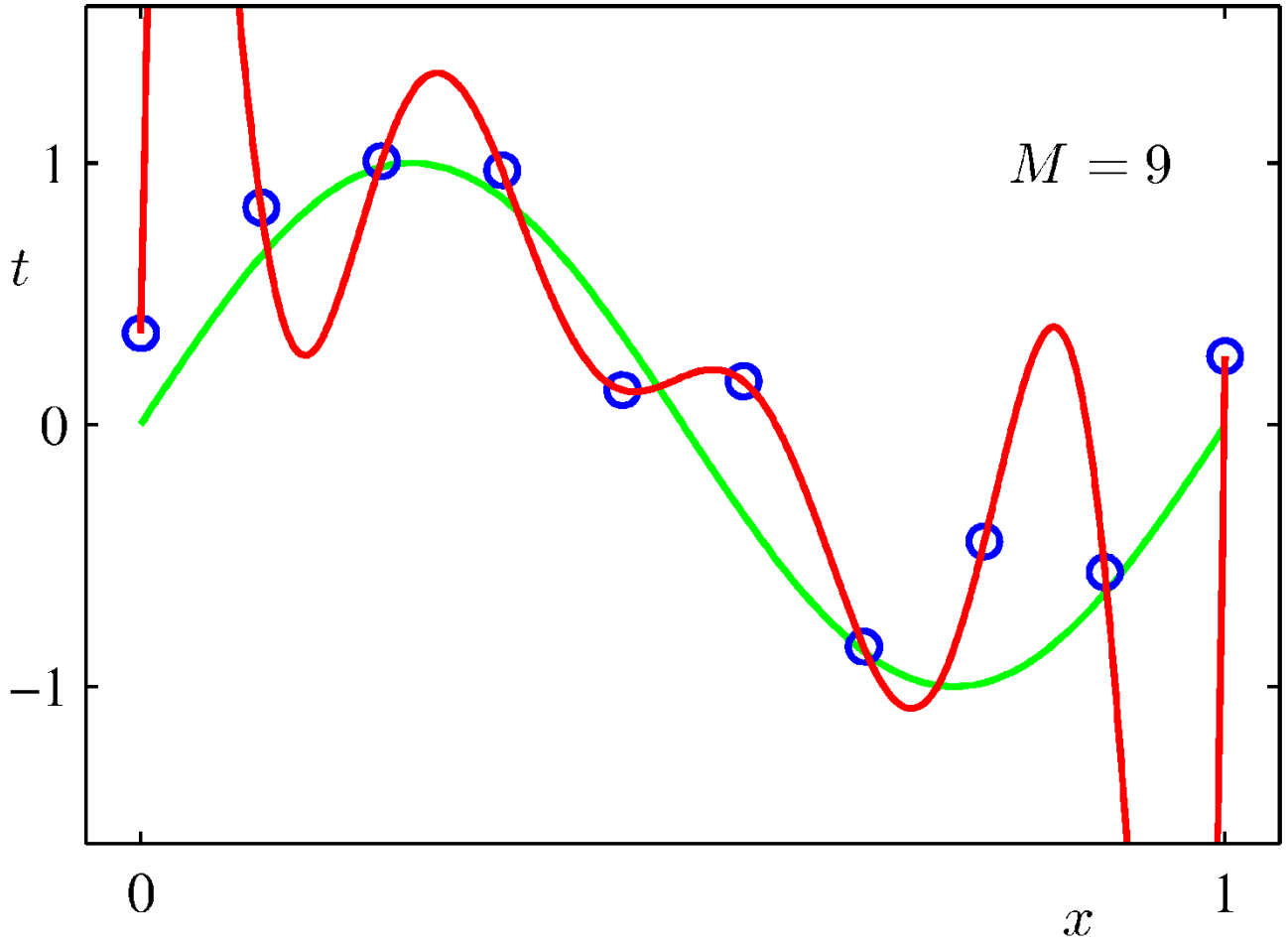
Polynom 1-ter Ordnung



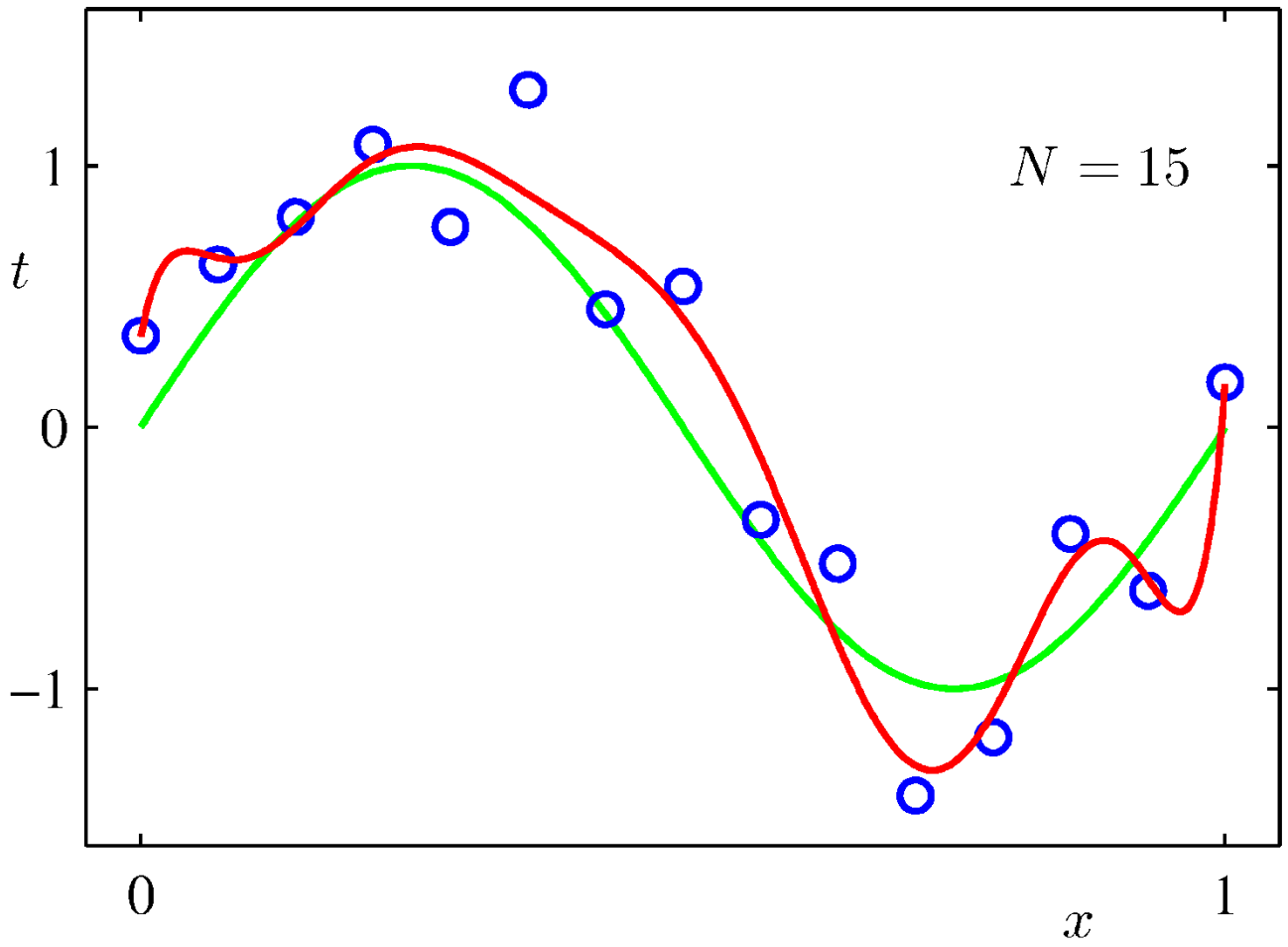
Polynom 3-ter Ordnung



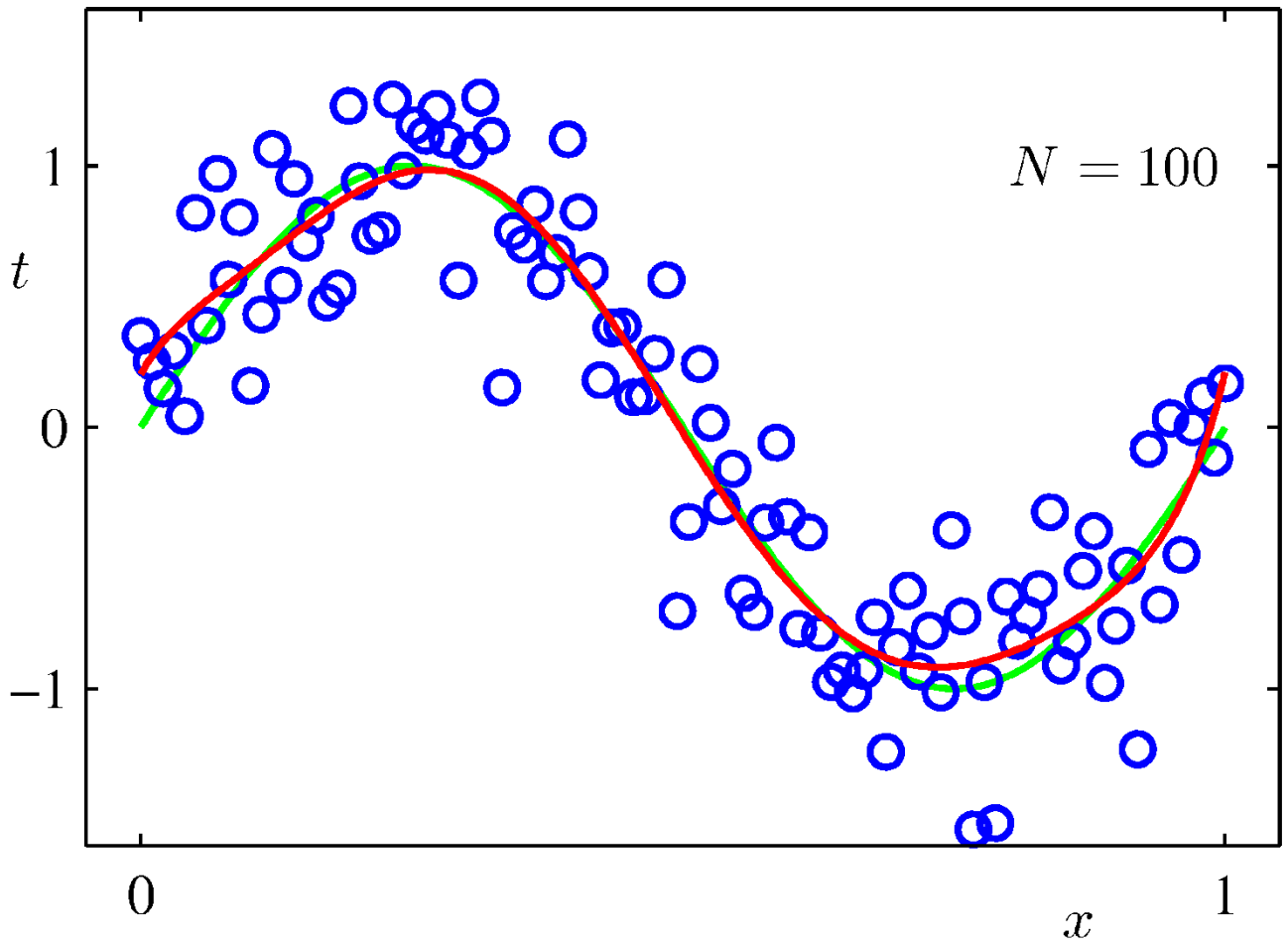
Polynom 9-ter Ordnung



Polynom 9-ter Ordnung, N=15



Polynom 9-ter Ordnung, N=100



Folgerungen

- Die Wahl des Grades n des Polynoms muss geeignet für die Aufgabe und die Anzahl der Trainingsbeispiele gewählt werden.
- Die Wahl des Grades n des Polynoms bestimmt, welche „Funktionen“ gelernt werden können. Sie bestimmt also die Menge, aus der die Lösung gezogen werden kann. Diese Menge nennen wir den *Hypothesenraum*.
- Da die Lösung stets aus dem Hypothesenraum kommt, nennt man diese Einschränkung of „*restriction bias*“

„Preference Bias“ – Motivation (1)

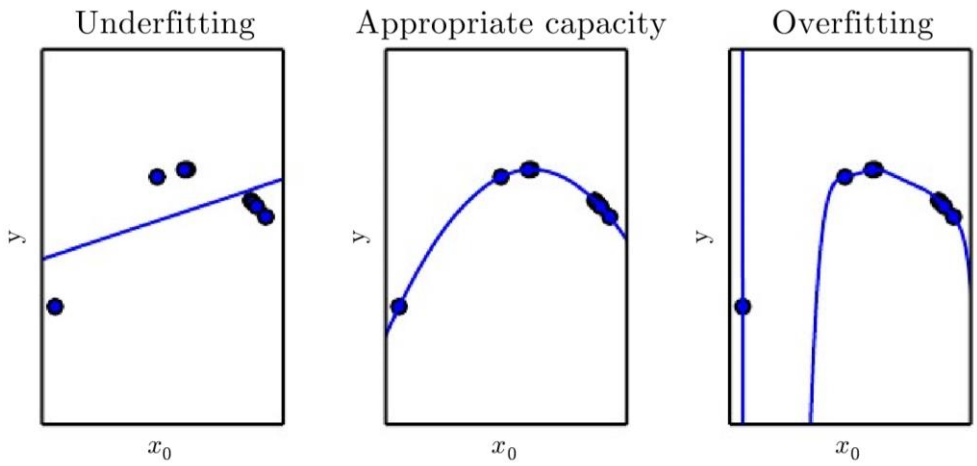


Figure 5.2: We fit three models to this example training set. The training data was generated synthetically, by randomly sampling x values and choosing y deterministically by evaluating a quadratic function. *(Left)*A linear function fit to the data suffers from underfitting—it cannot capture the curvature that is present in the data. *(Center)*A quadratic function fit to the data generalizes well to unseen points. It does not suffer from a significant amount of overfitting or underfitting. *(Right)*A polynomial of degree 9 fit to the data suffers from overfitting. Here we used the Moore-Penrose pseudoinverse to solve the underdetermined normal equations. The solution passes through all the training points exactly, but we have not been lucky enough for it to extract the correct structure. It now has a deep valley between two training points that does not appear in the true underlying function. It also increases sharply on the left side of the data, while the true function decreases in this area.

„Preference Bias“ – Beispiel *weight decay*

Ansatz bisher:

$$J(\boldsymbol{w}) = MSE_{train}$$

Ansatz jetzt:

$$J(\boldsymbol{w}) = MSE_{train} + \lambda \boldsymbol{w}^T \boldsymbol{w}$$

Regularization := any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.

Here: Regularizer equals $\Omega(\boldsymbol{w}) = \lambda \boldsymbol{w}^T \boldsymbol{w}$ and $\lambda \in \mathbb{R}$ chosen appropriately. It is called *weight decay*.

„Preference Bias“ – Motivation (2)

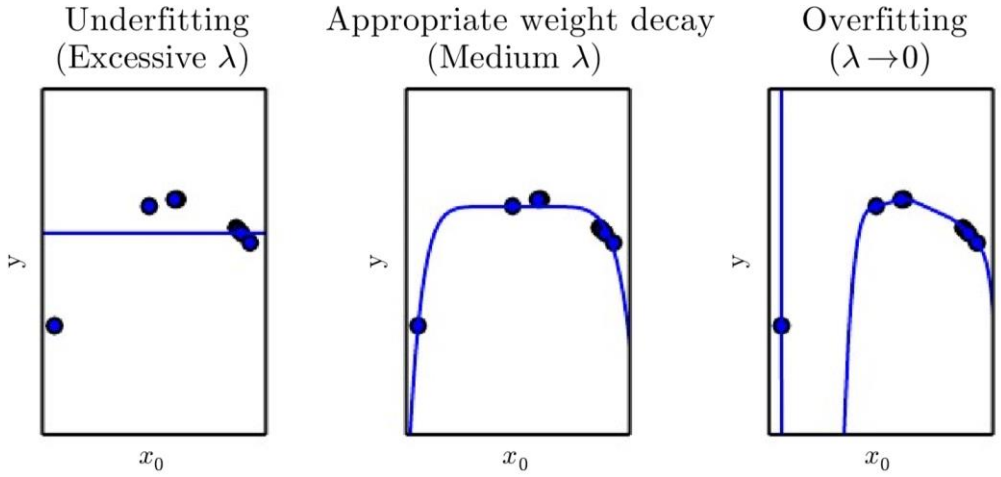


Figure 5.5: We fit a high-degree polynomial regression model to our example training set from figure 5.2. The true function is quadratic, but here we use only models with degree 9. We vary the amount of weight decay to prevent these high-degree models from overfitting. *(Left)*With very large λ, we can force the model to learn a function with no slope at all. This underfits because it can only represent a constant function. *(Center)*With a medium value of λ, the learning algorithm recovers a curve with the right general shape. Even though the model is capable of representing functions with much more complicated shapes, weight decay has encouraged it to use a simpler function described by smaller coefficients. *(Right)*With weight decay approaching zero (i.e., using the Moore-Penrose pseudoinverse to solve the underdetermined problem with minimal regularization), the degree-9 polynomial overfits significantly, as we saw in figure 5.2.