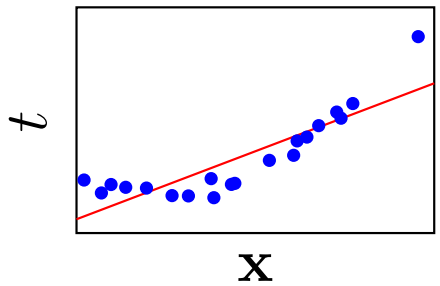
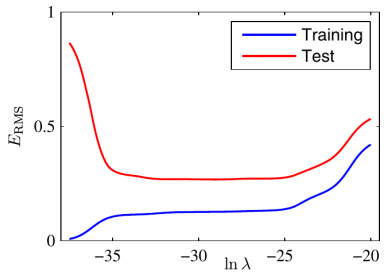
Machine Learning

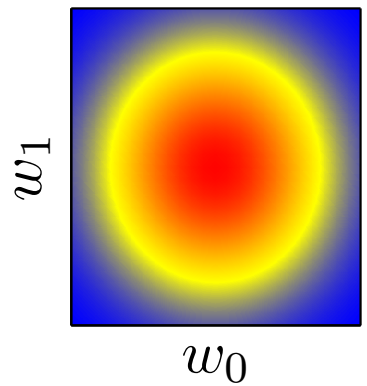
Linear Models for Regression

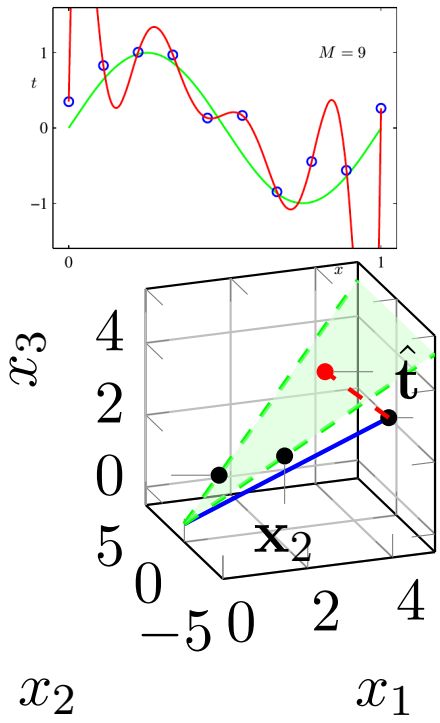


Marcello Restelli

February 20, 2024



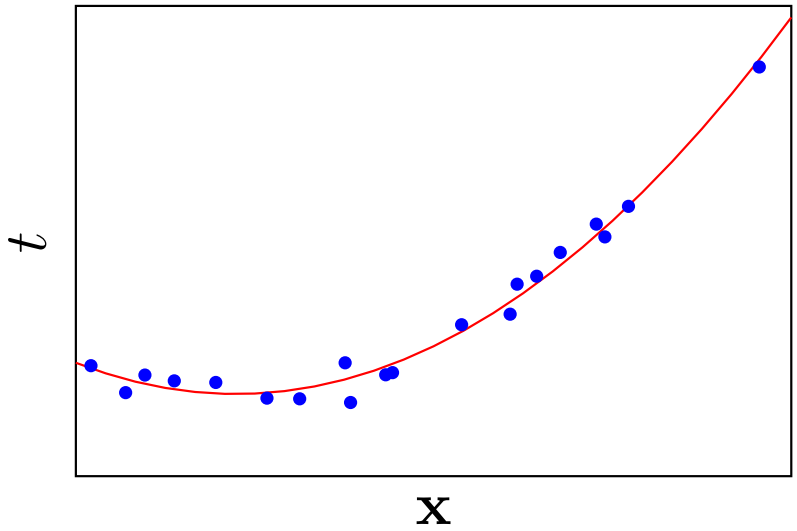




Outline

Regression Problems

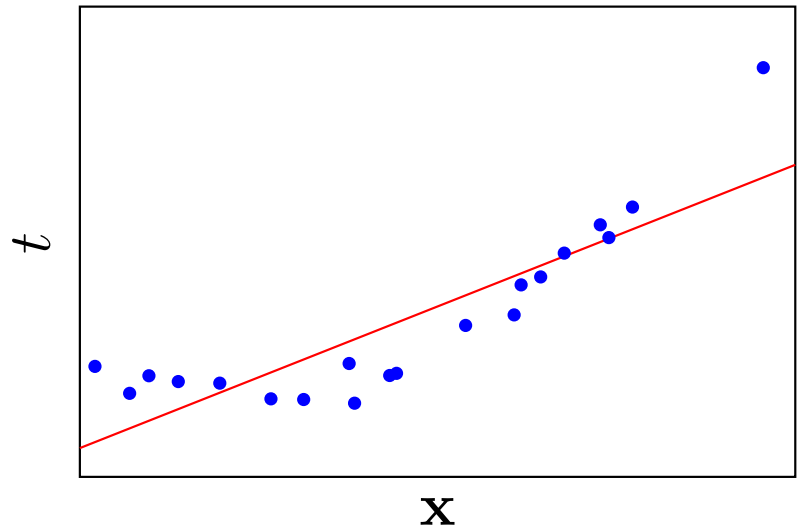
- The **goal** of regression is to learn a **mapping** from input $\mathbf{x}$ to a continuous output t



- Examples
  - Predict stock market price
  - Predict age of a web user
  - Predict effect of an actuation in robotics
  - Predict the value of a house
  - Predict the temperature in a building

Linear Models

- Many real processes can be **approximated** with linear models
- Linear regression often appears as a **module** of larger systems



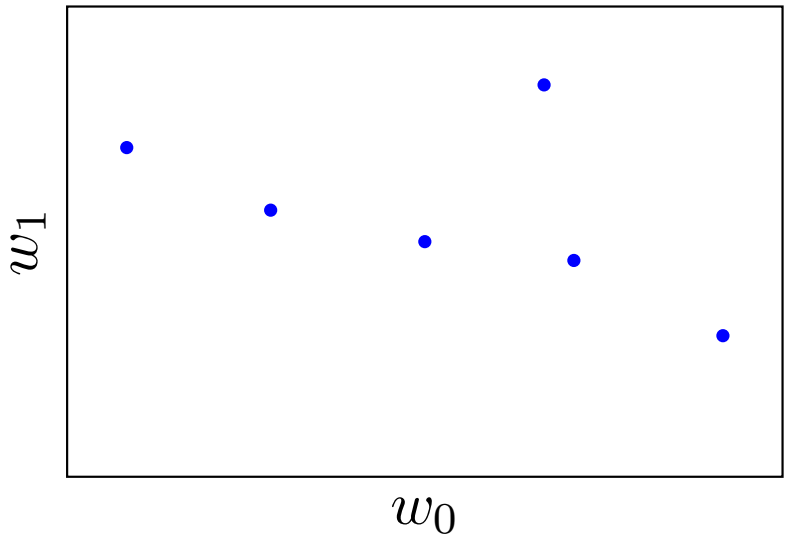
- Linear problems can be solved **analytically**
- Linear prediction provides an introduction to many of the **core concepts** of machine learning
- Augmented with kernels, it can model **non-linear** relationships

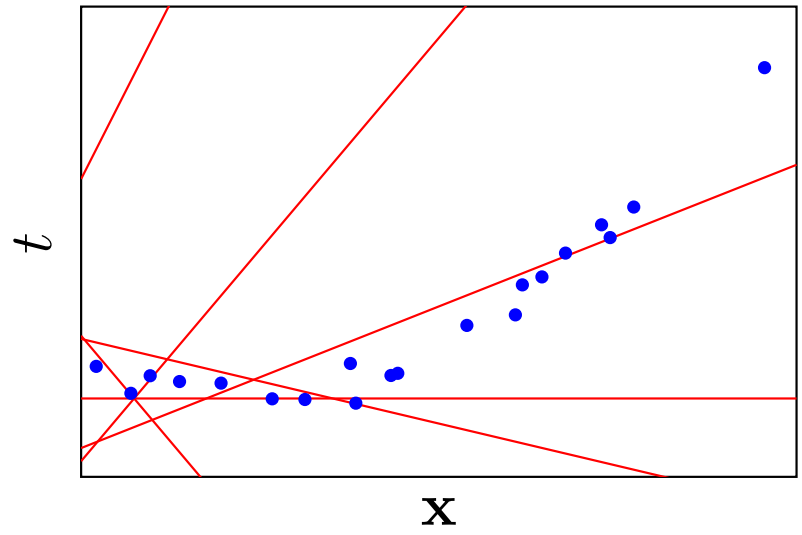
Linear Function

- Linear function in the **parameters** $\mathbf{w}$:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{D-1} w_j x_j = \mathbf{w}^T \mathbf{x}$$

- $\mathbf{x} = (1, x_1, \ldots, x_{D-1})$
- w_0 is the offset





Loss Functions for Regression

- We need to quantify what it means to do **well or poorly** on a task
- We need to define a **loss** (error) function: $L(t, y(\mathbf{x}))$
- The **average**, or expected, loss is given by:

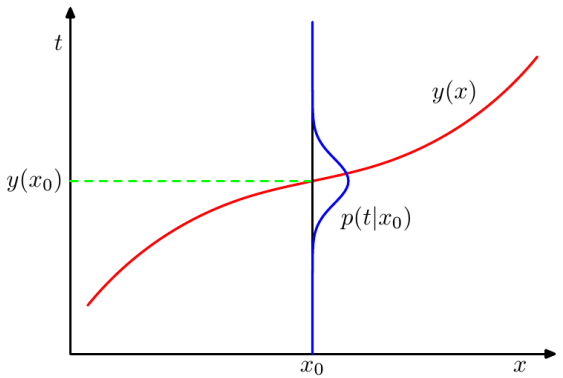
$$\mathbb{E}[L] = \int \int L(t, y(\mathbf{x}))p(\mathbf{x}, t)\mathrm{d}\mathbf{x}\mathrm{d}t$$

- A common choice is the **squared loss function**

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

- The optimal solution (if we assume a completely flexible function) is the **conditional average**:

$$y(\mathbf{x}) = \int tp(t|\mathbf{x})\mathrm{d}t = \mathbb{E}[t|\mathbf{x}]$$



Other Loss Functions

- Simple generalization of the squared loss, called the **Minkowski** loss:

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

- The minimum of $\mathbb{E}[L]$ is given by:
  - the conditional mean for $q = 2$
  - the conditional median for $q = 1$
  - the conditional mode for $q \to 0$

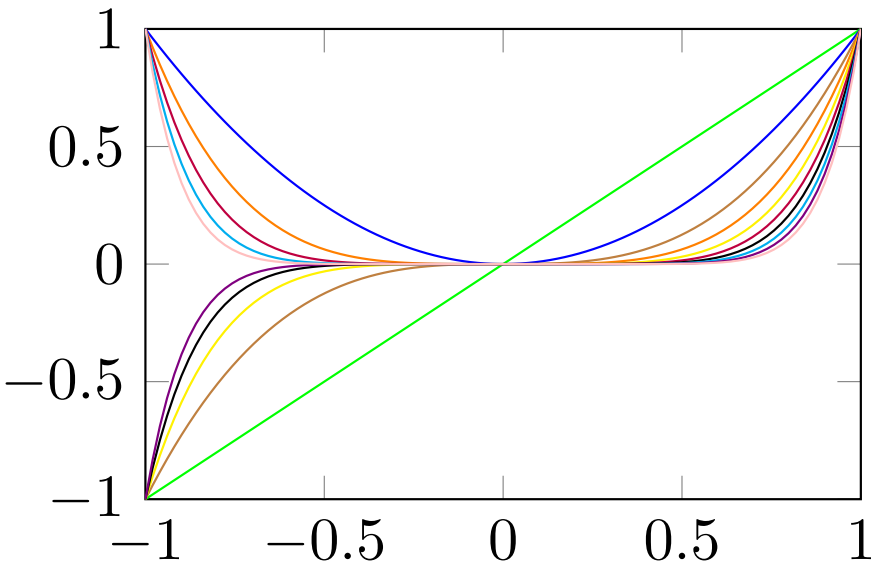
Basis Functions

- To consider non-linear functions we can use non-linear **basis function**:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$$

- $\boldsymbol{\phi}(\mathbf{x}) = (1, \phi_1(\mathbf{x}), \ldots, \phi_{M-1}(\mathbf{x}))^T$

- Examples:
  - Polynomial: $\phi_j(x) = x^j$
  - Gaussian: $\phi_j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2\sigma^2}\right)$
  - Sigmoidal: $\phi_j(x) = \frac{1}{1+\exp\left(\frac{\mu_j - x}{\sigma}\right)}$



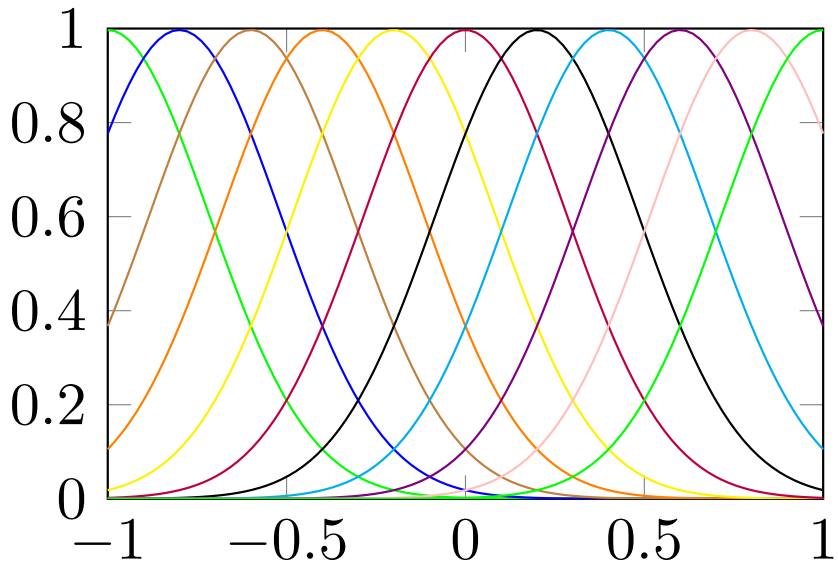
Basis Functions

- To consider non-linear functions we can use non-linear **basis function**:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$$

- $\boldsymbol{\phi}(\mathbf{x}) = (1, \phi_1(\mathbf{x}), \ldots, \phi_{M-1}(\mathbf{x}))^T$

- Examples:
  - Polynomial: $\phi_j(x) = x^j$
  - Gaussian: $\phi_j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2\sigma^2}\right)$
  - Sigmoidal: $\phi_j(x) = \frac{1}{1+\exp\left(\frac{\mu_j - x}{\sigma}\right)}$



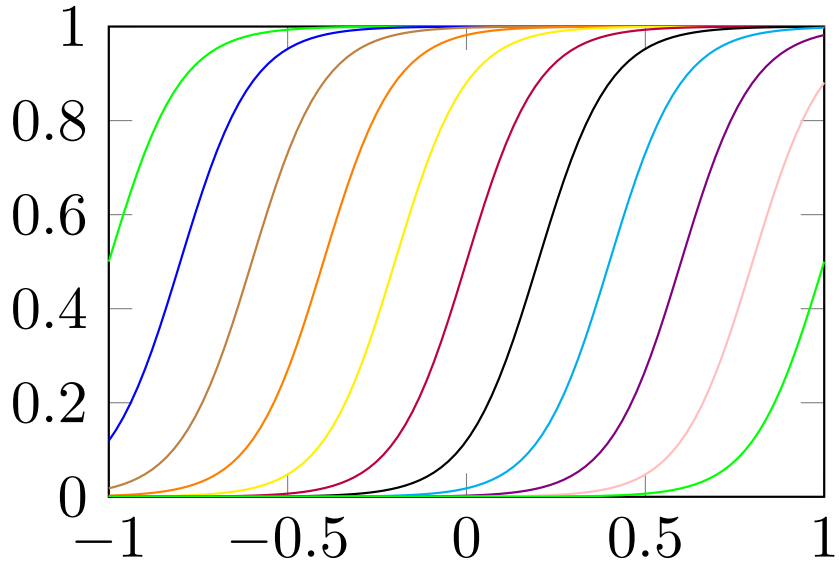
Basis Functions

- To consider non-linear functions we can use non-linear **basis function**:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$$

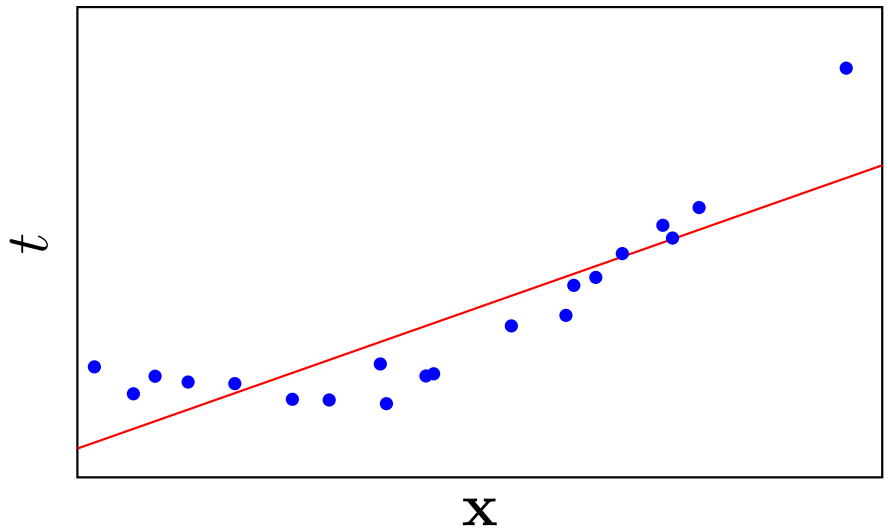
- $\boldsymbol{\phi}(\mathbf{x}) = (1, \phi_1(\mathbf{x}), \ldots, \phi_{M-1}(\mathbf{x}))^T$

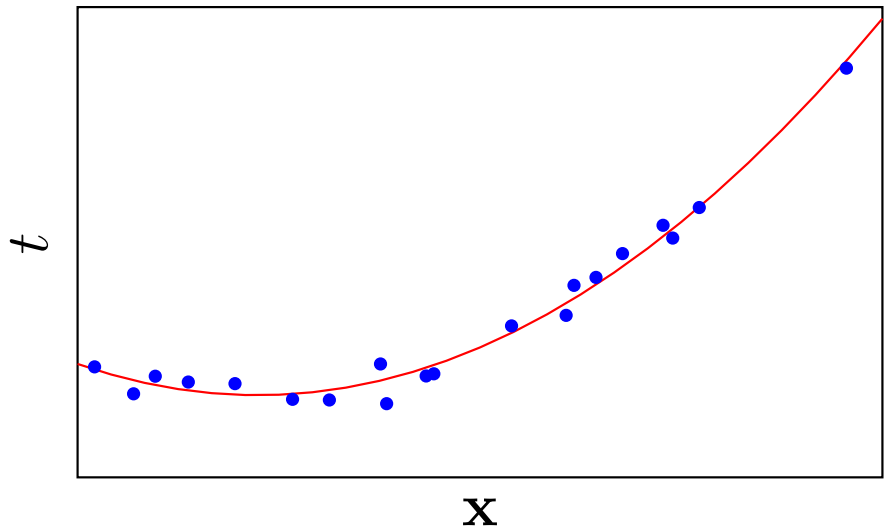
- Examples:
  - Polynomial: $\phi_j(x) = x^j$
  - Gaussian: $\phi_j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2\sigma^2}\right)$
  - Sigmoidal: $\phi_j(x) = \frac{1}{1+\exp\left(\frac{\mu_j - x}{\sigma}\right)}$

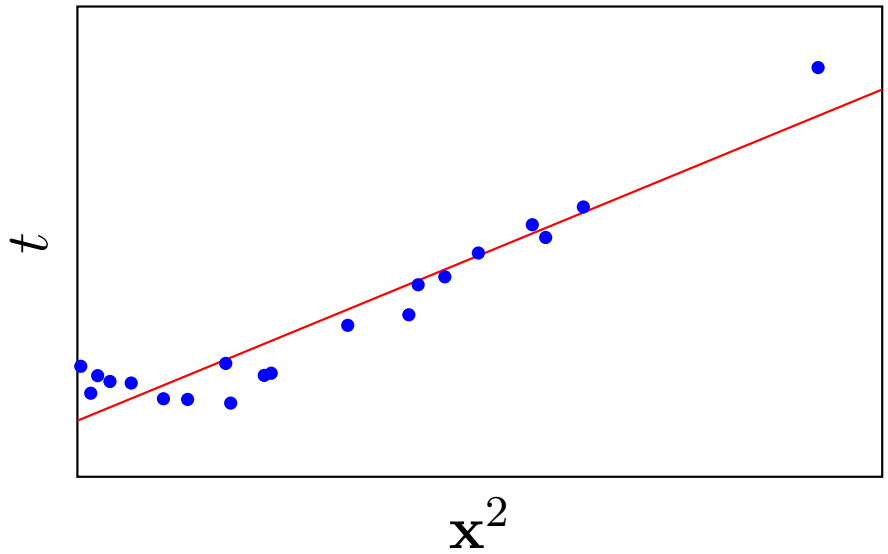


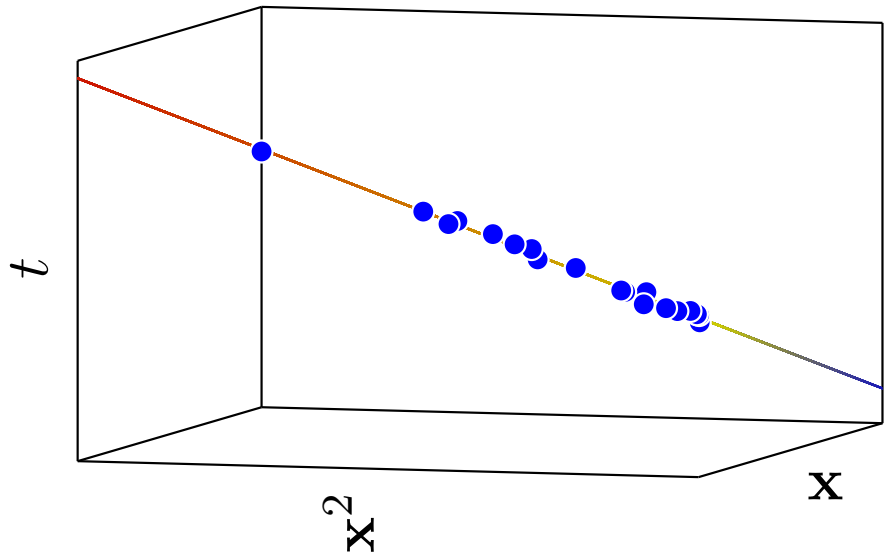
Basis Functions

Example









Discriminative vs Generative

- **Generative approach**:
  - Model the **joint density**: $p(\mathbf{x}, t) = p(\mathbf{x}|t)p(t)$
  - Infer **conditional density**: $p(t|\mathbf{x}) = \frac{p(\mathbf{x},t)}{p(\mathbf{x})}$
  - Marginalize to find **conditional mean**: $\mathbb{E}[t|\mathbf{x}] = \int t p(t|\mathbf{x}) \mathrm{d}t$
- **Discriminative approach**:
  - Model **conditional density** $p(t|\mathbf{x})$
  - Marginalize to find **conditional mean**: $\mathbb{E}[t|\mathbf{x}] = \int t p(t|\mathbf{x}) \mathrm{d}t$
- **Direct approach**
  - Find a regression function $y(\mathbf{x})$ directly from the training data

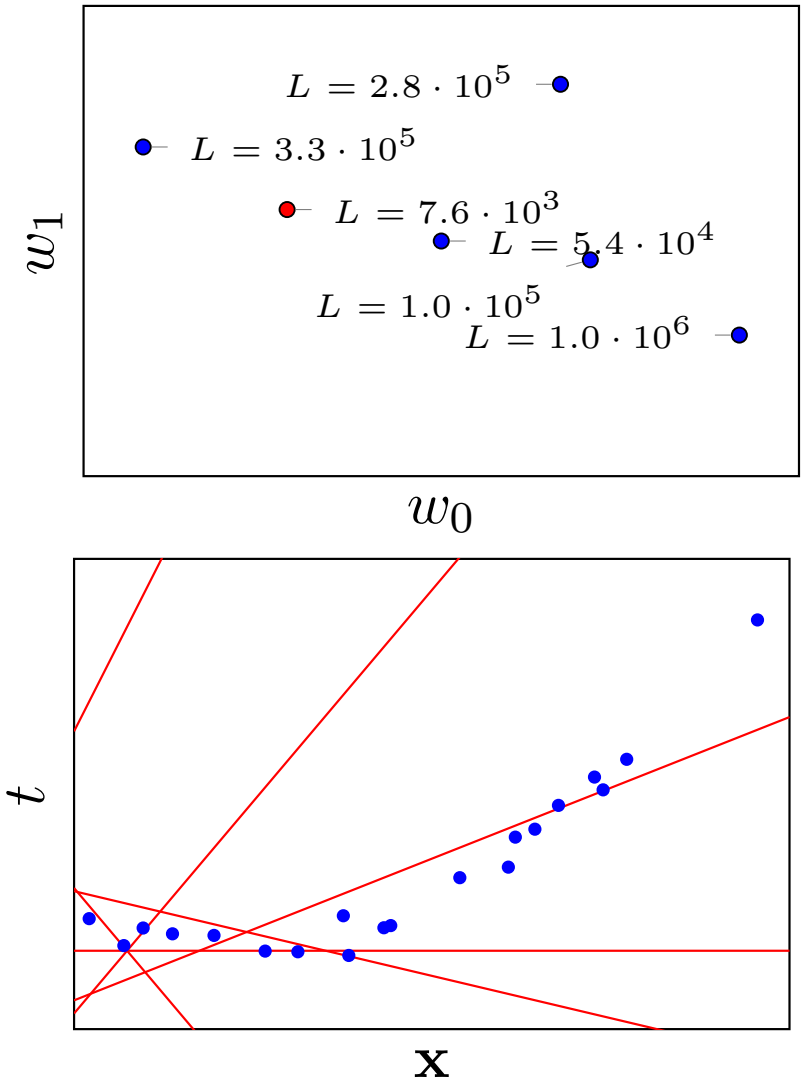
Minimizing Least Squares

- Given a data set with N samples, let us consider the following **error (loss) function**

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

- This is (half) the **residual sum of squares** (RSS), a.k.a. **sum of squared errors** (SSE)
- It can also be written as the sum of the ℓ_2-norm of the vector of **residual errors**

$$RSS(\mathbf{w}) = \|\boldsymbol{\epsilon}\|_2^2 = \sum_{i=1}^{N}\epsilon_i^2$$



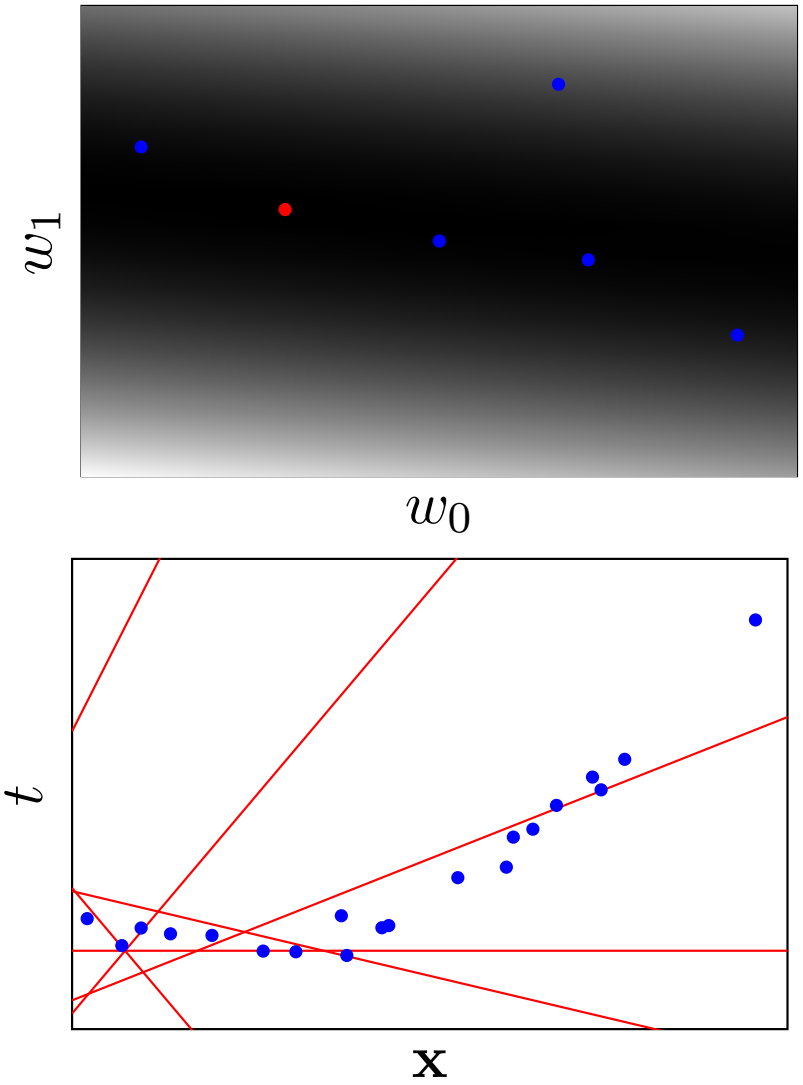
Minimizing Least Squares

- Given a data set with N samples, let us consider the following **error (loss) function**

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

- This is (half) the **residual sum of squares** (RSS), a.k.a. **sum of squared errors** (SSE)
- It can also be written as the sum of the ℓ_2-norm of the vector of **residual errors**

$$RSS(\mathbf{w}) = \|\boldsymbol{\epsilon}\|_2^2 = \sum_{i=1}^{N}\epsilon_i^2$$



Ordinary Least Squares

Closed-Form Optimization

- Let's write RSS in matrix form with $\mathbf{\Phi} = (\boldsymbol{\phi}(\mathbf{x}_1), \ldots, \boldsymbol{\phi}(\mathbf{x}_N))^T$ and $\mathbf{t} = (t_1, \ldots, t_N)^T$

$$L(\mathbf{w}) = \frac{1}{2} RSS(\mathbf{w}) = \frac{1}{2} (\mathbf{t} - \mathbf{\Phi}\mathbf{w})^T (\mathbf{t} - \mathbf{\Phi}\mathbf{w})$$

- Compute first and second derivative

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = -\mathbf{\Phi}^T (\mathbf{t} - \mathbf{\Phi}\mathbf{w}); \quad \frac{\partial^2 L(\mathbf{w})}{\partial \mathbf{w} \partial \mathbf{w}^T} = \mathbf{\Phi}^T \mathbf{\Phi}$$

- Assuming $\mathbf{\Phi}^T \mathbf{\Phi}$ in **nonsingular**

$$\hat{\mathbf{w}}_{OLS} = \left(\mathbf{\Phi}^T \mathbf{\Phi}\right)^{-1} \mathbf{\Phi}^T \mathbf{t}$$

- Complexity $O(NM^2 + M^3)$
  - Cholesky: $M^3 + NM^2/2$
  - QR: NM^2

Gradient Optimization

- Closed-form solution is not practical with **big data**
- We can use **sequential (online)** updates
- **Stochastic gradient descent**
  - If the loss function can be expressed as a **sum over samples** ($L(\mathbf{x}) = \sum_n L(x_n)$)

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \alpha^{(k)} \nabla L(x_n)$$

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \alpha^{(k)} \left({\mathbf{w}^{(k)}}^T \boldsymbol{\phi}(\mathbf{x}_n) - t_n \right) \boldsymbol{\phi}(\mathbf{x}_n)$$

  - where k is the iteration and α is a **learning rate**
  - For **convergence** the learning rate has to satisfy

$$\sum_{k=0}^{\infty} \alpha^{(k)} = +\infty$$

$$\sum_{k=0}^{\infty} {\alpha^{(k)}}^2 < +\infty$$

Geometric Interpretation

- $\mathbf{t}$ is an N-dimensional vector
- Let's denote with $\boldsymbol{\varphi}_j$ the j-th column of $\boldsymbol{\Phi}$
- Define $\hat{\mathbf{t}}$ the N-dimensional vector, whose n-th element is $y(\mathbf{x}_n, \mathbf{w})$
- $\hat{\mathbf{t}}$ is a linear combination of $\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_M$
- so $\hat{\mathbf{t}}$ lies in an M-subspace $\mathcal{S}$
- Since $\hat{\mathbf{t}}$ minimizes the SSE with respect to $\mathbf{t}$, it represents the projections of $\mathbf{t}$ onto the subspace $\mathcal{S}$

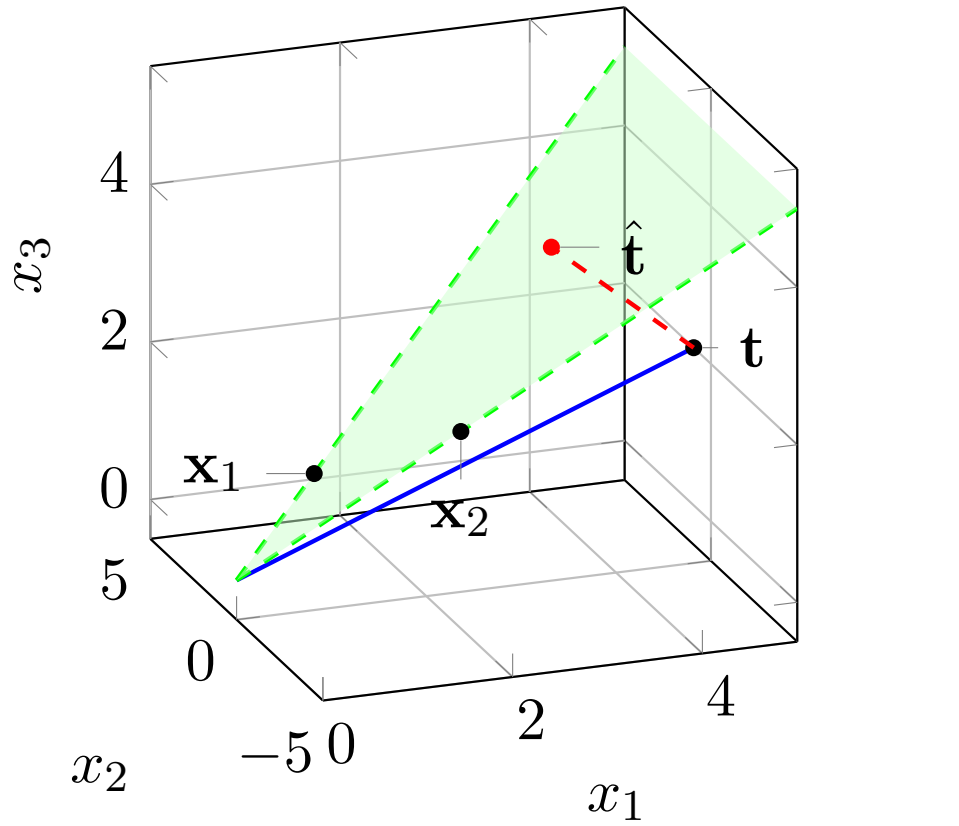
$$\hat{\mathbf{t}} = \boldsymbol{\Phi}\hat{\mathbf{w}} = \boldsymbol{\Phi}\left(\boldsymbol{\Phi}^T\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^T\mathbf{t}$$

- $H = \boldsymbol{\Phi}\left(\boldsymbol{\Phi}^T\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^T$ is called the **hat matrix**

Geometric Example

- Assume $N = 3$ and $M = D = 2$

$$\mathbf{\Phi} = \mathbf{X} = \begin{pmatrix} 1 & 2 \\ 1 & -2 \\ 1 & 2 \end{pmatrix} \quad \mathbf{t} = \begin{pmatrix} 5 \\ 1 \\ 2 \end{pmatrix} \quad \hat{\mathbf{t}} = \begin{pmatrix} 3.5 \\ 1 \\ 3.5 \end{pmatrix}$$



Maximum Likelihood (ML)

- The output variable t can be modeled as a deterministic function y of the input $\mathbf{x}$ and a random noise ϵ

$$t = f(\mathbf{x}) + \epsilon$$

- We want to approximate $f(\mathbf{x})$ with $y(\mathbf{x}, \mathbf{w})$
- We assume $\epsilon \sim \mathcal{N}\left(0, \sigma^2\right)$
- Given N samples, with inputs $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and outputs $\mathbf{t} = (t_1, \ldots, t_N)^T$, the likelihood function is

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}\left(t_n|\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n), \sigma^2\right)$$

Maximum Likelihood (ML)

- Assuming the samples to be **independent and identically distributed (iid)**, we consider the log-likelihood:

$$\ell(\mathbf{w}) = \ln p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \sigma^2) = \sum_{n=1}^{N} \ln p(t_n|\mathbf{x}_n, \mathbf{w}, \sigma^2)$$
$$= -\frac{N}{2} \ln\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2} RSS(\mathbf{w})$$

- To find the maximum likelihood, we equal the gradient to zero

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

Variance of the Parameters

- We assume
  - the observation t_i are **uncorrelated** and have **constant variance** σ^2
  - the x_i are fixed (non random)
- The variance-covariance matrix of the least-squares estimates is

$$\text{Var}(\hat{\mathbf{w}}_{OLS}) = \left(\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\sigma^2$$

- Usually, the variance σ^2 is estimated by

$$\hat{\sigma^2} = \frac{1}{N-M}\sum_{n=1}^{N}(t_n - \hat{\mathbf{w}}^T\boldsymbol{\phi}(\mathbf{x}_n))^2$$

- Assuming that the model is linear in the features $\phi_1(), \ldots, \phi_M()$ and that the noise is additive and Gaussian

$$\hat{\mathbf{w}} \sim \mathcal{N}\left(\mathbf{w}, (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\sigma^2\right) \quad (N-M)\hat{\sigma^2} \sim \sigma^2\chi^2_{N-M}$$

- Such properties can be used to form **test hypothesis** and **confidence intervals**

Gauss-Markov Theorem

Theorem (Gauss-Markov)

The least squares estimate of $\mathbf{w}$ *has the* ***smallest variance*** *among all linear* ***unbiased*** *estimates.*

- It follows that least squares estimator has the **lowest MSE** of all linear estimator with **no bias**
- However, there may exist a **biased** estimator with **smaller MSE**

Multiple Outputs

- Let now consider the case of **multiple outputs**
- We could use a **different** set of basis functions for each output, thus having **independent** regression problems
- Usually, a **single** set of basis functions is considered

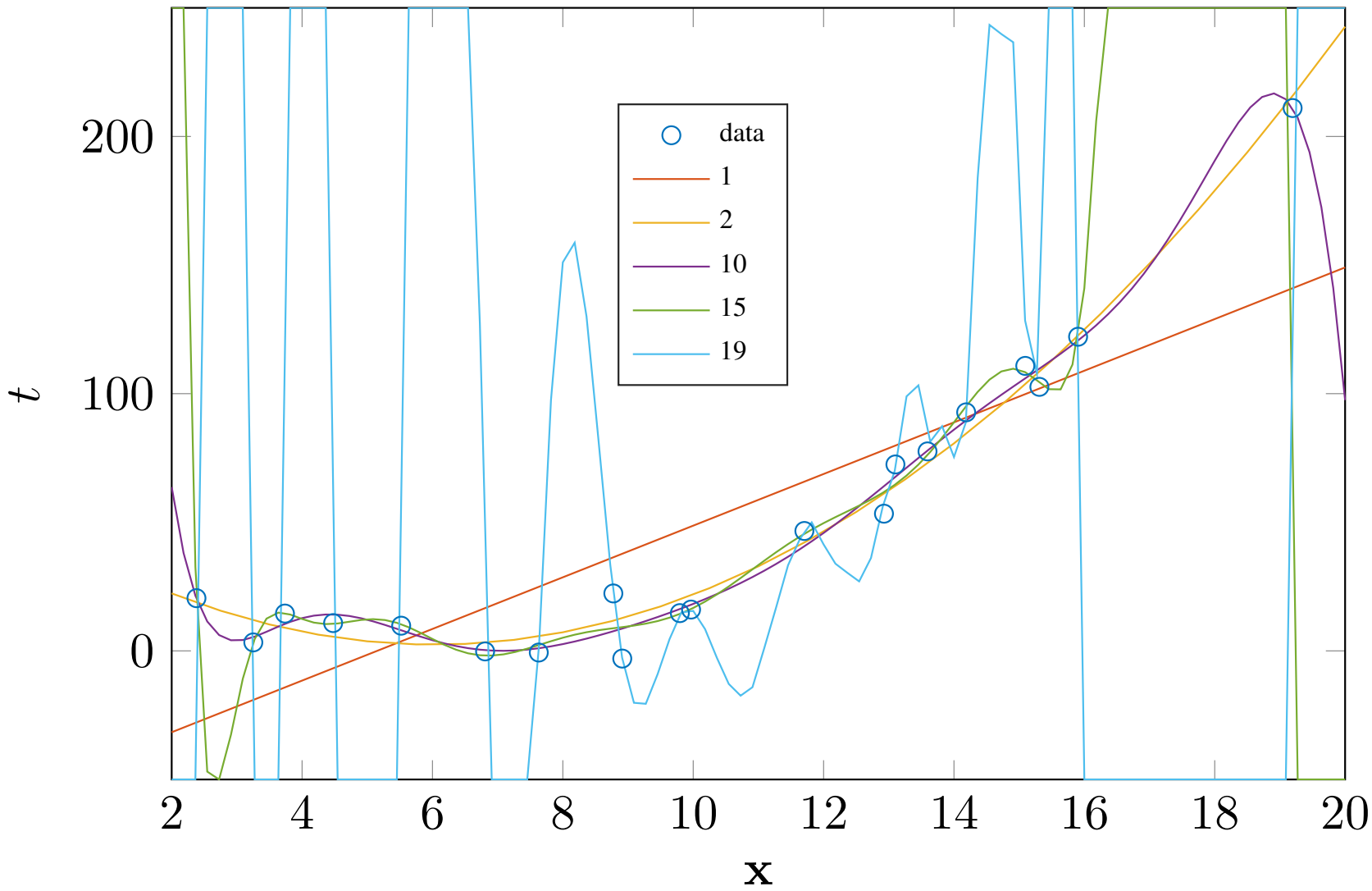
$$\hat{\mathbf{W}}_{ML} = \left(\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\mathbf{T}$$

- For each output t_k we have

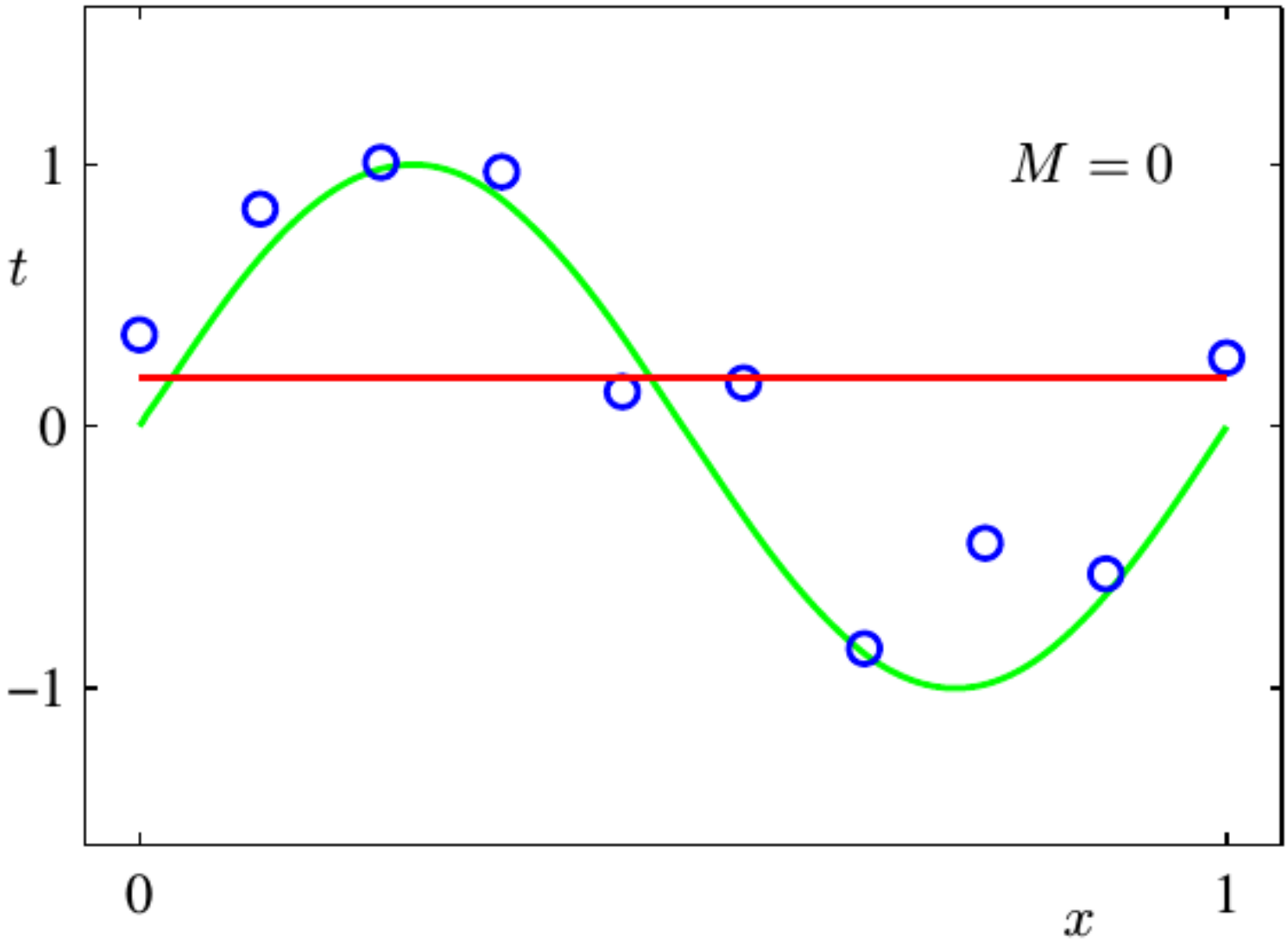
$$\hat{\mathbf{w}}_k = \left(\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\mathbf{t}_k$$

- where $\mathbf{t}_k$ is an N-dimensional column vector
- The solution **decouples** between the different outputs
- The pseudo-inverse $\mathbf{\Phi}^\dagger = \left(\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T$ needs to be computed only **once**

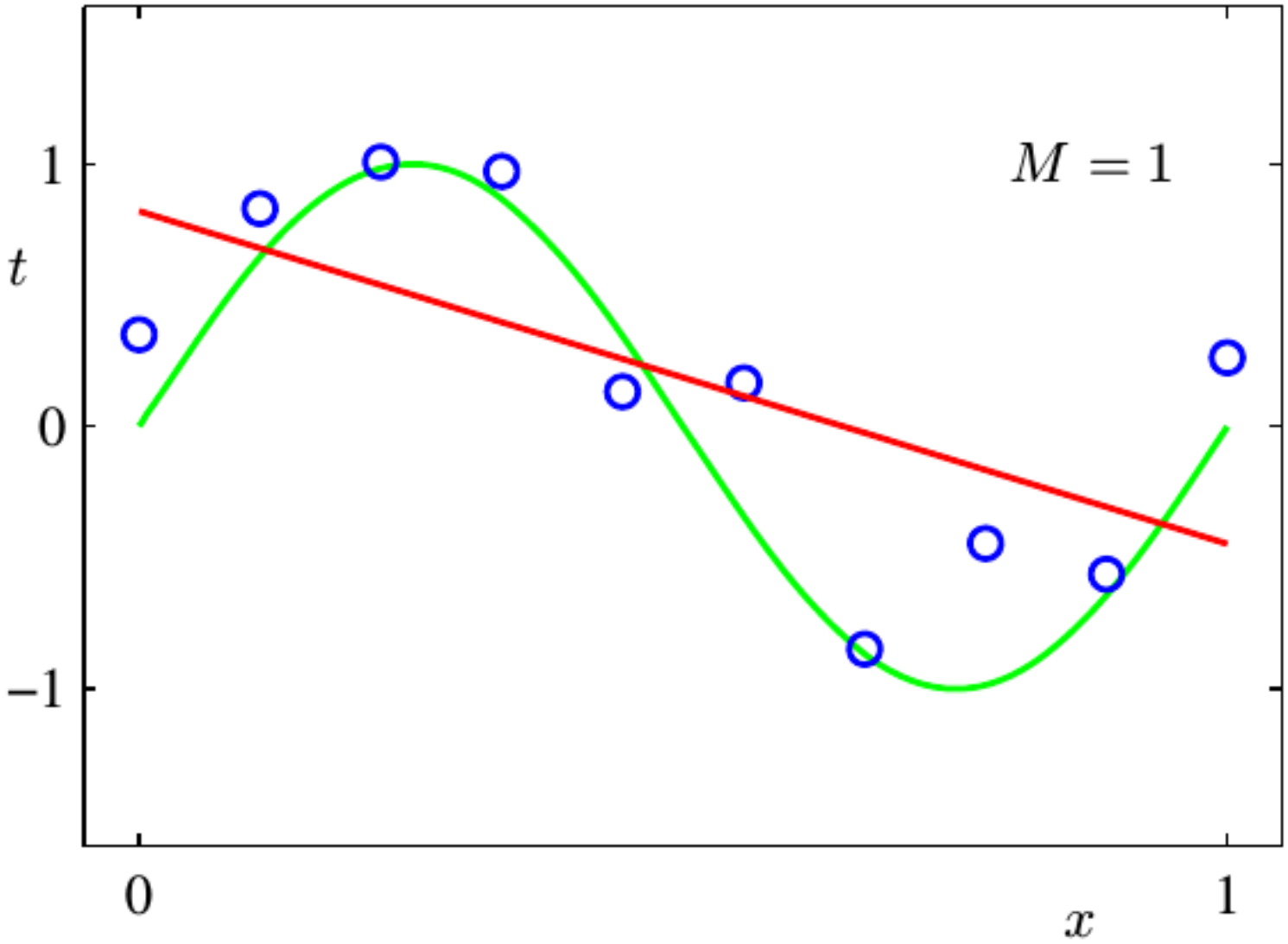
Increasing Model Complexity: Quadratic Function



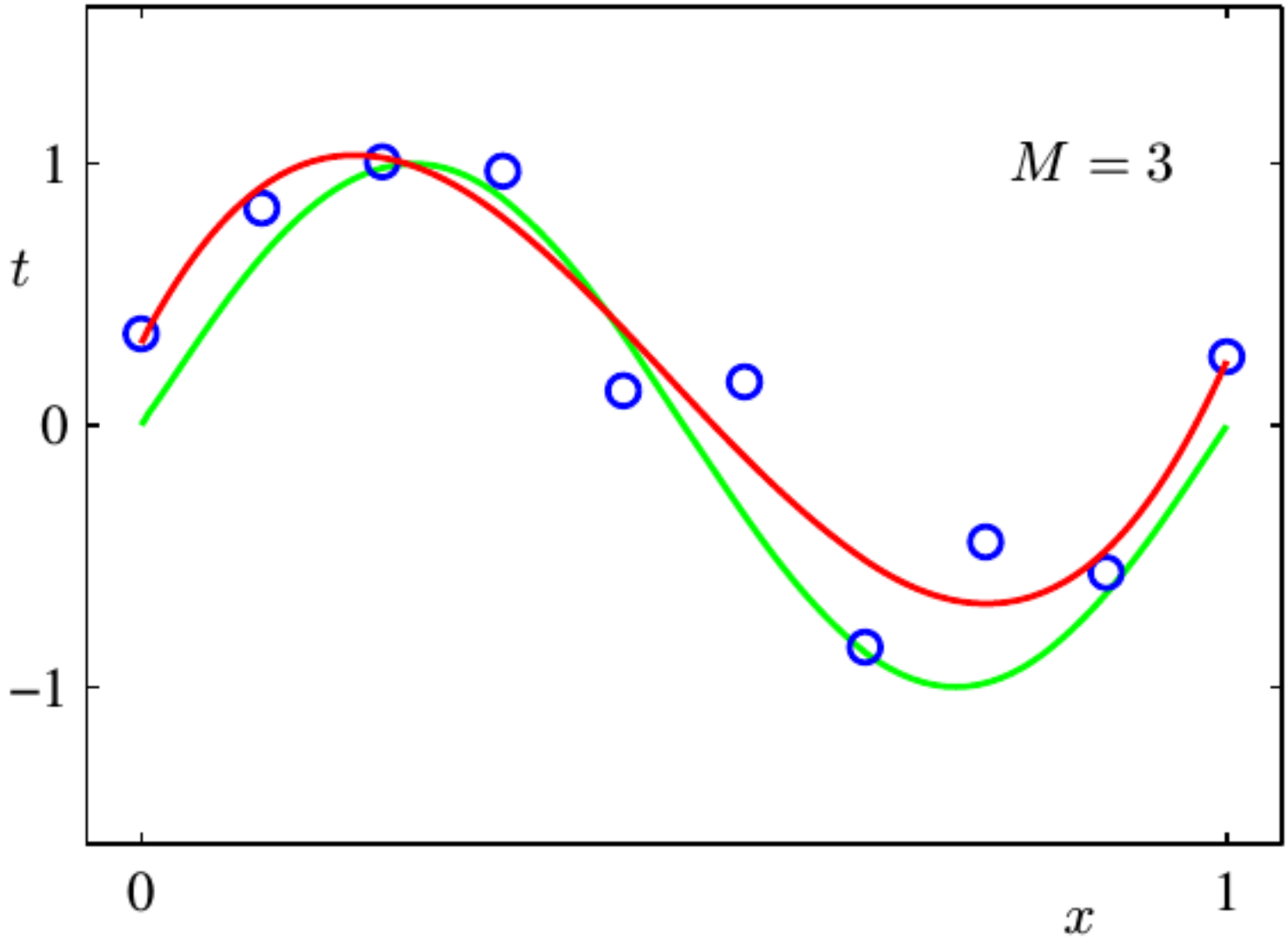
Increasing Model Complexity: Sinusoidal Function



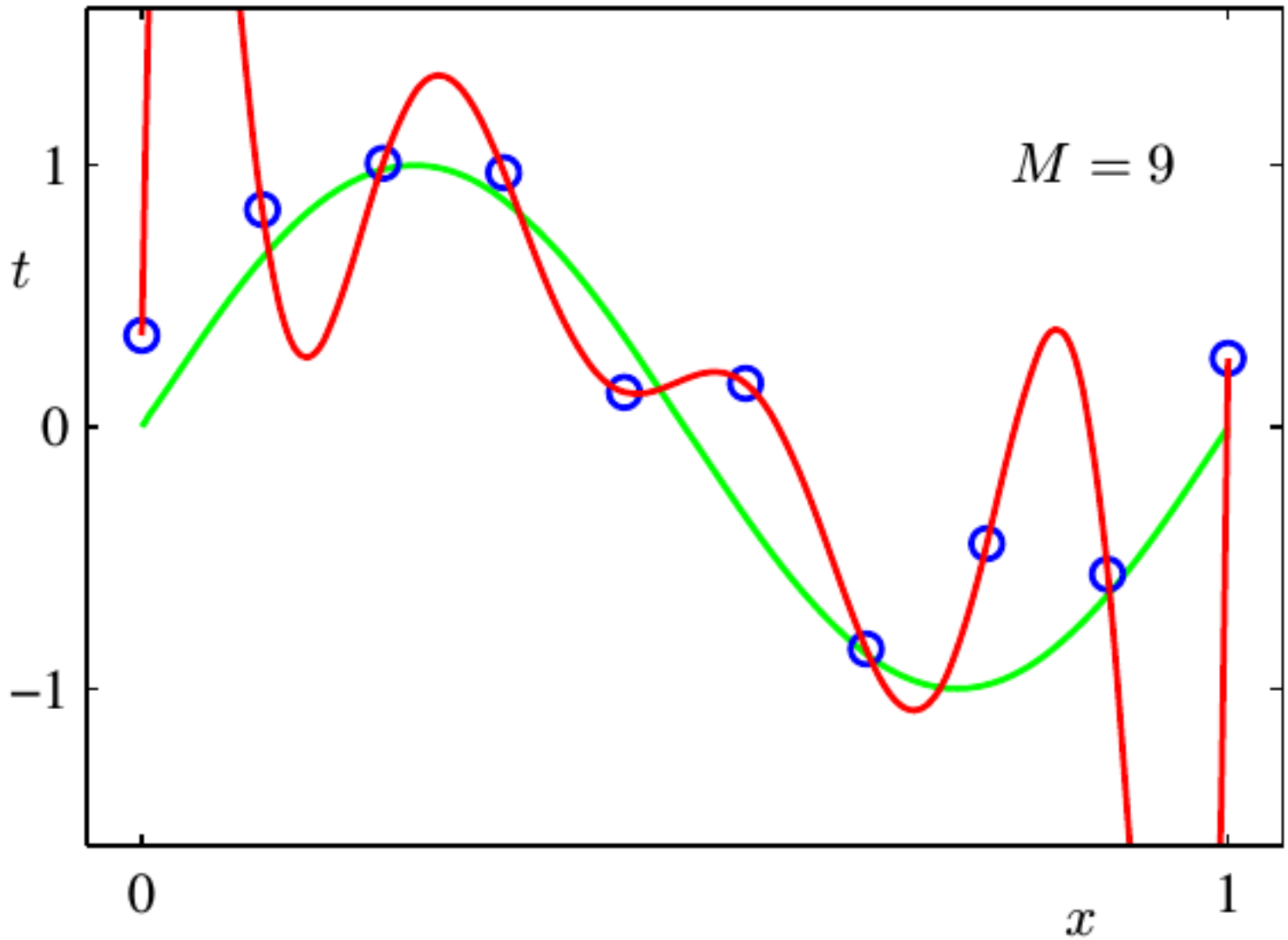
Increasing Model Complexity: Sinusoidal Function



Increasing Model Complexity: Sinusoidal Function



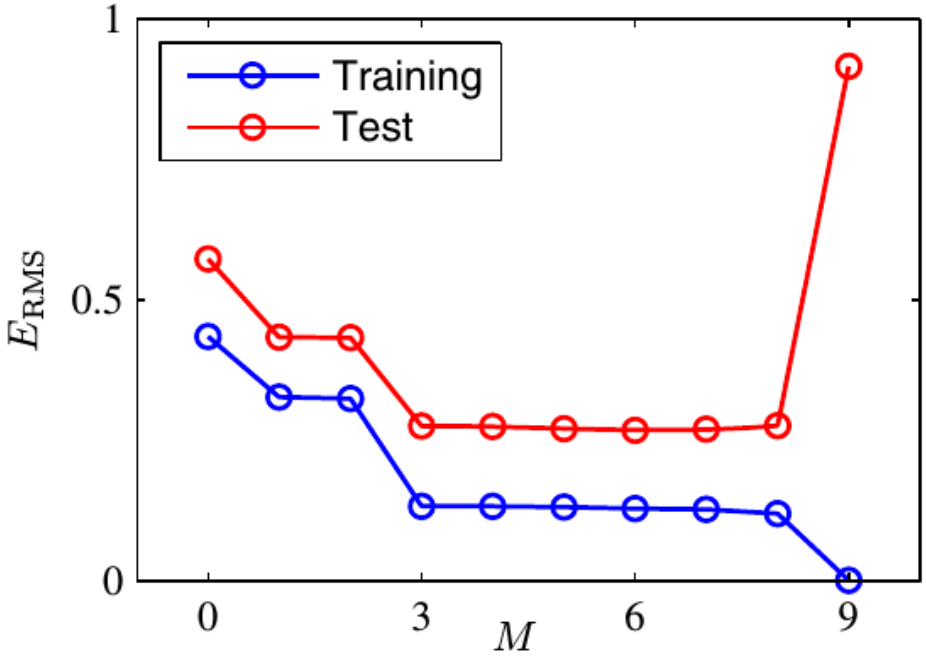
Increasing Model Complexity: Sinusoidal Function



Under-fitting vs Over-Fitting

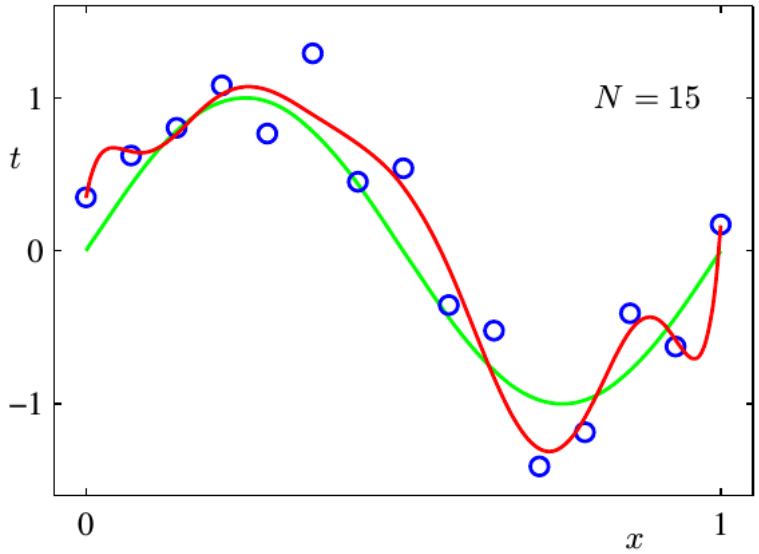
- With low-order polynomials we have **under-fitting**
- With high-order polynomials we get excellent fit over the training data, but a poor representation of the true function: **over-fitting**
- We want to have good **generalization**

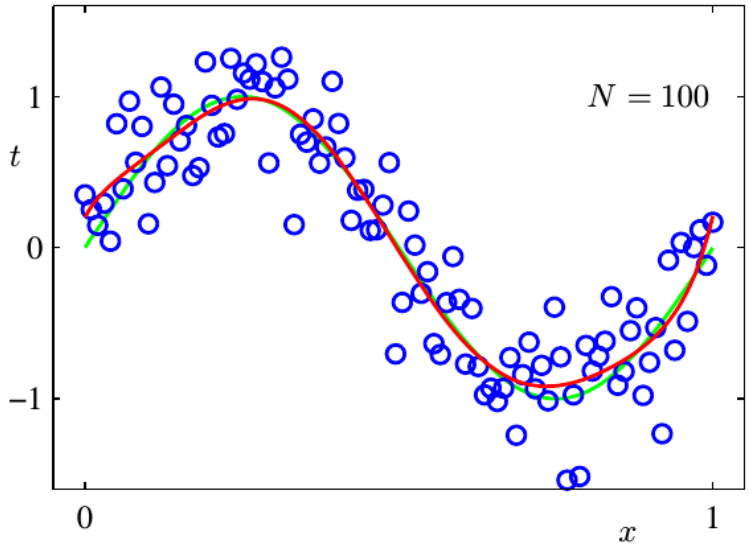
- We use a **test set** of 100 samples to evaluate generalization
- $E_{RMS} = \sqrt{\frac{2*RSS(\hat{\mathbf{w}})}{N}}$



How to Avoid Over-fitting?

- This is the problem of **model selection** (we will see this later)
- What happens when the number of training samples **increases**?





How to Avoid Over-fitting?

What happens to the parameters when the model gets more complex?

	$M=0$	$M=1$	$M=3$	$M=9$
$\hat{w}_0$	0.19	0.82	0.31	0.35
$\hat{w}_1$		-1.27	7.99	232.37
$\hat{w}_2$			-25.43	-5321.83
$\hat{w}_3$				48568.31
$\hat{w}_4$				-231639.30
$\hat{w}_5$				640042.26
$\hat{w}_6$				-1061800.52
$\hat{w}_7$				1042400.18
$\hat{w}_8$				-557682.99
$\hat{w}_9$				125201.43

Ridge Regression

- One way to reduce the MSE is to change the **loss function** as follows

$$L(\mathbf{w}) = L_D(\mathbf{w}) + \lambda L_W(\mathbf{w})$$

- $L_D(\mathbf{w})$: error on data (e.g., RSS)
- $L_W(\mathbf{w})$: model complexity
- By taking $L_W(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} = \frac{1}{2}\|\mathbf{w}\|_2^2$ we get

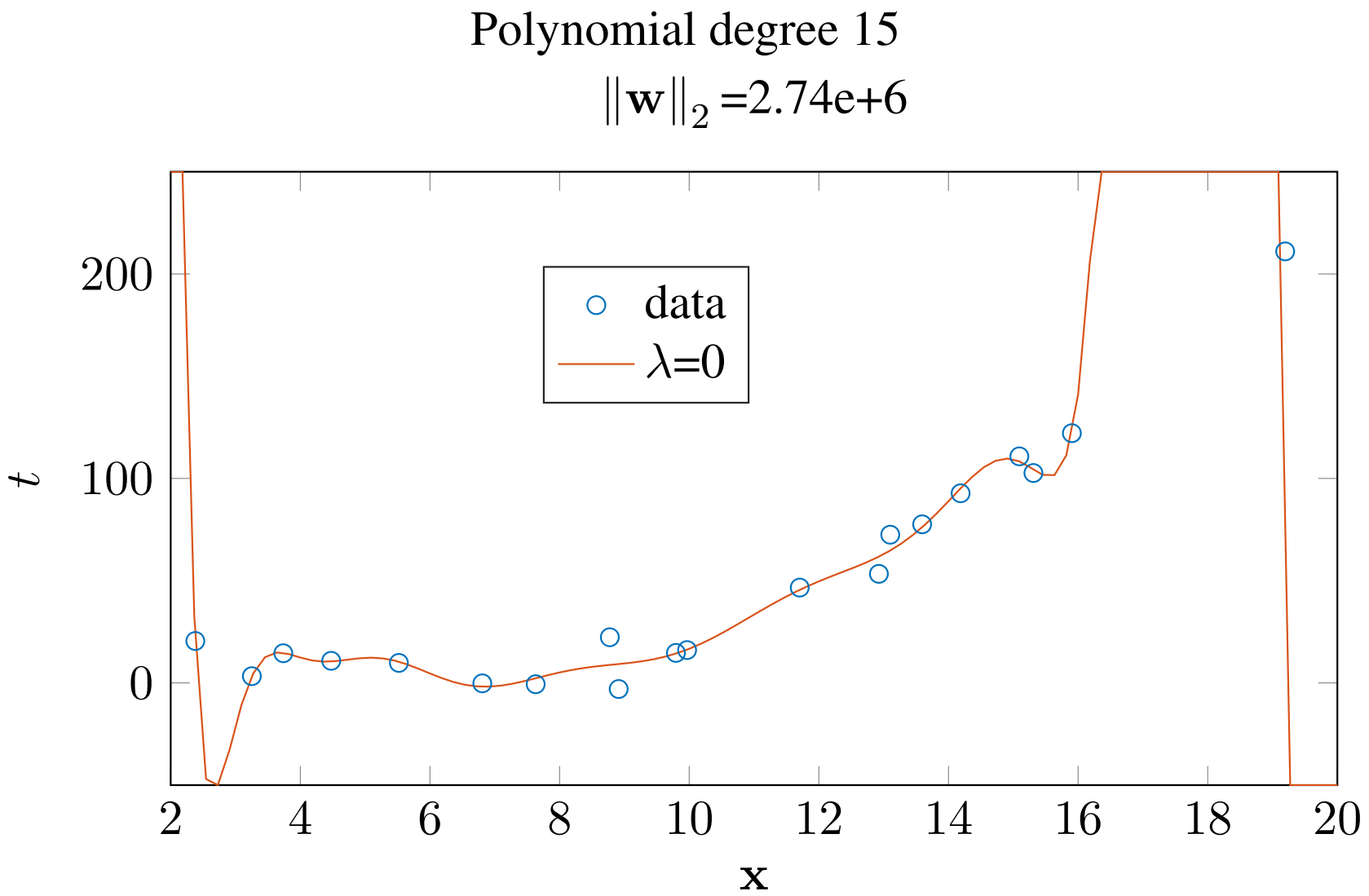
$$L(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\left(t_i - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i)\right)^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

- It is called **ridge regression** (or weight decay)
- It is a **regularization** (or parameter shrinkage) method
- The loss function is still quadratic in $\mathbf{w}$:

$$\hat{\mathbf{w}}_{ridge} = \left(\lambda\mathbf{I} + \boldsymbol{\Phi}^T\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^T\mathbf{t}$$

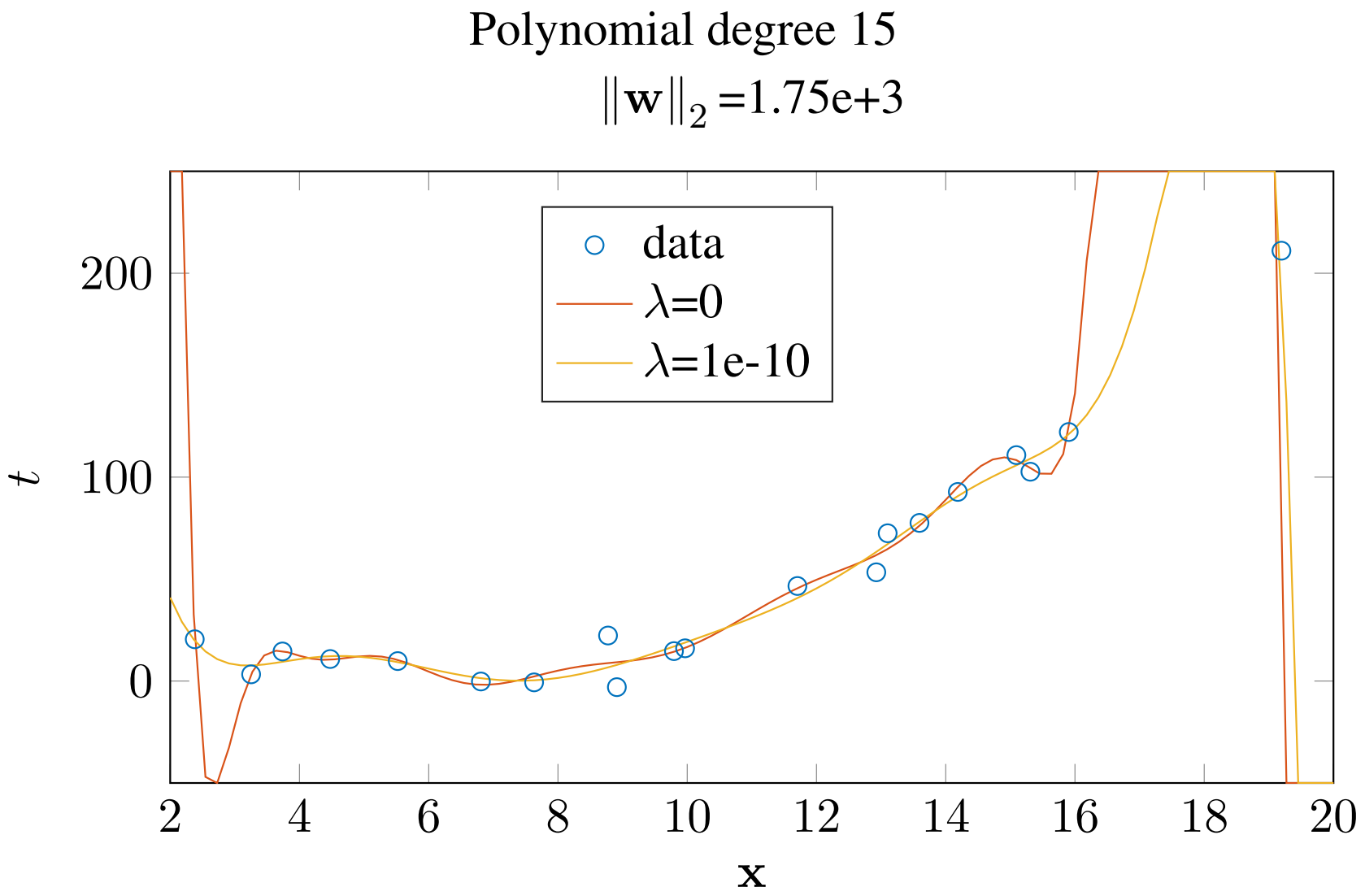
Ridge Regression

Quadratic Example



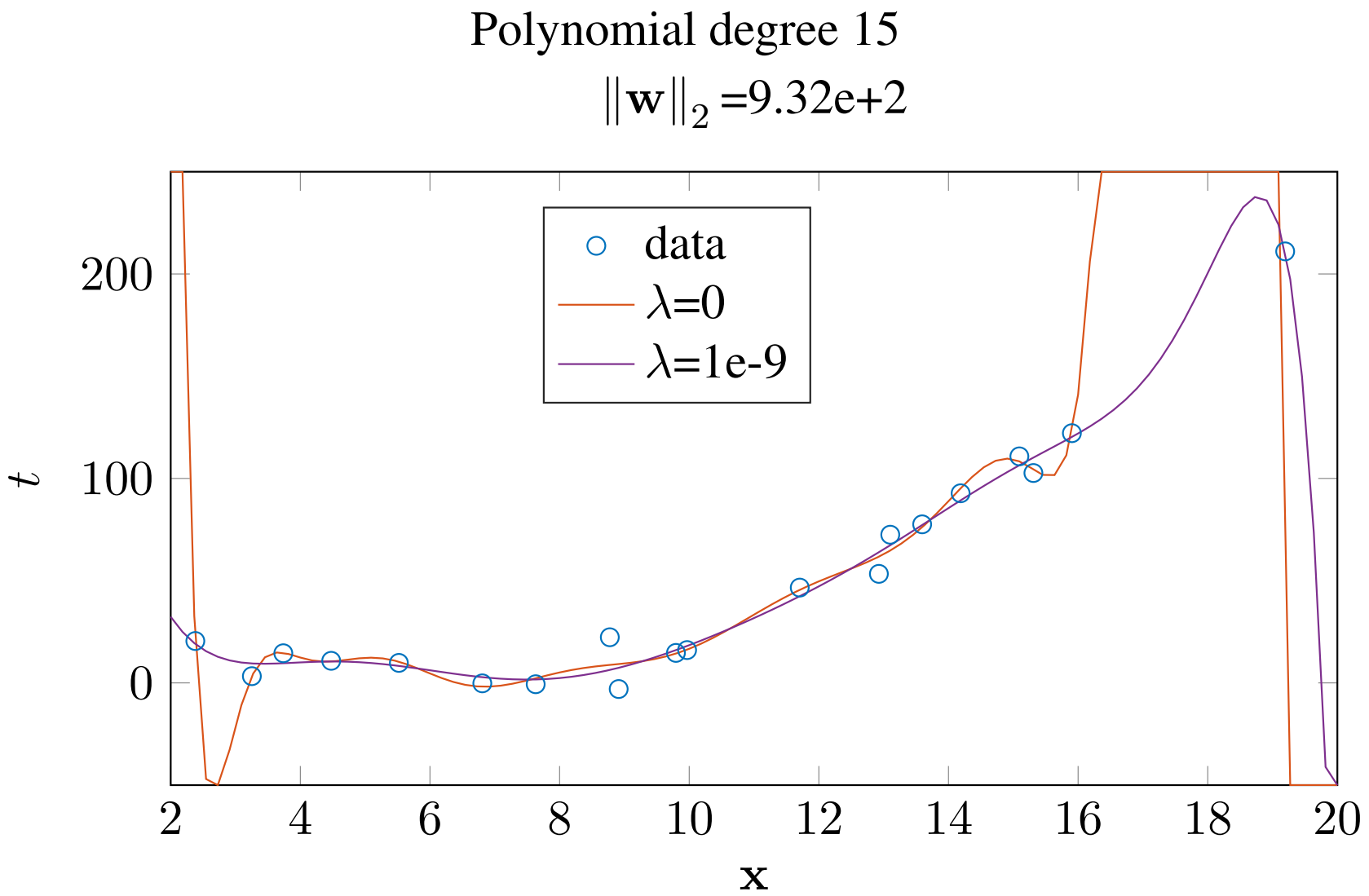
Ridge Regression

Quadratic Example



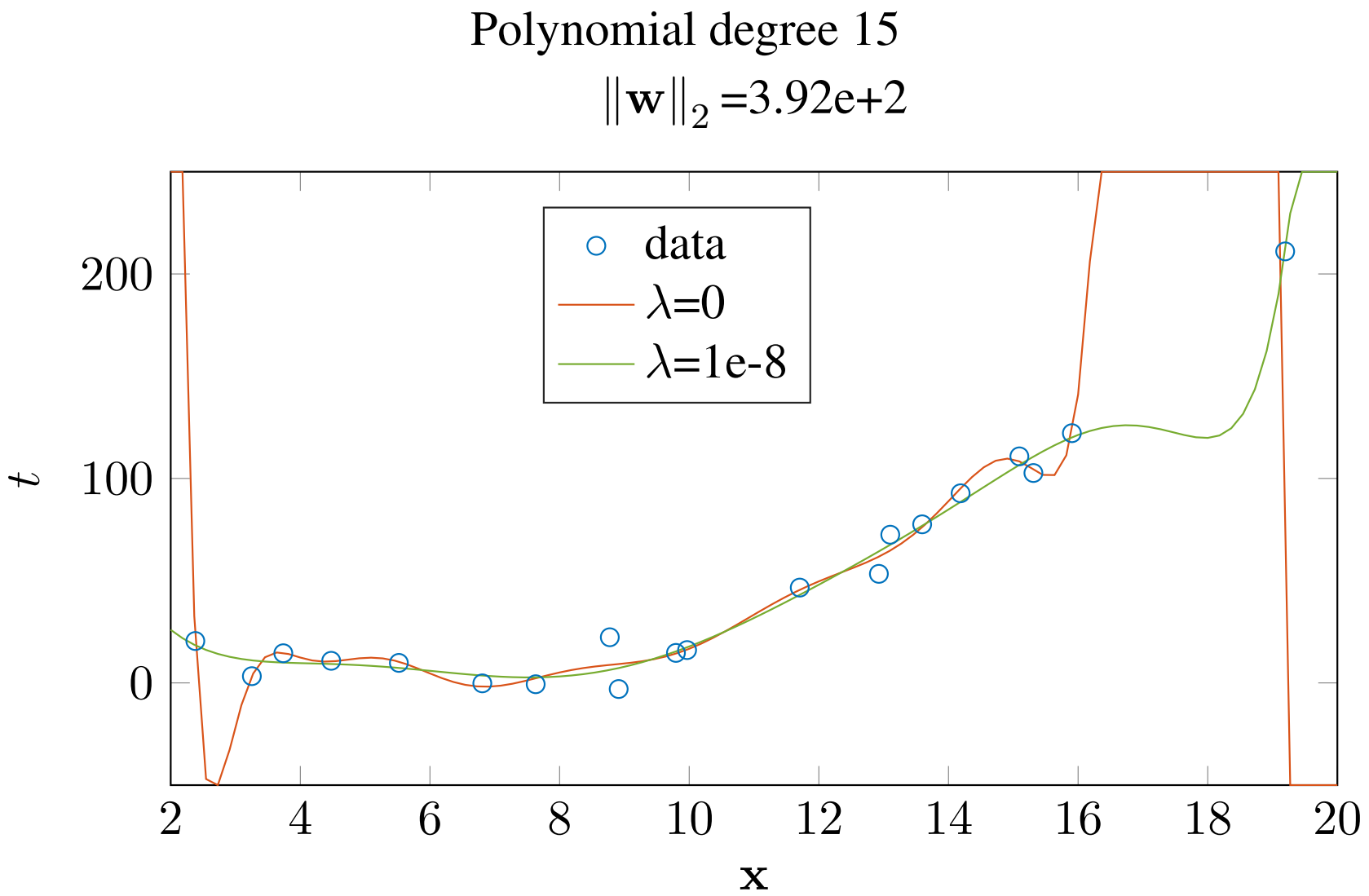
Ridge Regression

Quadratic Example



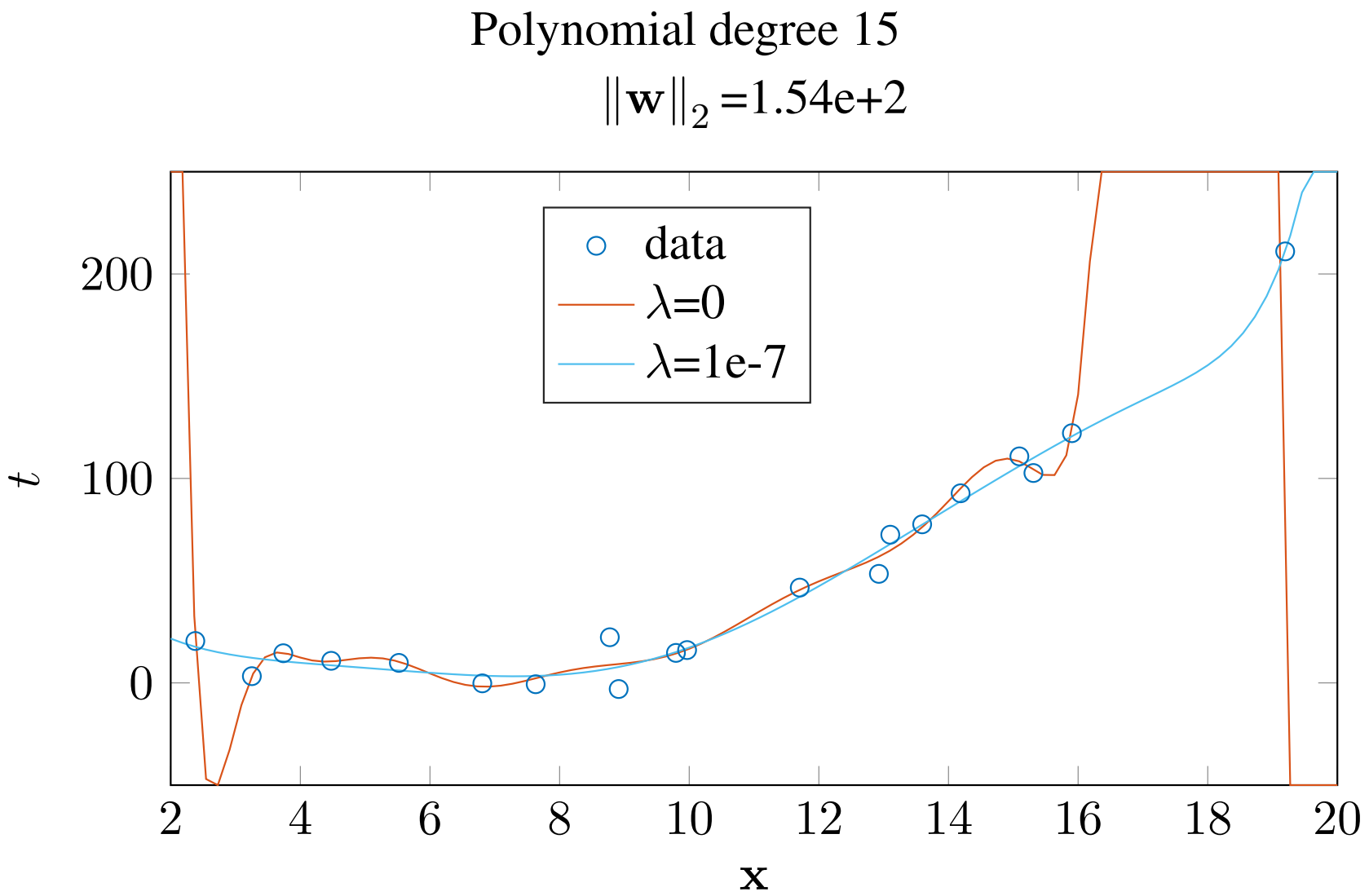
Ridge Regression

Quadratic Example



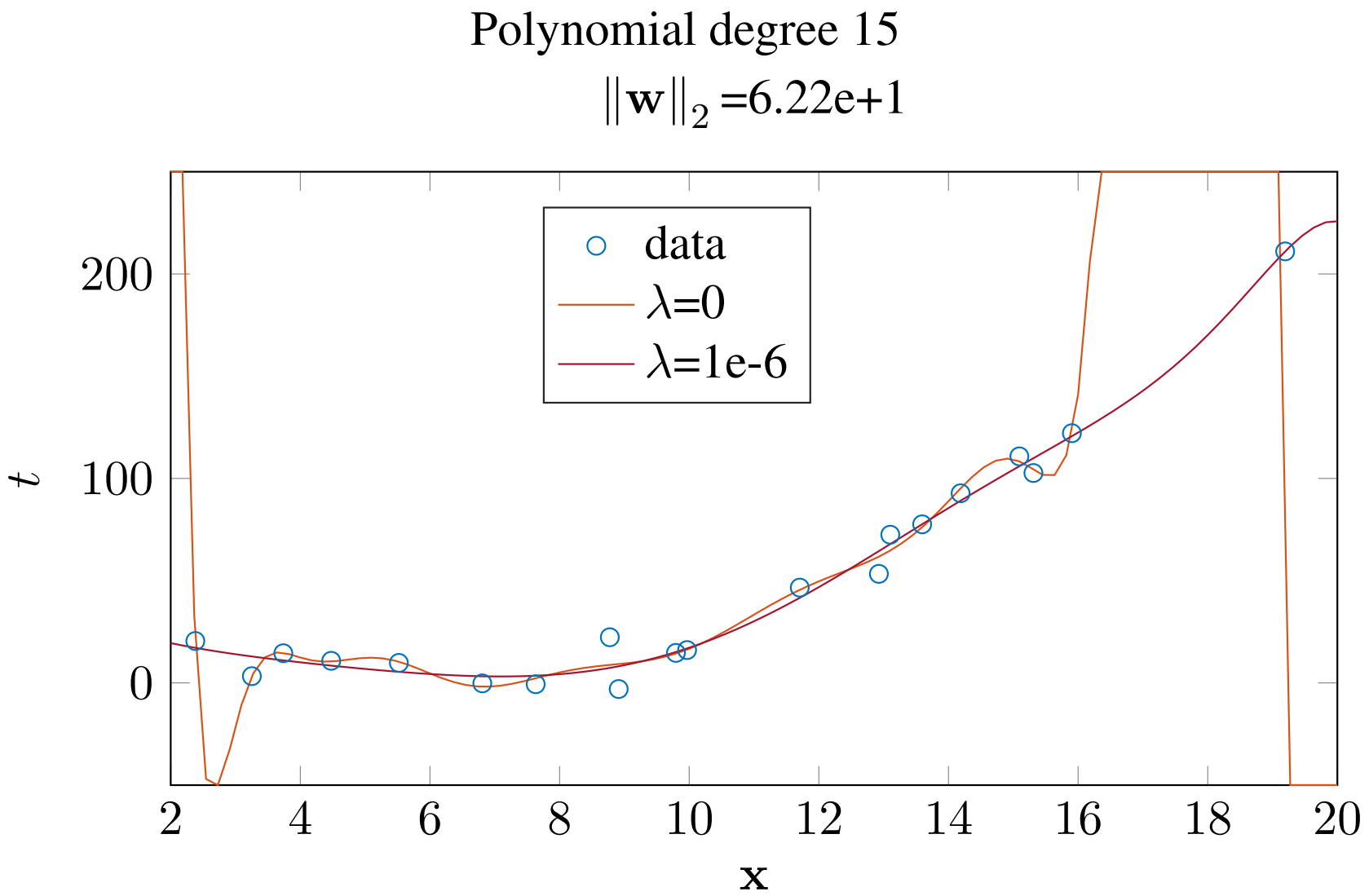
Ridge Regression

Quadratic Example



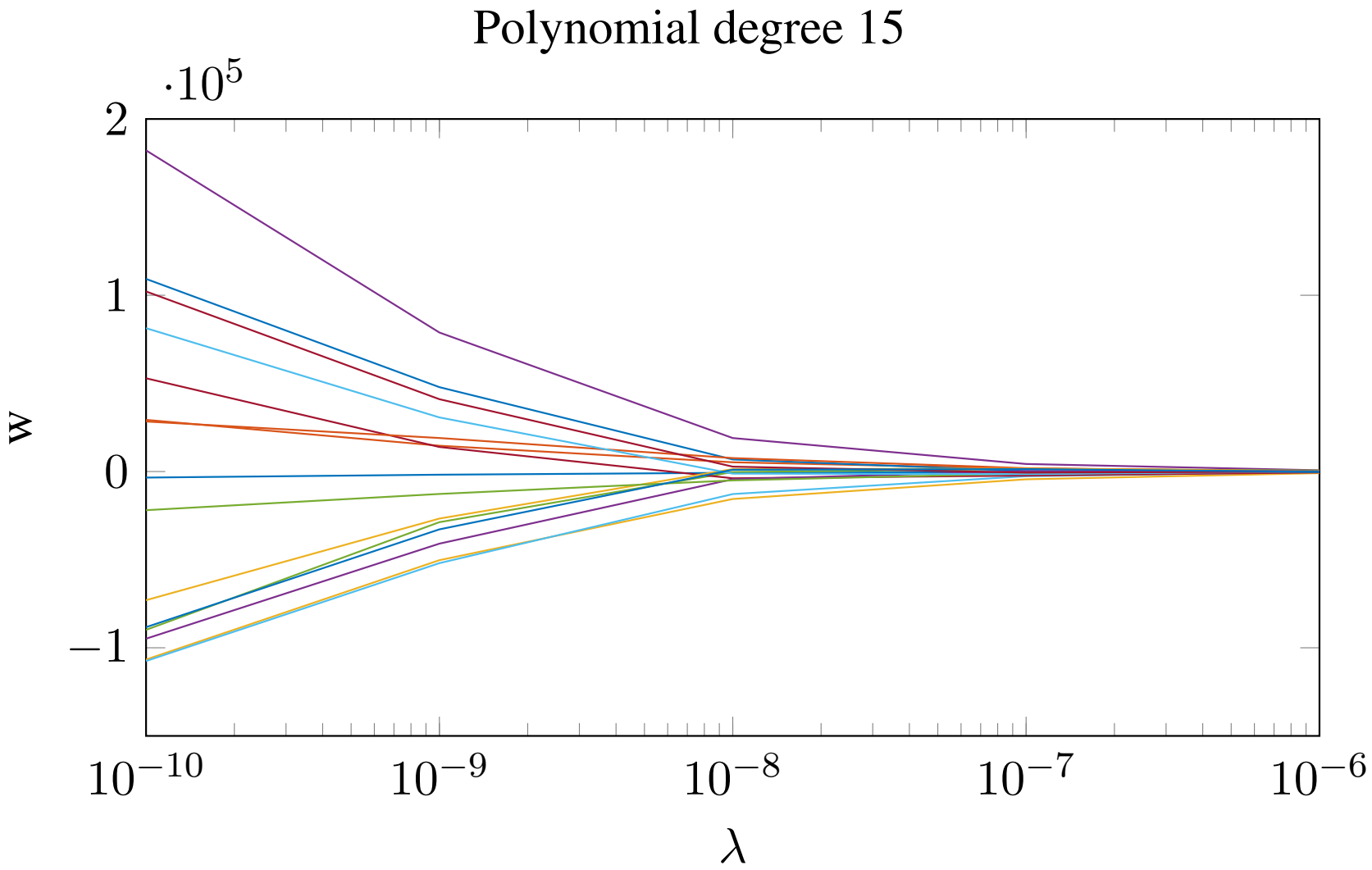
Ridge Regression

Quadratic Example



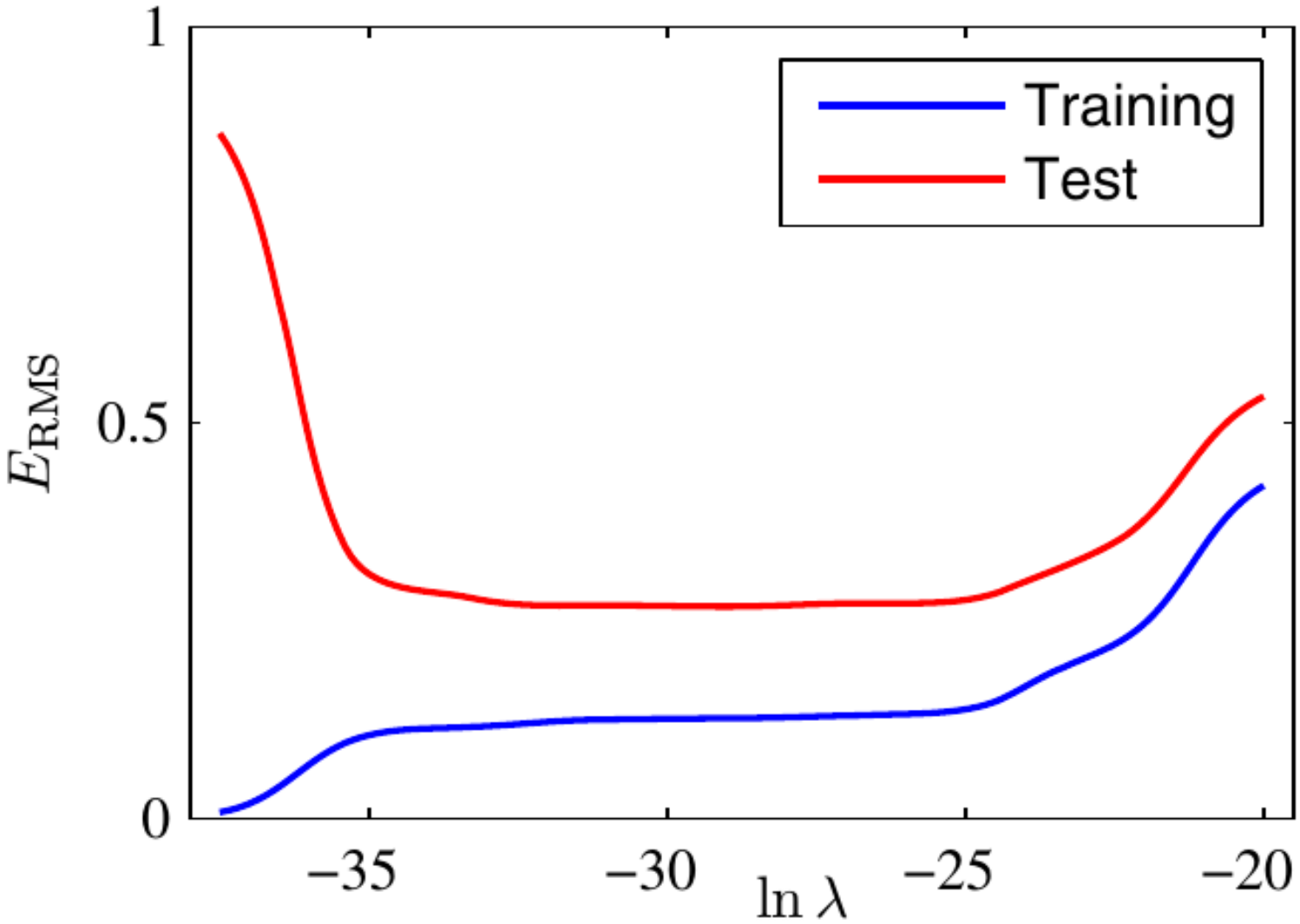
Ridge Regression

Quadratic Example: Weights



Ridge Regression

Sinusoidal Example



Lasso

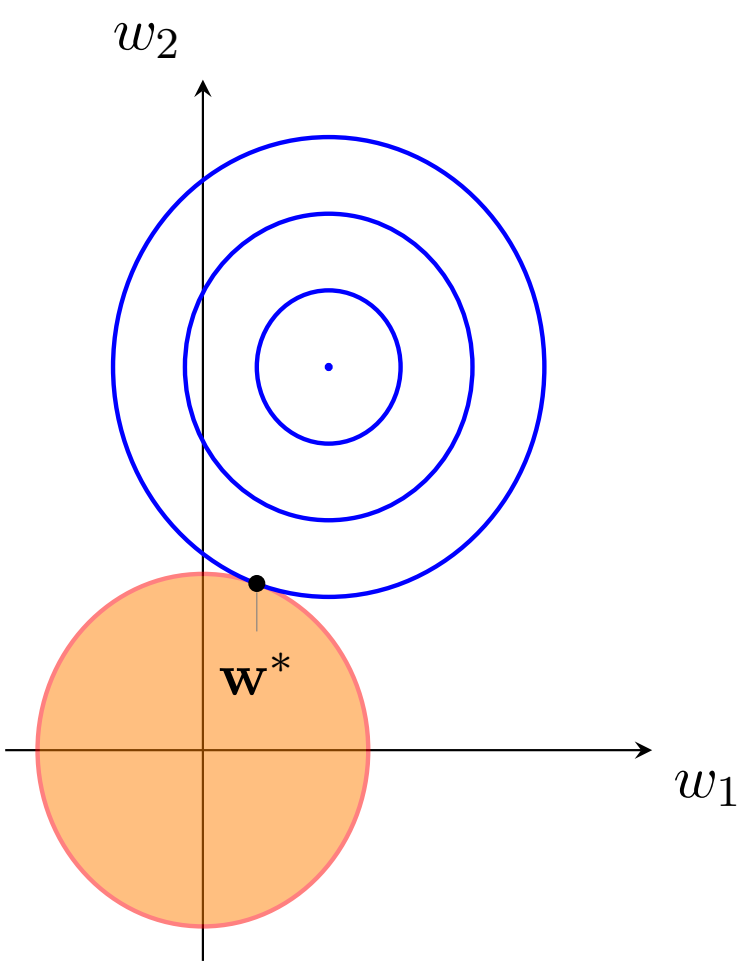
- Another popular regularization method is **lasso**

$$L(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\left(t_i - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i)\right)^2 + \frac{\lambda}{2}\|\mathbf{w}\|_1$$

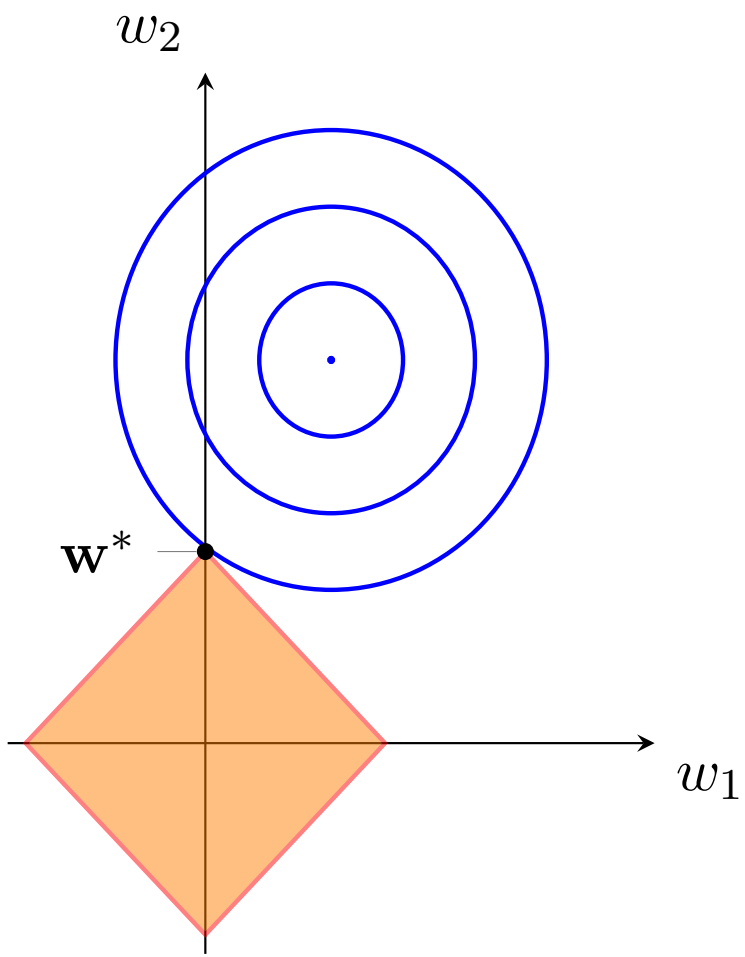
- where $\|\mathbf{w}\|_1 = \sum_{j=1}^{M}|w_j|$
- Differently from ridge, lasso is **nonlinear** in the t_i and **no closed-form** solution exists (quadratic programming problem)
- Nonetheless, it has the advantage of making some weights equal to **zero** for values of λ sufficiently large
- Lasso yields **sparse** models

Lasso vs Ridge Regression

Ridge



Lasso



Lasso tends to generate **sparser solutions** than quadratic regularizer

Bayesian Approach

- We formulate our knowledge about the world in a **probabilistic way**
  - We define the **model** that expresses our knowledge **qualitatively**
  - Our model will have some **unknown parameters**
  - We capture our assumptions about unknown parameters by specifying the **prior distribution** over those parameters before seeing the data
- We **observe the data**
- We compute the **posterior probability distribution** for the parameters, given observed data
- We use the posterior distribution to:
  - **Make predictions** by averaging over the posterior distribution
  - **Examine/Account for uncertainty** in the parameter values
  - **Make decisions** by minimizing expected posterior loss

Posterior Distribution

- The **posterior** distribution for the model parameters can be found by combining the **prior** with the **likelihood** for the parameters given data
- This is accomplished using **Bayes' Rule**:

$$P(\text{parameters}|\text{data}) = \frac{P(\text{data}|\text{parameters})P(\text{parameters})}{P(\text{data})}$$

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})P(\mathbf{w})}{P(\mathcal{D})}$$

where

  - $p(\mathbf{w}|\mathcal{D})$ is the **posterior** probability of parameters $\mathbf{w}$ given training data $\mathcal{D}$
  - $p(\mathcal{D}|\mathbf{w})$ is the probability (**likelihood**) of observing $\mathcal{D}$ given $\mathbf{w}$
  - $P(\mathbf{w})$ is the **prior** probability over the parameters
  - $P(\mathcal{D})$ is the marginal likelihood (**normalizing constant**): $P(\mathcal{D}) = \int p(\mathcal{D}|\mathbf{w})P(\mathbf{w})\mathrm{d}\mathbf{w}$

- Stating Bayes' rule in words: posterior $\propto$ likelihood $\times$ prior
- We want the most probable value of $\mathbf{w}$ given the data: **maximum a posteriori** (MAP). It is the **mode** of the posterior.

Bayesian Linear Regression

- Another approach to avoid the **over-fitting** problem of ML is to use **Bayesian** linear regression
- In the Bayesian approach the parameters of the model are considered as drawn from some **distribution**
- Assuming Gaussian likelihood model, the **conjugate prior** is Gaussian too

$$p(\mathbf{w}) = \mathcal{N}\left(\mathbf{w}|\mathbf{w}_0, \mathbf{S}_0\right)$$

- Given the data $\mathcal{D}$, the **posterior** is still Gaussian

$$p(\mathbf{w}|\mathbf{t}, \mathbf{\Phi}, \sigma^2) \propto \mathcal{N}\left(\mathbf{w}|\mathbf{w}_0, \mathbf{S}_0\right) \mathcal{N}\left(\mathbf{t}|\mathbf{\Phi}\mathbf{w}, \sigma^2\mathbf{I}_N\right) = \mathcal{N}\left(\mathbf{w}|\mathbf{w}_N, \mathbf{S}_N\right)$$

$$\mathbf{w}_N = \mathbf{S}_N\left(\mathbf{S}_0^{-1}\mathbf{w}_0 + \frac{\mathbf{\Phi}^T\mathbf{t}}{\sigma^2}\right)$$

$$\mathbf{S}_N^{-1} = \mathbf{S}_0^{-1} + \frac{\mathbf{\Phi}^T\mathbf{\Phi}}{\sigma^2}$$

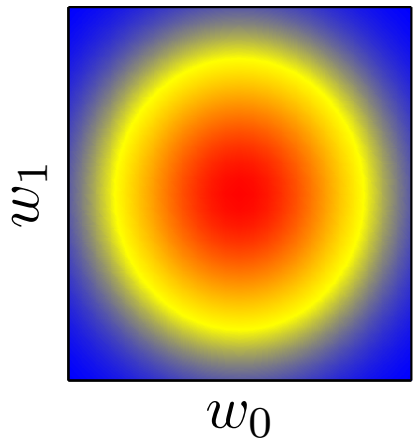
- For **sequential** data, the posterior acts as prior for the next iteration

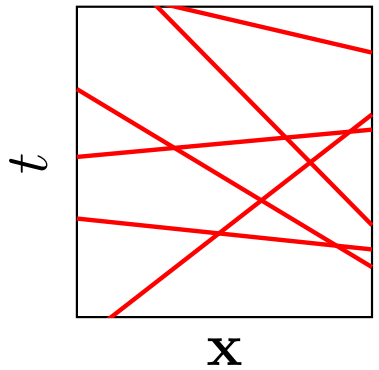
Relation to Ridge Regression

- In Gaussian distributions the **mode** coincides with the **mean**
- It follows that $\mathbf{w}_N$ is the **MAP estimator** (Maximum a posteriori)
- If the prior has infinite variance, $\mathbf{w}_N$ reduces to the **ML estimator**
- If $\mathbf{w}_0 = 0$ and $\mathbf{S}_0 = \tau^2\mathbf{I}$, then $\mathbf{w}_N$ reduces to the **ridge estimate**, where $\lambda = \frac{\sigma^2}{\tau^2}$

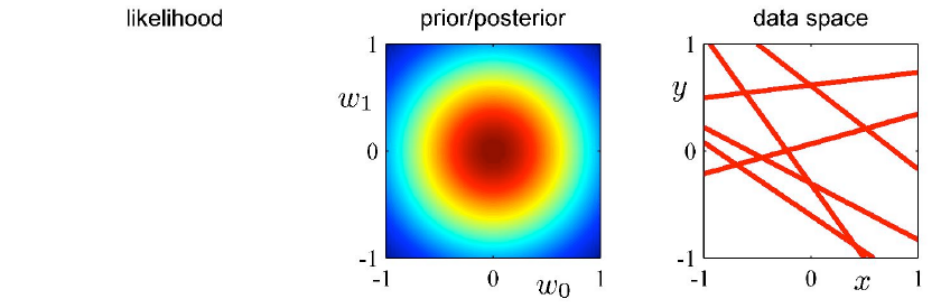
1D Example

- Data generated from: $t(x) = -0.3 + 0.5x + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 0.04)$
- x values taken uniformly from $[-1, 1]$
- Model: $y(x, \mathbf{w}) = w_0 + w_1 x$
- We assume to know $\sigma^2 = 0.04$ and $\tau^2 = 0.5$

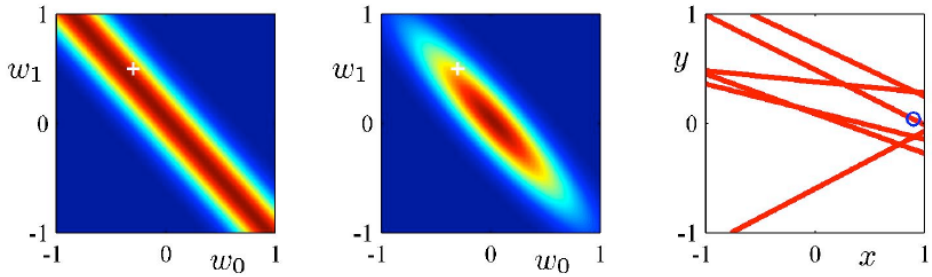




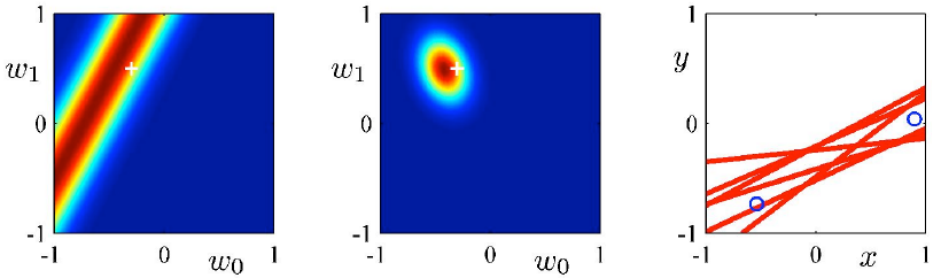
1D Example



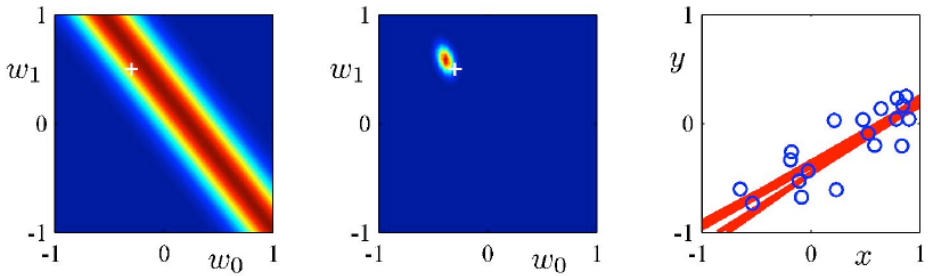
0 data points observed



1 data point observed



2 data points observed



20 data points observed

Predictive Distribution

- We are interested in the **posterior predictive distribution**

$$
\begin{aligned}
p(t|\mathbf{x}, \mathcal{D}, \sigma^2) &= \int \mathcal{N}\left(t|\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}), \sigma^2\right) \mathcal{N}\left(\mathbf{w}|\mathbf{w}_N, \mathbf{S}_N\right) \mathrm{d}\mathbf{w} \\
&= \mathcal{N}\left(t|{\mathbf{w}_N}^T\boldsymbol{\phi}(\mathbf{x}), \sigma_N^2(\mathbf{x})\right) \\
\sigma_N^2(\mathbf{x}) &= \sigma^2 + \boldsymbol{\phi}(\mathbf{x})^T\mathbf{S}_N\boldsymbol{\phi}(\mathbf{x})
\end{aligned}
$$

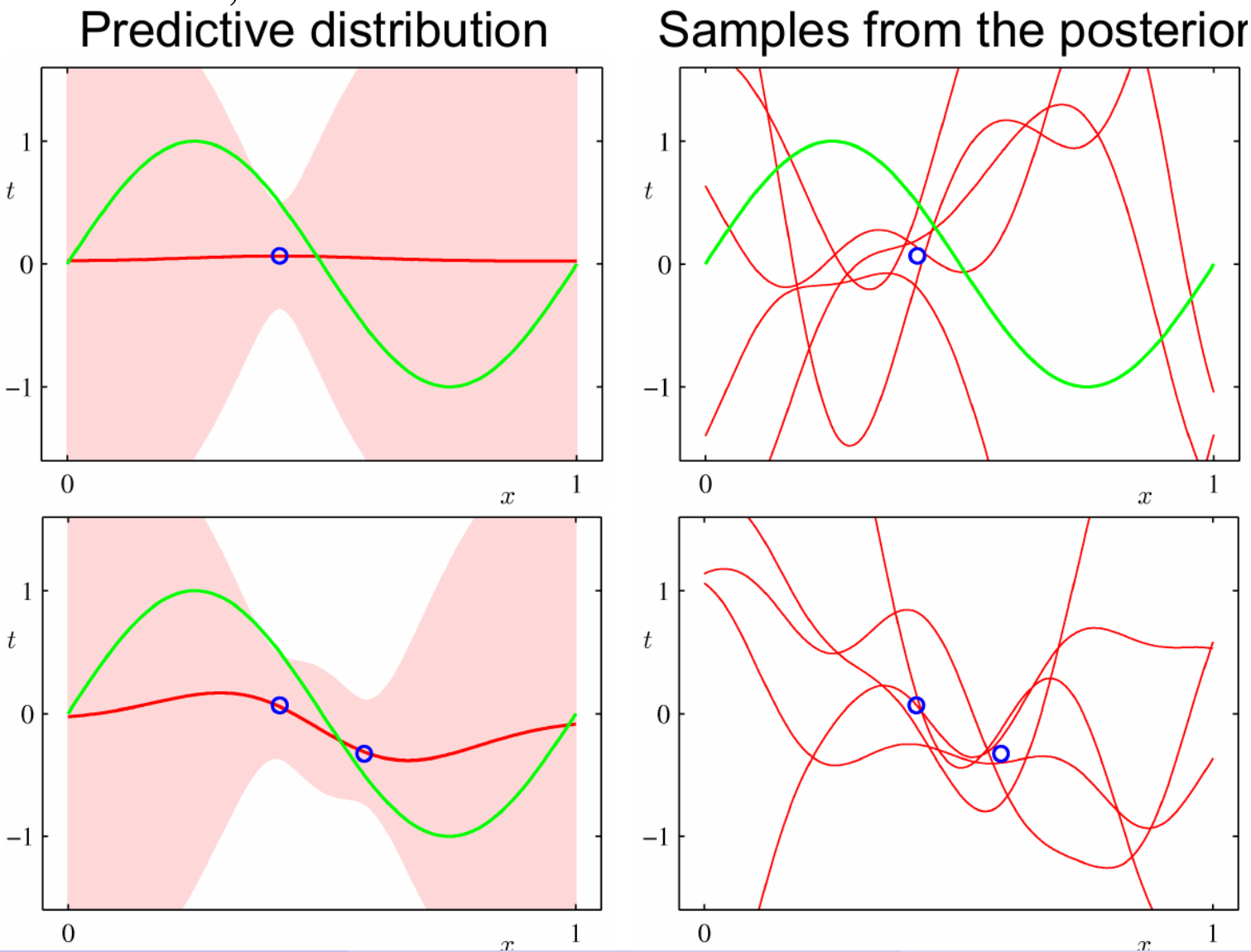
where

$$
\sigma_N^2(\mathbf{x}) = \underbrace{\sigma^2}_{\substack{\text{noise in the} \\ \text{target values}}} + \underbrace{\boldsymbol{\phi}(\mathbf{x})^T\mathbf{S}_N\boldsymbol{\phi}(\mathbf{x})}_{\substack{\text{Uncertainty associated} \\ \text{with parameter values}}}
$$

- In the limit, as $N \to \infty$, **the second term goes to zero**
- The variance of the predictive distribution arises only from the additive noise governed by parameter σ

Example

Sinsusoidal dataset, 9 Gaussian basis functions



Modeling Challenges

- The first challenge is in **specifying suitable model** and **suitable prior distributions**
  - A suitable model should admit all the possibilities that thought to be at all **likely**
  - A suitable prior should **avoid** giving zero or very small probabilities to **possible events**, but should also avoid spreading out the probability over all possibilities
- To avoid uninformative priors, we may need to model **dependencies** between parameters
- One strategy is to introduce **latent variables** into the model and **hyperparameters** into the prior
- Both of these represent the ways of modeling dependencies in a **tractable way**

Computational Challenges

The other big challenge is **computing the posterior distribution**. There are several approaches:

- **Analytical integration**: If we use **"conjugate priors"**, the posterior distribution can be computed **analytically**. Only works for **simple models**
- **Gaussian (Laplace) approximation**: Approximate the posterior distribution with a Gaussian. Works well when there a lot of data compared to the model complexity
- **Monte Carlo integration**: Once we have a sample from the posterior distribution, we can do many things. Currently, the common approach is Markov Chain Monte Carlo (MCMC), that consists in simulating a Markov chain that converges to the posterior distribution
- **Variational approximation**: A cleverer way of approximating the posterior. It is usually faster than MCMC, but it is less general

Pros and Cons of Fixed Basis Functions

- **Advantages**
  - Closed-form solution
  - Tractable Bayesian treatment
  - Arbitrary non-linearity with the proper basis functions
- **Limitations**
  - Basis functions are chosen independently from the training set
  - Curse of dimensionality