

# Linear Regression

## Machine Learning

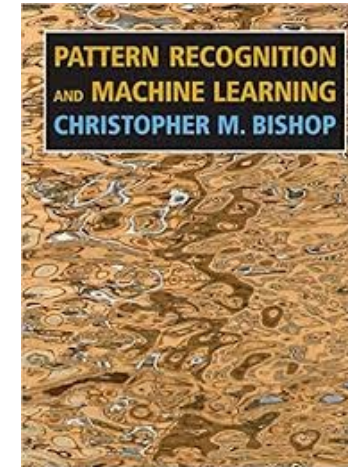
Daniele Loiacono



**POLITECNICO**  
MILANO 1863

# References

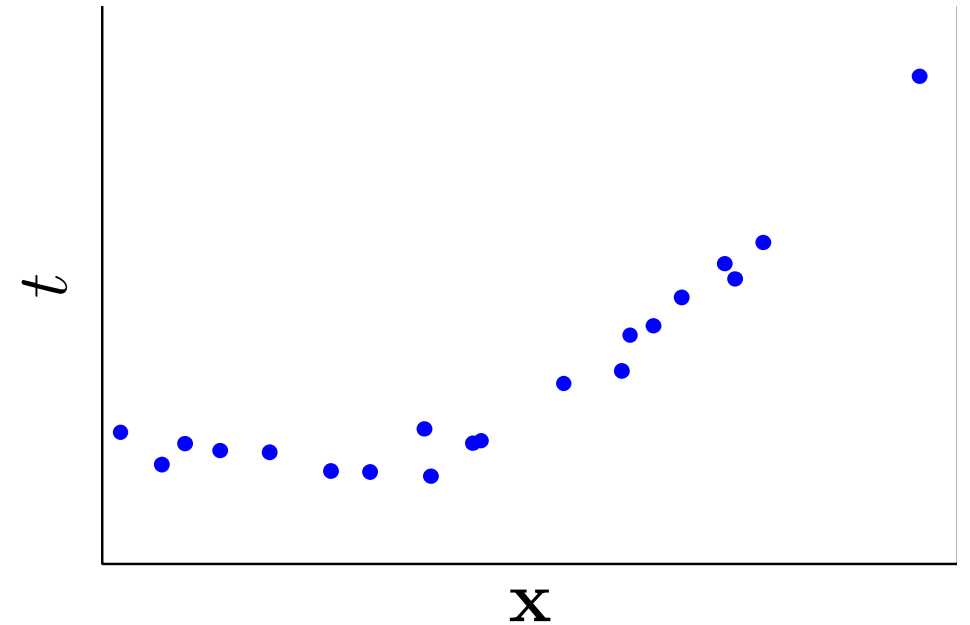
- ❑ This slides are based on material of [prof. Marcello Restelli](#)
  
- ❑ *Pattern Recognition and Machine Learning*, Bishop
  - ▶ Chapter 1 (1.1, 1.2, 1.3)
  - ▶ Chapter 3 (3.1, 3.3)



# What is regression?

- Learn an **approximation** of function  $f(x)$  that maps input  $x$  to a continuous output  $t$  from a dataset  $\mathcal{D}$

$$\mathcal{D} = \{\langle x, t \rangle\} \Rightarrow t = f(x)$$

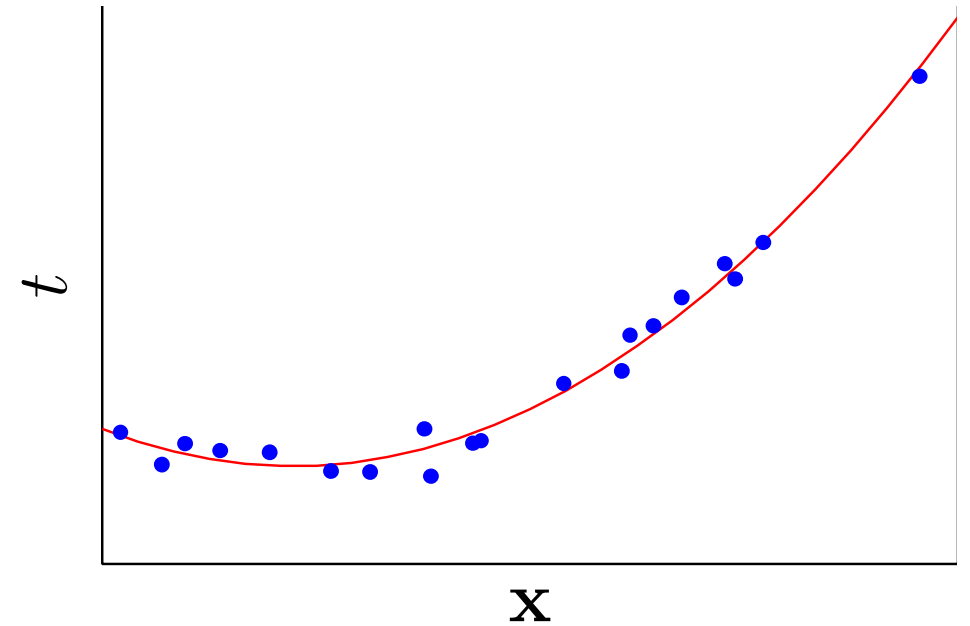


# What is regression?

- Learn an **approximation** of function  $f(x)$  that maps input  $x$  to a continuous output  $t$  from a dataset  $\mathcal{D}$

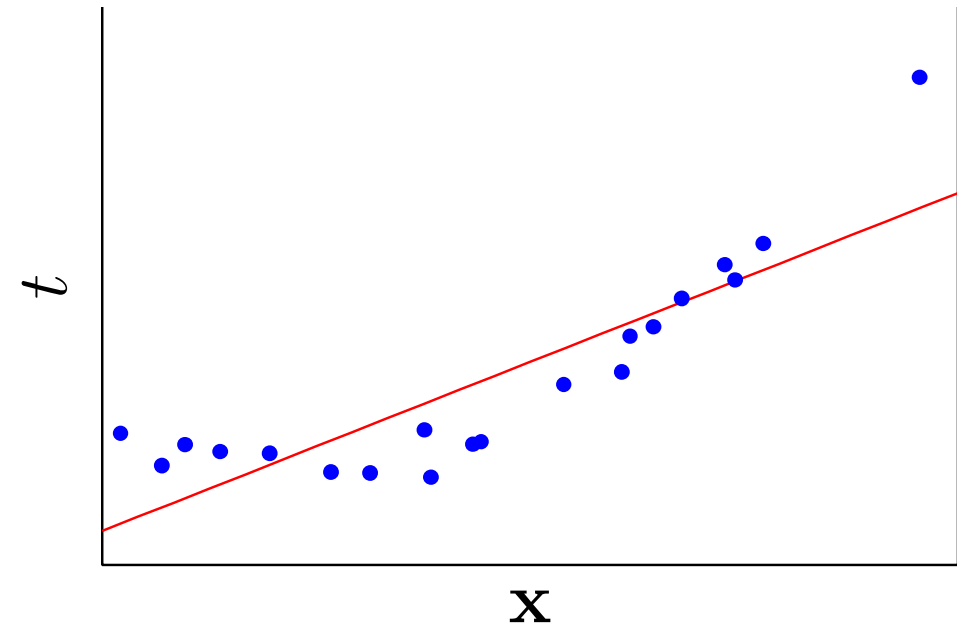
$$\mathcal{D} = \{\langle x, t \rangle\} \Rightarrow t = f(x)$$

- ▶ How do we model  $f$ ?
- ▶ How do we evaluate our approximation?
- ▶ How do we optimize our approximation?



# Linear Regression

- In linear regression,  $f(x)$  is modeled with linear functions
  - ▶ Linear models can be easily **explained**
  - ▶ A linear regression problem can be solved **analytically**
  - ▶ Linear functions can be extended to model also **non-linear relationships**
  - ▶ More sophisticated methods are based on linear regression

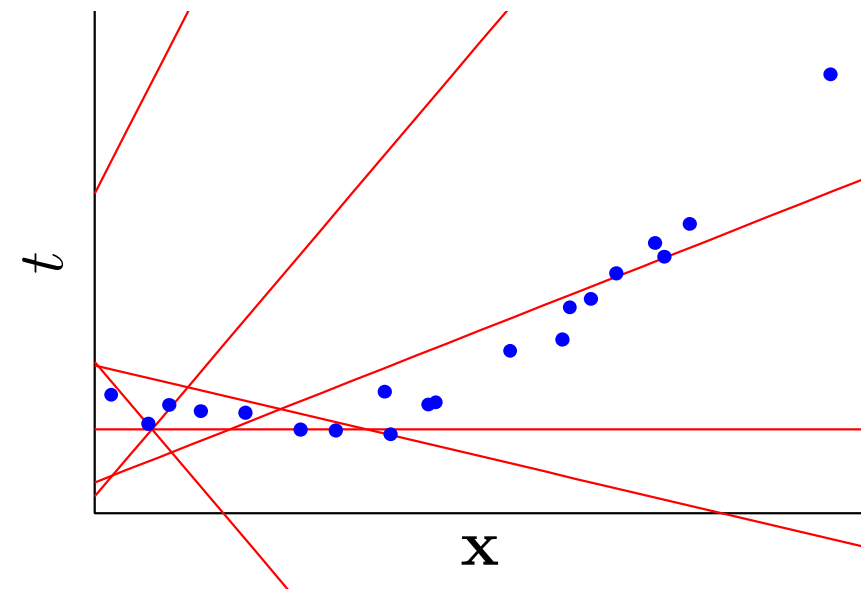
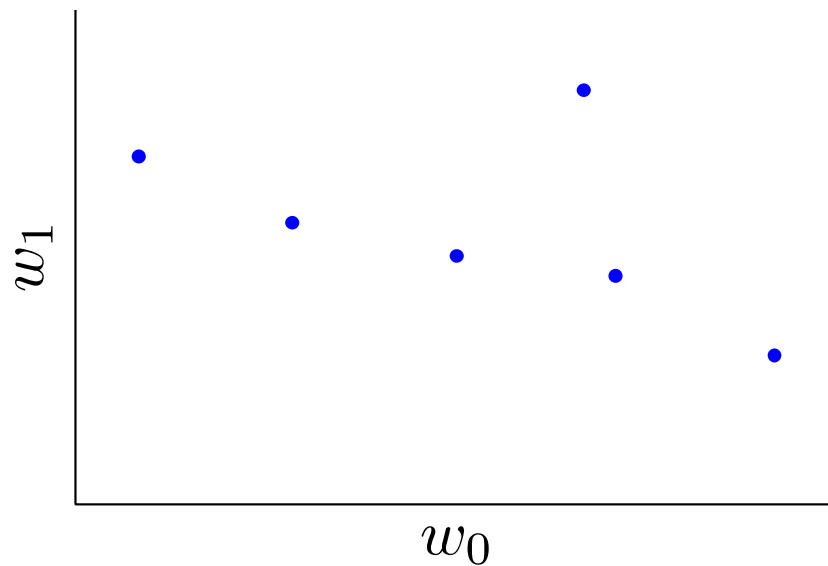


# Linear Regression: model

□ The simplest linear model can be defined as:

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

- ▶  $\mathbf{x} = (1, x_1, \dots, x_{D-1})$
- ▶  $w_0$  is called **bias parameter**



# Linear Regression: loss function and optimization

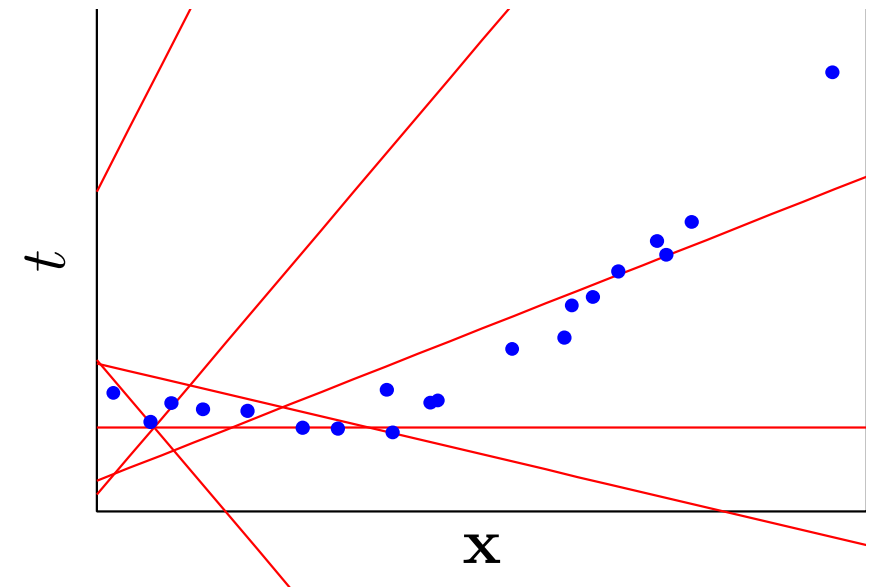
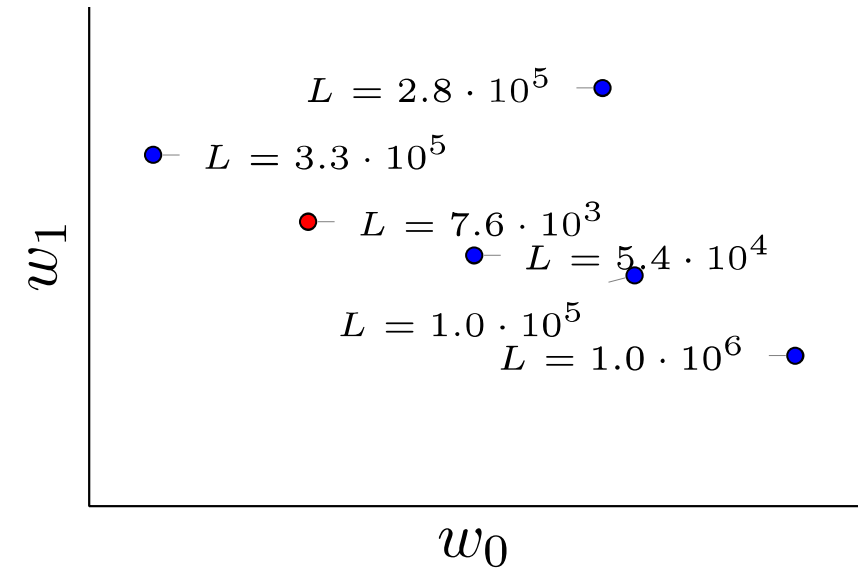
- A convenient **error loss function** is the **sum of squared errors (SSE)**:

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

- the sum in  $\mathcal{L}$  is also called **residual sum of squares (RSS)** and can be written as the sum of **residual errors**:

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

- closed-form optimization of  $\mathcal{L}$  can be easily obtained



# Linear Models and Basis Functions

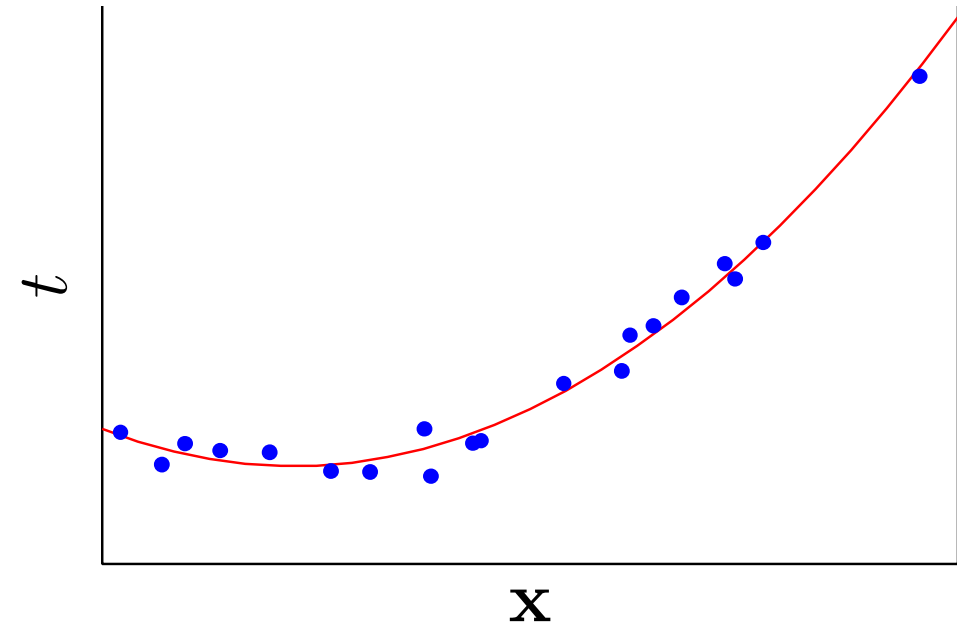


# Linear Models

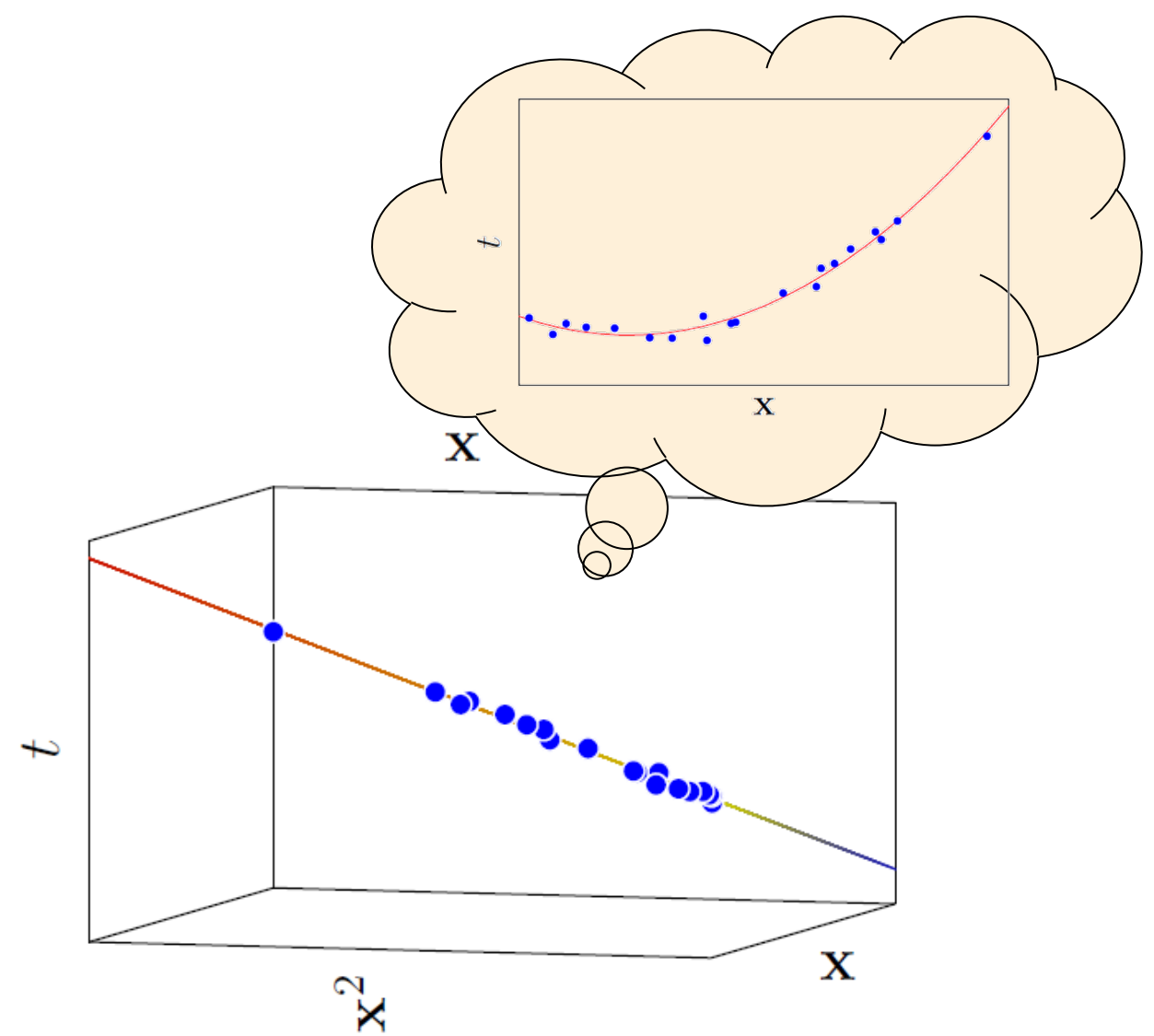
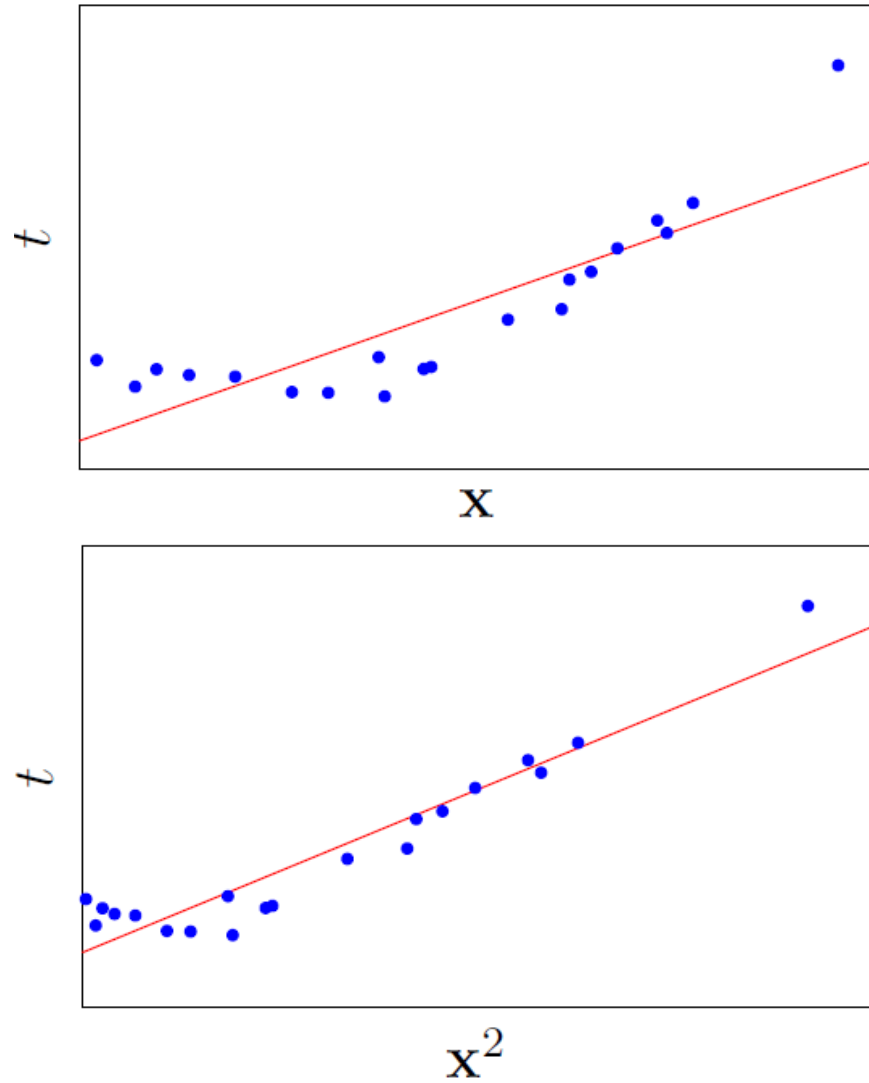
- ❑ A linear combination of the input variables is not enough to model data...
- ❑ ... but we just need a regression model that is **linear in the parameters**
- ❑ We can define a model using non-linear basis functions:

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

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



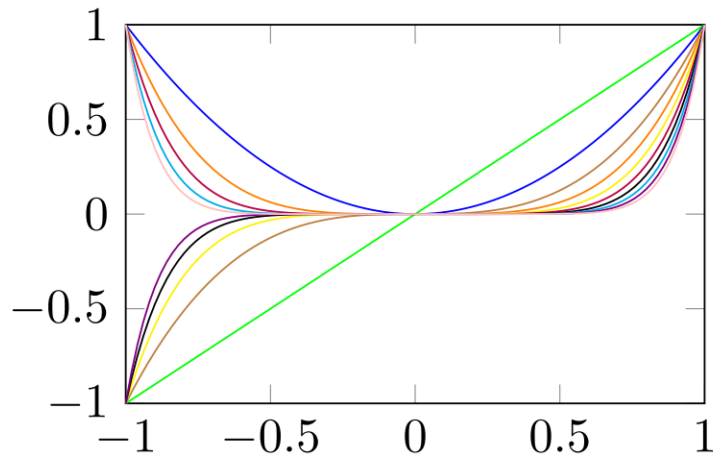
Let see it in feature space...



# Basis Functions

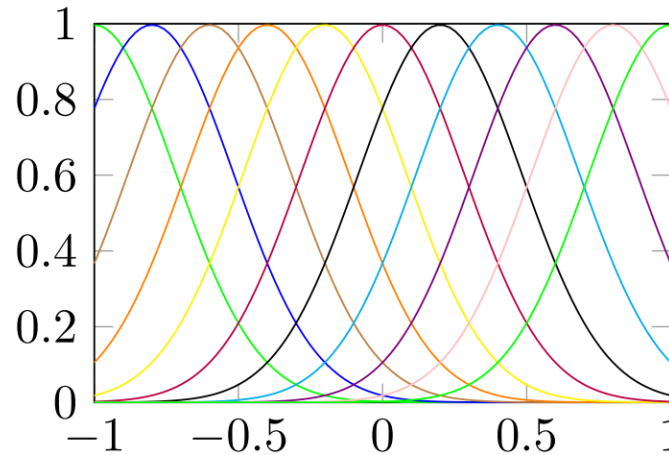
- Some examples of basis function (assuming single-variable input)

Polynomial



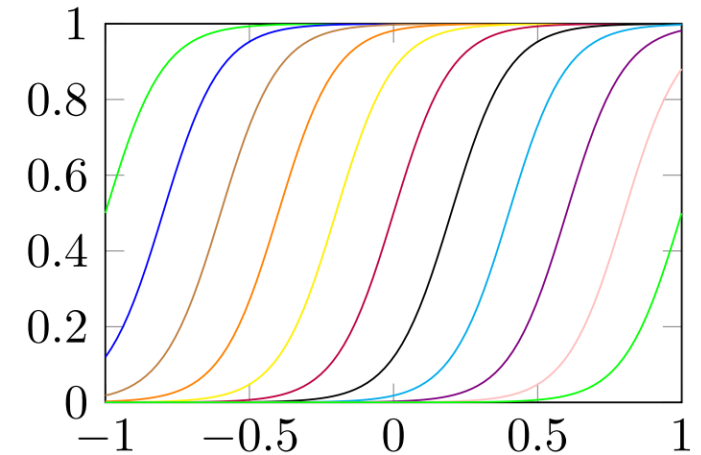
$$\phi_j(x) = x^j$$

Gaussian



$$\phi_j(x) = \exp\left(-\frac{(x - \mu_j)^2}{2\sigma^2}\right)$$

Sigmoidal

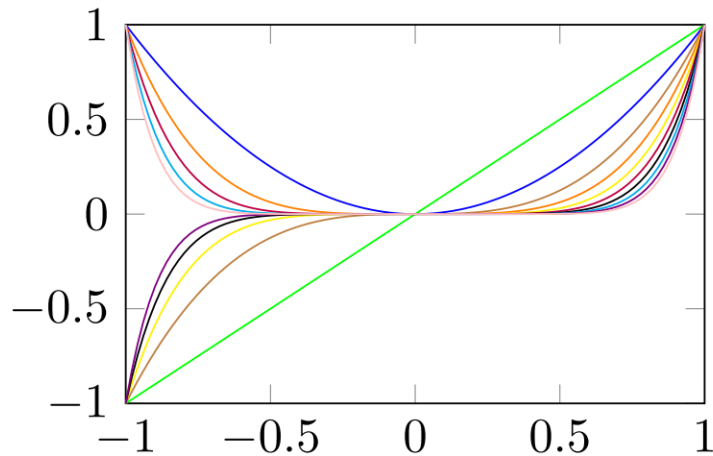


$$\phi_j(x) = \frac{1}{1 + \exp\left(\frac{\mu_j - x}{\sigma}\right)}$$

# Basis Functions

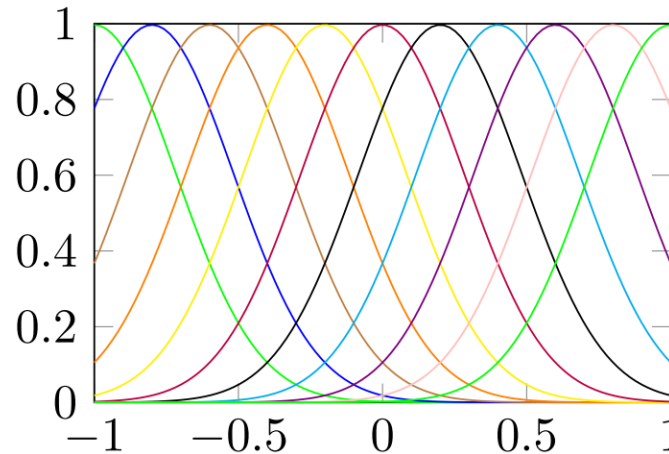
- Some examples of basis function (assuming single-variable input)

Polynomial



$$\phi_j(x) = x^j$$

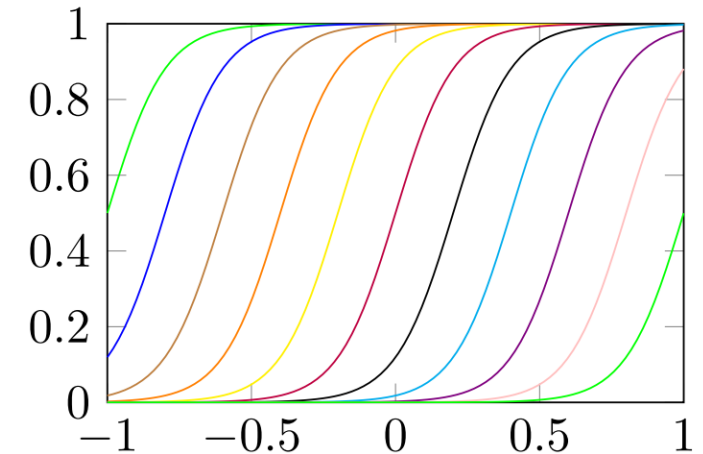
Gaussian



$$\phi_j(x) = \exp\left(-\frac{(x - \mu_j)^2}{2\sigma^2}\right)$$

**LOCAL**

Sigmoidal



$$\phi_j(x) = \frac{1}{1 + \exp\left(\frac{\mu_j - x}{\sigma}\right)}$$

# Least Squares

# Ordinary Least Squares

- For linear models, a closed-form optimization of the RSS, known as **least squares**, starting from the matrix form of the loss function:

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

► where  $\Phi = (\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_N))^T$  and  $\mathbf{t} = (t_1, \dots, t_N)^T$

# Ordinary Least Squares

- For linear models, a closed-form optimization of the RSS, known as **least squares**, starting from the matrix form of the loss function:

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

► where  $\Phi = (\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_N))^T$  and  $\mathbf{t} = (t_1, \dots, t_N)^T$

- We can compute first a second derivative of  $\mathcal{L}(w)$  to find the optimal  $w$

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

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

# Ordinary Least Squares

- For linear models, a closed-form optimization of the RSS, known as **least squares**, starting from the matrix form of the loss function:


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

► where  $\Phi = (\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_N))^T$  and  $\mathbf{t} = (t_1, \dots, t_N)^T$

- We can compute first a second derivative of  $\mathcal{L}(w)$  to find the optimal  $w$

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

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


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

Assuming  $(\Phi^T \Phi)$  non singular  
Complexity  $O(NM^2 + M^3)$



# Sequential Learning

- ❑ Closed-form optimization (OLS) is not feasible with large dataset
- ❑ Instead, a **stochastic** (or **sequential**) gradient descent is possible
- ❑ **Least Mean Square (LMS)** algorithm:

$$L(\mathbf{x}) = \sum_n L(x_n)$$

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

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

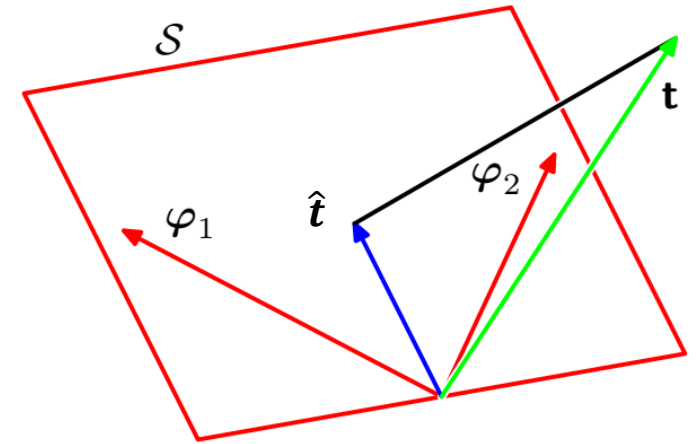
- $\alpha$  is called learning rate and to guarantee convergence:

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

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

# Geometric Interpretation of OLS

- Let  $\mathbf{t}$  be the N-dimensional target vector
- Let  $\varphi_j$  be the  $j$ -th column of matrix  $\Phi$ 
  - ▶  $\varphi_1, \dots, \varphi_M$  identify a linear subspace  $\mathcal{S}$
- Let  $\hat{\mathbf{t}}$  be the N-dimensional vector computed as  $\Phi \mathbf{w}$ 
  - ▶  $\hat{\mathbf{t}}$  is a linear combination of  $\varphi_j$  and lies in  $\mathcal{S}$
- OLS finds  $\hat{\mathbf{t}}$  minimizing the SSE with respect to  $\mathbf{t}$ 
  - ▶  $\hat{\mathbf{t}}$  represents the projection of  $\mathbf{t}$  onto the subspace  $\mathcal{S}$



$$\hat{\mathbf{t}} = \Phi \hat{\mathbf{w}} = \underbrace{\Phi \left( \Phi^T \Phi \right)^{-1} \Phi^T}_{\text{Hat Matrix (H)}} \mathbf{t}$$

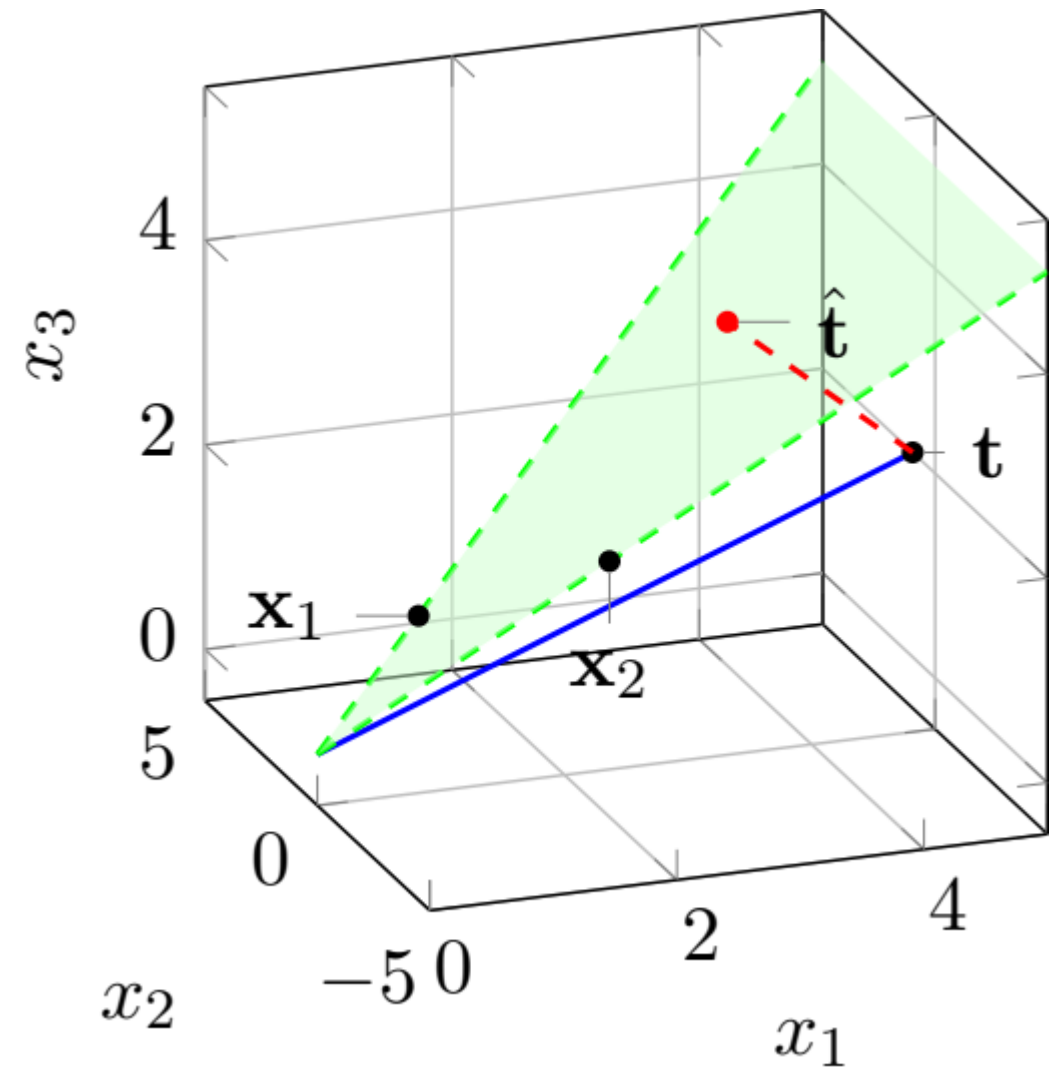
# Geometric Interpretation of OLS: an example

□ Let  $N=3$  and  $M=2$

$$\Phi = \mathbf{X} = \begin{pmatrix} 1 & 2 \\ 1 & -2 \\ 1 & 2 \end{pmatrix}$$

$$\mathbf{t} = \begin{pmatrix} 5 \\ 1 \\ 2 \end{pmatrix}$$

$$\hat{\mathbf{t}} = \begin{pmatrix} 3.5 \\ 1 \\ 3.5 \end{pmatrix}$$



# Multiple Outputs

- ❑ What happens if our regression problem has multiple outputs, i.e.,  $\mathbf{t}$  is not scalar
- ❑ It is possible to solve independently a regression problem for each problem
- ❑ Yet, it is possible to use the same set of basis functions:

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

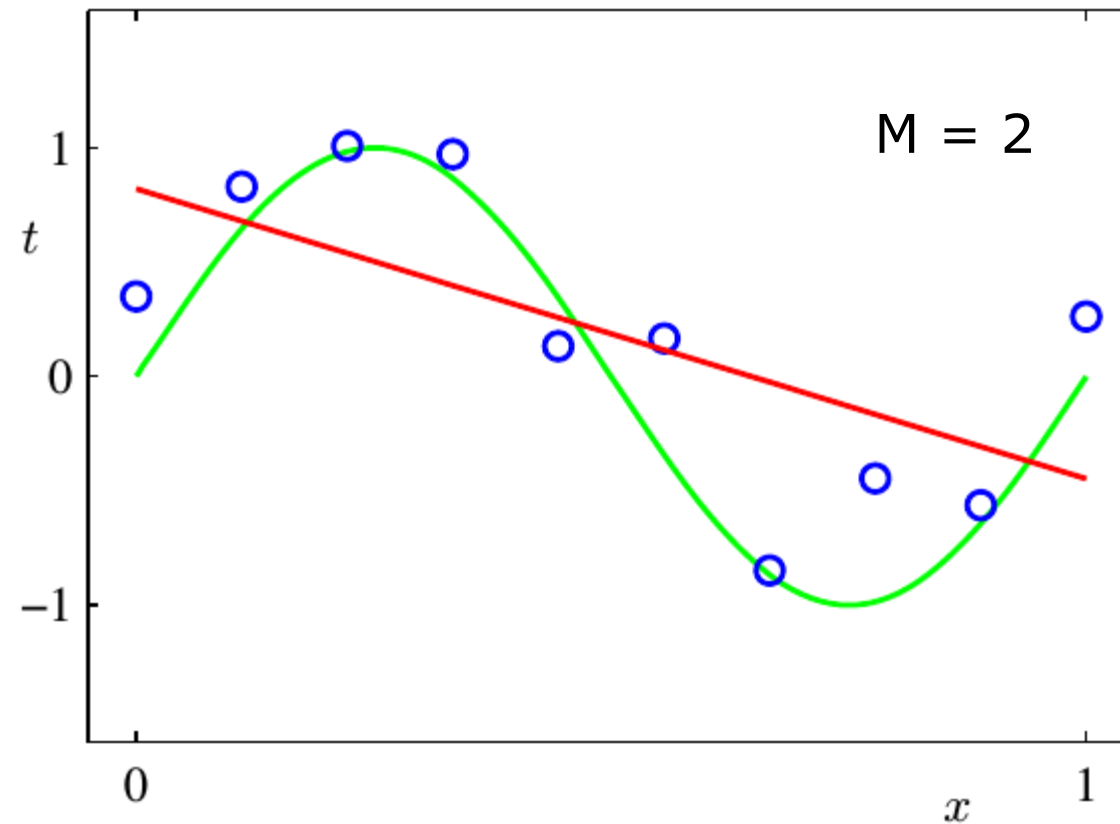
- ▶ Where each column of matrix  $\mathbf{T}$  and  $\hat{\mathbf{W}}$  are respectively the target vector of each and the weight vector for each output
- ❑ The solution above can be easily **decoupled** for each output  $k$ :

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

- ▶ as a benefit  $(\Phi^T \Phi)^{-1}$  can be computed only **once**

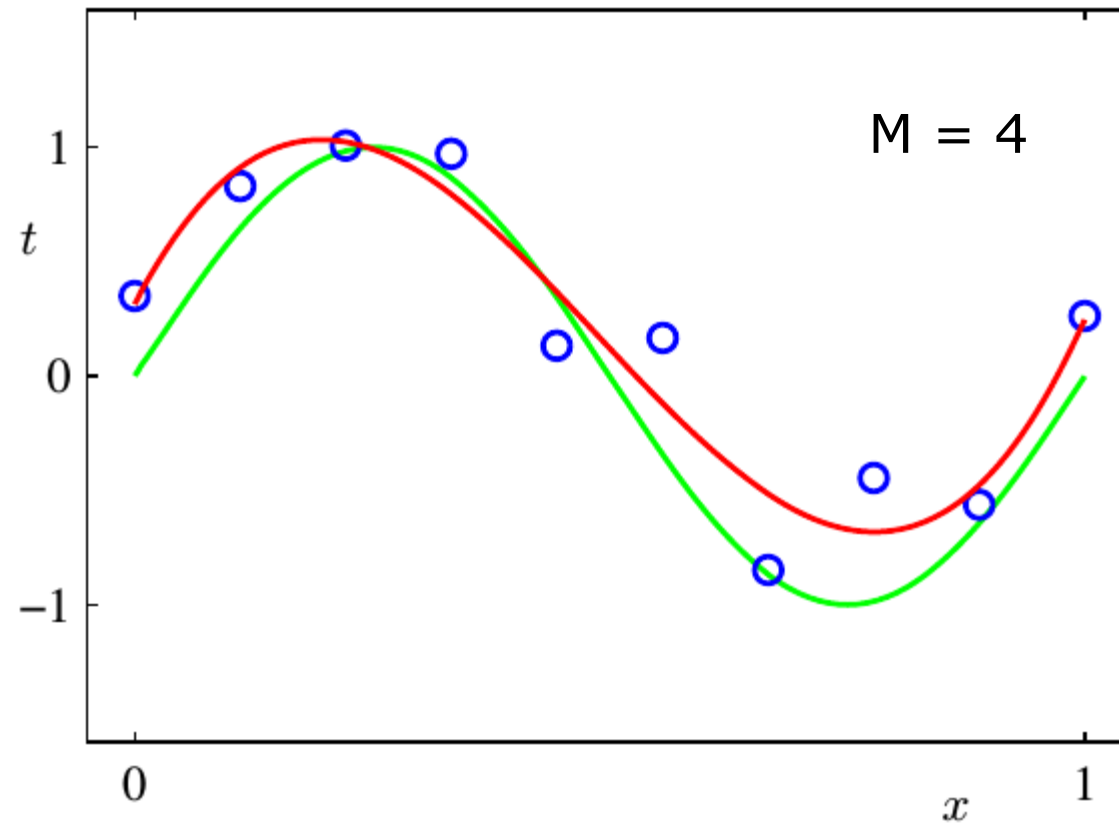
# Regularization

# How do we design the linear models? An example



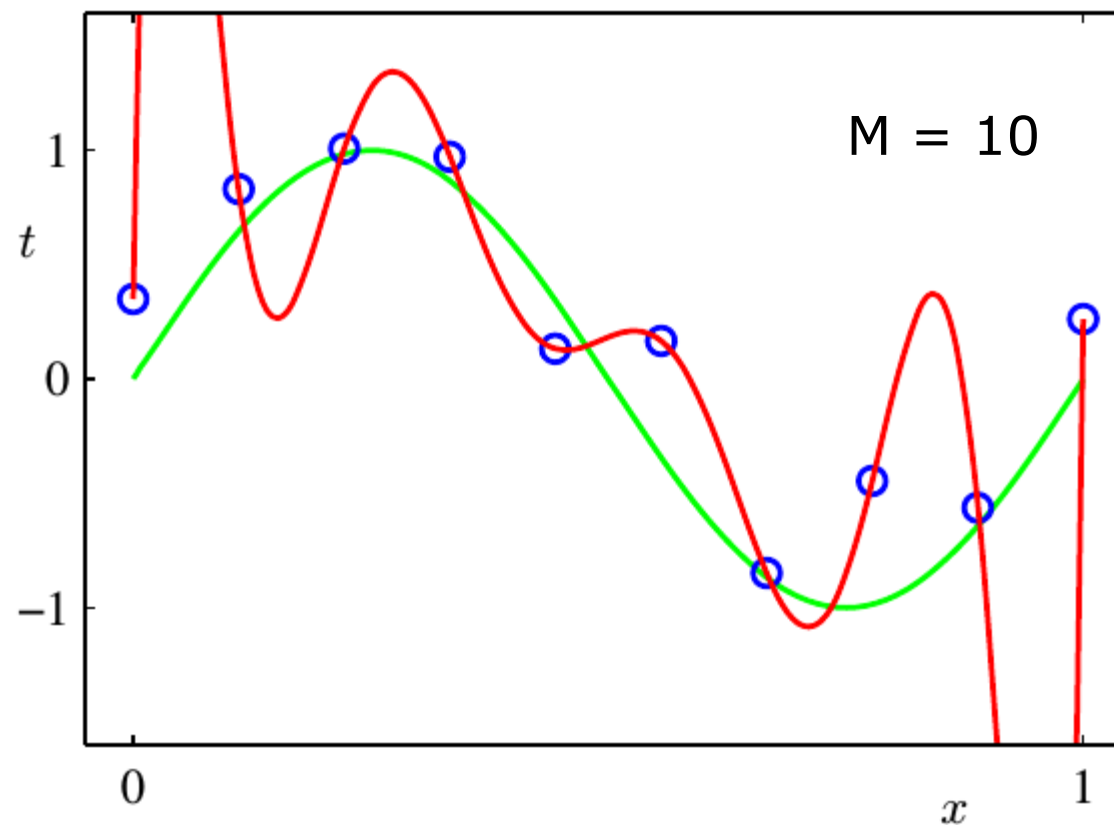
1-order polynomial model

# How do we design the linear models? An example



3-order polynomial model

# How do we design the linear models? An example



9-order polynomial model



# What is regularization?

- What happens to model parameters when complexity increases?

	$M = 1$	$M = 2$	$M = 3$	$M = 10$
$\hat{w}_0$	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

# Regularization

□ How do we extend the loss function?

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

- ▶  $\mathcal{L}_D(w)$  is the usual loss function (e.g., RSS)
- ▶  $\mathcal{L}_w(w)$  accounts for model complexity
- ▶  $\lambda$  is the **regularization** coefficient

□ How do we design  $\mathcal{L}_w(w)$  ?

- ▶ Ridge Regression
- ▶ Lasso

# Ridge Regression

- In ridge regression:

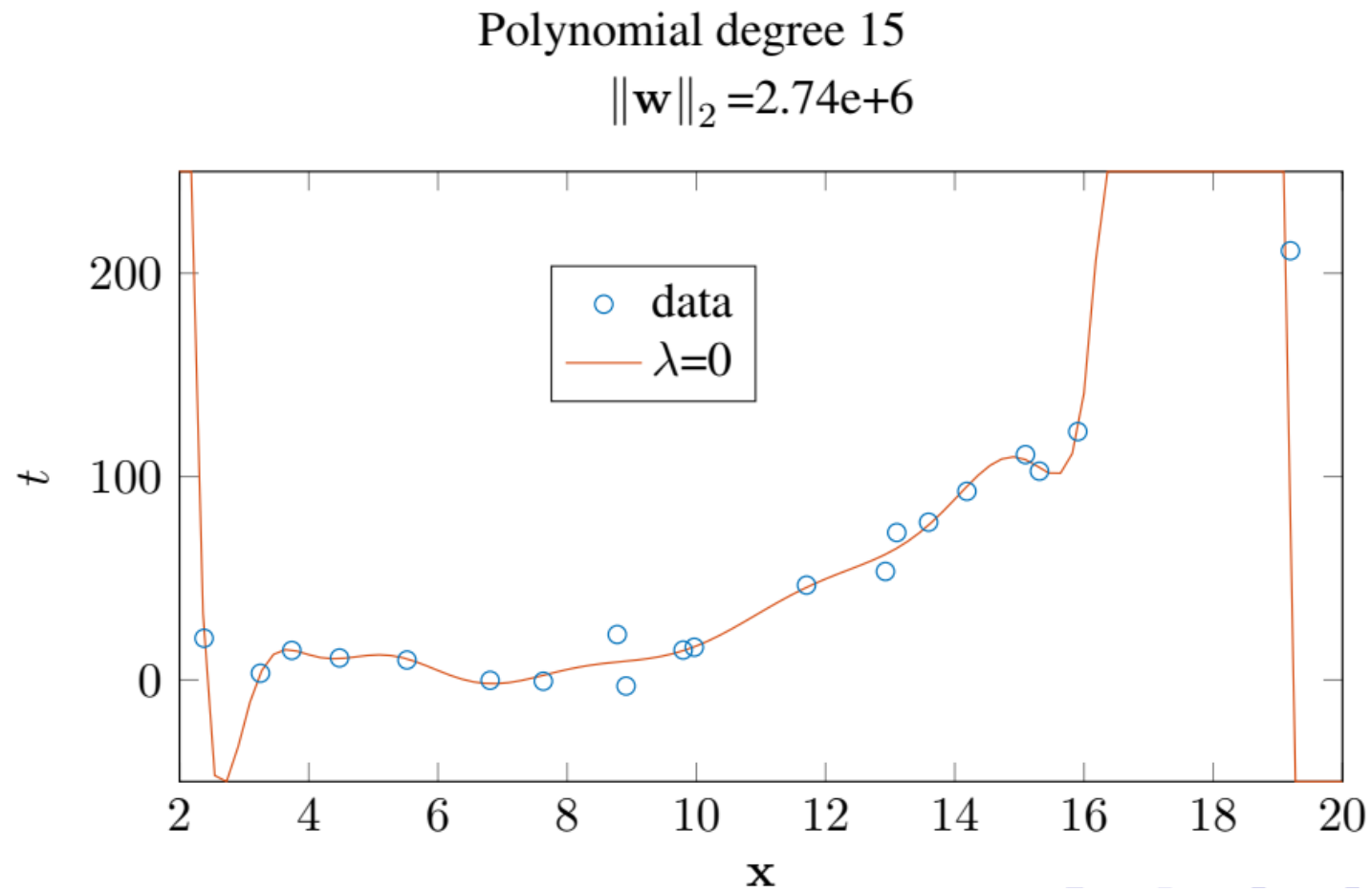
$$L_W(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} = \frac{1}{2} \|\mathbf{w}\|_2^2$$

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

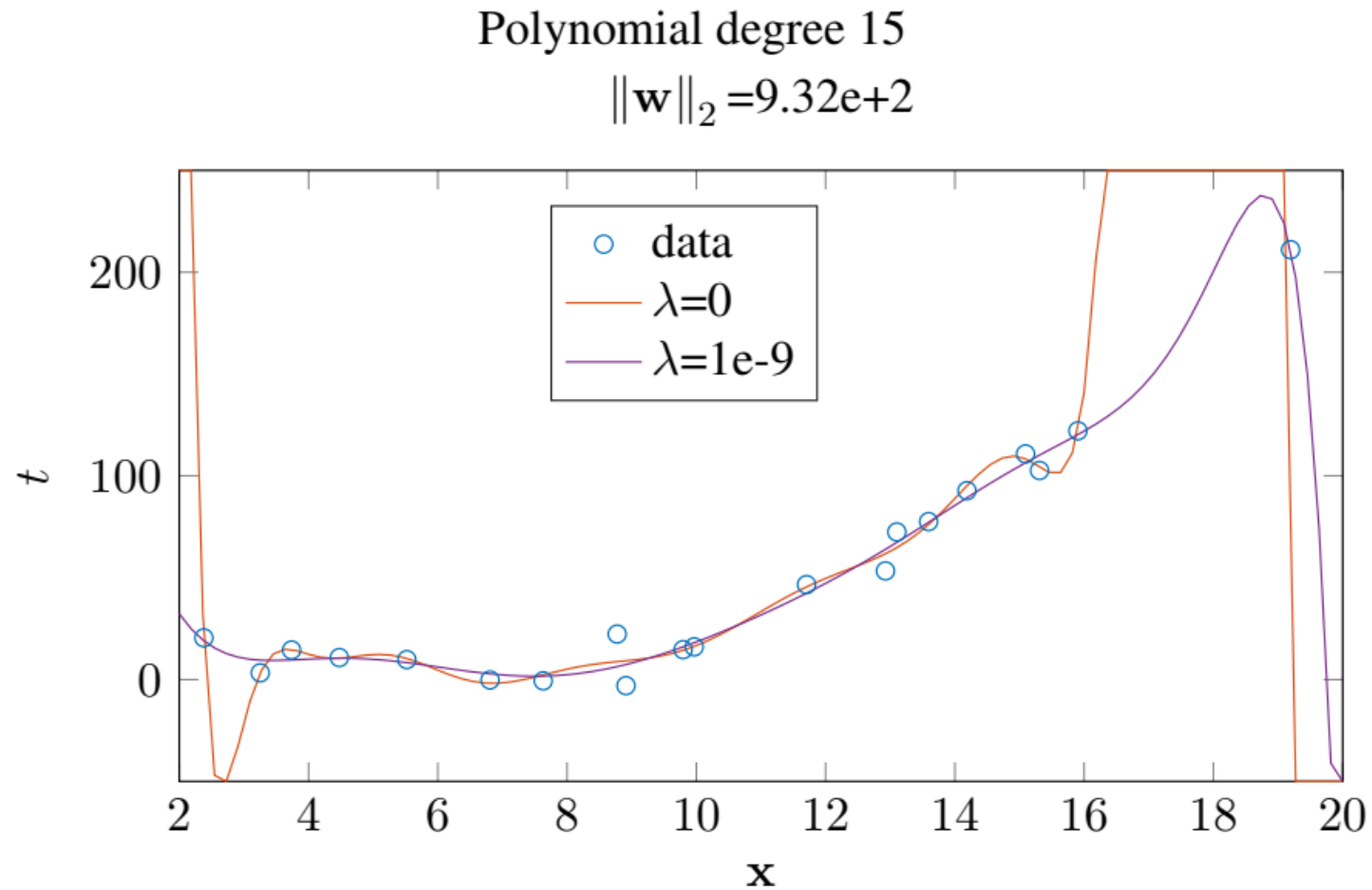
- The loss function is still quadratic with respect to  $w$  and closed-form optimization is still possible:

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

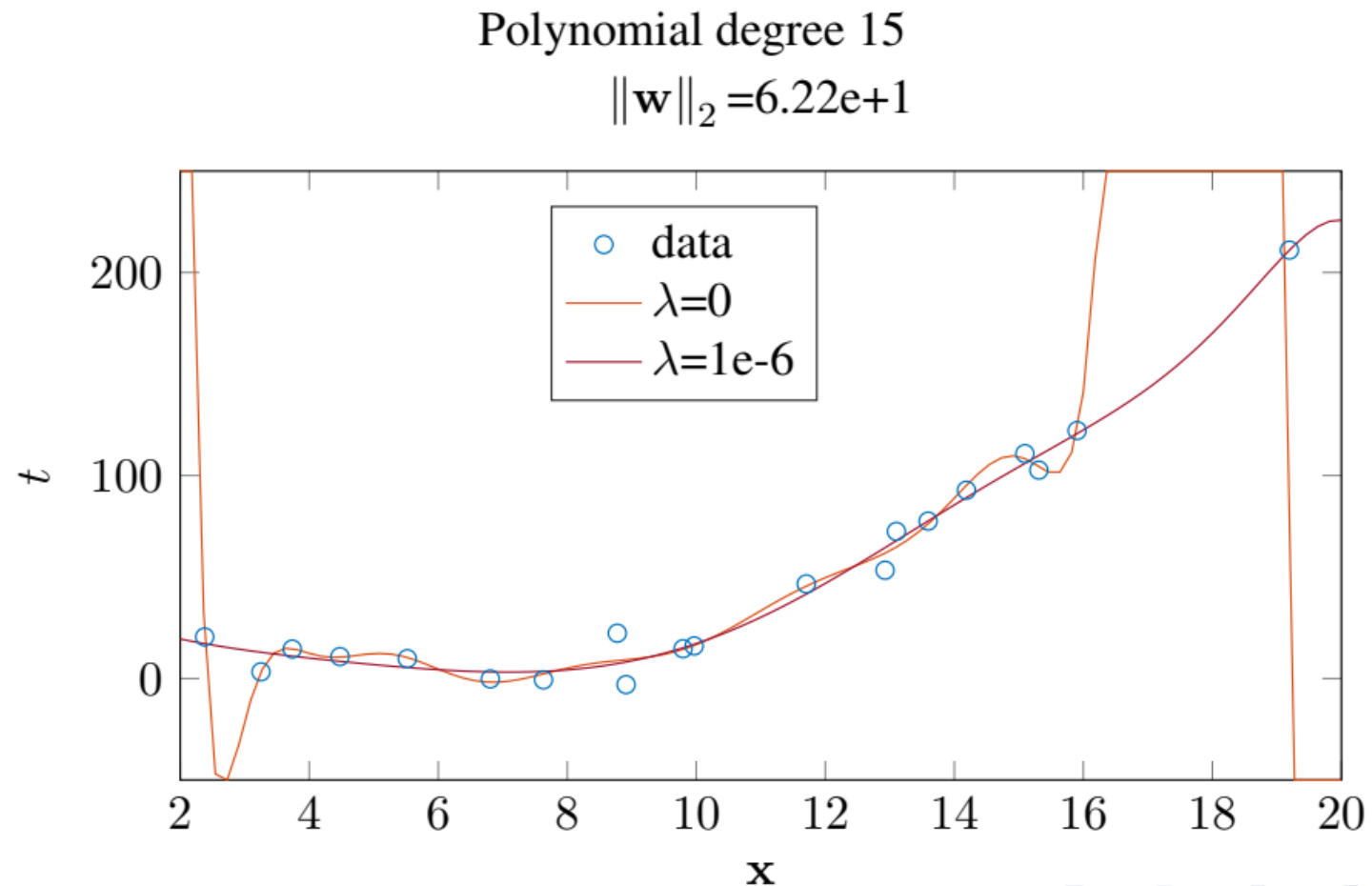
# Ridge Regression: quadratic example



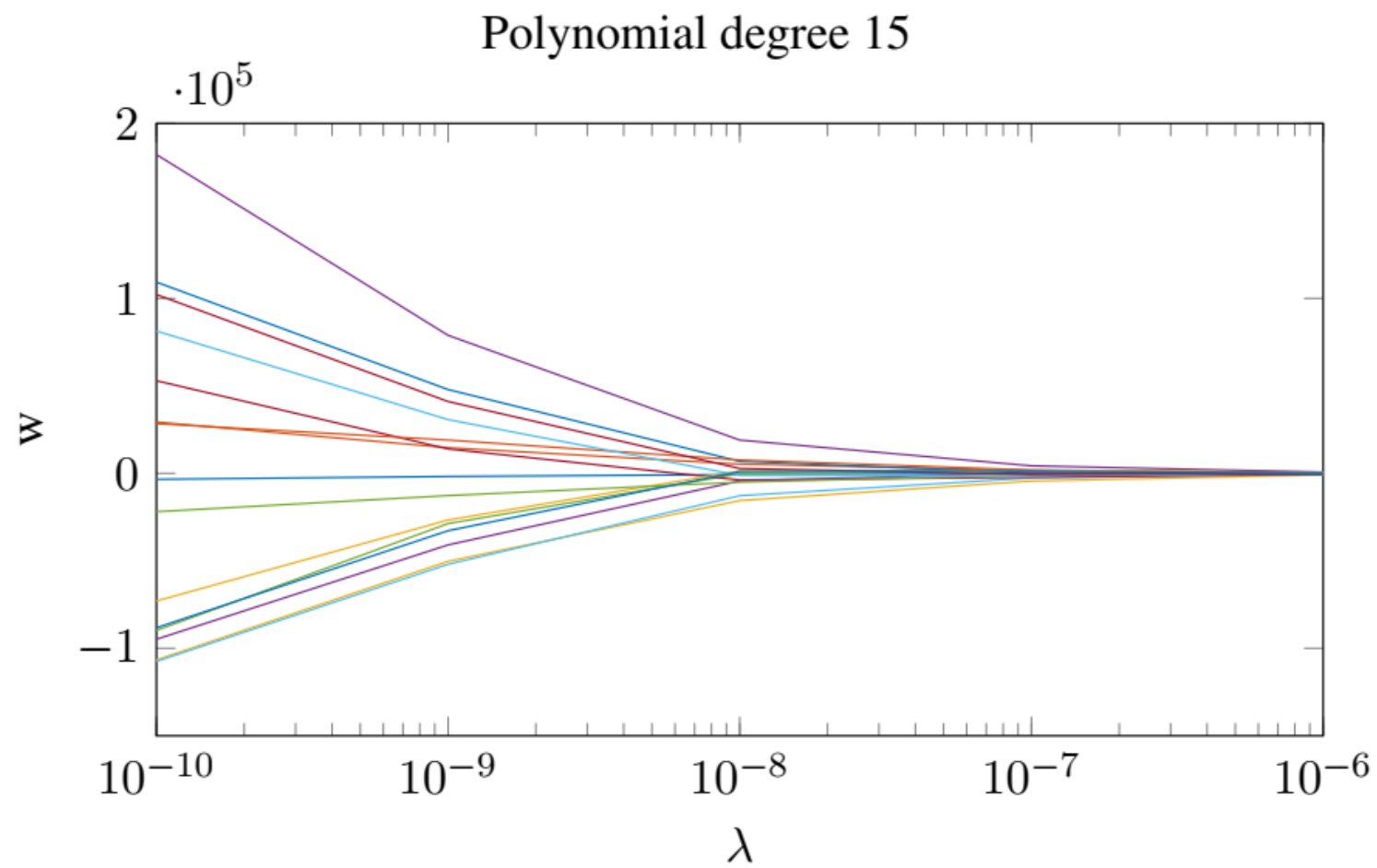
# Ridge Regression: quadratic example



# Ridge Regression: quadratic example



# Ridge Regression: quadratic example



# Lasso

- Another common regularization method is **lasso**:

$$L_W(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|_1 = \frac{1}{2} \sum_{j=0}^{M-1} |w_j|$$

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

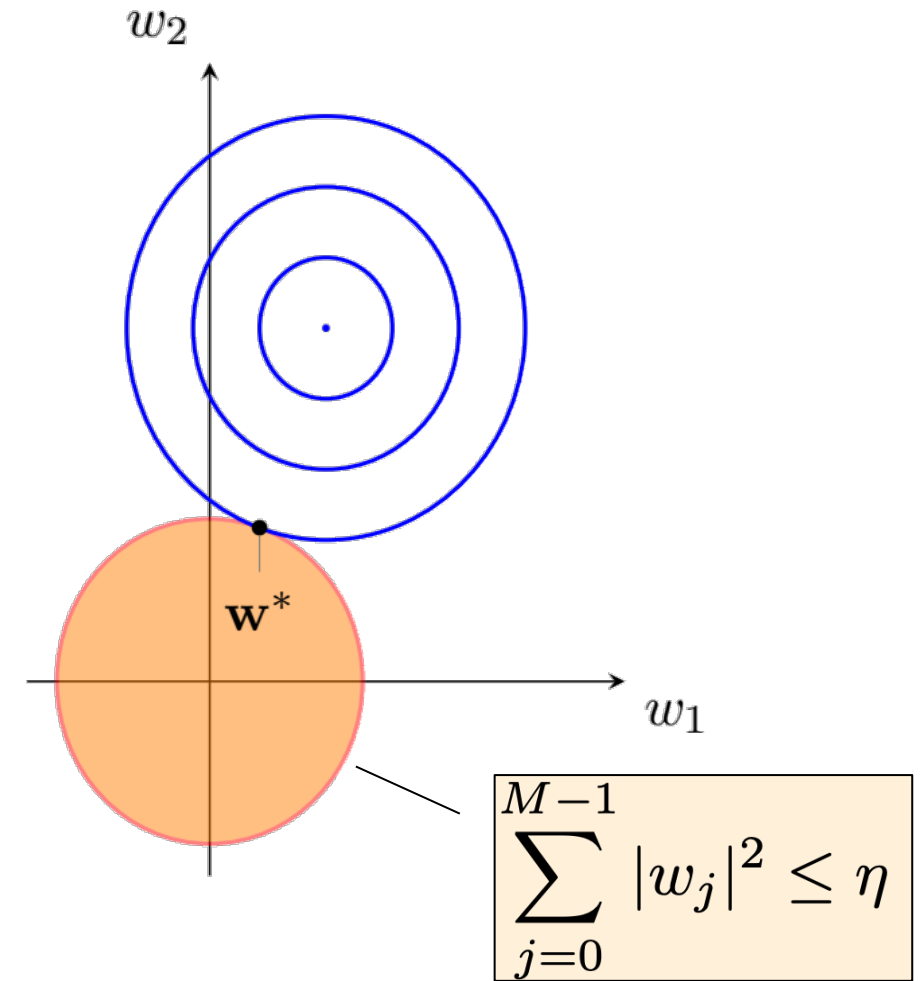
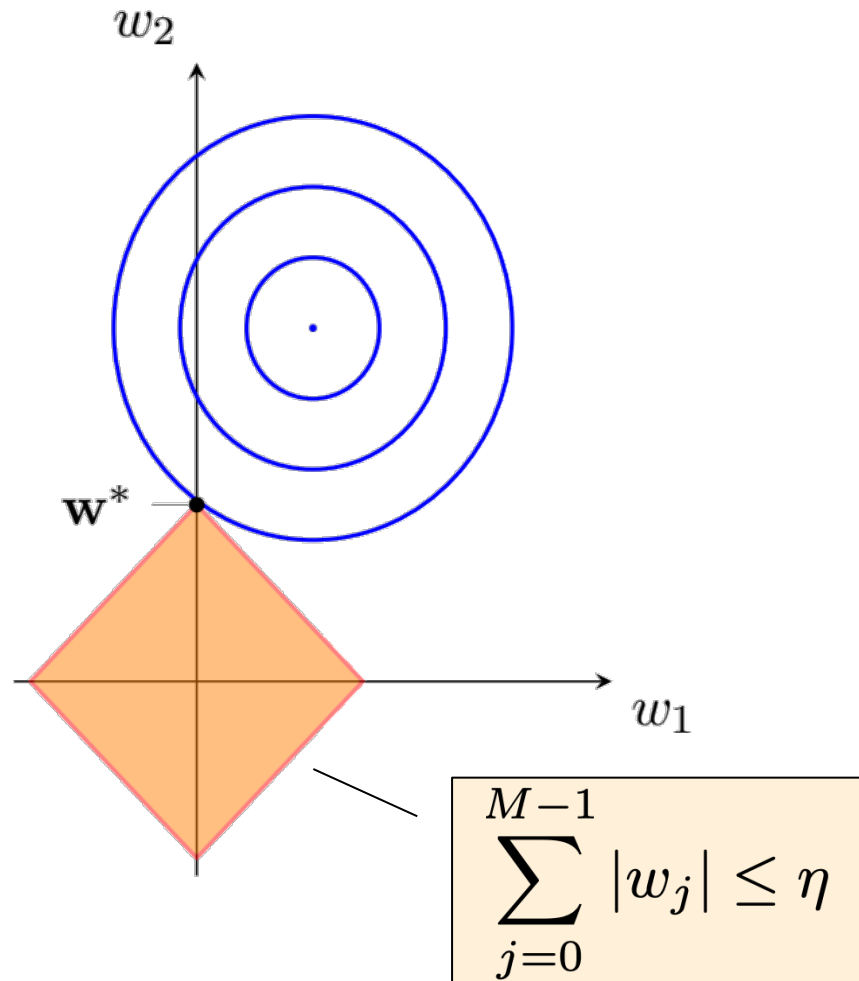
- In this case, closed-form optimization is not possible
- Nevertheless, lasso typically leads to **sparse** regression models: when regularization coefficient ( $\lambda$ ) is large enough, some **components** of  $\hat{\mathbf{w}}$  become **equal to zero**
- We can see regularization equivalent to minimizing  $\mathcal{L}_D(w)$  subject to constraint:

$$\sum_{j=0}^{M-1} |w_j| \leq \eta$$



# Lasso vs Ridge Regression

□ Why lasso leads to **sparse** model? Let's visualize constraint of lasso and ridge



## Least Squares and Maximum Likelihood

# Maximum Likelihood (ML)

- We can deal with regression in a **probabilistic way**
  - ▶ We define a probabilistic model that maps inputs ( $x$ ) to outputs ( $t$ )
  - ▶ Such probabilistic model,  $y(x, w)$ , will include some **unknown parameters** ( $w$ )
  - ▶ Then, we model the **likelihood**, i.e., the probability that observed data  $\mathcal{D}$  is generated by a given set of **parameters** ( $w$ ):

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

- ▶ Finally, we can estimate **parameters** ( $w$ ) by maximizing the likelihood:

$$\mathbf{w}_{ML} = \arg \max_{\mathbf{w}} p(\mathcal{D}|\mathbf{w})$$

# Maximum Likelihood (ML) for linear regression

- Our probabilistic model can be defined as:

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon = \mathbf{w}^T \phi(\mathbf{x}) + \epsilon$$

- ▶ we assumed a linear model for  $y(x, w)$
- ▶ we assumed  $\epsilon \sim \mathcal{N}(0, \sigma^2)$

- Given a dataset  $\mathcal{D}$  of  $N$  samples with inputs  $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  and outputs  $\mathbf{t} = \{t_1, \dots, t_N\}^T$ :

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

## Maximum Likelihood (ML) for linear regression (cont.)

□ To find  $\mathbf{w}_{ML}$  it is convenient to maximize the log-likelihood:

$$\mathcal{N}(t_n | \mathbf{w}^T \phi(\mathbf{x}_n), \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp \left\{ -\frac{1}{2\sigma^2} (t_n - \mathbf{w}^T \phi(\mathbf{x}_n))^2 \right\}$$

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

□ To solve the optimization problem, we equal the gradient to zero:

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

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

**OLS**



# Bayesian Linear Regression

# Bayesian approach

1. We formulate our knowledge about the world in a **probabilistic way**:
  - i. We define the **model** that expresses our knowledge qualitatively
  - ii. Our model will have some **unknown parameters**
  - iii. We capture our **assumptions** about unknown parameters with the **prior distribution** over those parameters before seeing the data
2. We observe the **data**
3. We compute the posterior probability distribution for the parameters, given observed data

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

4. We use the posterior distribution to:
  - a. Make predictions by averaging over the posterior distribution
  - b. Examine/Account for uncertainty in the parameter values
  - c. Make decisions by minimizing expected posterior loss

# Parameters Posterior Distribution

- The **posterior distribution** for the model parameters can be found by combining the prior with the likelihood for the parameters given data:

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



# Parameters Posterior Distribution

- The **posterior distribution** for the model parameters can be found by combining the prior with the likelihood for the parameters given data:

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

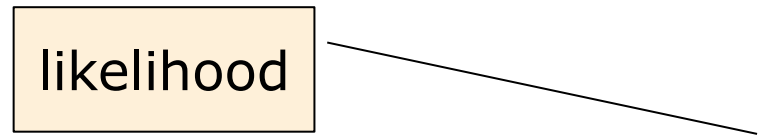


prior

- ▶  $p(\mathbf{w})$  is the **prior** probability over the parameter – what we know before observing the data

# Parameters Posterior Distribution

- The **posterior distribution** for the model parameters can be found by combining the prior with the likelihood for the parameters given data:


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

- ▶  $p(D|\mathbf{w})$  is the **likelihood** – the probability of observing the data ( $D$ ) given some value of the parameters ( $\mathbf{w}$ )

# Parameters Posterior Distribution

- The **posterior distribution** for the model parameters can be found by combining the prior with the likelihood for the parameters given data:

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

**normalizing constant**

$$p(\mathcal{D}) = \int p(\mathcal{D}|\mathbf{w})p(\mathbf{w})d\mathbf{w}$$

- $P(D)$  is the **marginal likelihood** and acts as **normalizing constant**

# Parameters Posterior Distribution

- The **posterior distribution** for the model parameters can be found by combining the prior with the likelihood for the parameters given data:

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

posterior



- ▶  $p(\mathbf{w}|\mathcal{D})$  is the **posterior probability** of parameters  $\mathbf{w}$  given training data
- ▶ the most probable value of  $\mathbf{w}$  given the data will be the mode of the posterior, also known as **maximum a posteriori (MAP)**.

# Bayesian Linear Regression

- How to model the **prior**? Assuming a Gaussian likelihood, a **conjugate prior** is the most convenient choice:

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

# Bayesian Linear Regression

- How to model the **prior**? Assuming a Gaussian likelihood, a **conjugate prior** is the most convenient choice:

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

- As a result, the **posterior** is still Gaussian:

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


# Bayesian Linear Regression

- How to model the **prior**? Assuming a Gaussian likelihood, a **conjugate prior** is the most convenient choice:

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

- As a result, the **posterior** is still Gaussian:

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




$$\left[ \begin{array}{l} p(\mathbf{w} | \mathbf{t}, \Phi, \sigma^2) = \mathcal{N}(\mathbf{w} | \mathbf{w}_N, \mathbf{S}_N) \\ \mathbf{w}_N = \mathbf{S}_N \left( \mathbf{S}_0^{-1} \mathbf{w}_0 + \frac{\Phi^T \mathbf{t}}{\sigma^2} \right) \\ \mathbf{S}_N^{-1} = \mathbf{S}_0^{-1} + \frac{\Phi^T \Phi}{\sigma^2} \end{array} \right.$$

# Bayesian Linear Regression and Maximum Likelihood

## □ Which parameters?

- ▶ The maximum a-posteriori (**MAP**) is the obvious choice
- ▶ When posterior is gaussian the **MAP** is equal to the mean

## □ When prior is **infinitely broad** MAP is equal to ML solution:

$$\lim_{S_0 \rightarrow \infty} \mathbf{w}_N = \left( \Phi^T \Phi \right)^{-1} \Phi^T \mathbf{t}$$

$$\lim_{S_0 \rightarrow \infty} \mathbf{S}_N^{-1} = \frac{\Phi^T \Phi}{\sigma^2}$$


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

- ▶ The ML estimate of **w** has the **smallest variance** among linear unbiased estimates and the **lowest MSE** among linear unbiased estimates (Gauss-Markov).



# Bayesian Linear Regression and Regularization

□ What about regularization?

► When  $\mathbf{w}_0=0$  and  $\mathbf{S}_0=\tau^2\mathbf{I}$ , we have:

$$\ln p(\mathbf{w}|\mathbf{t}) = -\frac{1}{2\sigma^2} \sum_{i=1}^N (t_i - \mathbf{w}^T \phi(\mathbf{x}_i))^2 - \frac{1}{2\tau^2} \|\mathbf{w}\|_2^2$$

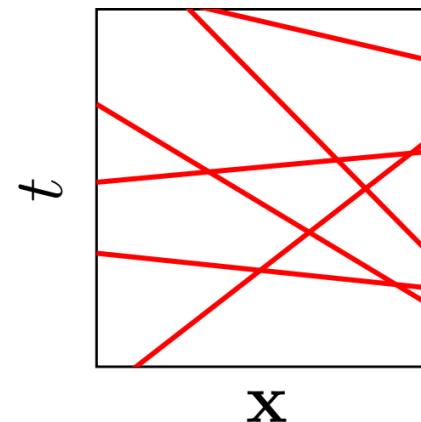
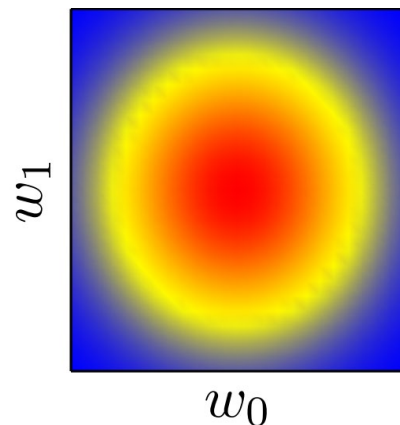
► In this case, **MAP** ( $\mathbf{w}_N$ ) is equivalent to the solution of ridge regression ( $\hat{\mathbf{w}}_{ridge}$ ) with  $\lambda = \frac{\sigma^2}{\tau^2}$

# Bayesian Linear Regression: an example

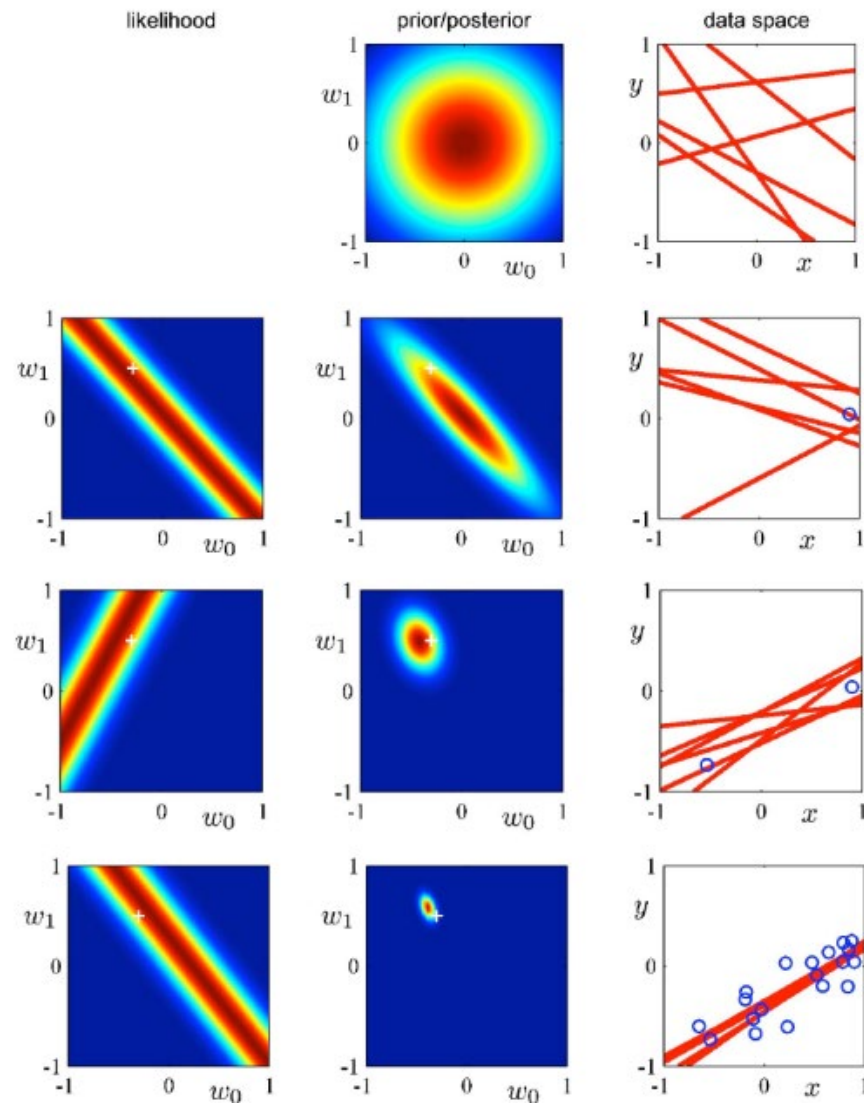
- How to exploit the Bayesian approach for sequential learning?
  - ▶ We compute **posterior** with initial data
  - ▶ When additional data is available, the **posterior** becomes the **prior**

# Bayesian Linear Regression: an example

- How to exploit the Bayesian approach for sequential learning?
  - ▶ We compute **posterior** with initial data
  - ▶ When additional data is available, the **posterior** becomes the **prior**
- Let see an example
  - ▶ Let assume data is generated as  $t(x) = -0.3 + 0.5x + \varepsilon$
  - ▶ Let assume that  $x \sim U(-1,1)$  and  $\varepsilon \sim \mathcal{N}(0,0.04)$
  - ▶ Let use as model:  $y(x, \mathbf{w}) = w_0 + w_1 x$
  - ▶ Let assume as prior:  $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}_0, \tau^2 \mathbf{I})$  with  $\tau^2 = 0.5$  and  $\mathbf{w}_0 = [0,0]^T$



# Bayesian Linear Regression: an example (2)



Step 0: 0 samples observed

Step 1: 1 samples observed

Step 2: 2 samples observed

Step 20: 20 samples observed

# Predictive Distribution

- With a Bayesian framework, it is possible to compute the probability distribution of the target for a new sample  $\mathbf{x}^*$  (given the training data  $\mathcal{D}$ ), by **integrating over the posterior distribution**:

$$p(t^*|\mathbf{x}^*, \mathcal{D}) = \mathbb{E}[t^*|\mathbf{x}^*, \mathbf{w}, \mathcal{D}] = \int p(t^*|\mathbf{x}^*, \mathbf{w}, \mathcal{D})p(\mathbf{w}|\mathcal{D})d\mathbf{w}$$

- This is often called **predictive distribution**
- Unfortunately, computing the predictive distribution requires knowledge of the posterior distribution, which is usually **intractable**

# Predictive Distribution for Bayesian Regression

□ With our assumptions, we can compute the **predictive distribution**:

$$p(t|\mathbf{x}, \mathcal{D}, \sigma^2) = \int p(t|\mathbf{x}, \mathbf{w}, \sigma^2) p(\mathbf{w}|\mathbf{w}_N, \mathbf{S}_N) d\mathbf{w} = \int \mathcal{N}(t|\mathbf{w}^T \phi(\mathbf{x}), \sigma^2) \mathcal{N}(\mathbf{w}|\mathbf{w}_N, \mathbf{S}_N) d\mathbf{w}$$

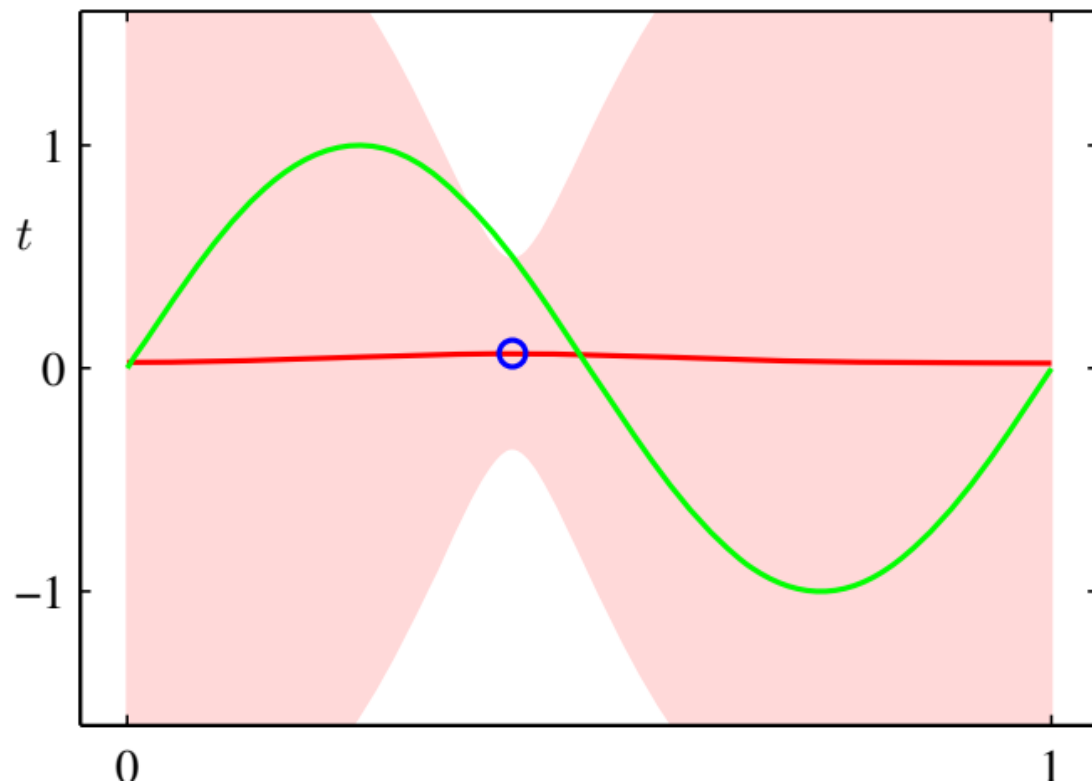
$$p(t|\mathbf{x}, \mathcal{D}, \sigma^2) = \mathcal{N}(t|\mathbf{w}_N^T \phi(\mathbf{x}), \sigma_N^2(\mathbf{x}))$$

$$\Rightarrow \sigma_N^2(\mathbf{x}) = \underbrace{\sigma^2}_{\text{data}} + \underbrace{\phi(\mathbf{x})^T \mathbf{S}_N \phi(\mathbf{x})}_{\text{parameters}}$$

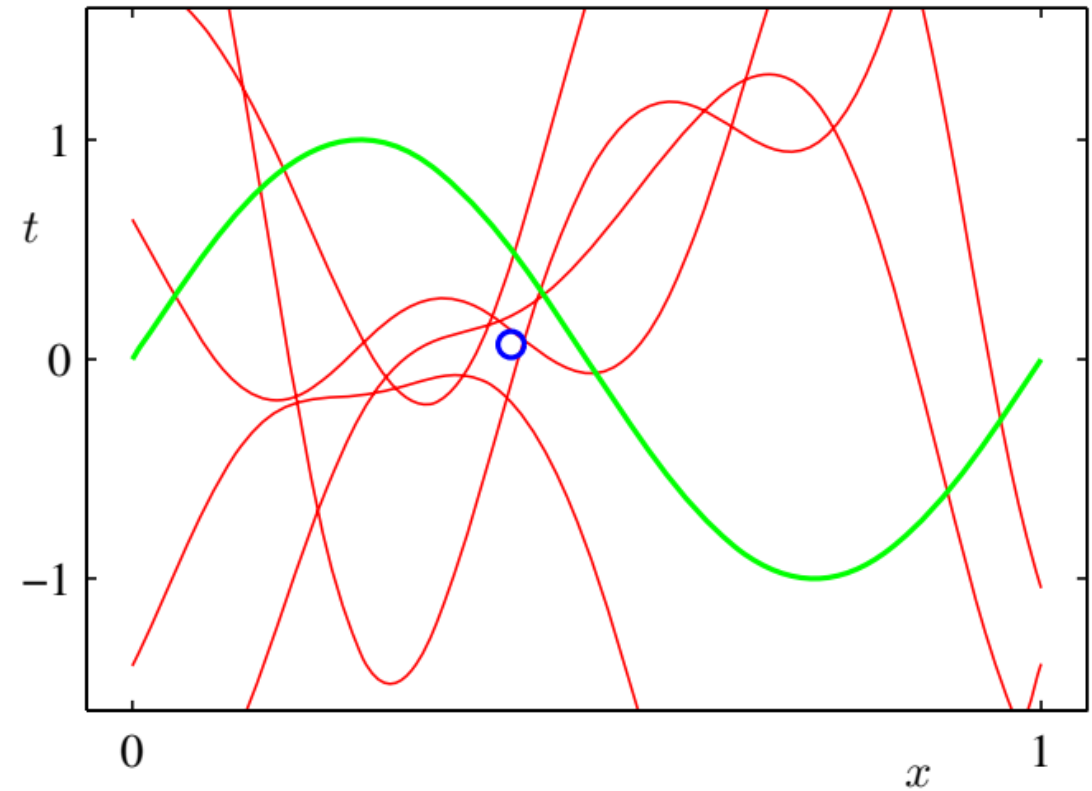
- ▶ when  $N \rightarrow \infty$  the uncertainty associated to parameters (second term) goes to zero and the variance of predictive distribution depends only on variance of data ( $\sigma^2$ )

# Predictive Distribution: an example

- Approximating a sinusoidal dataset with a linear model w/ 9 Gaussian basis functions: 1 sample observed



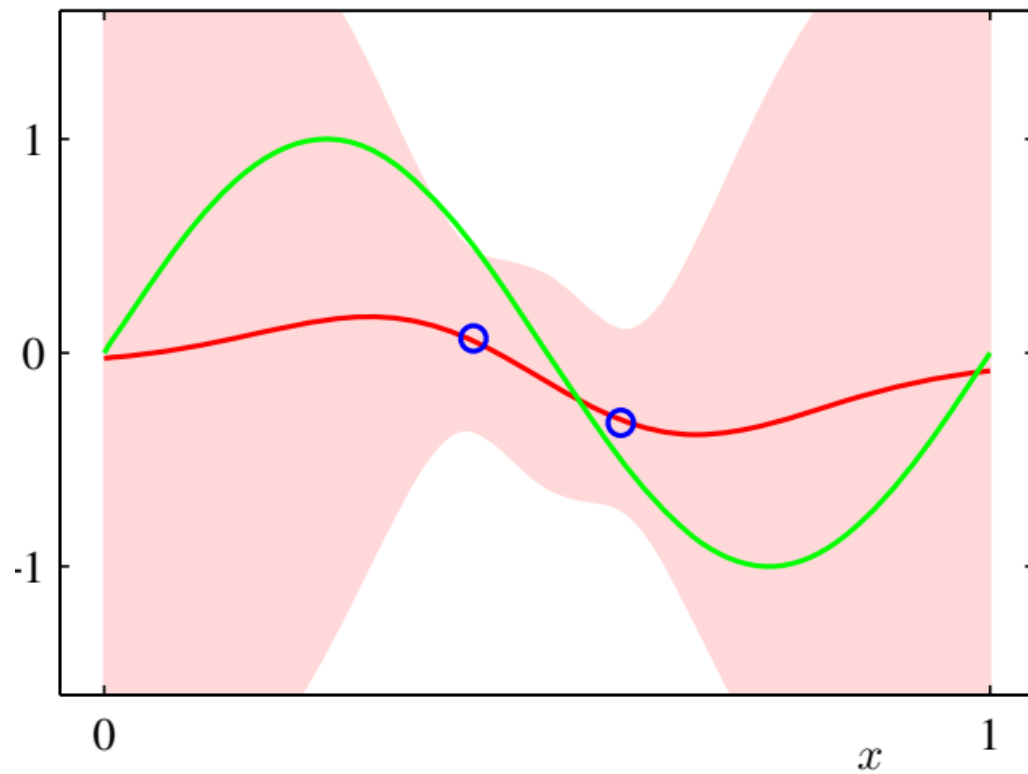
Predictive Distribution



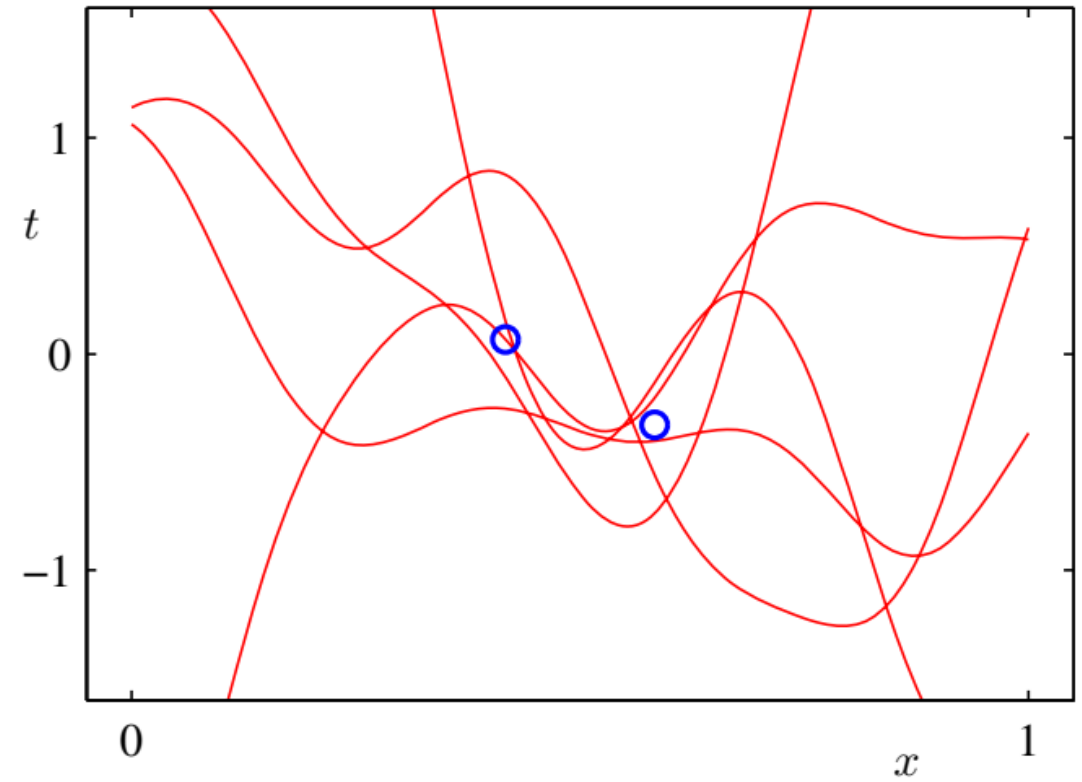
Sampled Models

# Predictive Distribution: an example

- Approximating a sinusoidal dataset with a linear model w/ 9 Gaussian basis functions: 2 samples observed



Predictive Distribution

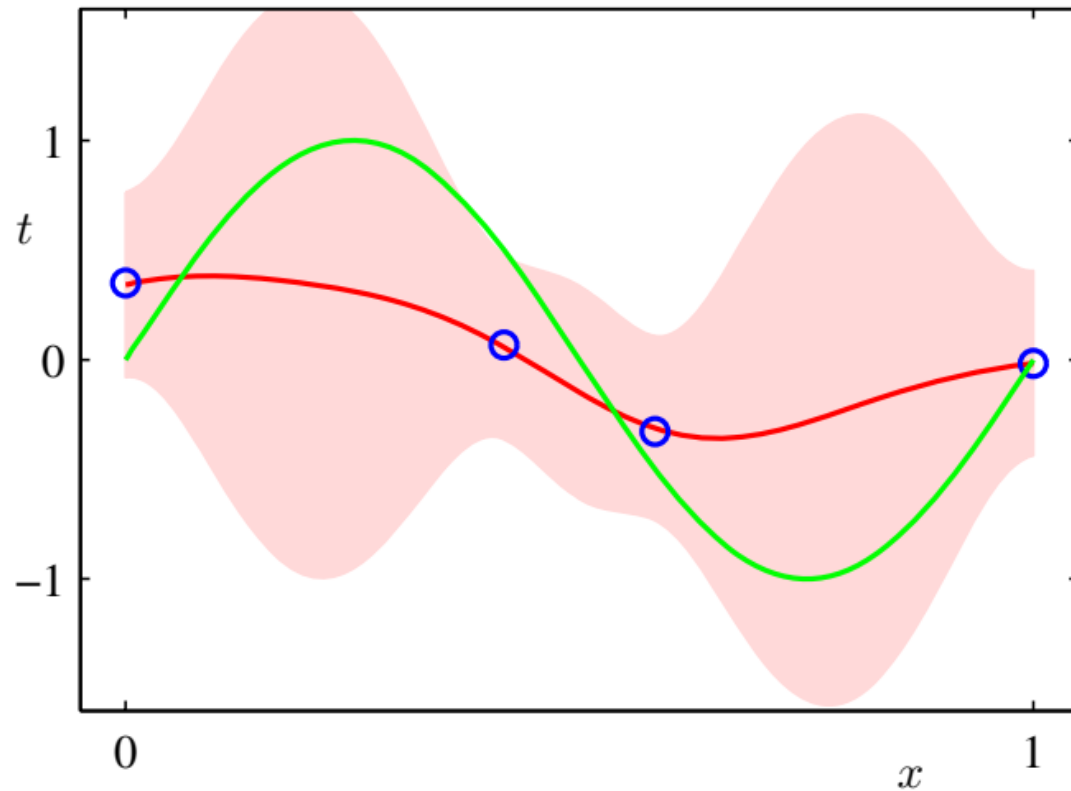


Sampled Models

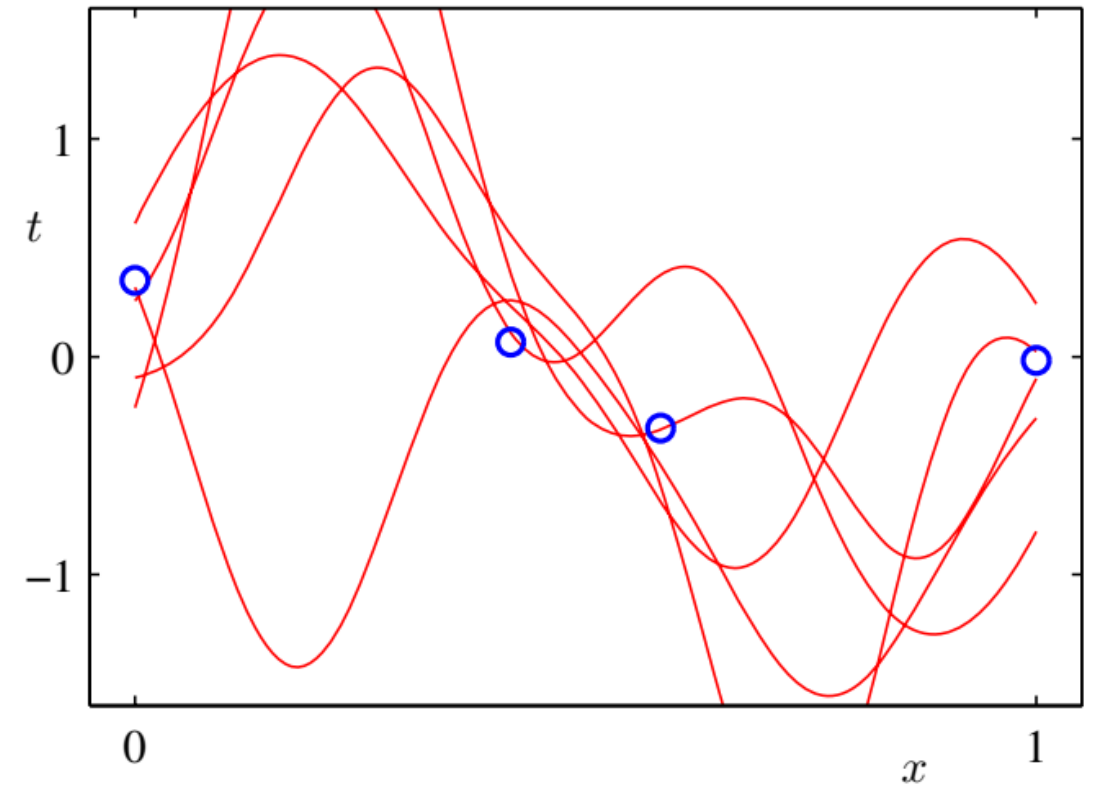


# Predictive Distribution: an example

- Approximating a sinusoidal dataset with a linear model w/ 9 Gaussian basis functions: 4 samples observed



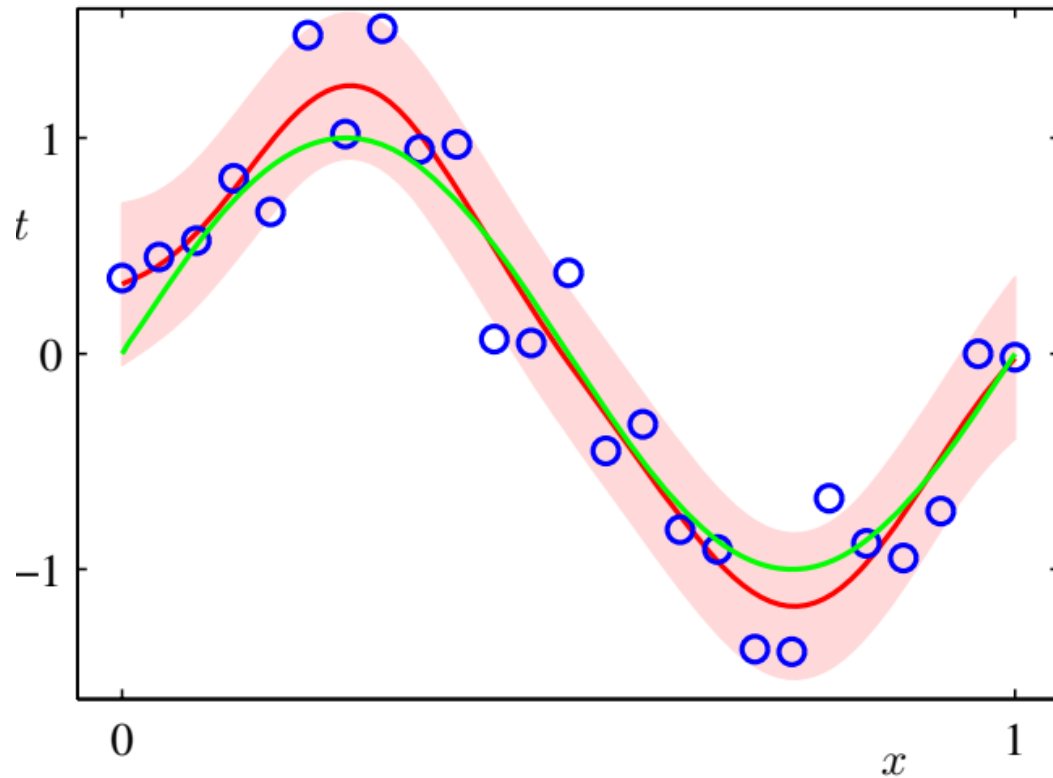
Predictive Distribution



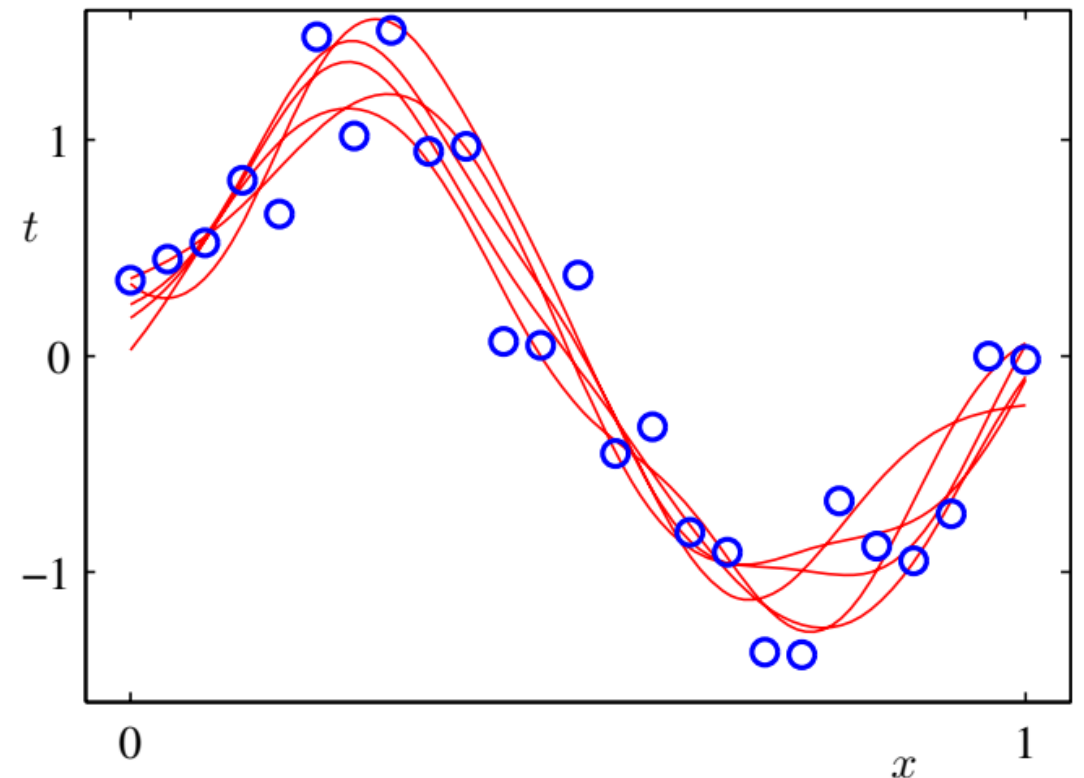
Sampled Models

# Predictive Distribution: an example

- Approximating a sinusoidal dataset with a linear model w/ 9 Gaussian basis functions: 25 samples observed



Predictive Distribution



Sampled Models

## Challenges and Limitation of Linear Regression

# Challenges

## ❑ Modeling Challenges

- ▶ Our model should fit well all the functions (we think) are **likely**
- ▶ The prior should not give zero or small probabilities to possible values, but at the same time also avoid spreading out the probability (**uninformative**)

## ❑ Computational Challenges

- ▶ **Analytical integration** is possible only using **conjugate priors** and works for **simple models**
- ▶ **Approximated** approaches are instead to be used in the more general case, such as Gaussian (Laplace) approximation, Monte Carlo integration, and Variational approximation

# Limitation of Fixed Basis Functions

- ❑ Linear models with fixed basis functions have several advantages
  - ▶ Allow closed-form solution
  - ▶ Tractable **Bayesian treatment**
  - ▶ Can model non-linear relationship with proper basis function
- ❑ However, they have also several limitations
  - ▶ Basis functions are not *adaptive* with respect to the training data
  - ▶ These models suffer of the **curse of dimensionality**