LECTURE 6

SPRING 2021

APPLIED MACHINE LEARNING

CIHANG XIE

SLIDE CREDIT:
NARGES NOROUZI

BASIS FUNCTIONS EXERCISE

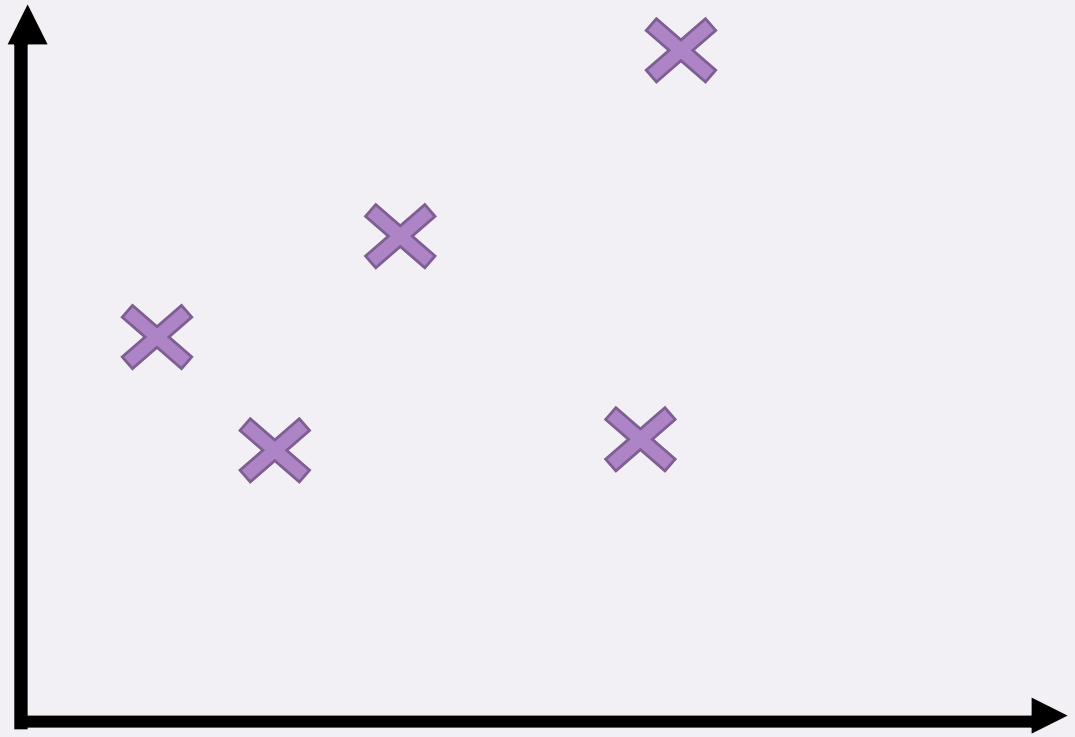
1.2.2 Implementation of the basis function

We have already set up the data for linear regression. In the following cell, we add another dimension to our data to accommodate the θ_0 intercept term. Do NOT execute this cell more than once. Define the basis function in the following block. For example, if your basis function is log(x), you should change the basis into np.log(x).

```
[ ]  def basis_func(x):
       ## define your function here
       x_new = x ** 2
       return np.stack([x, x_new], axis=1)
```

1) do not discuss with your peers about how to perform gradient descent if basis functions are applied;
2) do not include any codes related to basis functions in your week 3 group activities

LINEAR REGRESSION



GRADIENT DESCENT

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial Cost(\theta)}{\partial \theta_j} \qquad \text{(simultaneous update for } \theta_0, \theta_1, \dots, \theta_d)$$

- For linear regression:

$$\frac{\partial Cost(\theta)}{\partial \theta_j} = \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^{n} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)^2$$

With $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_d x_d = \sum_{j=0}^{d} \theta_j x_j$

GRADIENT DESCENT

$$\frac{\partial Cost(\theta)}{\partial \theta_j} = \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^{n} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)^2$$

$$= \frac{1}{2n} \quad \frac{\partial}{\partial \theta_j} \sum_{i=1}^{n} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)^2$$

Scalar multiple rule

$$= \frac{1}{2n} \sum_{i=1}^{n} \quad \frac{\partial}{\partial \theta_j} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)^2$$

Sum rule

$$= \frac{1}{2n} \sum_{i=1}^{n} \quad 2\left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) \frac{\partial}{\partial \theta_j} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)$$

Power rule

$$= \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right) \quad \frac{\partial}{\partial \theta_j} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right)$$

$$= \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right) x_j^{(i)}$$

LINEAR BASIS FUNCTION MODEL

- Generally,

$$h_\theta(x) = \sum_{j=0}^{d} \theta_j \phi_j(x)$$

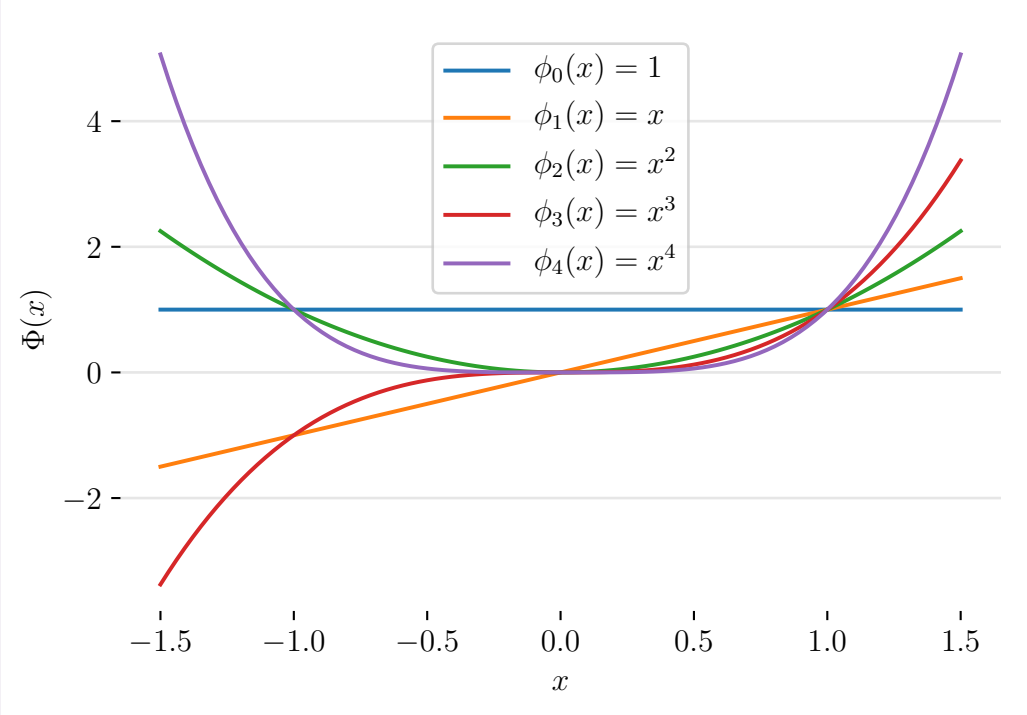
Basis Function

- Typically, $\phi_0(x) = 1$ so that θ_0 acts as a bias.
- In the simplest case, we can use linear basis function:

$$\phi_j(x) = x_j$$

- Polynomial basis function: $\phi_j(x) = x^j$
- Gaussian basis function: $\phi_j(x) = e^{-\frac{\left(x-\mu_j\right)^2}{2s^2}}$

EXAMPLE - POLYNOMIAL BASIS FUNCTION

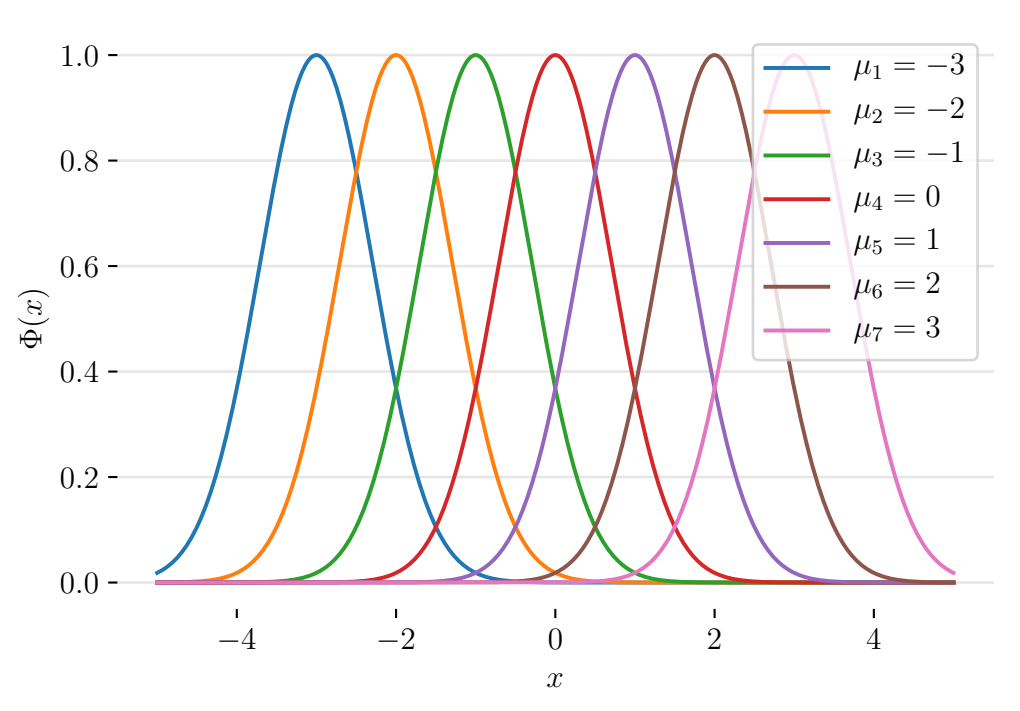


(a) Polynomial basis out to degree 4.

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 = \sum_{j=0} \theta_j x^j$$

Image credit: https://www.cs.princeton.edu/courses/archive/fall18/cos324/files/basis-functions.pdf

EXAMPLE - GAUSSIAN BASIS FUNCTION



(a) Examples of Gaussian-type radial basis functions.

$$y = \theta_0 + \theta_1 e^{-\frac{(x-\mu_1)^2}{2s^2}} + \cdots + \theta_7 e^{-\frac{(x-\mu_7)^2}{2s^2}}$$

Image credit: https://www.cs.princeton.edu/courses/archive/fall18/cos324/files/basis-functions.pdf

TODAY

- Vector Calculations
- Regularization
- Overfitting and Underfitting

VECTOR CALCULATIONS

LINEAR ALGEBRA CONCEPTS

- Vector in $\mathbb{R}^d$ is a set of d real numbers.

$$v = [1, 2, 3, 4] \text{ is in } \mathbb{R}^4 \text{ and is a column vector.}$$

- An m-by-n matrix is an object with m rows and n columns, where each entry is a real number

$$\begin{bmatrix} 1 & 0 & 2 \\ 4 & 0.5 & 7.8 \end{bmatrix}$$

- Transposing the matrix

$$\begin{bmatrix} a \\ b \end{bmatrix}^T = [a \quad b], \qquad \begin{bmatrix} a & b \\ c & d \end{bmatrix}^T = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

Note: $(Ax)^T = x^T A^T$

LINEAR ALGEBRA CONCEPTS

- Vector norms:
  - L_p norm of $v = (v_1, v_2, \ldots, v_k) = (\sum_i |v_i|^p)^{\frac{1}{p}}$
  - Common norms L_1 and L_2
  - Length of the vector v is $L_2(v)$

LINEAR ALGEBRA CONCEPTS

- Vector products:
  - Inner or Dot product:

$$u^T v = [u_1 \;\; u_2].\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = u_1 \times v_1 + u_2 \times v_2$$

  - Outer product:

$$uv^T = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}[v_1 \;\; v_2] = \begin{bmatrix} u_1 v_1 & u_1 v_2 \\ u_2 v_1 & u_2 v_2 \end{bmatrix}$$

LINEAR ALGEBRA CONCEPTS

- Matrix product:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \qquad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

$$A \times B = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

VECTORIZATION

- Benefits of vectorization
  - More compact equations
  - Faster code (using optimized matrix libraries)
- Consider our model: $h_\theta(x) = \sum_{j=0}^{d} \theta_j x_j$
- Let $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}$, $x = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}$

We can write our model in vectorized form as $h_\theta(x) = x^T\theta$

VECTORIZATION

- Consider our model for n instances: $h\left(x^{(i)}\right) = \sum_{j=0}^{d} \theta_j x_j^{(i)}$
- Let

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \in \mathbb{R}^{d+1 \times 1}, \qquad X = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_d^{(2)} \\ \vdots & & \ddots & \vdots \\ 1 & x_1^{(n)} & \cdots & x_d^{(n)} \end{bmatrix} \in \mathbb{R}^{n \times (d+1)}$$

- We can write the vectorized form as $h_\theta(X) = X \times \theta$

VECTORIZATION

- For the linear regression cost function:

$$Cost(\theta) = \frac{1}{2\times n}\sum_{i=1}^{n}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$$= \frac{1}{2\times n}\sum_{i=1}^{n}\left(x^{(i)^T}\theta - y^{(i)}\right)^2$$

$$= \frac{1}{2\times n}(X\times\theta - Y)^T(X\times\theta - Y)$$

- Note that:

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^{n\times 1}, \qquad X \in \mathbb{R}^{n\times d+1}, \qquad \theta \in \mathbb{R}^{d+1\times 1} \rightarrow \quad X\times\theta \in \mathbb{R}^{n\times 1}$$

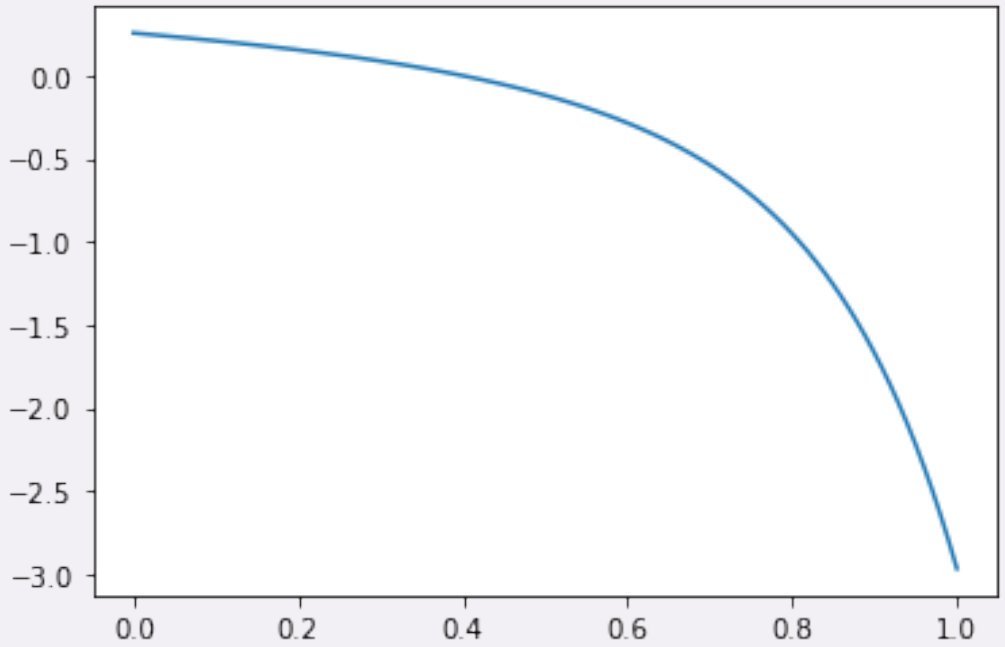
$$X\times\theta - Y \in \mathbb{R}^{n\times 1}, \quad (X\times\theta - Y)^T \quad \in \mathbb{R}^{1\times n} \rightarrow Cost(\theta) \in \mathbb{R}$$

REGULARIZATION

POLYNOMIAL BASIS FUNCTION

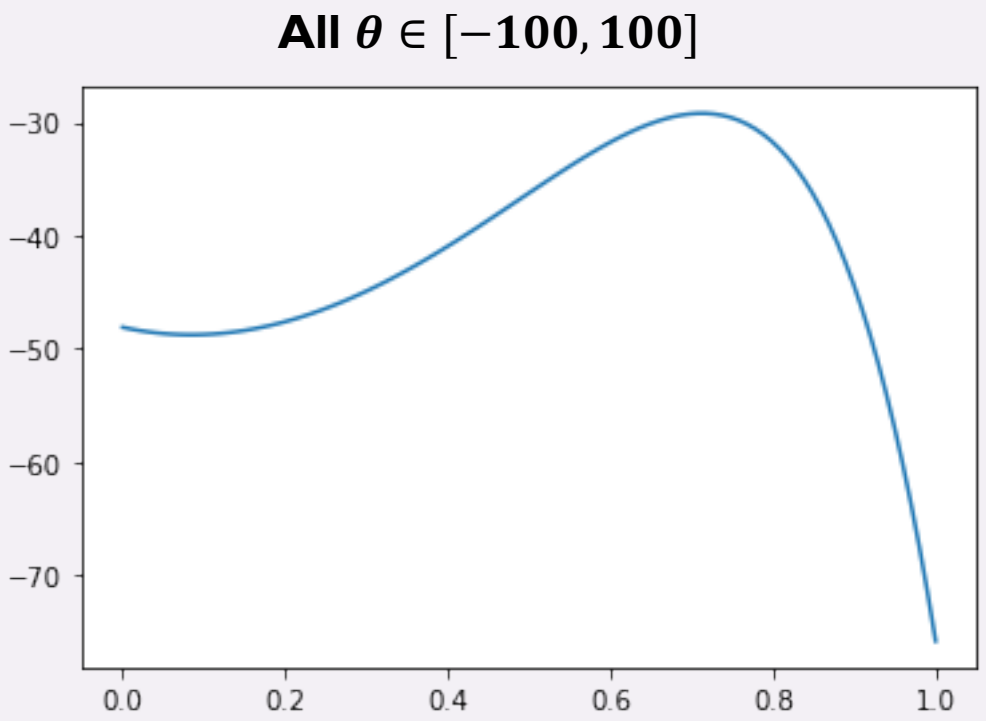
$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5 + \theta_6 x^6 + \theta_7 x^7 + \theta_8 x^8 + \theta_9 x^9$$

All $\theta \in [-1, 1]$



POLYNOMIAL BASIS FUNCTION

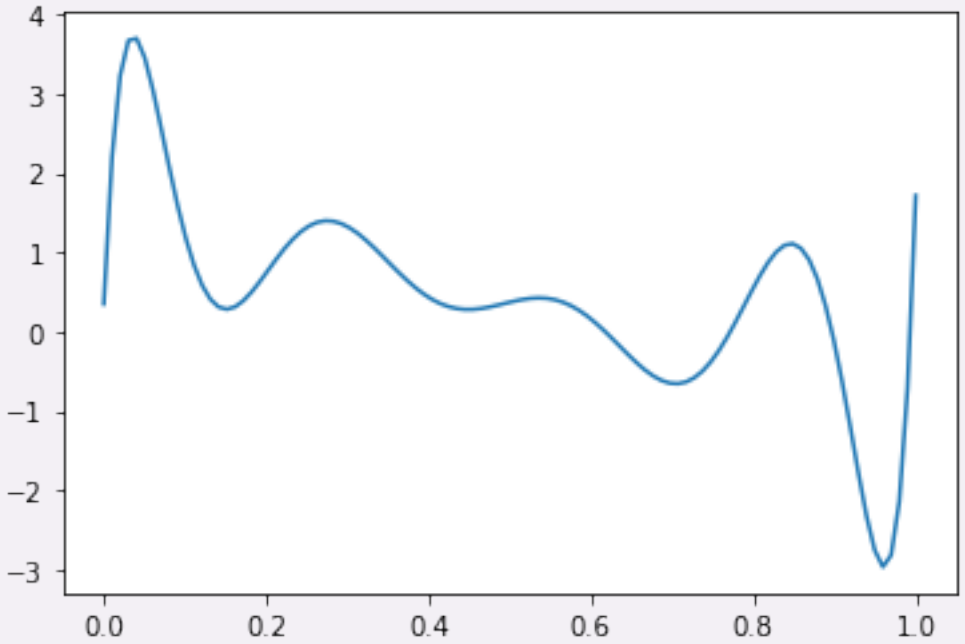
$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5 + \theta_6 x^6 + \theta_7 x^7 + \theta_8 x^8 + \theta_9 x^9$$



POLYNOMIAL BASIS FUNCTION

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5 + \theta_6 x^6 + \theta_7 x^7 + \theta_8 x^8 + \theta_9 x^9$$

All $\theta \in [-100,000, 100,000]$



REGULARIZERS

- Generally, we don't want huge weights
- If weights are large, a small change in a feature can result in a large change in the prediction
- Also gives too much weight to any one feature
- Might also prefer weights of 0 for features that aren't useful

How do we encourage small weights? or penalize large weights?

REGULARIZATION

- A method for automatically controlling the complexity of the learned hypothesis
- Idea: penalize large values of θ_j
- There are other techniques we will learn later such as dropout technique in neural networks or dimensionality reduction.

REGULARIZATION

- Linear regression objective function

$$Cost(\theta) = \left\{ \frac{1}{2 \times n} \sum_{i=1}^{n} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)^2 \right\}$$

Model fit to data

Regularization

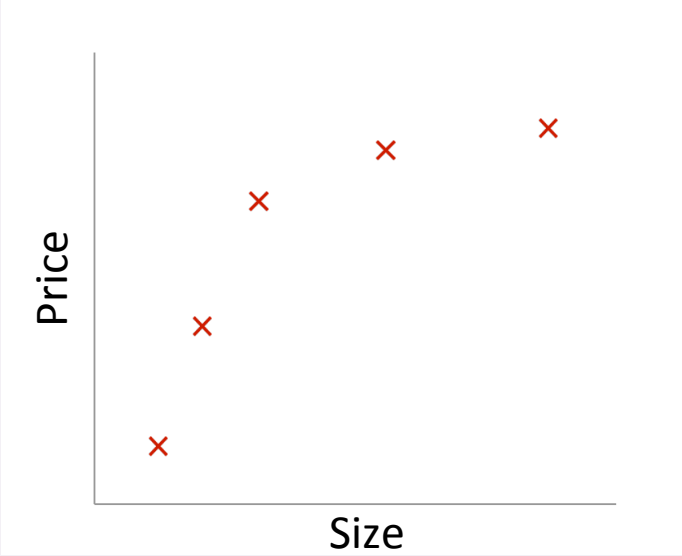
- λ is the regularization parameter ($\lambda \geq 0$)
- No regularization on θ_0
- This exact regularizer pulls coefficients/parameters to 0

5 MINUTES BREAK

UNDERSTANDING REGULARIZATION

- What happens if we set λ too huge?

$$Cost(\theta) = \frac{1}{2\times n}\sum_{i=1}^{n}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{d}\theta_j^2$$

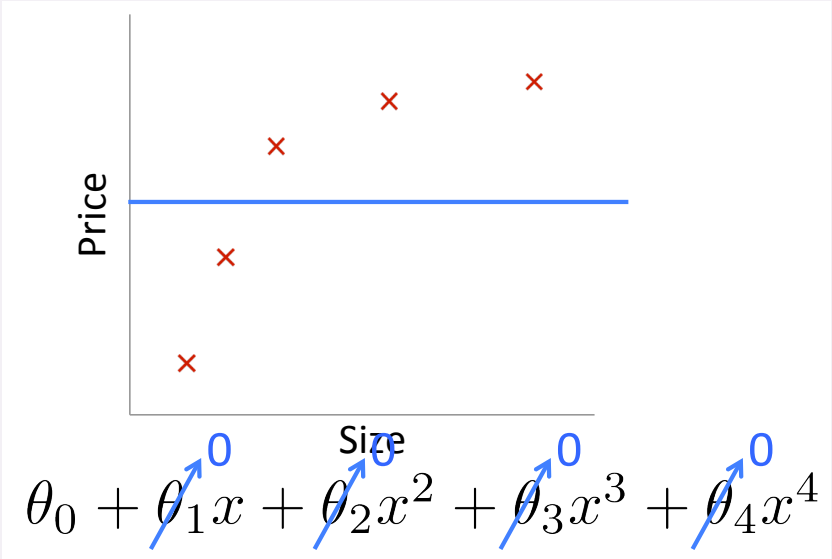


$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

UNDERSTANDING REGULARIZATION

- What happens if we set λ too huge?

$$Cost(\theta) = \frac{1}{2\times n}\sum_{i=1}^{n}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{d}\theta_j^2$$



REGULARIZATION

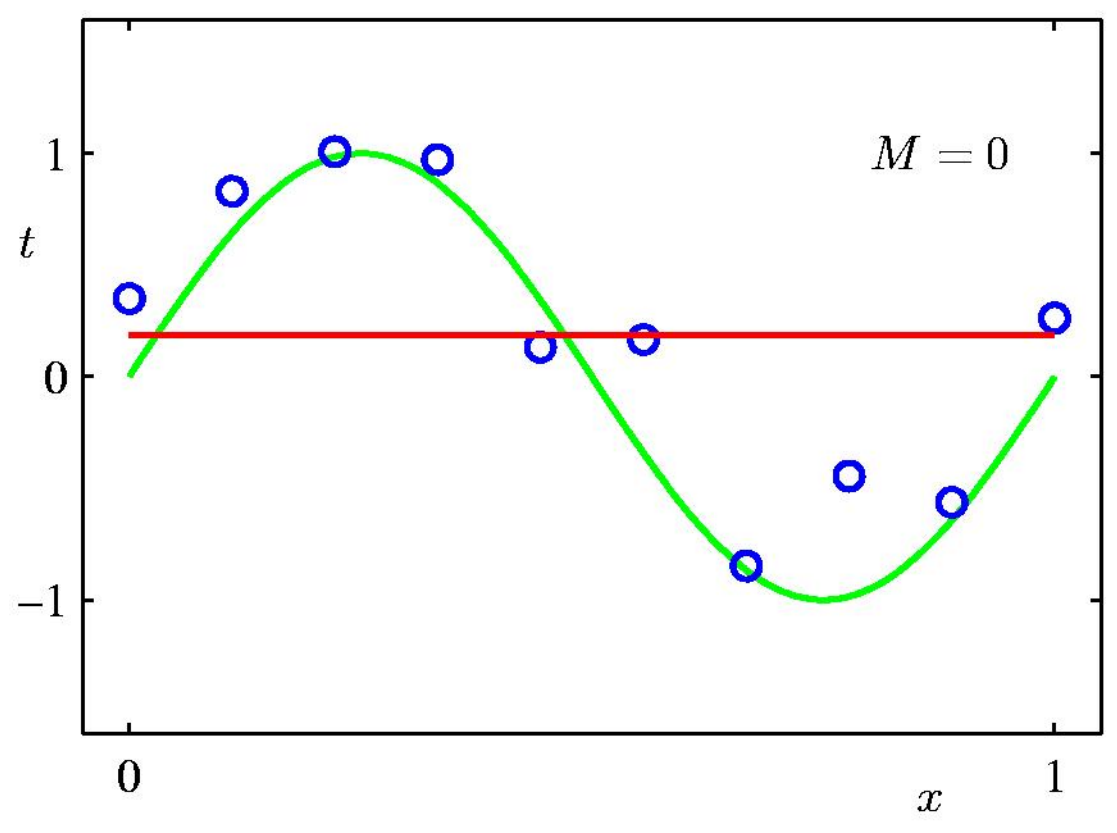
- Linear regression objective function

$$Cost(\theta) = \frac{1}{2\times n}\sum_{i=1}^{n}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{d}\theta_j^2$$

Model fit to data ⟺ Regularization

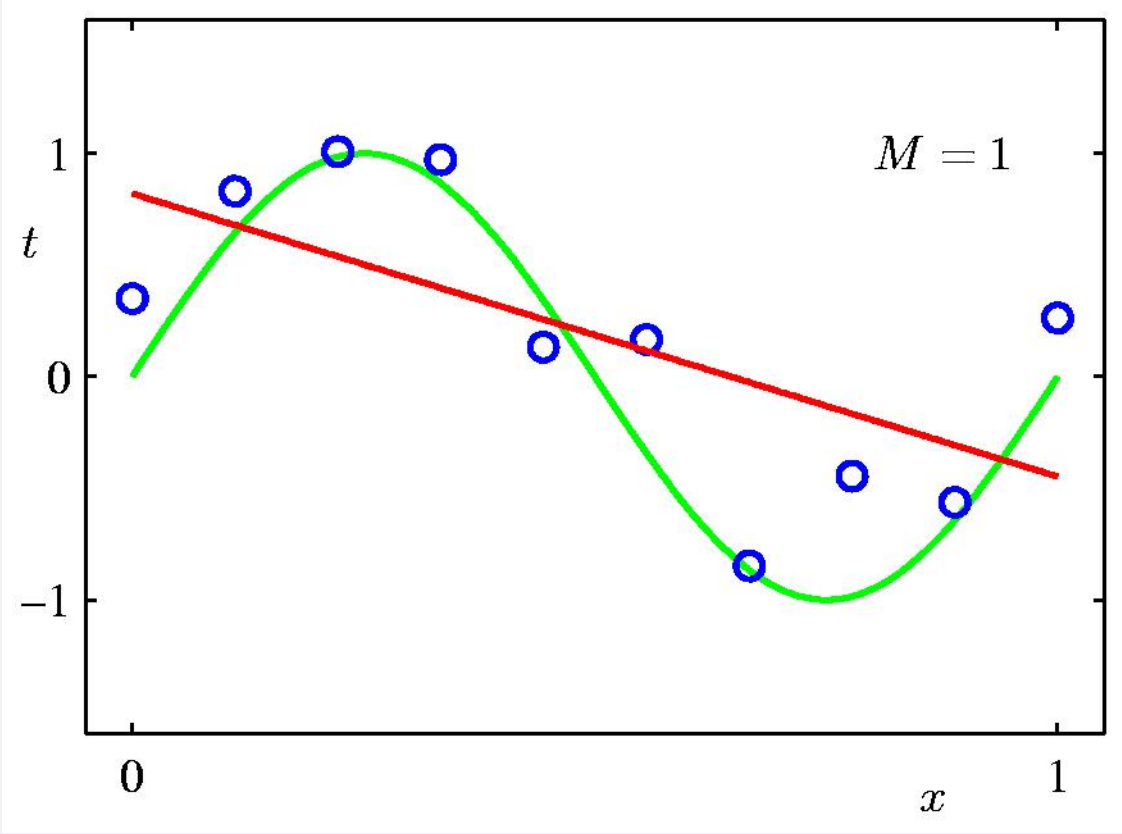
Need a balance

0TH ORDER POLYNOMIAL

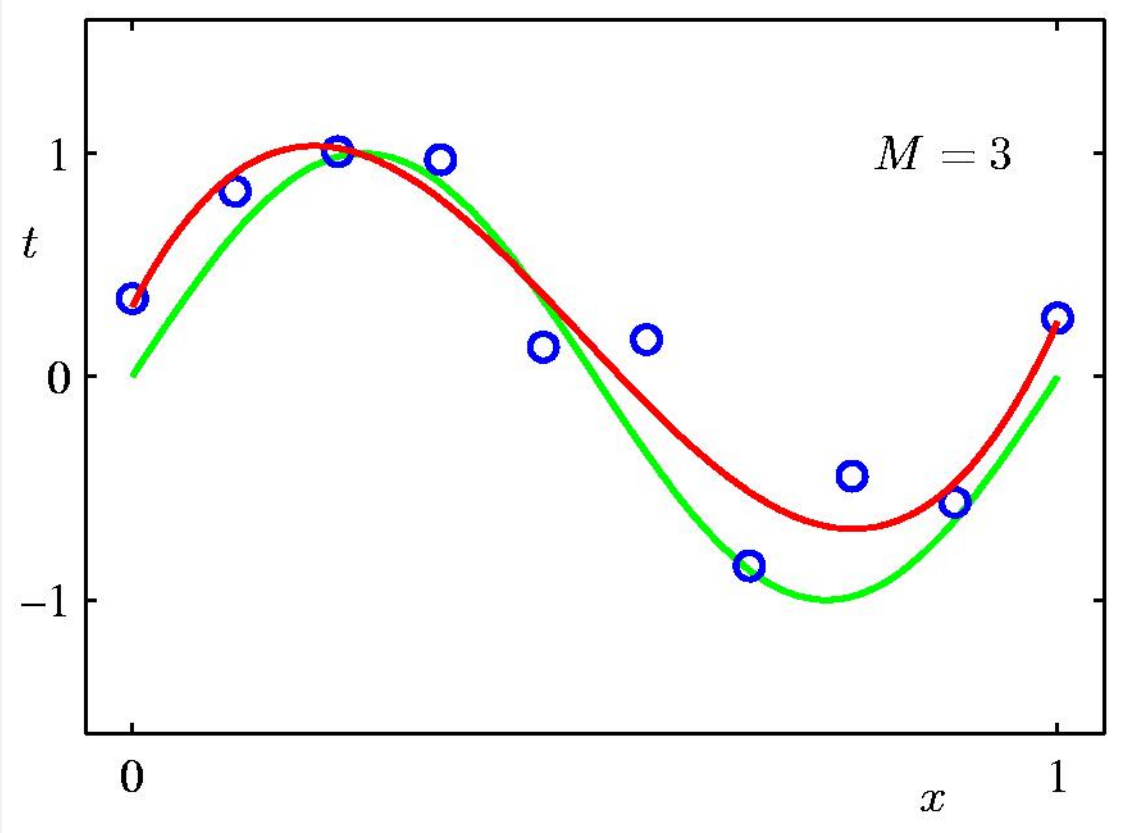


N = 10

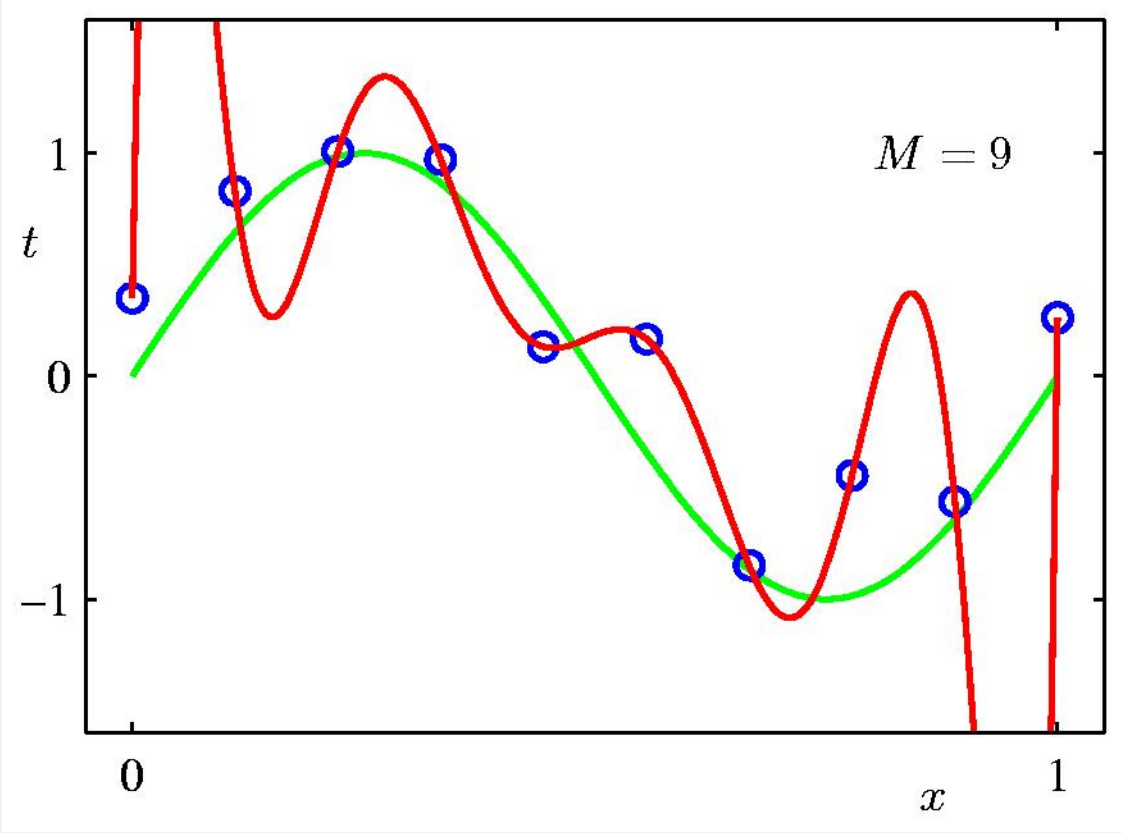
1ST ORDER POLYNOMIAL



3RD ORDER POLYNOMIAL



9TH ORDER POLYNOMIAL



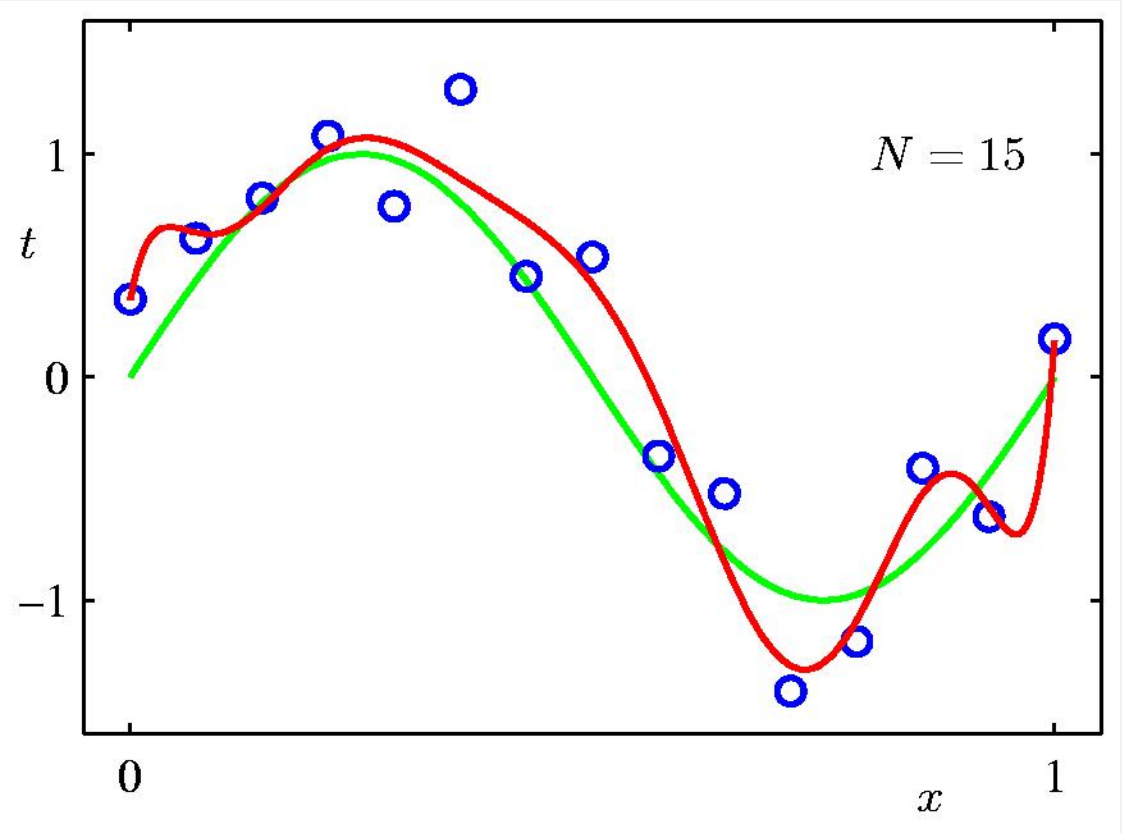
POLYNOMIAL COEFFICIENTS

	$M=0$	$M=1$	$M=3$	$M=9$
θ_0	0.19	0.82	0.31	0.35
θ_1		-1.27	7.99	232.37
θ_2			-25.43	-5321.83
θ_3			17.37	48568.31
θ_4				-231639.30
θ_5				640042.26
θ_6				-1061800.52
θ_7				1042400.18
θ_8				-557682.99
θ_9				125201.43

DATA SET SIZE

$$N = 15$$

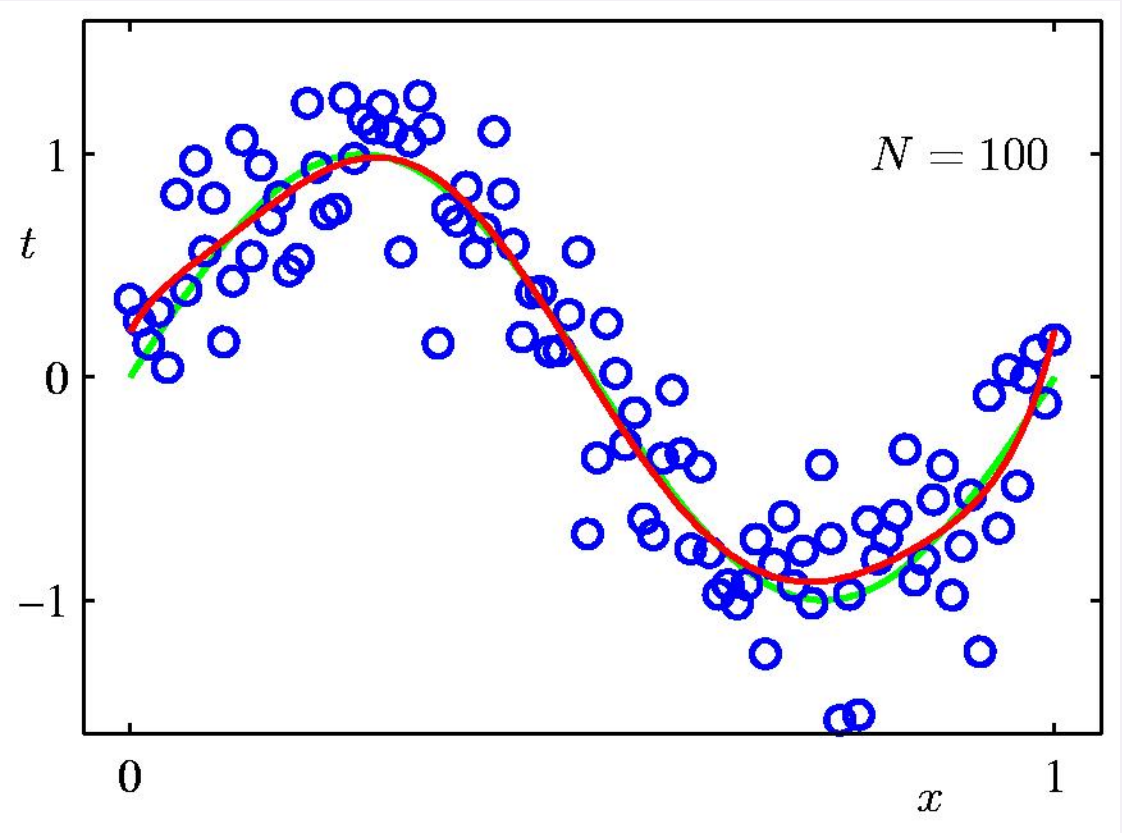
9th Order Polynomial



DATA SET SIZE

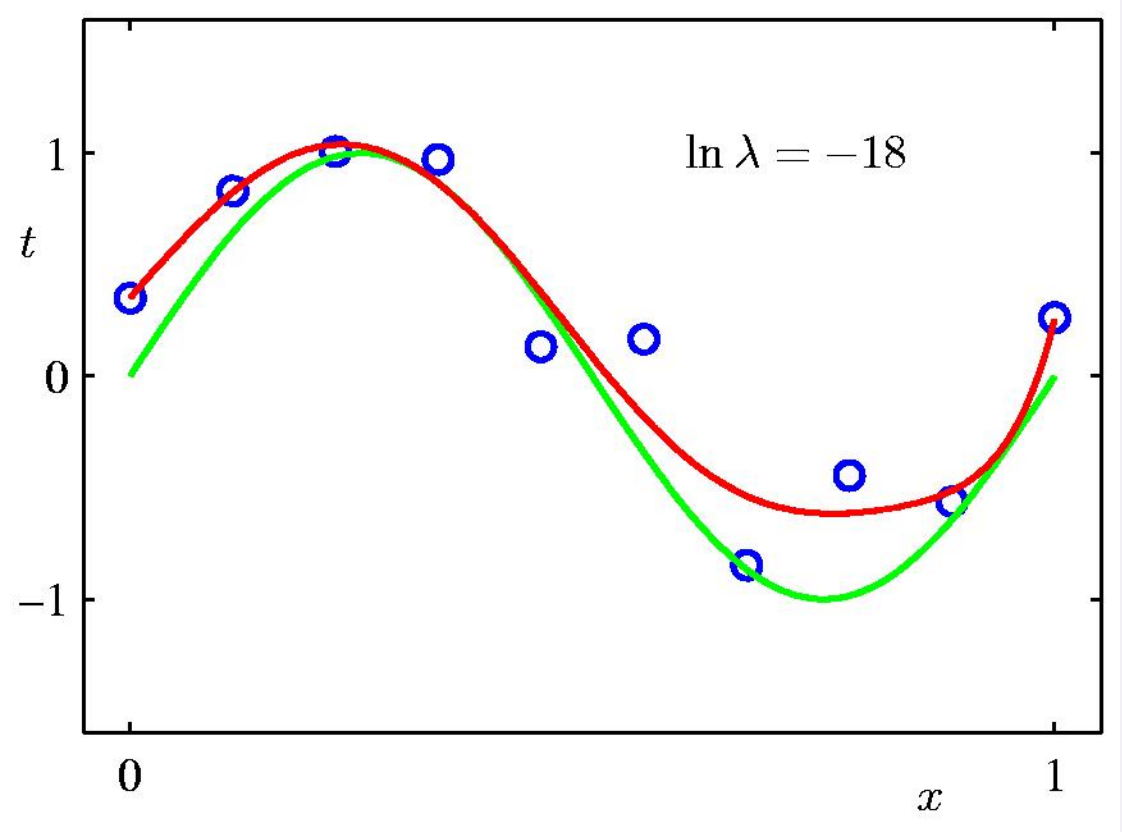
$$N = 100$$

9th Order Polynomial



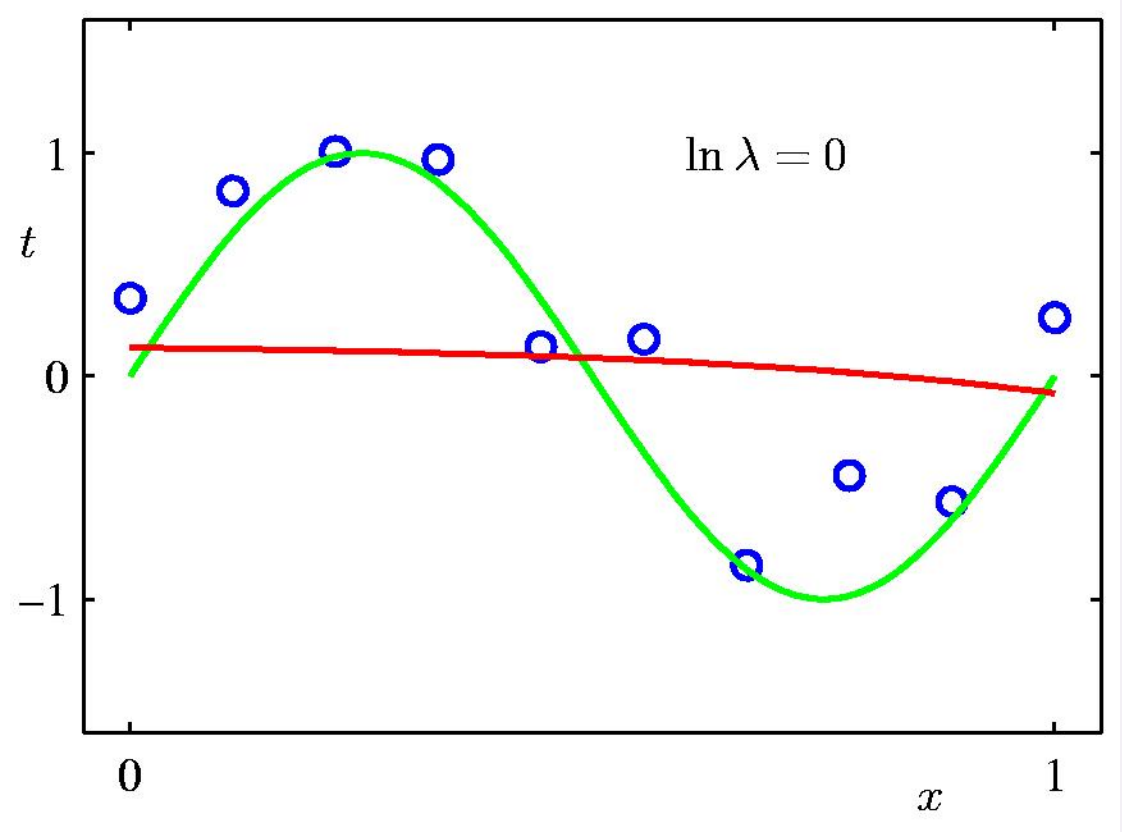
REGULARIZATION

$$\ln \lambda = -18$$

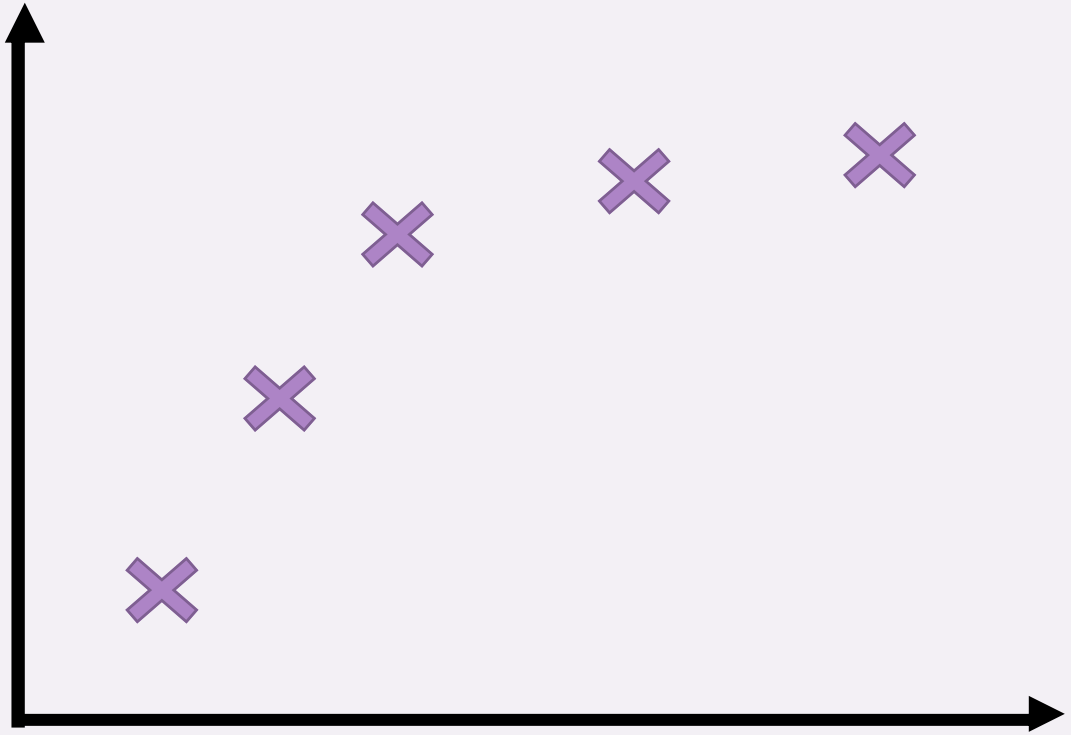


OVER-REGULARIZATION

$$\ln \lambda = 0$$

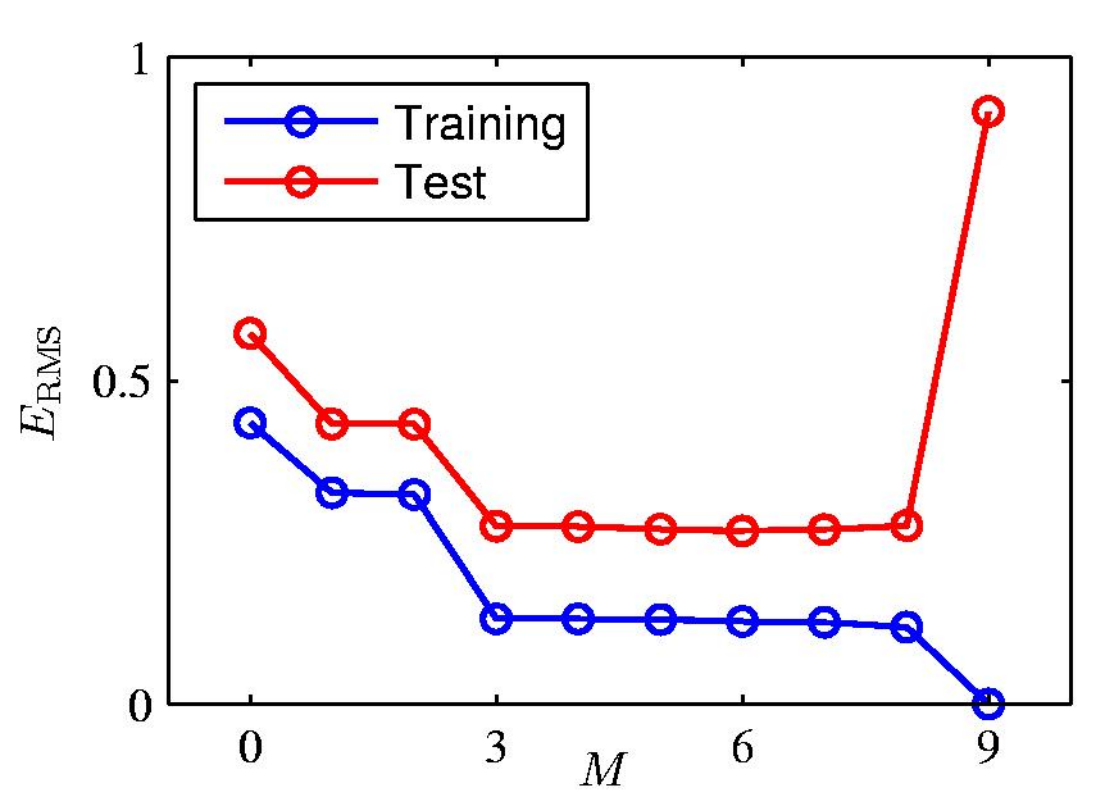


QUALITY OF FIT



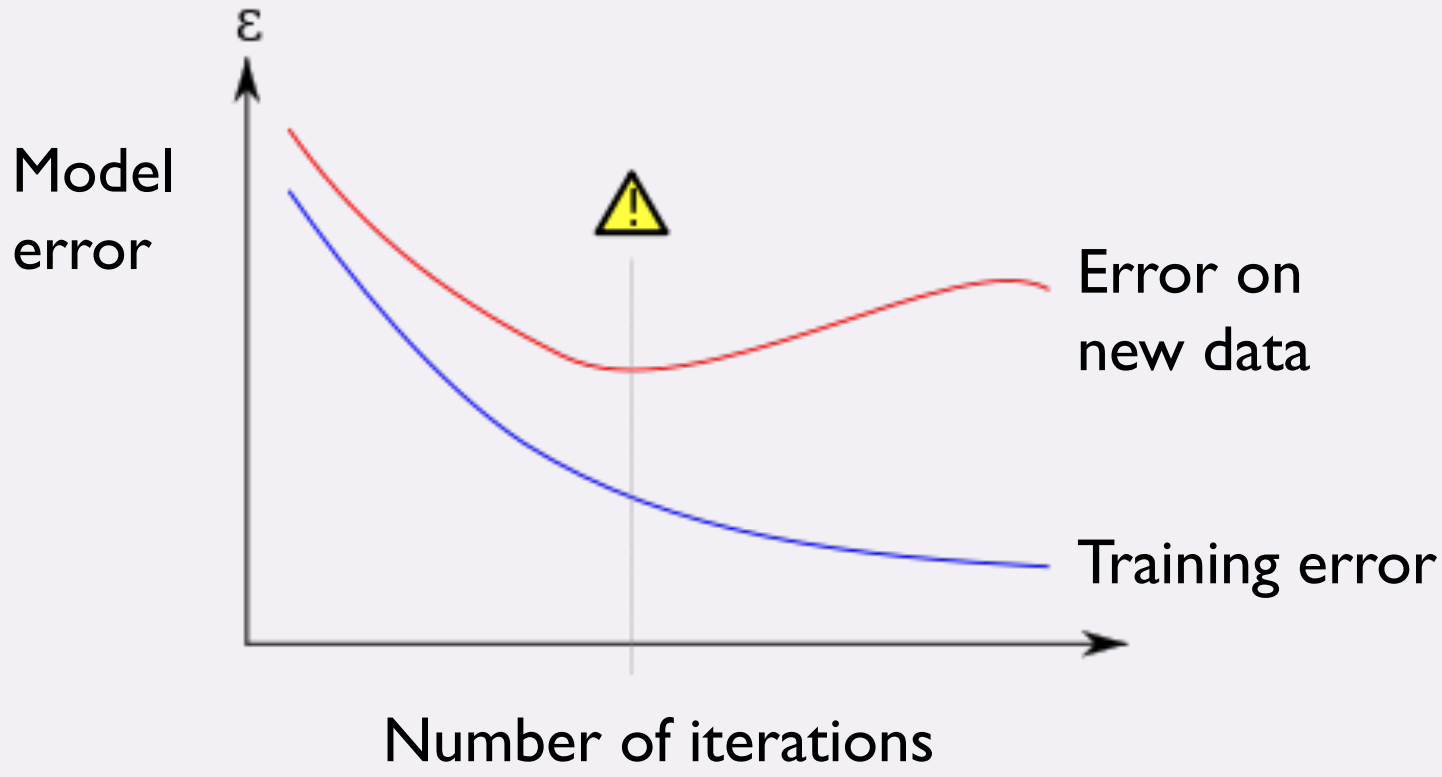
- Overfitting: the learned hypothesis may fit the training set very well
  - but fails to generalize to new examples

OVER-FITTING



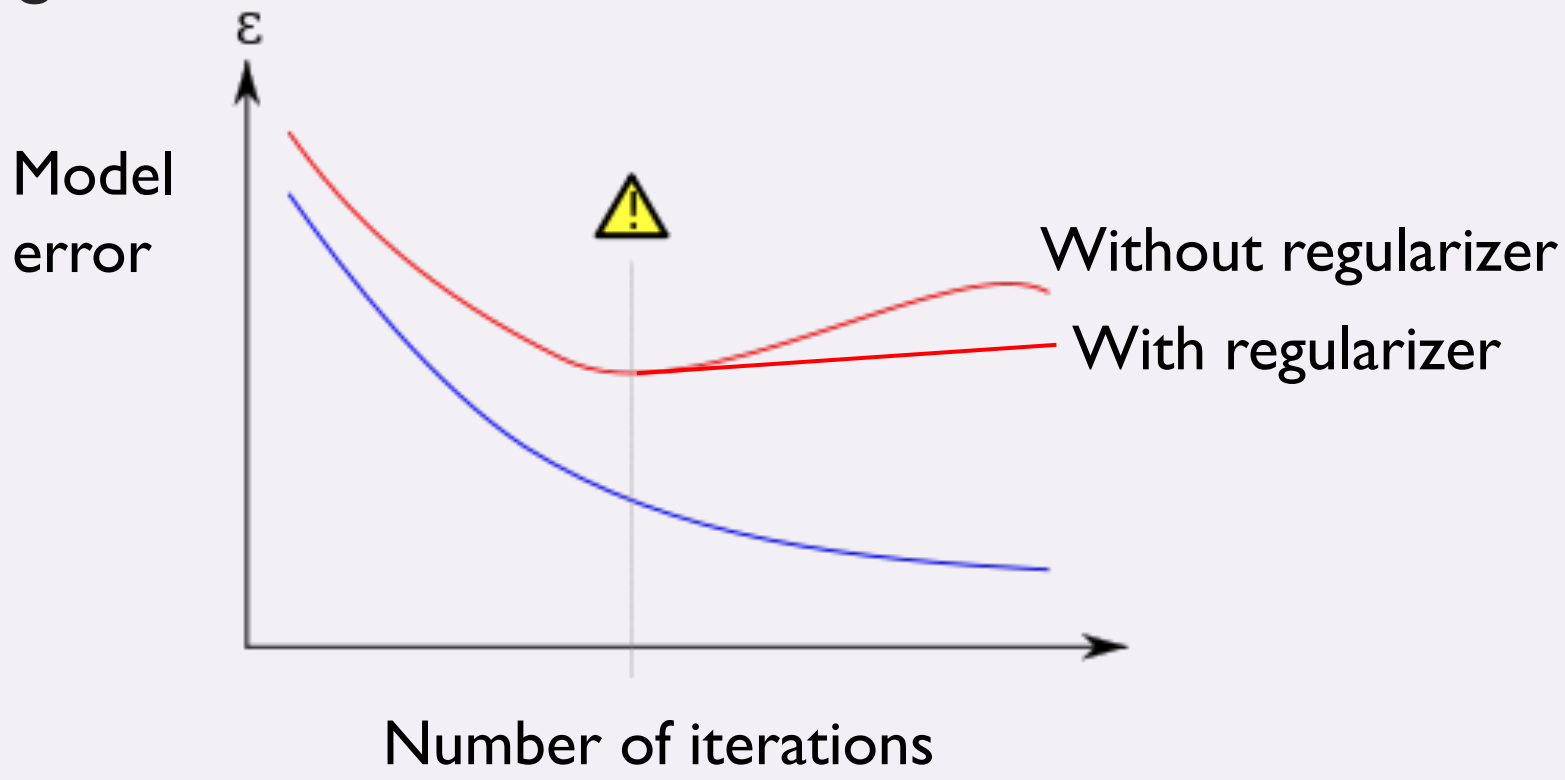
OVER-FITTING

Over-fitting during training



REGULARIZATION AND OVER-FITTING

Adding a regularizer:



QUESTIONS?