Training Linear Model

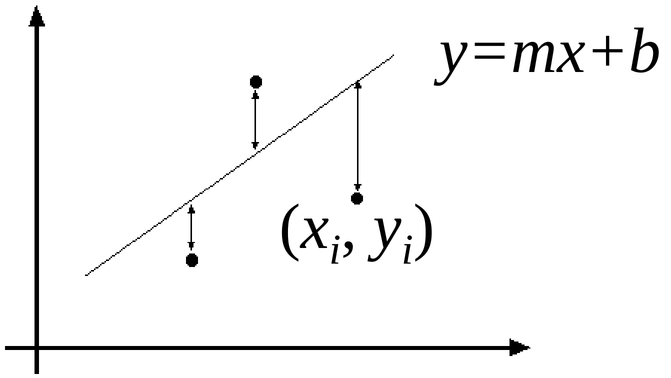
Chul Min Yeum
Assistant Professor
Civil and Environmental Engineering
University of Waterloo, Canada

CIVE 497 – CIVE 700: Smart Structure Technology

Last updated: 2021-04-01

UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING

Recall: Least Squares Line Fitting



Data (measurement): $(x_1, y_1), \ldots, (x_n, y_n)$

Model: Line $f(x_i, m, b) = mx_i + b$

Task: Find (m, b)

Minimize $$E = J(m,b) = \sum_{i=1}^{n} (y_i - f(x_i, m, b))^2 = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$

$$\frac{\partial(E)}{\partial m} = -2 \sum_{i=1}^{n} [y_i - mx_i - b] x_i = 0$$

$$\mathrm{m} = \frac{\sum_{i=1}^{n} x_i y_i - 1/n\left(\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i\right)}{\sum_{i=1}^{n} x_i^2 - 1/n\left(\sum_{i=1}^{n} x_i\right)^2}$$

$$\frac{\partial(E)}{\partial b} = -2 \sum_{i=1}^{n} [y_i - mx_i - b] = 0$$

$$\mathrm{b} = \frac{1/n\left(\sum_{i=1}^{n} y_i\right)\left(\sum_{i=1}^{n} x_i^2\right) - 1/n \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2 - 1/n\left(\sum_{i=1}^{n} x_i\right)^2}$$

Linear Regression

The linear model for regression is one that involves a linear combination of the input variables

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \ \dots + w_D x_D$$

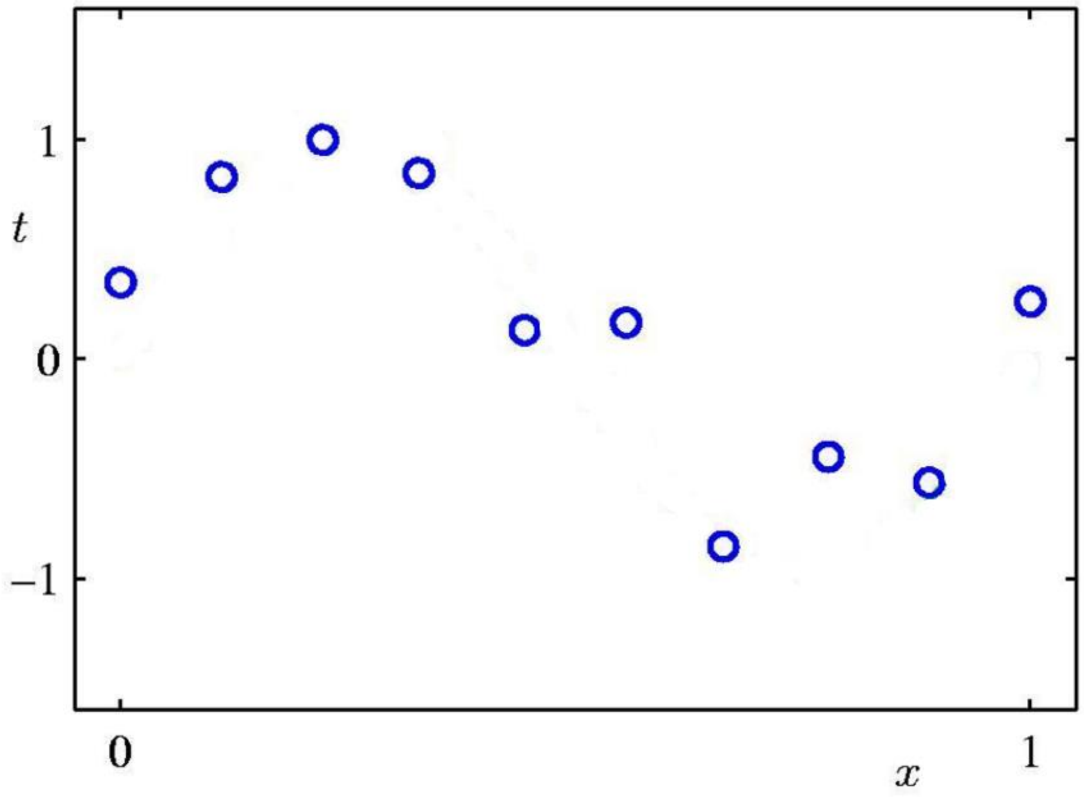
where $\mathbf{x} = (x_1, \dots, x_D)^T$. This is often simply known as ***linear regression***. The key property of this model is that it is a linear function of the parameters, $w_1, \dots, w_D$. When D becomes 1, it is called ***simple linear regression.***

The polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modelled as an *nth degree polynomial in x*

$$y(x, \boldsymbol{\beta}) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_M x^N$$

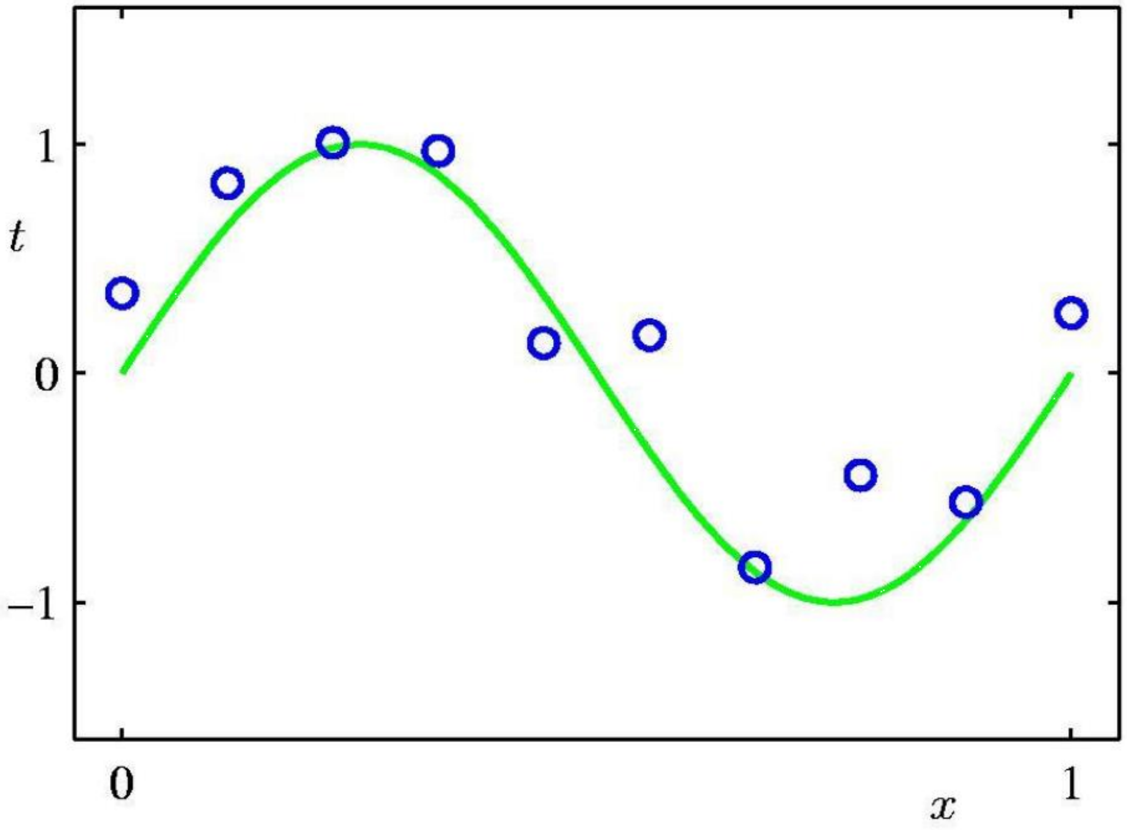
Although polynomial regression fits a nonlinear model to the data, the function is linear in terms of parameters and thus, it is considered as linear model.

Polynomial Curve Fitting



- Suppose we are given *N* observations $(x_1, t_1), (x_2, t_2), \dots, (x_N, t_N)$
- We are going to find a function $y = f(x)$ to estimate t from x

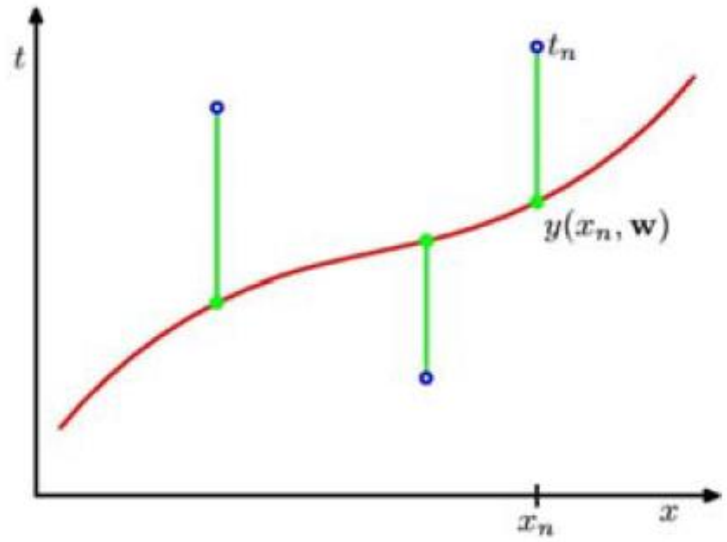
Polynomial Curve Fitting (Continue)



- The green curve is the true function.
- It looks like a polynomial but the true curve is generated from a sin function.

Polynomial Curve Fitting (Continue)

Data (measurement): $(x_1, t_1), \ldots, (x_N, t_N)$



Model: M^{th} order Polynomial

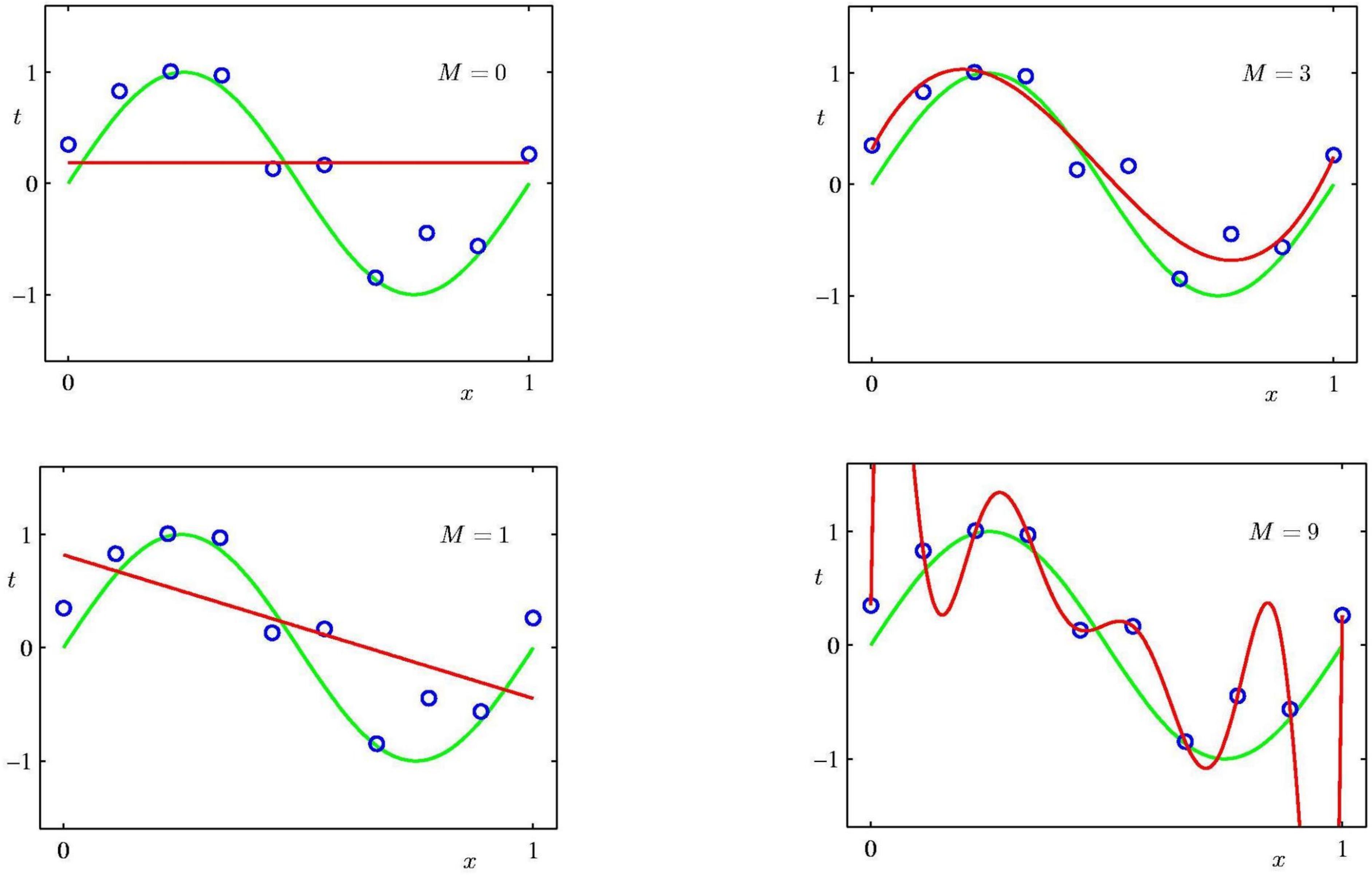
$$y(x, \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x$$

Task: Find the order of the polynomial and its coefficients

Minimize $$E = J(\boldsymbol{w}) = \frac{1}{2} \sum_{i=0}^{N} \{y(x_i, \boldsymbol{w}) - t_i\}^2$$

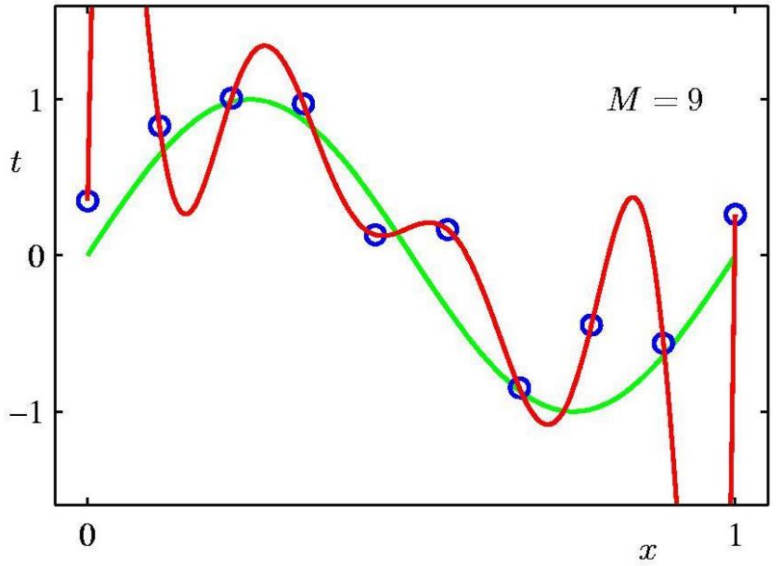
A loss function that measures the squared error in the prediction of y(x) from x.

Some Fits to the Data: Which is Best?



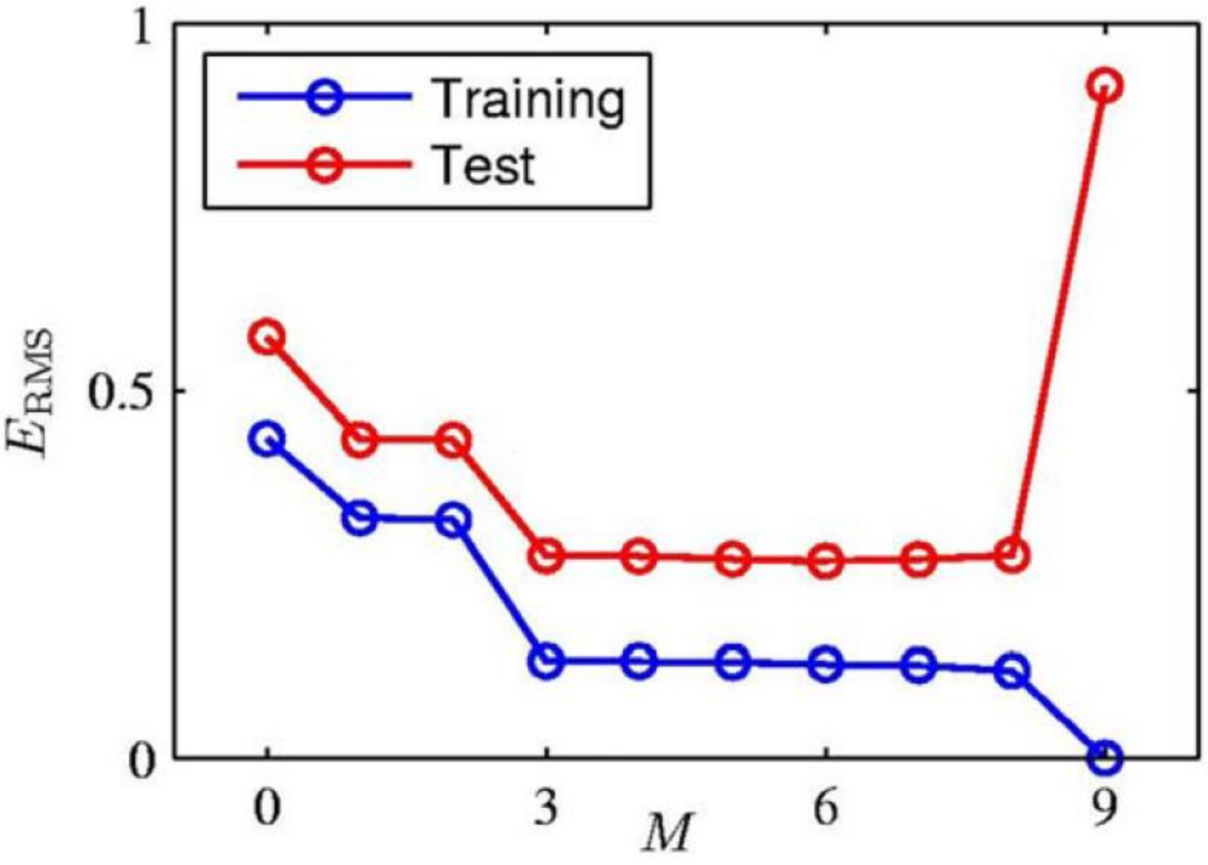
Polynomial Coefficients

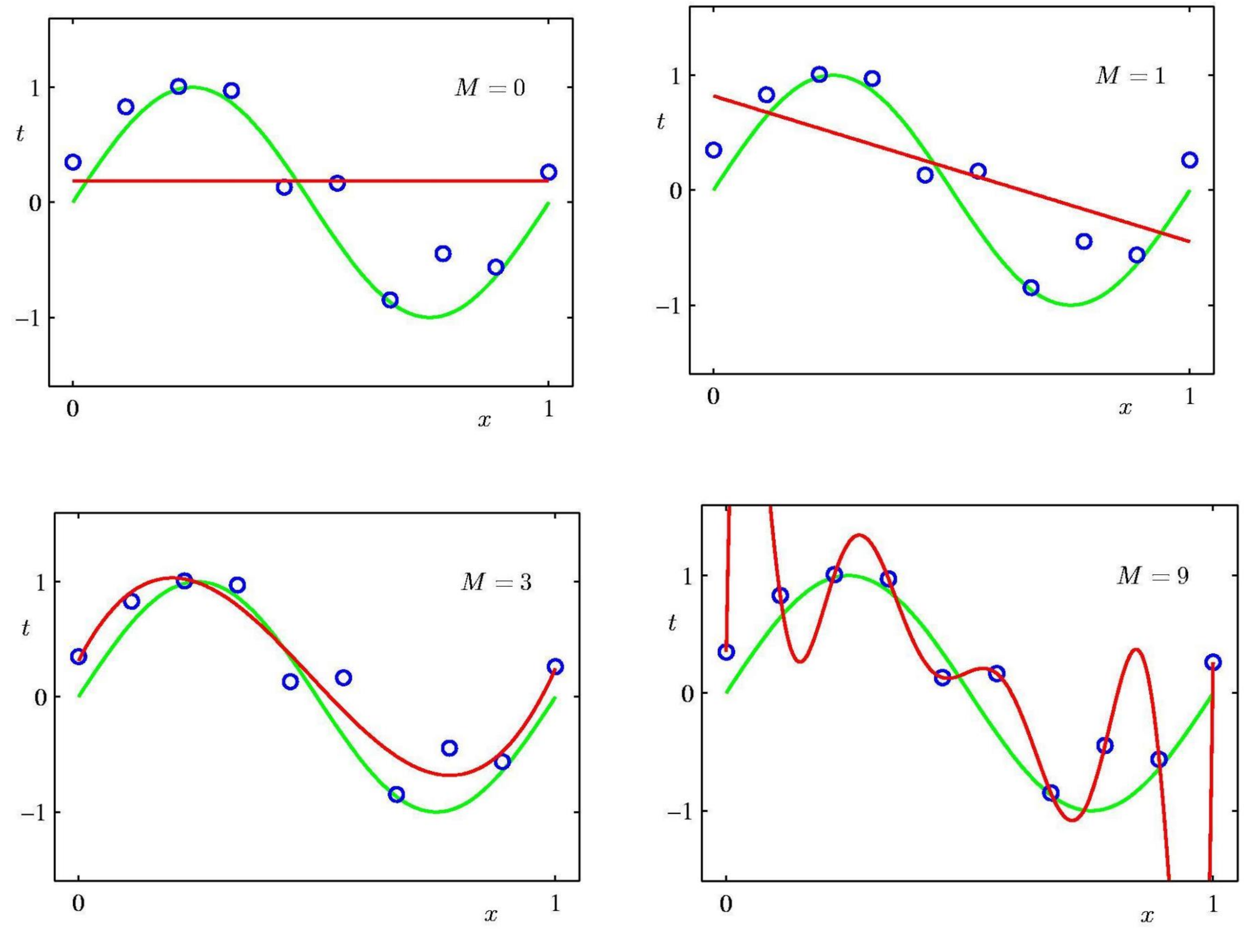
$$y(x, \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x$$



	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

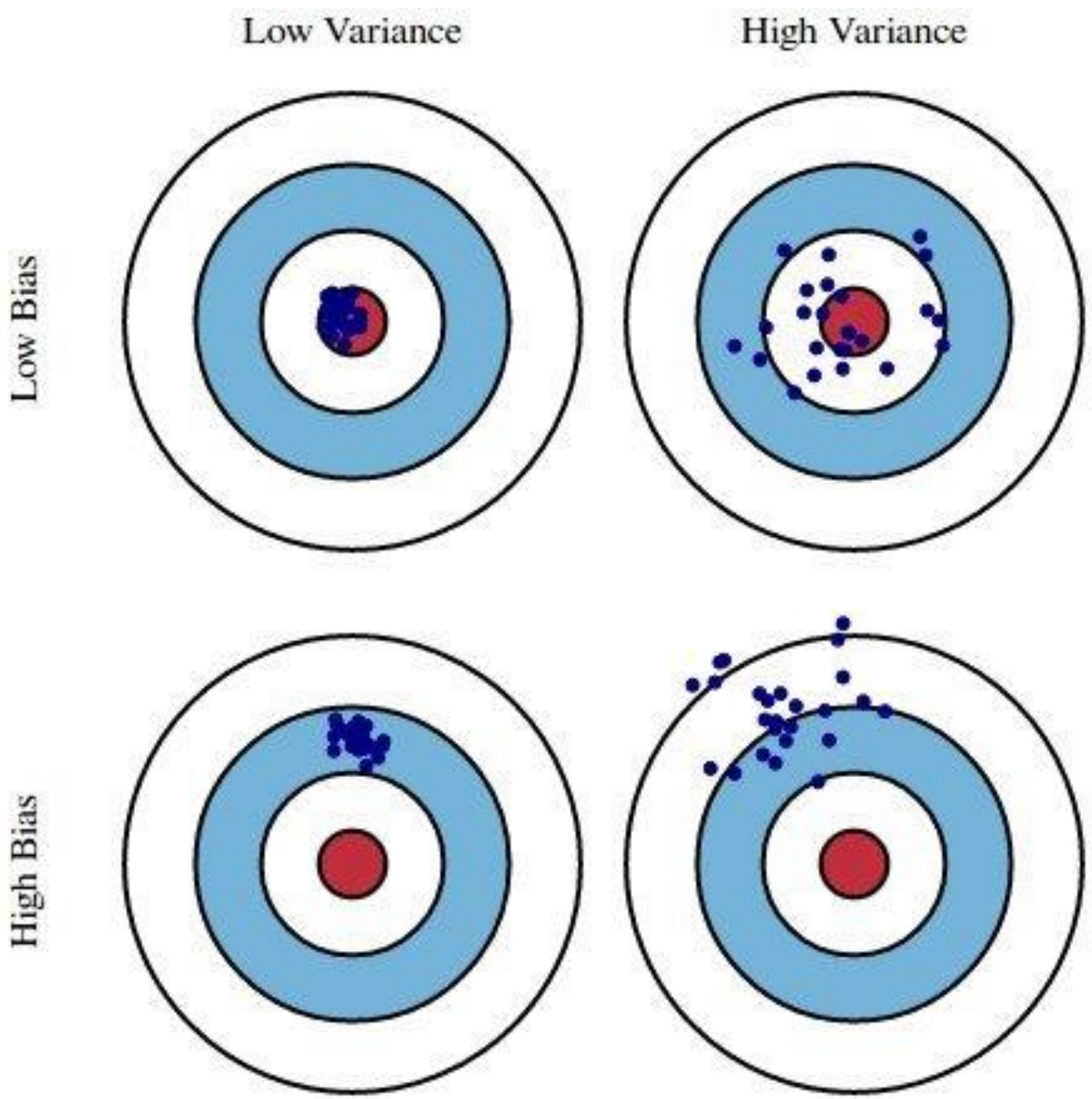
Overfitting

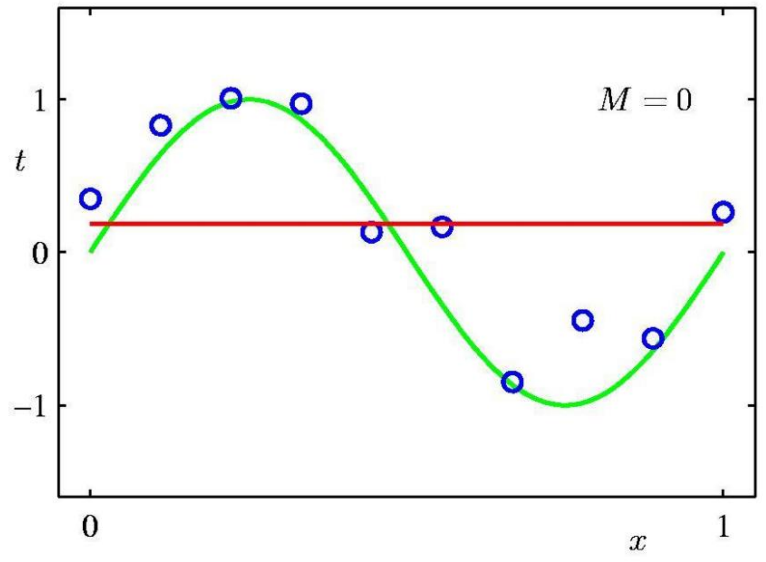


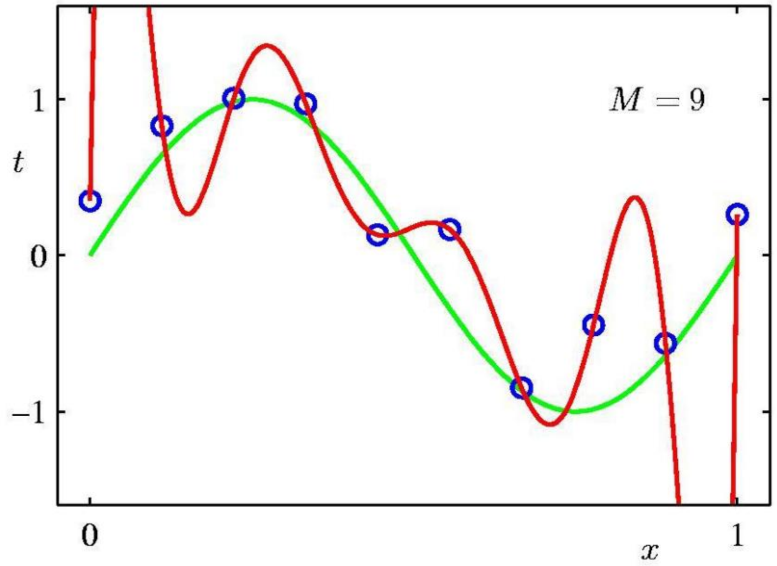


Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^{\star})/N}$

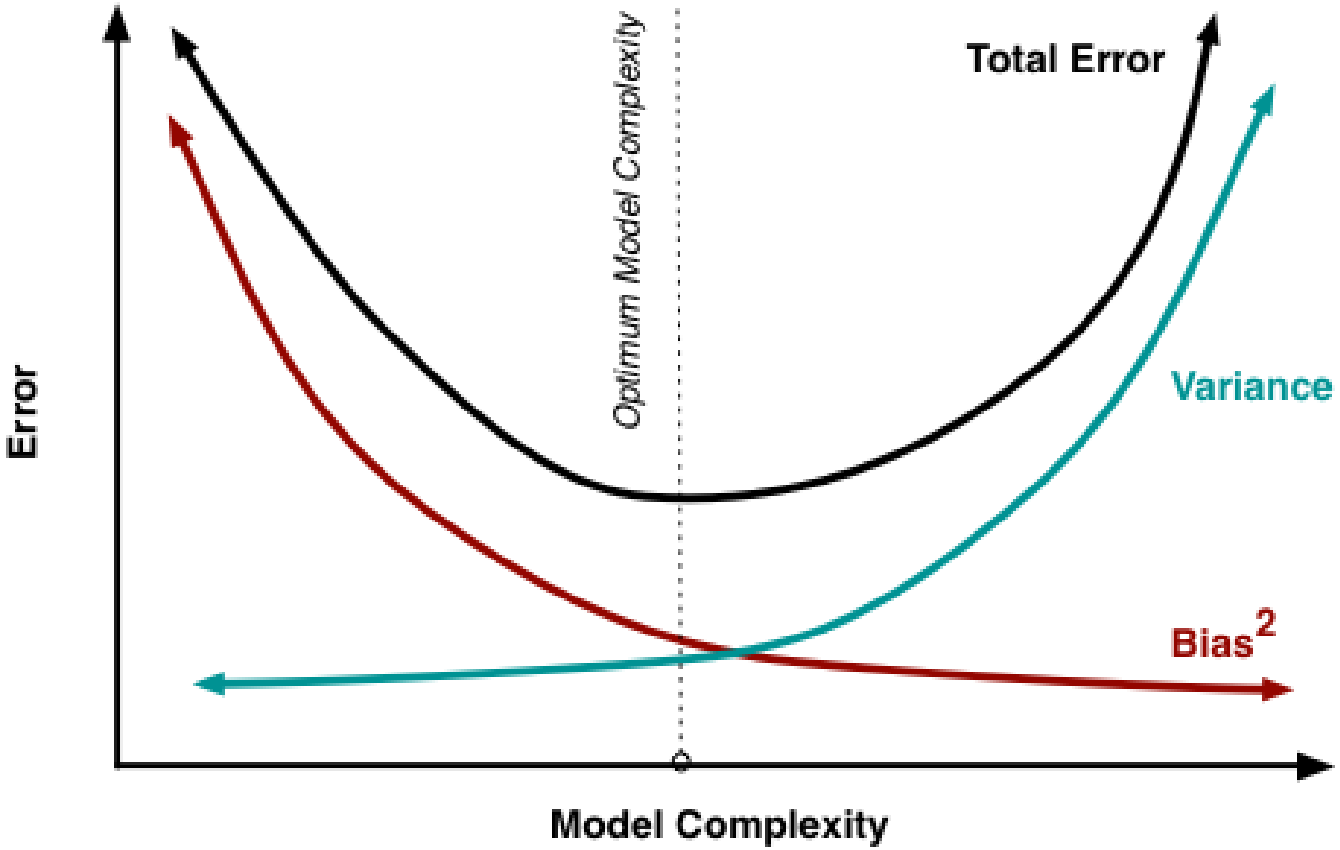
Bias and Variance using Bulls-eye Diagram







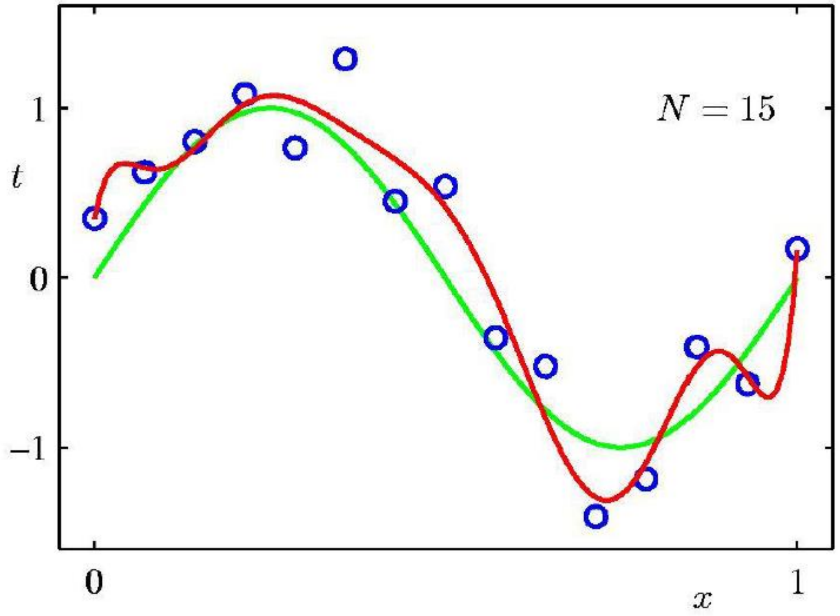
Trading off Goodness of Fit against Model Complexity (Bias-Variance Tradeoff)



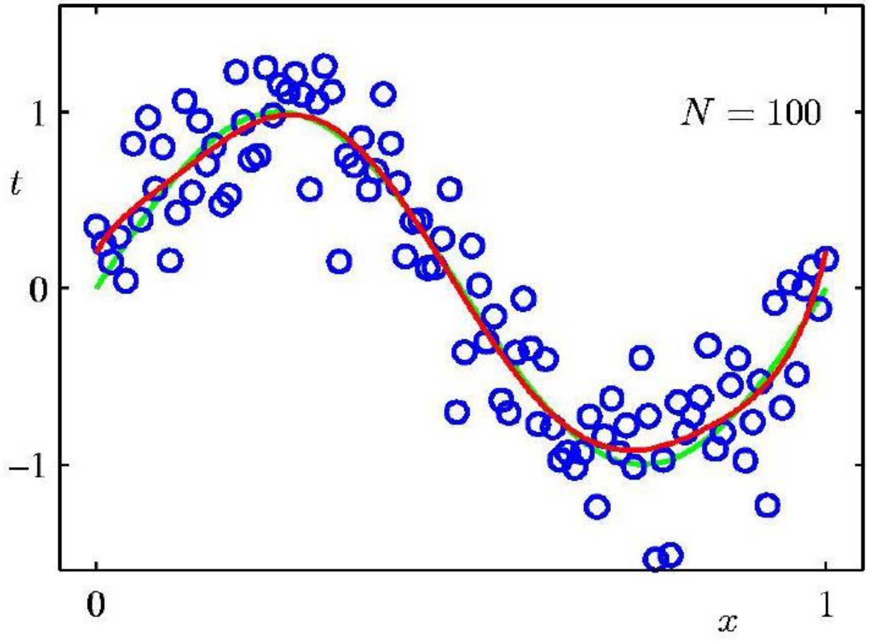
- If the model has many degrees of freedom as the data, it can fit the training data perfectly
- But, the objective in ML is generalization
- Can expect a model to generalize well if it explains the training well given the complexity of the model

How to Avoid Overfitting

9th Order Polynomial



9th Order Polynomial



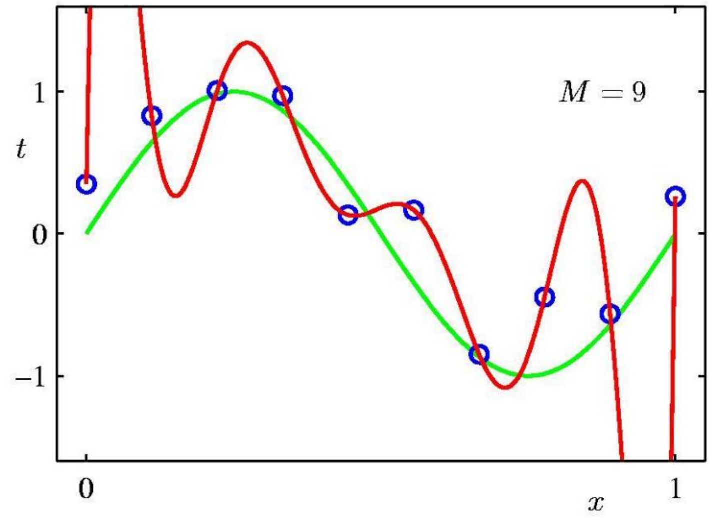
Regularization

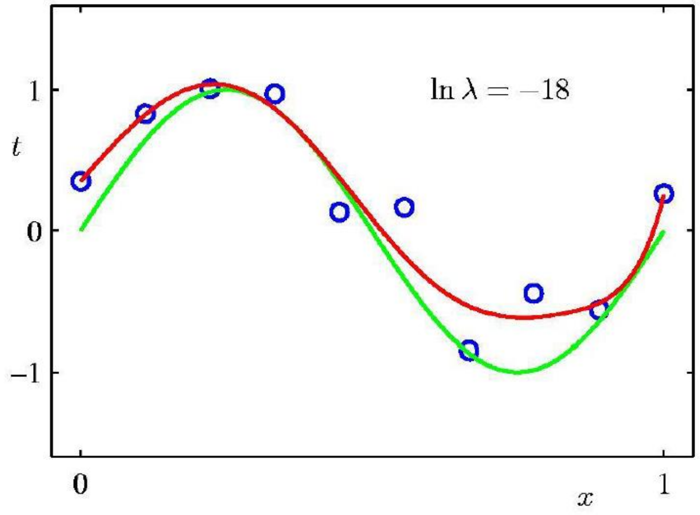
Penalize large coefficient values

$$E = J(\boldsymbol{w}) = \frac{1}{2}\sum_{i=0}^{N}\{y(x_i, \boldsymbol{w}) - t_i\}^2 + \boxed{\frac{\lambda}{2}\|\boldsymbol{w}\|^2}$$

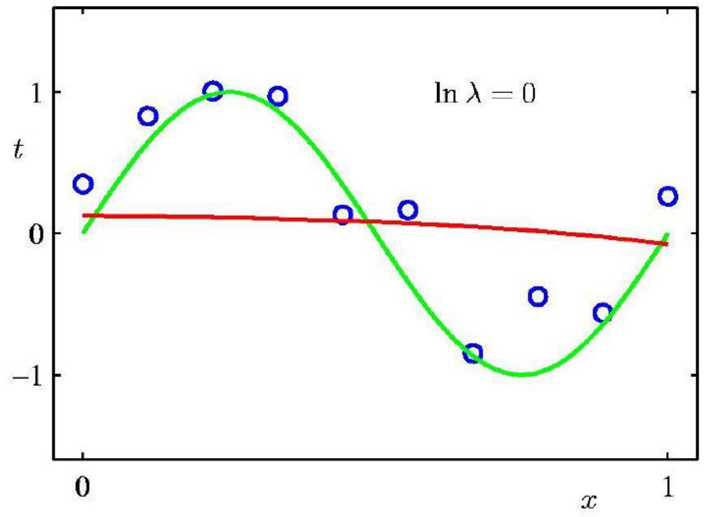
	$M = 0$	$M = 1$	$M = 3$	$M = 9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Regularization with Different λ



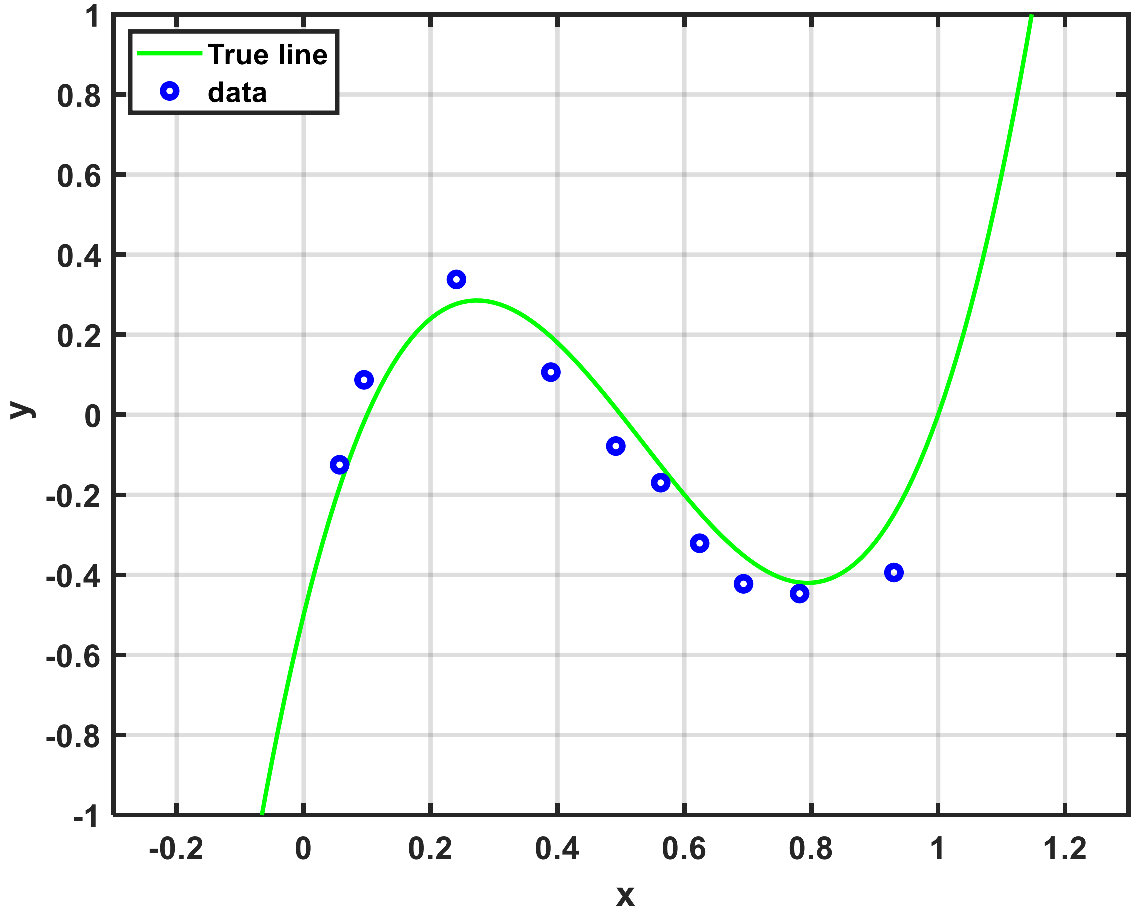


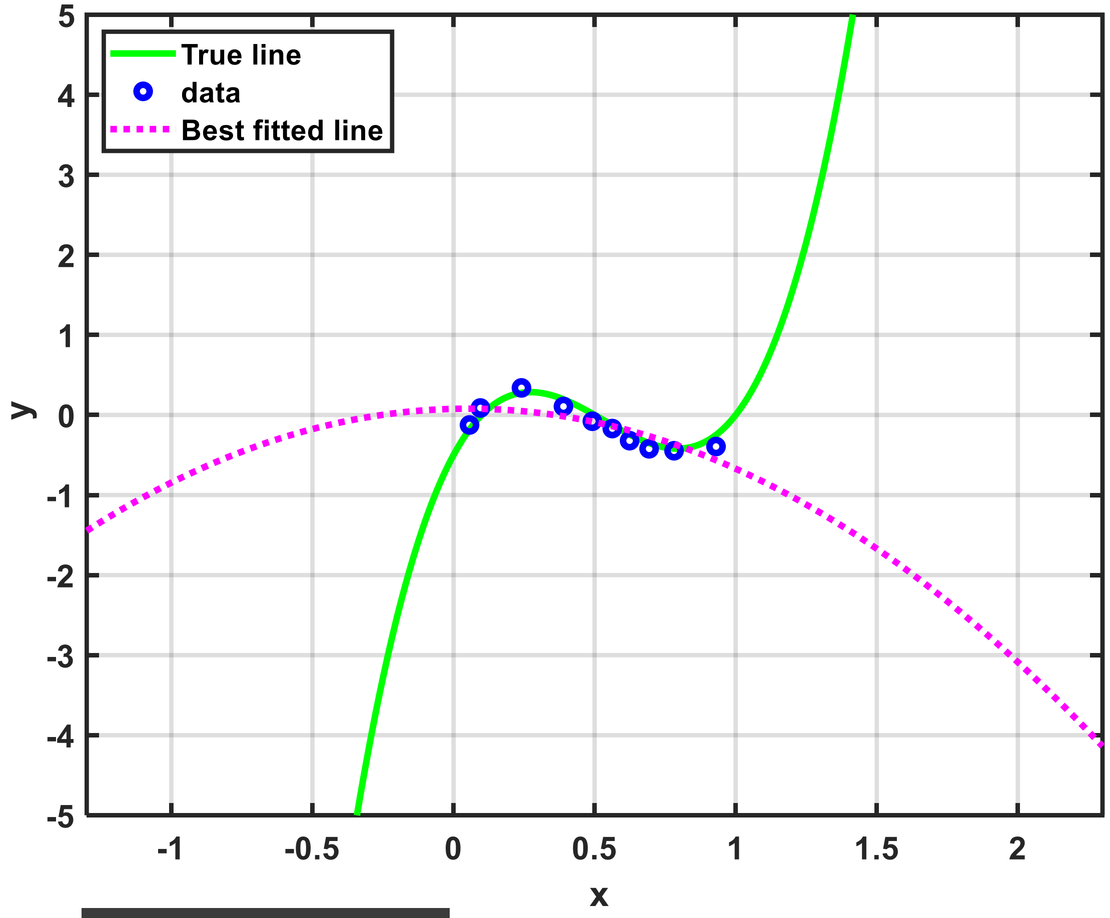
	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01



$$E = J(\boldsymbol{w}) = \frac{1}{2}\sum_{i=0}^{N}\{y(x_i, \boldsymbol{w}) - t_i\}^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$$

Example: Training a Linear Model





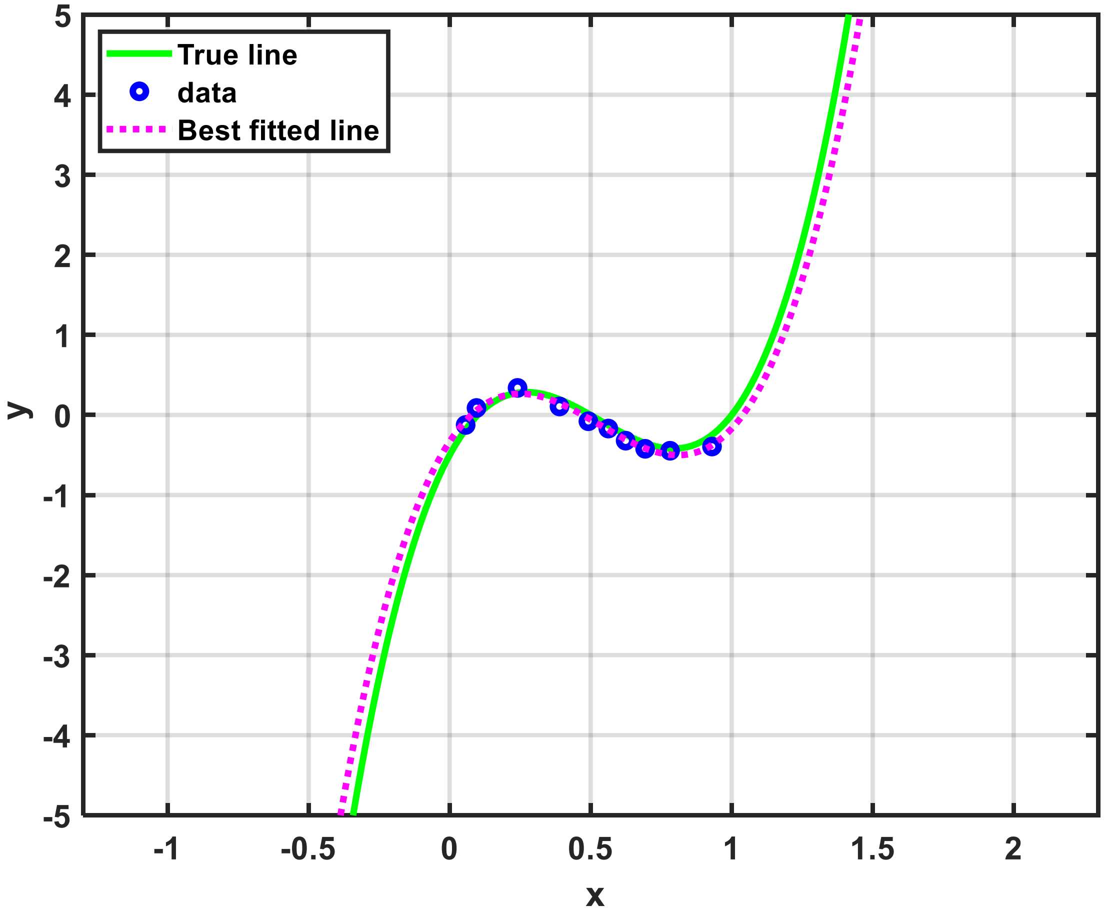
Data point: $N = 10$

True line: $y = 10 * (x - 1) * (x - 0.5) * (x - 1)$

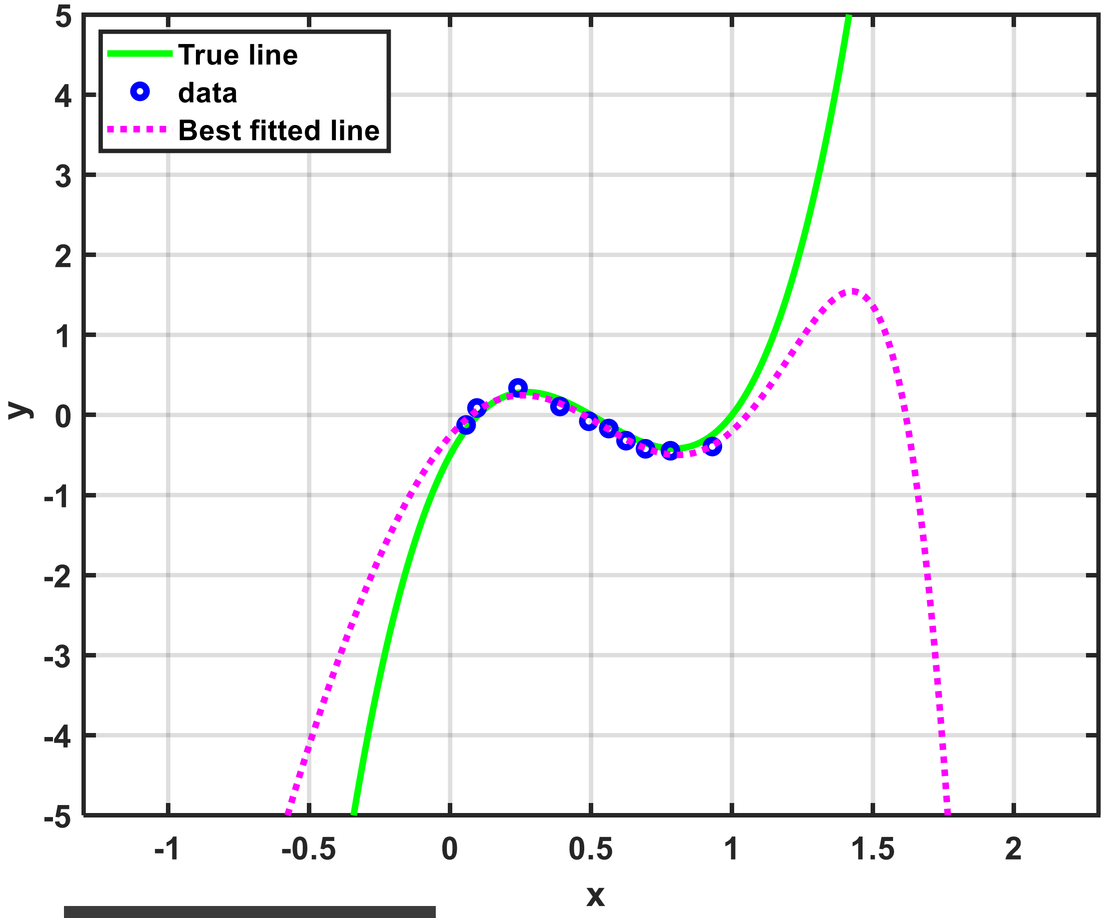
```
n_iter = 50000;
l_r = 0.05;
lambda = 0;
order = 2;
```

Tutorial

Example: Training a Linear Model (Continue)



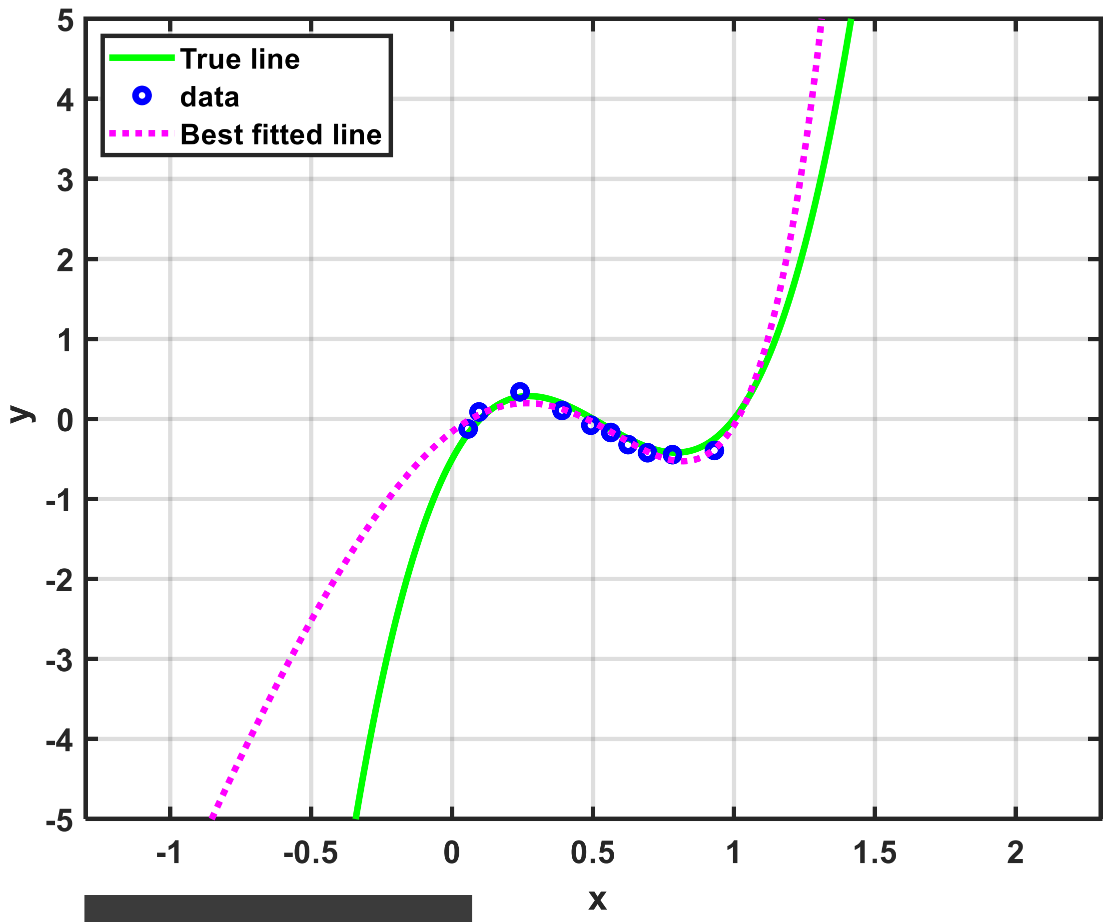

```
n_iter = 50000;
l_r = 0.05;
lambda = 0;
order = 3;
```




```
n_iter = 50000;
l_r = 0.05;
lambda = 0;
order = 6;
```

Tutorial

Example: Training a Linear Model (Continue)




```
n_iter = 50000;
l_r = 0.05;
lambda = 0.001;
order = 6;
```

Tutorial