INTRODUCTION TO MACHINE LEARNING COMPSCI 4ML3

Lecture 4

Hassan Ashtiani

MATRIX FORM OLS

- $\Delta = \begin{pmatrix} \Delta_1 \\ \dots \\ \Delta_n \end{pmatrix} = \begin{pmatrix} x_1^1 & \cdots & x_d^1 \\ \vdots & \ddots & \vdots \\ x_1^n & \cdots & x_d^n \end{pmatrix} \begin{pmatrix} w_1 \\ \dots \\ w_d \end{pmatrix} - \begin{pmatrix} y^1 \\ \dots \\ y^n \end{pmatrix}$

$$\underset{W \in \mathbb{R}^{d\times 1}}{\mathrm{MIN}} \sum_{i=1}^{n} (\Delta_i)^2 = \underset{W \in \mathbb{R}^{d\times 1}}{\mathrm{MIN}} \|\Delta\|_2^2 =$$

$$\min_{W \in \mathbb{R}^{d\times 1}} \|XW - Y\|_2^2$$

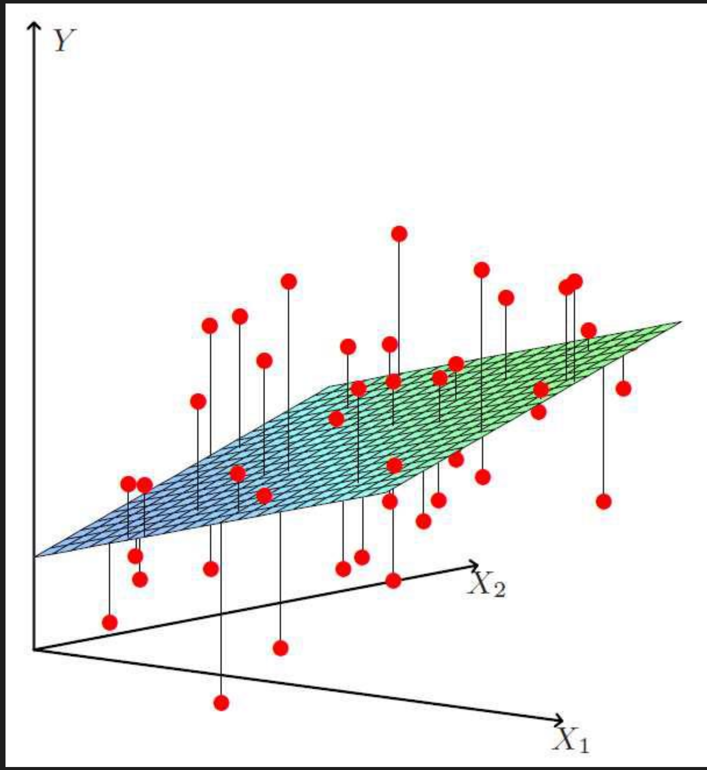
$$W^{LS} = (X^T X)^{-1} X^T Y$$

BIAS/INTERCEPT TERM

WE ARE MISSING THE BIAS TERM (w_0)

$$\min_{w_0, w_1, \ldots, w_d \in \mathbb{R}} \sum_{i=1}^{n} (w_1 x_1^i + \cdots + w_d x_d^i + w_0 - y^i)^2$$

$$\min_{w_0 \in \mathbb{R}, W \in \mathbb{R}^{d\times 1}} \left\| XW + \begin{pmatrix} w_0 \\ w_0 \\ \cdots \\ w_0 \end{pmatrix} - Y \right\|_2^2$$

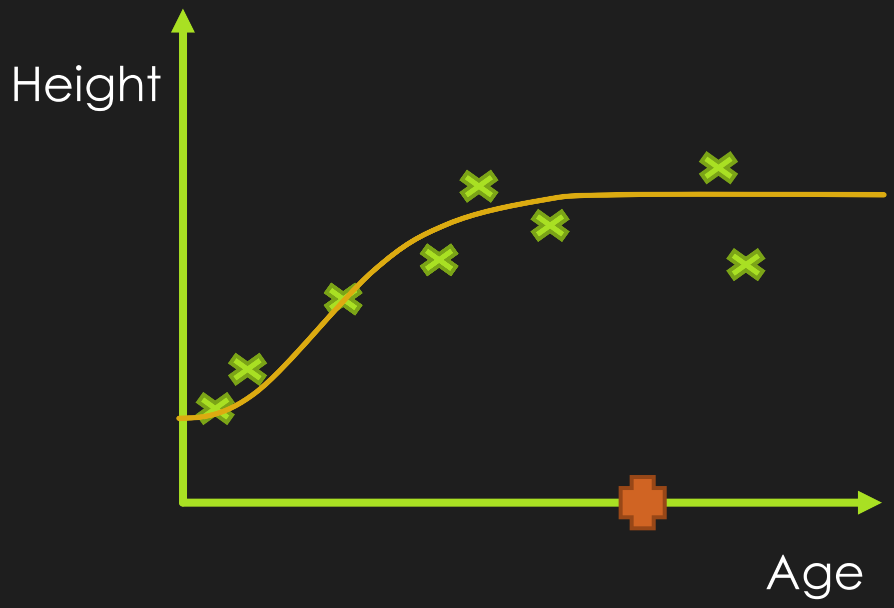


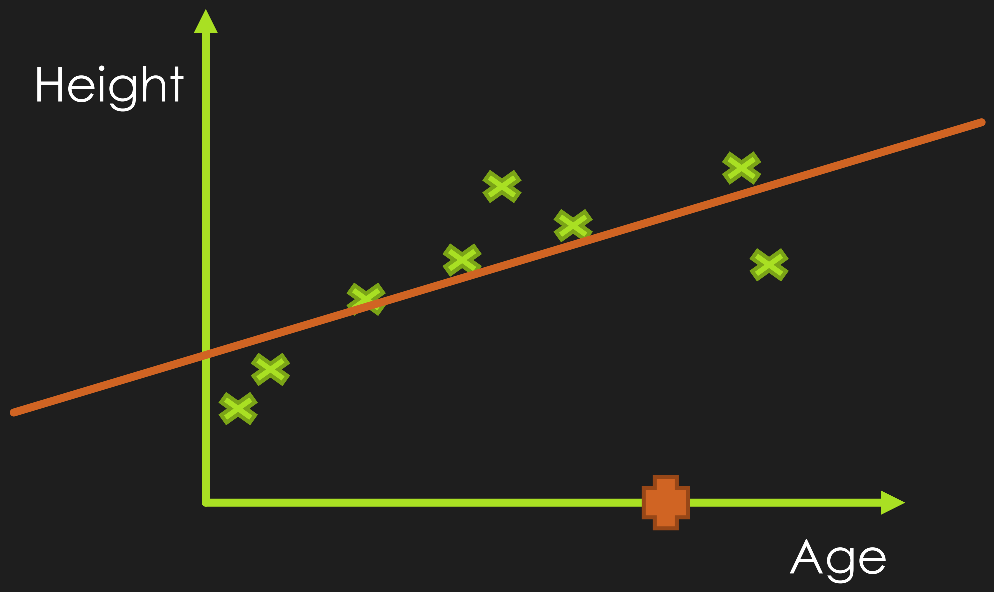
BIAS/INTERCEPT TERM

- ADD A NEW AUXILIARY DIMENSION TO THE DATA
- $X_{n\times(d+1)} = \begin{pmatrix} x_1^1 & \cdots & x_d^1 & 1 \\ \vdots & \ddots & \vdots & 1 \\ x_1^n & \cdots & x_d^n & 1 \end{pmatrix}$, $W_{(d+1)\times 1} = \begin{pmatrix} w_1 \\ \dots \\ w_d \\ w_0 \end{pmatrix}$
- SOLVE OLS: $\min_{W\in\mathbb{R}^{(d+1)\times 1}} \|XW - Y\|_2^2$
- w_0 WILL BE THE BIAS TERM!

"NON-LINEAR" DATA?

- For example, what is the best degree 2 polynomial?





- How can we reuse the "least-squares machinery"?

IDEA: DATA TRANSFORMATION

- We increased the flexibility of our predictor by a form of data transformation/augmentation
- $X'_{n\times(d+1)} = \begin{pmatrix} x_1^1 & \cdots & x_d^1 & 1 \\ \vdots & \ddots & \vdots & 1 \\ x_1^n & \cdots & x_d^n & 1 \end{pmatrix}$
- Can we use the same idea to make our predictor even more flexible (non-linear)?

EXAMPLE

LEAST-SQUARES FOR POLYNOMIALS

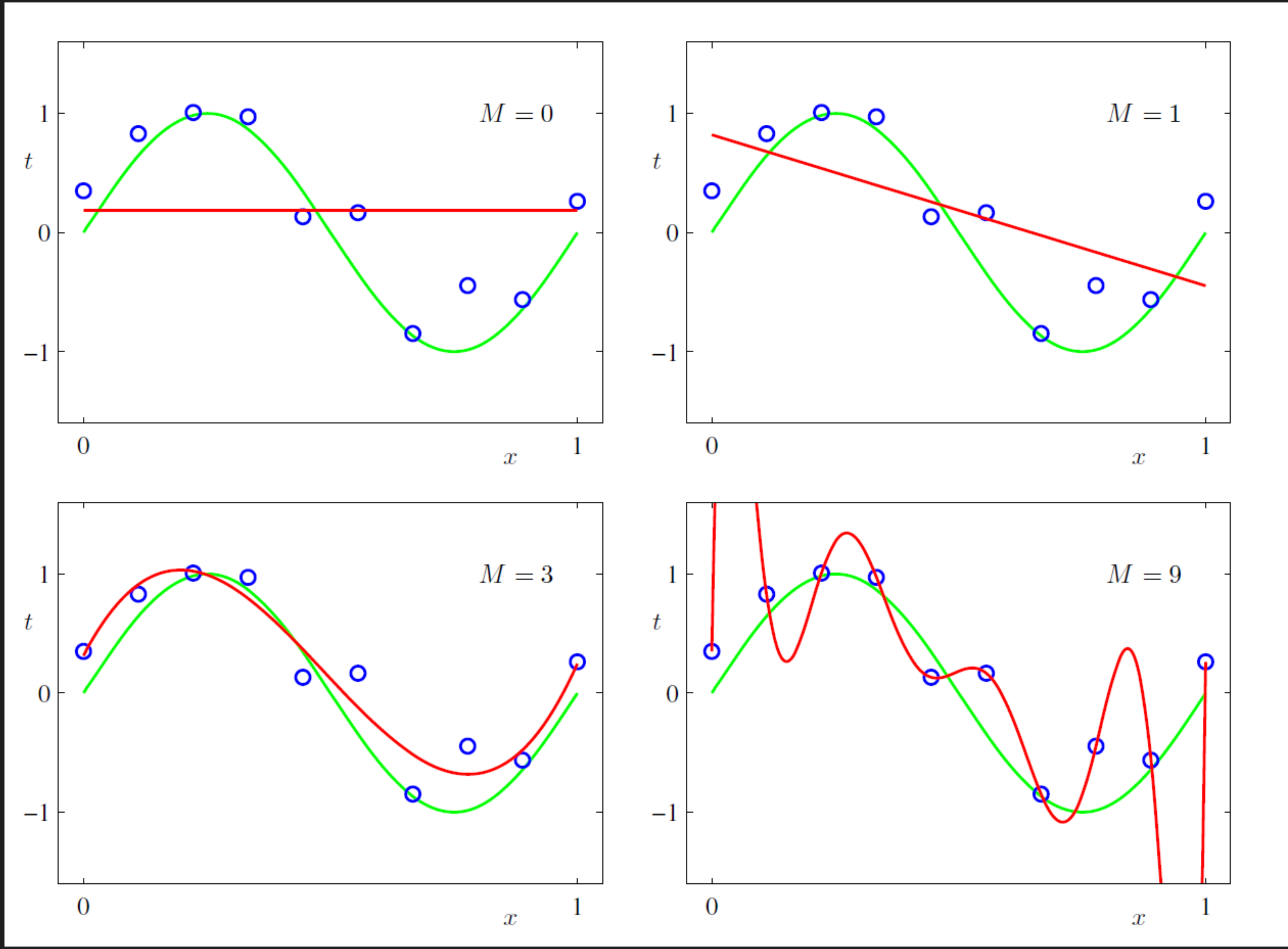
- IDEA: $ax^2 + bx + c$ IS STILL LINEAR WITH RESPECT TO THE PARAMETERS! (W.R.T. a, b AND c)
- INSTEAD OF $X_{n\times 1} = \begin{pmatrix} x^1 \\ \dots \\ x^n \end{pmatrix}$ USE $X'_{n\times 3} = \begin{pmatrix} x^1 & (x^1)^2 & 1 \\ \dots & \dots & \dots \\ x^n & (x^n)^2 & 1 \end{pmatrix}$
- TREAT $X_{n\times 3}$ AS IF IT WAS YOUR ORIGINAL INPUT DATA
- WE CAN EXTEND THIS TO HIGHER DEGREE POLYNOMIALS SIMILARLY, E.G., $ax^3 + bx^2 + cx + d$
- NOTEBOOK EXAMPLE

MULTIVARIATE POLYNOMIALS

- HOW ABOUT WHEN x IS MULTIVARIATE ITSELF?
  - $w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 (x_1)^2 + w_5 (x_2)^2 + w_6$
  - INSTEAD OF (x_1, x_2) USE $(x_1 \quad x_2 \quad x_1 x_2 \quad (x_1)^2 \quad (x_2)^2 \quad 1)$
- TREAT THE NEW X AS (A HIGHER-DIMENSIONAL) INPUT

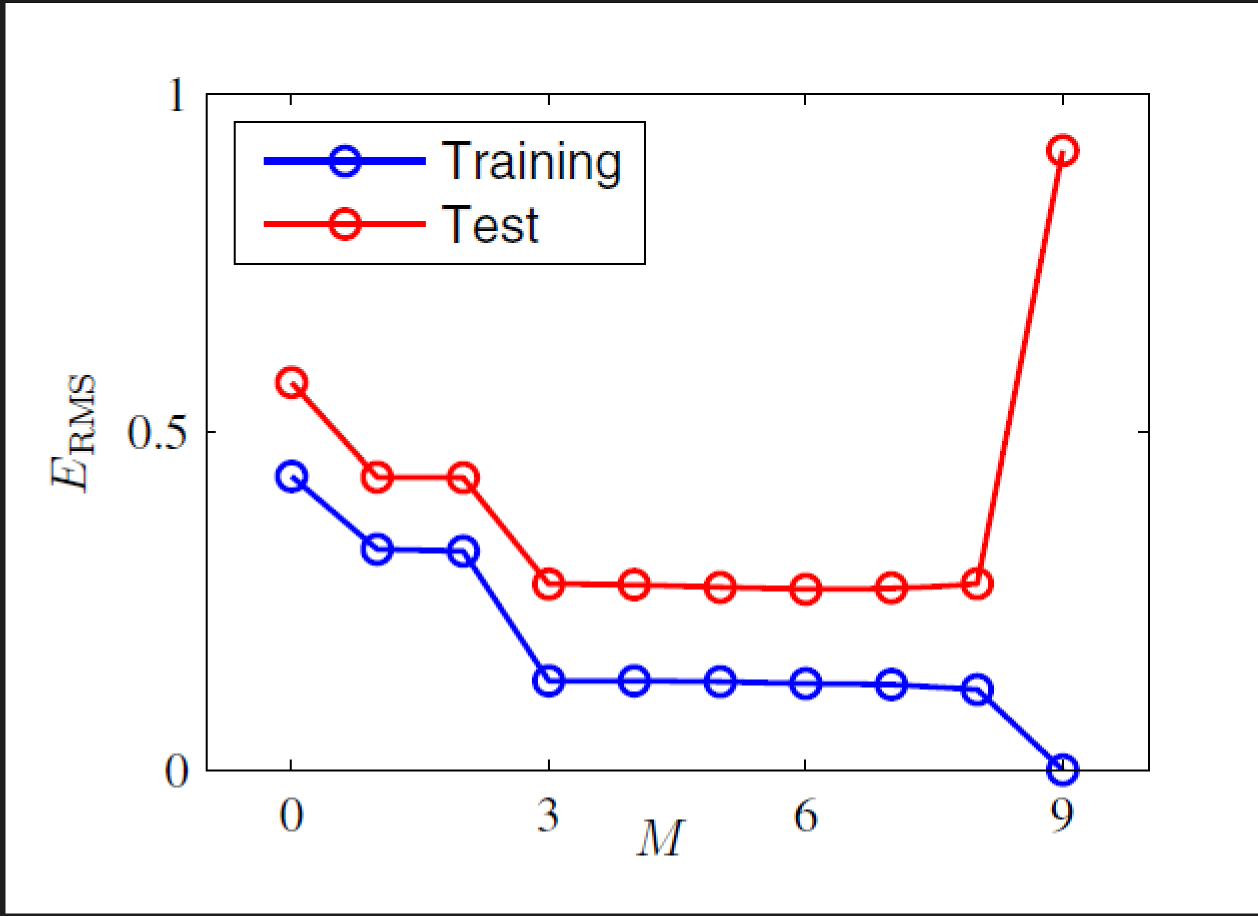
- INPUT DIMENSION: d
- DEGREE OF POLYNOMIAL: M
- NUMBER OF TERMS (MONOMIALS) OF DEGREE AT MOST M ≈ $\binom{M+d}{d} = \binom{M+d}{M}$

OVERFITTING



OVERFITTING

- DIVIDE THE DATA RANDOMLY TO "TRAIN" AND "TEST" SETS
- ROOT-MEAN-SQUARE ERROR FOR EACH SET:
- $\sqrt{\frac{\|\hat{Y}-Y\|_2^2}{n}} = \sqrt{\frac{\sum_{i=1}^{n}(y_i-\widehat{y_i})^2}{n}}$



MORE DATA, LESS OVER-FITTING

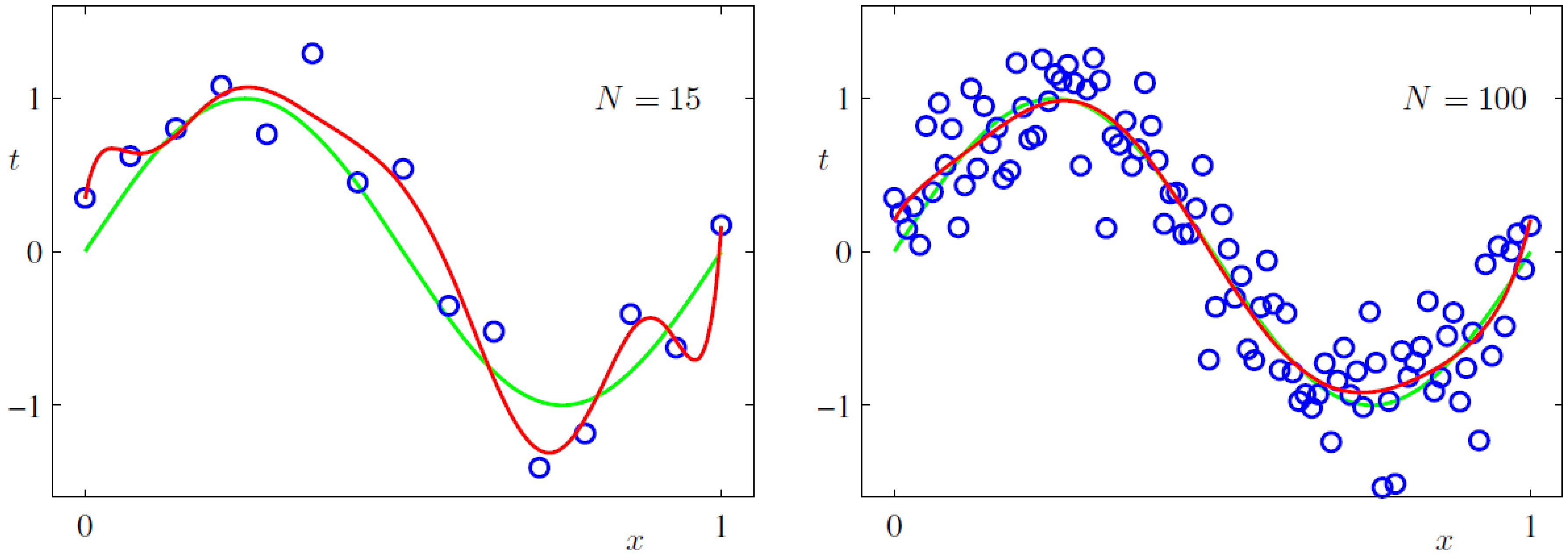


Figure 1.6 Plots of the solutions obtained by minimizing the sum-of-squares error function using the $M = 9$ polynomial for $N = 15$ data points (left plot) and $N = 100$ data points (right plot). We see that increasing the size of the data set reduces the over-fitting problem.

THE TRADE-OFF

- A powerful/flexible curve-fitting method
  - Small training error
  - Requires more training data to generalize
  - Otherwise large test error
- A less flexible curve-fitting Method
  - Larger training error
  - Requires less training data
  - Smaller difference between training and test error
- The so-called "bias-variance" trade-off

THE CASE OF MULTIVARIATE POLYNOMIALS

- Assume $M \gg d$
- Number of terms (monomials): $\approx (\frac{M}{d})^d$
- #training samples $\approx$ #parameters $\approx (\frac{M}{d})^d$
  - #training samples should increase exponentially with d
  - Susceptible to over-fitting…
  - An example of **curse of dimensionality**!
- We can say **sample complexity** of learning multivariate polynomials is exponential in d
  - Orthogonal to computational complexity