So Far…

- Our goal (supervised learning):
  - To learn a set of discriminant functions
- Bayesian framework
  - We could design an optimal classifier if we knew:
    - $P(\omega_i)$: priors and $P(x \mid \omega_i)$: class-conditional densities
    - Using training data to estimate $P(\omega_i)$ and $P(x \mid \omega_i)$
    - $P(\omega_i \mid x)$ is computed and be used as the discriminant functions
- Other possible ways?
  - Directly learning discriminant functions from the training data

Linear Methods for Regression

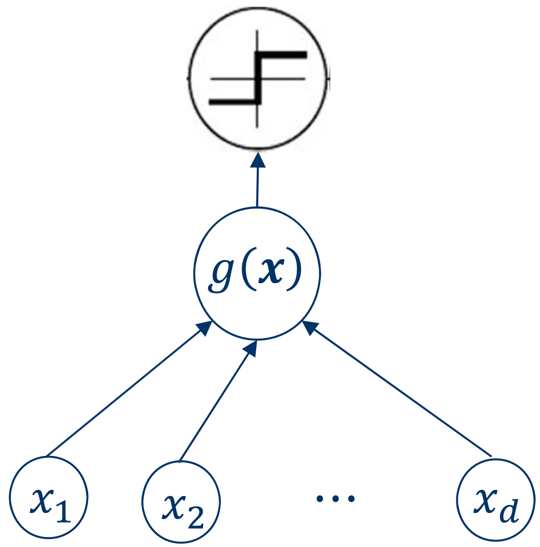
Deng Cai (蔡登)

College of Computer Science
Zhejiang University

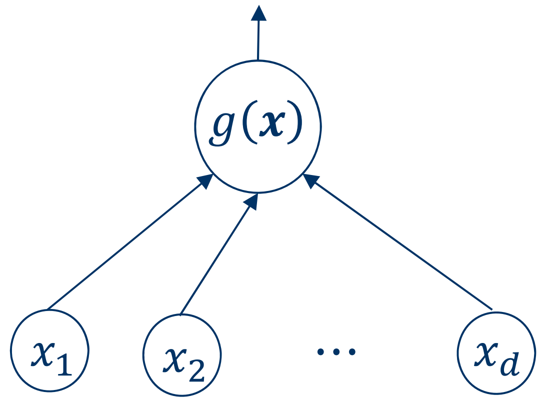
dengcai@gmail.com

Classification VS Regression

- Both are supervised learning methods
  - Goal: learn a mapping from inputs $\boldsymbol{x}$ to outputs y
- Classification (Categorization, Decision making…)
  - y is a categorical variable



- Regression
  - y is real-valued



Linear model

- Sample: $\boldsymbol{x} \in R^d, \boldsymbol{x} = [x_1, x_2, \cdots x_d]^T$
- Finds a linear function $\boldsymbol{w} = [w_1, w_2, \cdots, w_d]^T \in R^d, b$

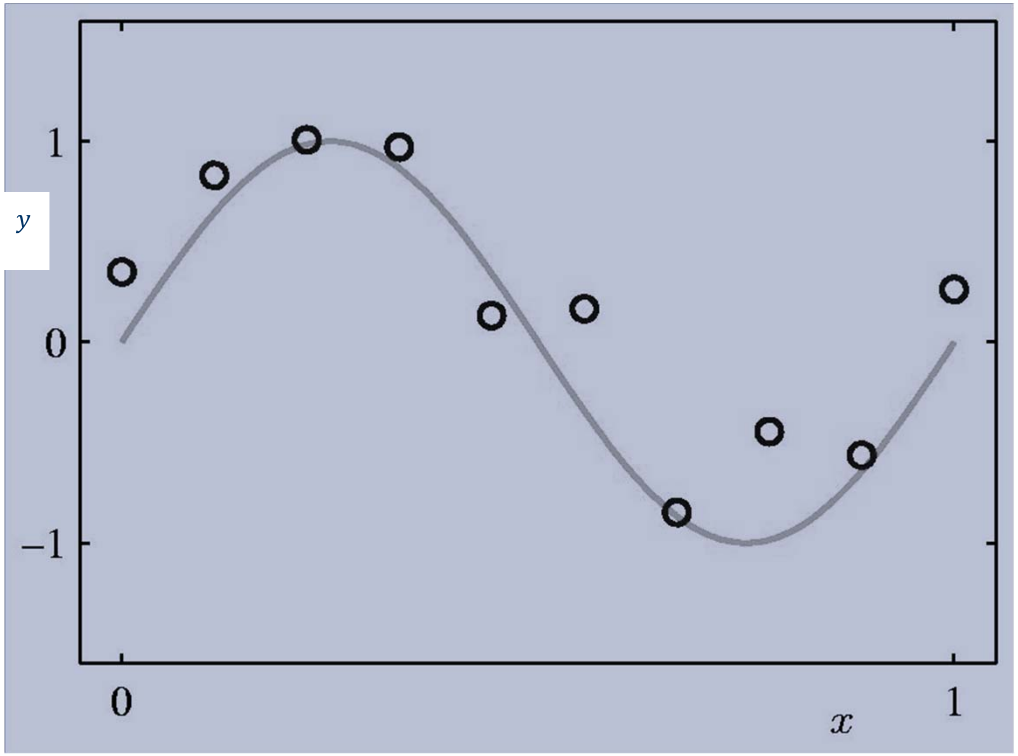
$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b$$

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$$

$$\boldsymbol{x} = [x_1, x_2, \cdots x_d, 1]^T \in R^{d+1}$$

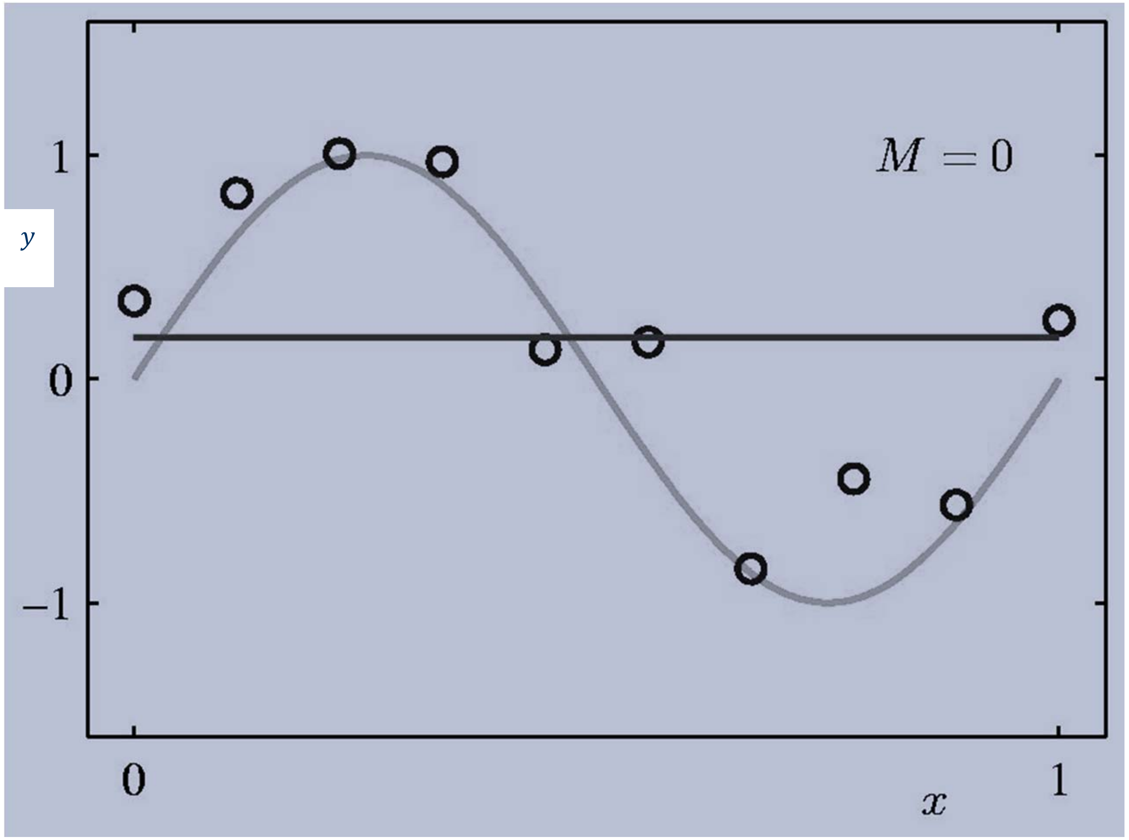
$$\boldsymbol{w} = [w_1, w_2, \cdots w_d, b]^T \in R^{d+1}$$

Polynomial Curve Fitting

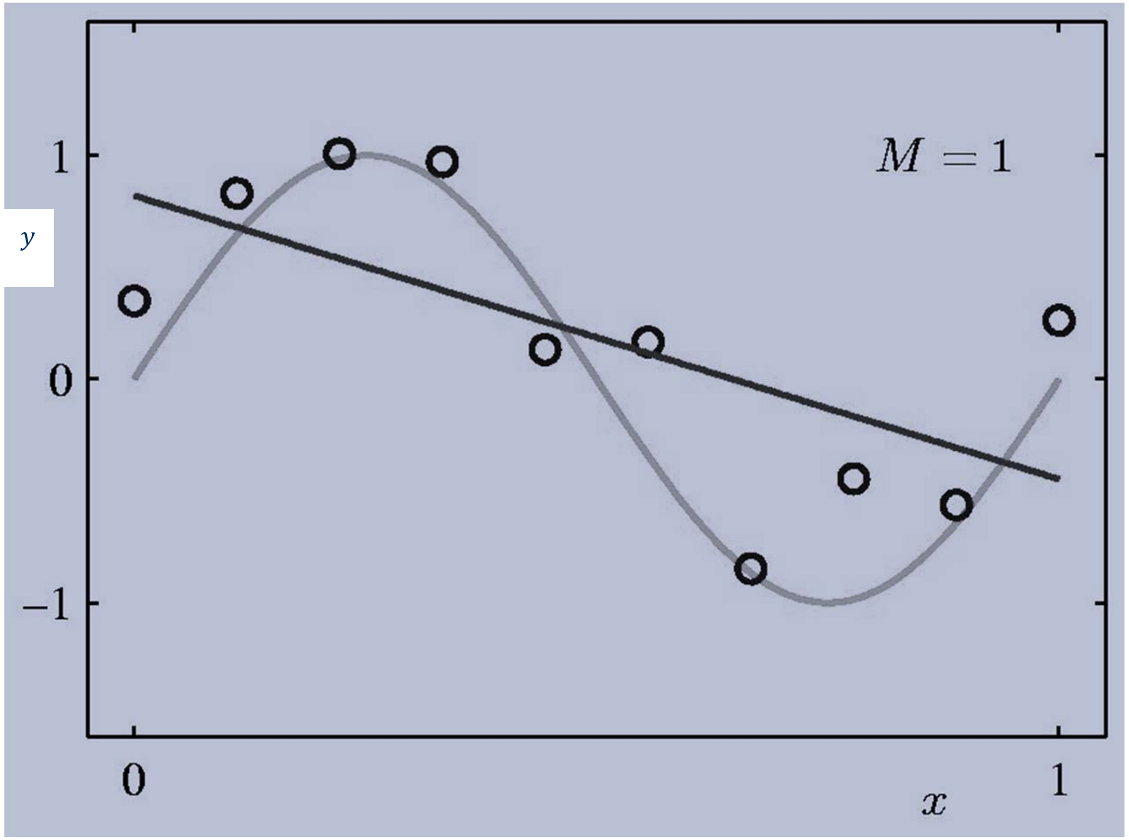


$$f(x, \boldsymbol{a}) = a_0 + a_1 x + a_2 x^2 + \cdots + a_M x^M = \sum_{j=0}^{M} a_j x^j$$

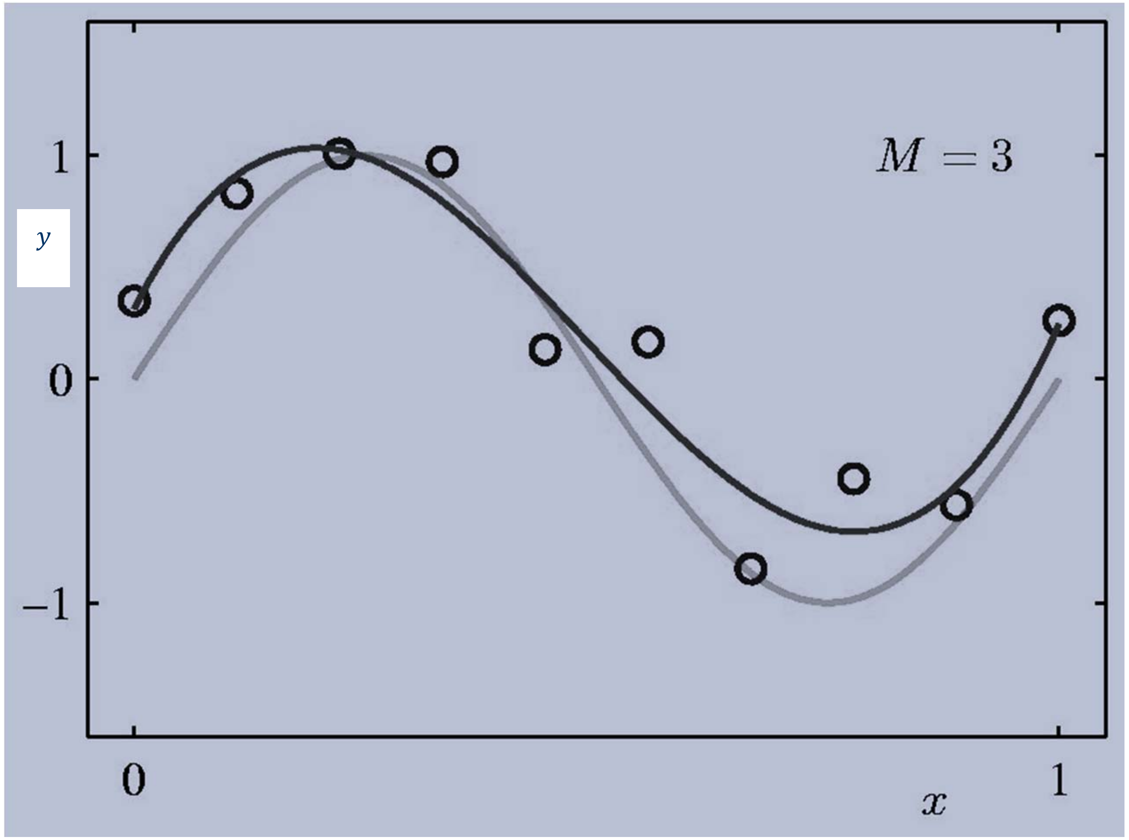
0^{th} Order Polynomial



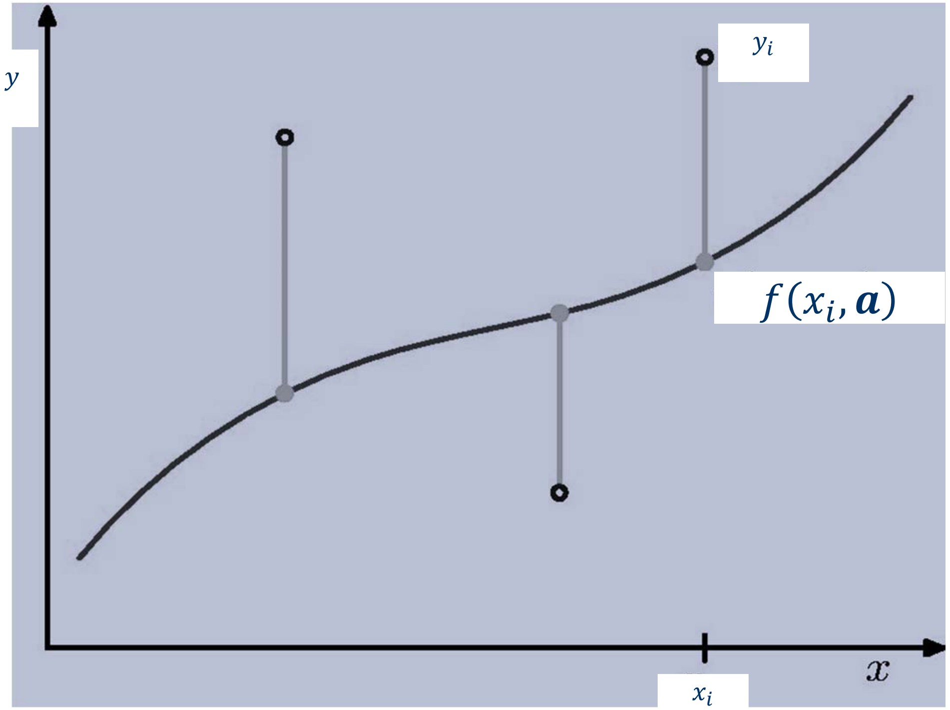
1^{st} Order Polynomial



3rd Order Polynomial



Sum-of-Squares Error Function



- Training data:
  - $(x_1, y_1), (x_2, y_2), \cdots (x_n, y_n)$
- To learn f which $f(x) = y$
- Criterion function:
  - $\text{MSE}(\boldsymbol{a}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i, \boldsymbol{a}))^2$

Polynomial Curve Fitting → Linear Regression

$$f(x, \boldsymbol{a}) = a_0 + a_1 x + a_2 x^2 + \cdots + a_M x^M = \sum_{j=1}^{M} a_M x^M$$

- $\boldsymbol{x} = [1, x, x^2, \cdots, x^M]^T$
- $\boldsymbol{a} = [a_0, a_1, a_2, \cdots, a_M]^T$
- $f(x, \boldsymbol{a}) = \boldsymbol{a}^T \boldsymbol{x}$

Linear Regression Model

- Training data: $(\boldsymbol{x}_i, y_i)$
- $f(\boldsymbol{x}) = a_0 + \sum_{i=1}^{p} a_i x_i = a_0 + \boldsymbol{a}^T \boldsymbol{x}$
  - $\boldsymbol{a} = [a_1, \cdots, a_p]^T$ and a_0 : unknown parameters or coefficients
  - $\boldsymbol{x}$: Feature vector, the outcome of feature extraction.
- Minimize the **mean-squared error** :

$$J_n = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(\boldsymbol{x}_i))^2$$

- Minimize the **residual sum of squares**

$$RSS(f) = \sum_{i=1}^{n} \left(y_i - f(\boldsymbol{x}_i)\right)^2$$

The MSE Criterion

$$J_n(\boldsymbol{a}) = \sum_{i=1}^{n} \left(y_i - \boldsymbol{a}^T \boldsymbol{x}_i\right)^2$$

- MSE Criterion: Minimize the sum of squared differences between $\boldsymbol{a}^T \boldsymbol{x}_i$ and y_i
- Using matrix notation for convenience

$$X = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n], \qquad \boldsymbol{y} = [y_1, \cdots, y_n]^T$$

$$J_n(\boldsymbol{a}) = \left(\boldsymbol{y} - X^T \boldsymbol{a}\right)^T \left(\boldsymbol{y} - X^T \boldsymbol{a}\right)$$

- How to optimize it (finds the optimal solution)?

Optimizing the MSE Criterion

- Computing the gradient gives:

$$\nabla J_n = -2X\left(\boldsymbol{y} - X^T\boldsymbol{a}\right)$$

- Setting the gradient to zero,

$$XX^T\boldsymbol{a} = X\boldsymbol{y}$$

$$\boldsymbol{a} = \left(XX^T\right)^{-1}X\boldsymbol{y}$$

- Any problems?
- What is the rank of the matrix XX^T?
- The solution for **a** can be obtained uniquely if XX^T is non-singular.
- The fitted values at the training inputs are

$$\hat{\boldsymbol{y}} = X^T\boldsymbol{a} = X^T\left(XX^T\right)^{-1}X\boldsymbol{y}$$

Geometry of least-squares fitting

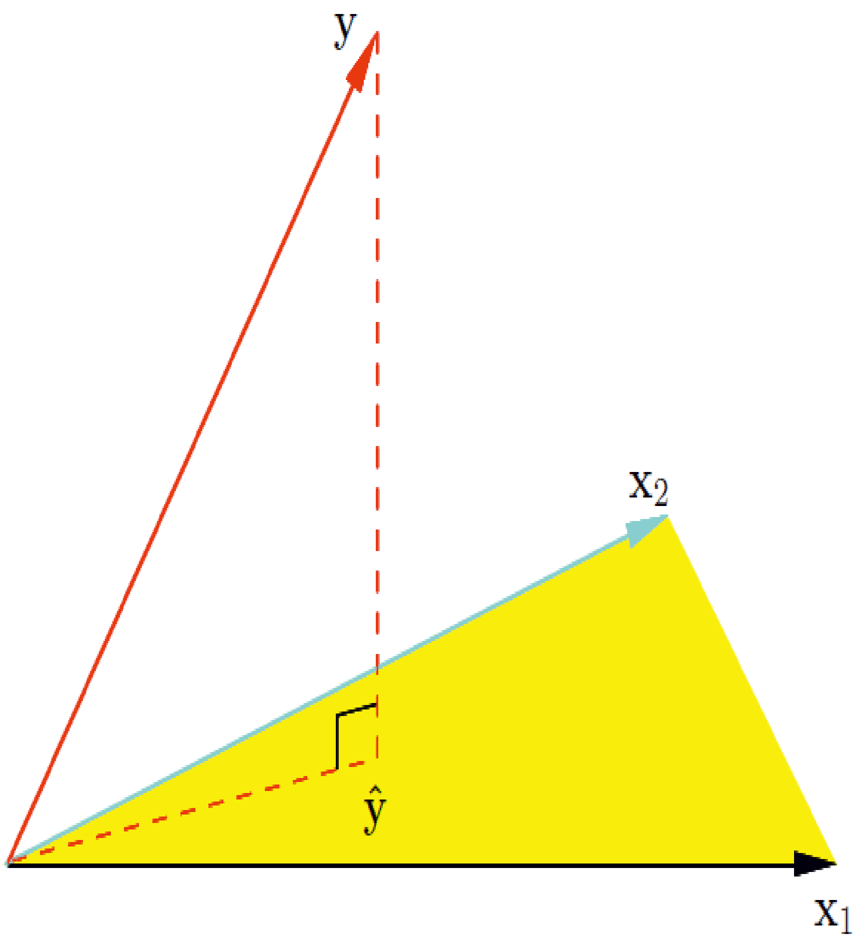
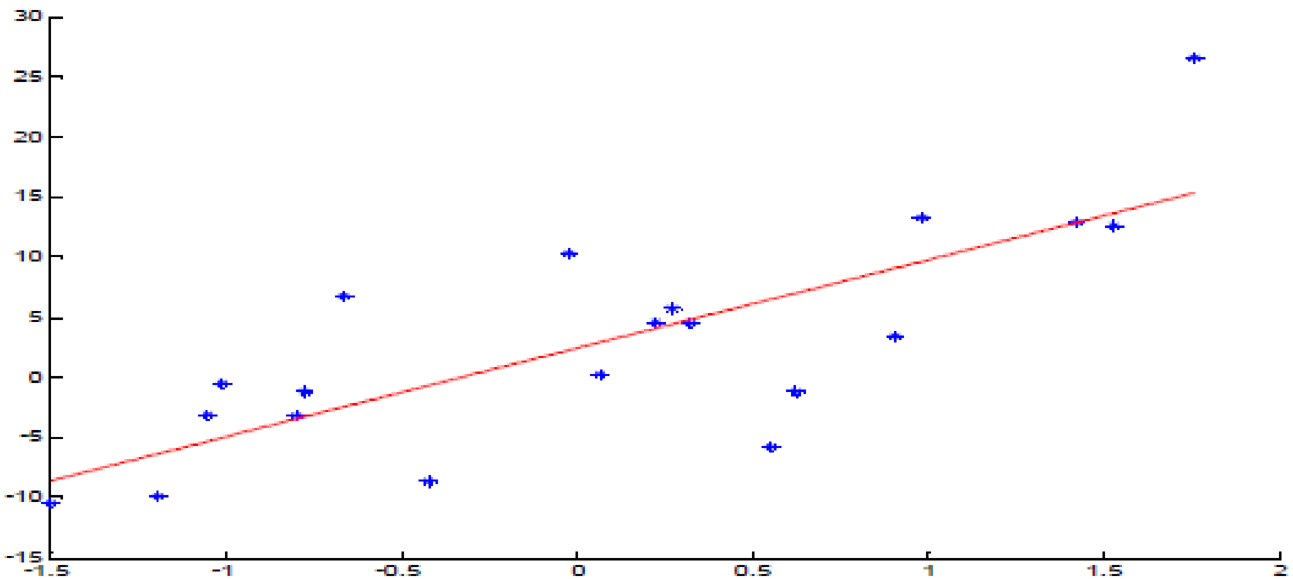


*Figure 2:The N-dimensional geometry of least squares regression with two predictors. The outcome vector **y*** is orthogonally projected onto the hyperplane *spanned by the input vectors **x1*** and ***x2*.** The projection $\hat{y}$ represents the vector *of the least squares predictions*

Statistical model of regression

- **A generative model:** $y = f(\boldsymbol{x}, \boldsymbol{a}) + \epsilon$
- $f(\boldsymbol{x}, \boldsymbol{a})$ is a deterministic function
- ϵ is a random noise , it represents things we cannot capture with ,
  e.g. $\epsilon \sim N(0, \sigma^2)$



Statistical model of regression

- A generative model: $y = f(\boldsymbol{x}, \boldsymbol{a}) + \epsilon$
- Assume: $\epsilon \sim N(0, \sigma^2)$
- $p(y|\boldsymbol{x}, \boldsymbol{a}, \sigma)$ =?

$$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(y-f(\boldsymbol{x},\boldsymbol{w}))^2}$$

- **Likelihood of predictions**
  - The probability of observing outputs $\boldsymbol{y}$ in $\boldsymbol{D}$ given $\boldsymbol{a}$, $\boldsymbol{X}$, and σ.

Maximum Likelihood Estimation

- Likelihood of predictions

$$L(\boldsymbol{D}, \boldsymbol{a}, \sigma) = \prod_{i=1}^{n} p(y_i | \boldsymbol{x}_i, \boldsymbol{a}, \sigma)$$

- Maximum likelihood estimation of parameters
  - Parameters maximizing the likelihood of predictions

$$\boldsymbol{a}^* = \operatorname{argmax} \prod_{i=1}^{n} p(y_i | \boldsymbol{x}_i, \boldsymbol{a}, \sigma)$$

- Log-likelihood

Maximum Likelihood Estimation

- Log-likelihood

$$l(\boldsymbol{D}, \boldsymbol{a}, \sigma) = \log(L(\boldsymbol{D}, \boldsymbol{a}, \sigma)) = \log \prod_{i=1}^{n} p(y_i | \boldsymbol{x}_i, \boldsymbol{a}, \sigma)$$

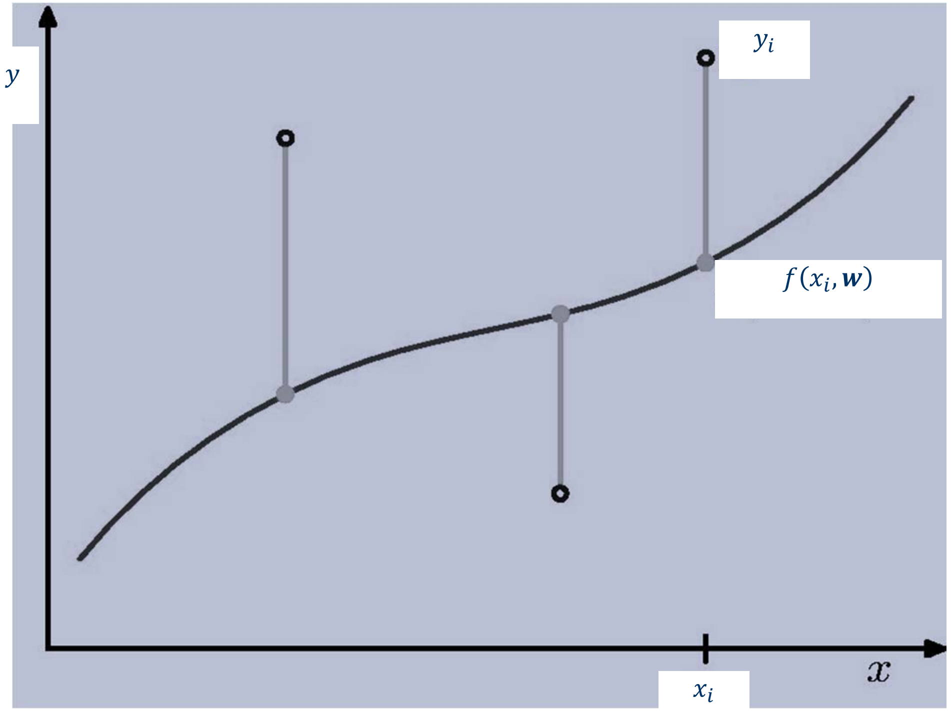
$$= \sum_{i=1}^{n} \log p(y_i | \boldsymbol{x}_i, \boldsymbol{a}, \sigma)$$

$$p(y_i | \boldsymbol{x}_i, \boldsymbol{a}, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}\left(y - f(\boldsymbol{x}, \boldsymbol{a})\right)^2}$$

$$l(\boldsymbol{D}, \boldsymbol{a}, \sigma) = -\frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y - f(\boldsymbol{x}, \boldsymbol{a})\right)^2 + c(\sigma)$$

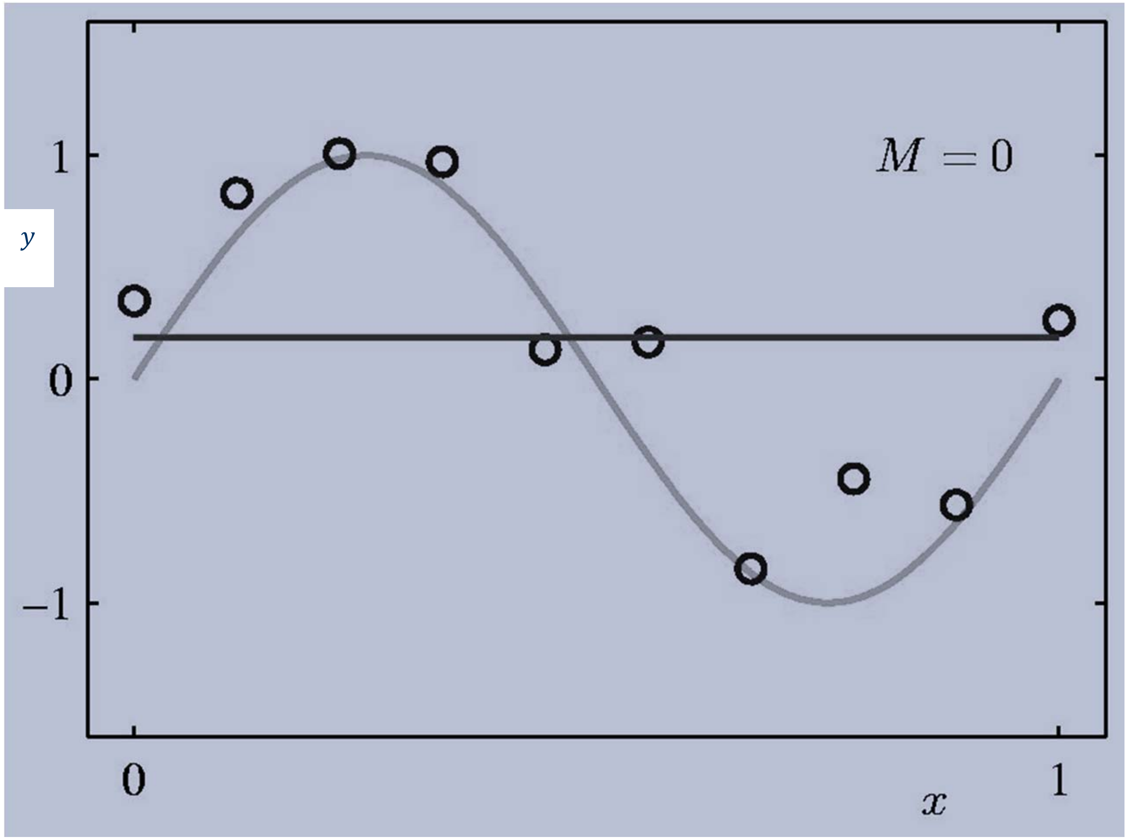
$$\boxed{RSS(f) = \sum_{i=1}^{n} \left(y_i - f(\boldsymbol{x}_i)\right)^2}$$

Sum-of-Squares Error Function

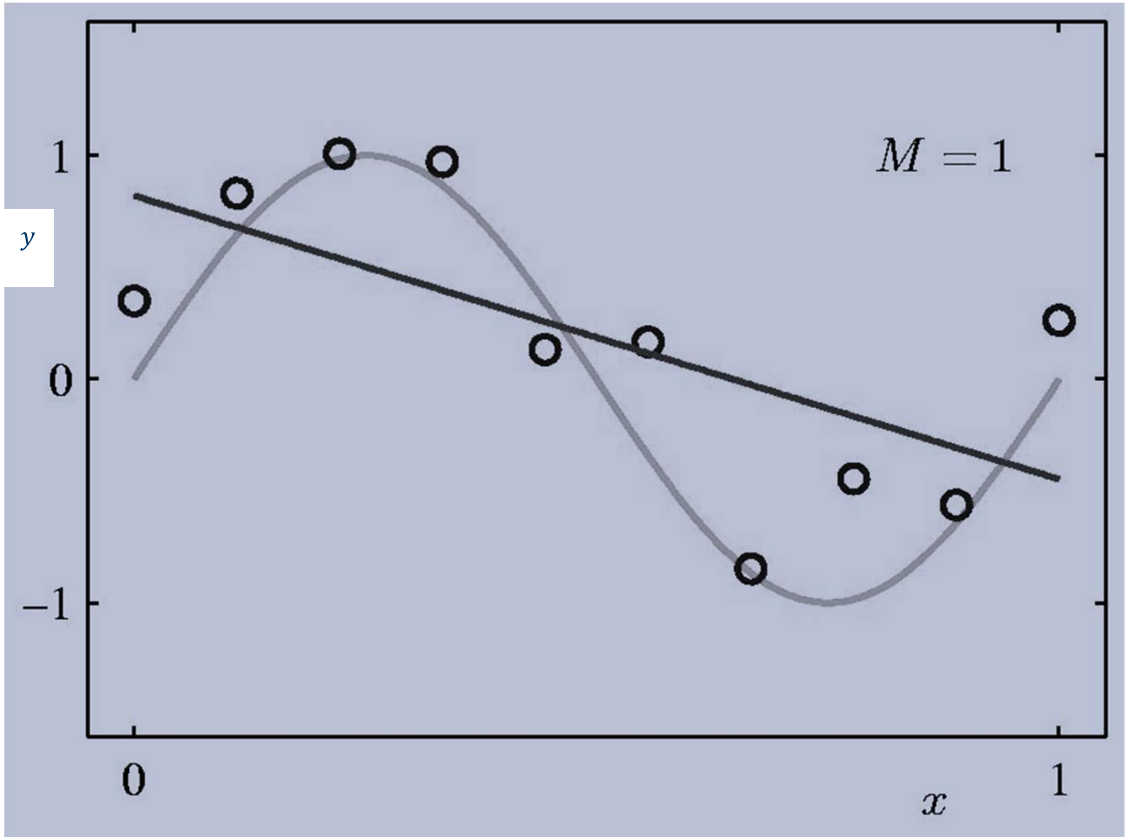


$$\mathrm{MSE}(\boldsymbol{a}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i, \boldsymbol{a}))^2$$

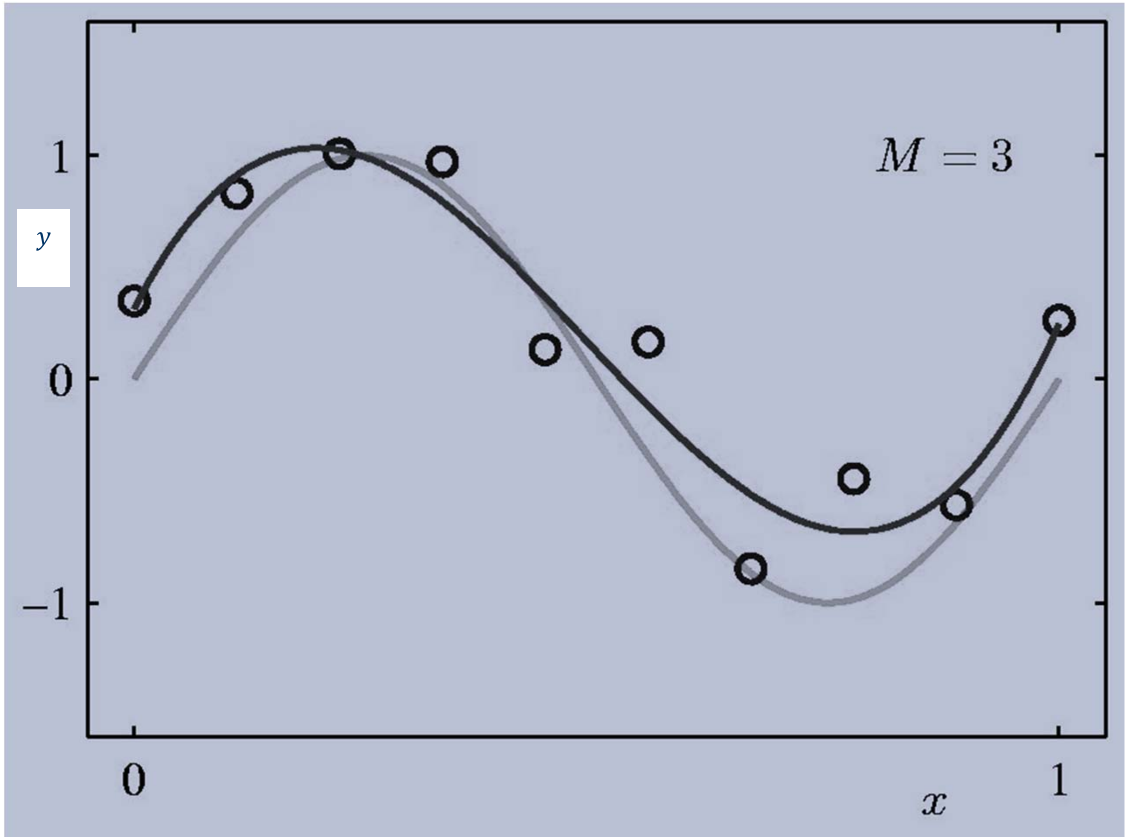
0^{th} Order Polynomial



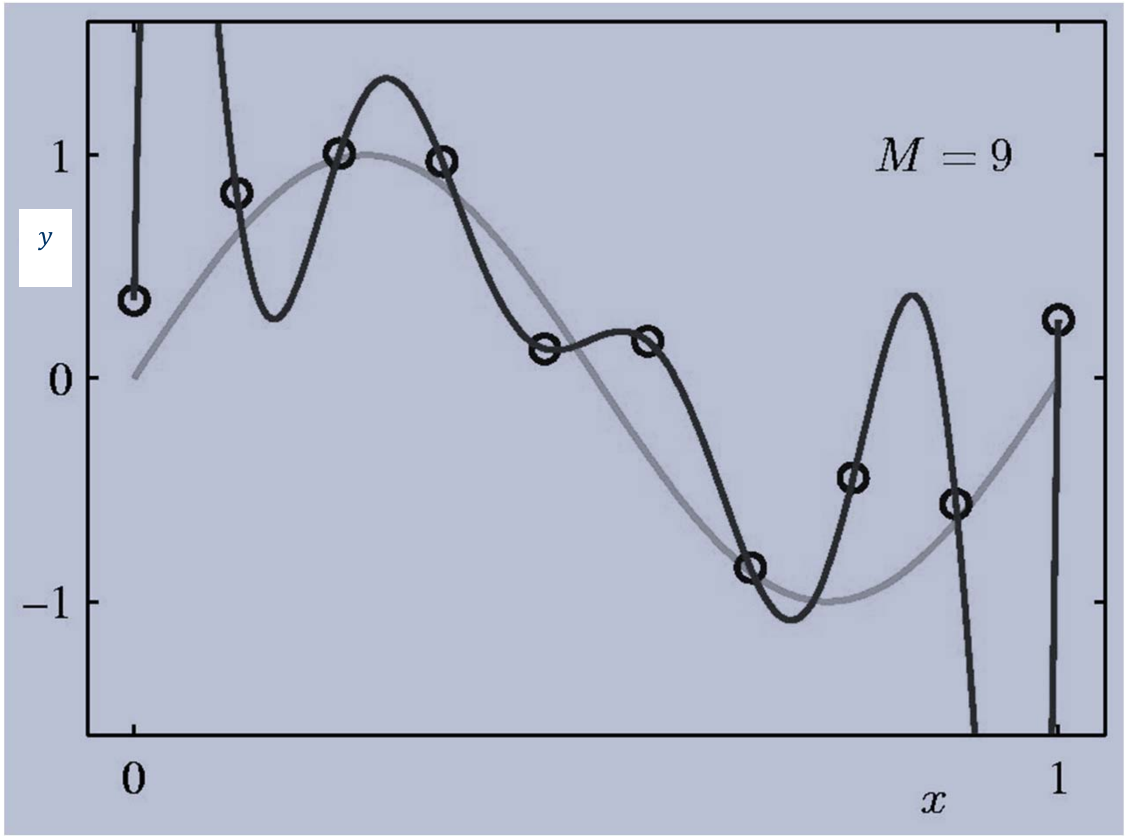
1^{st} Order Polynomial



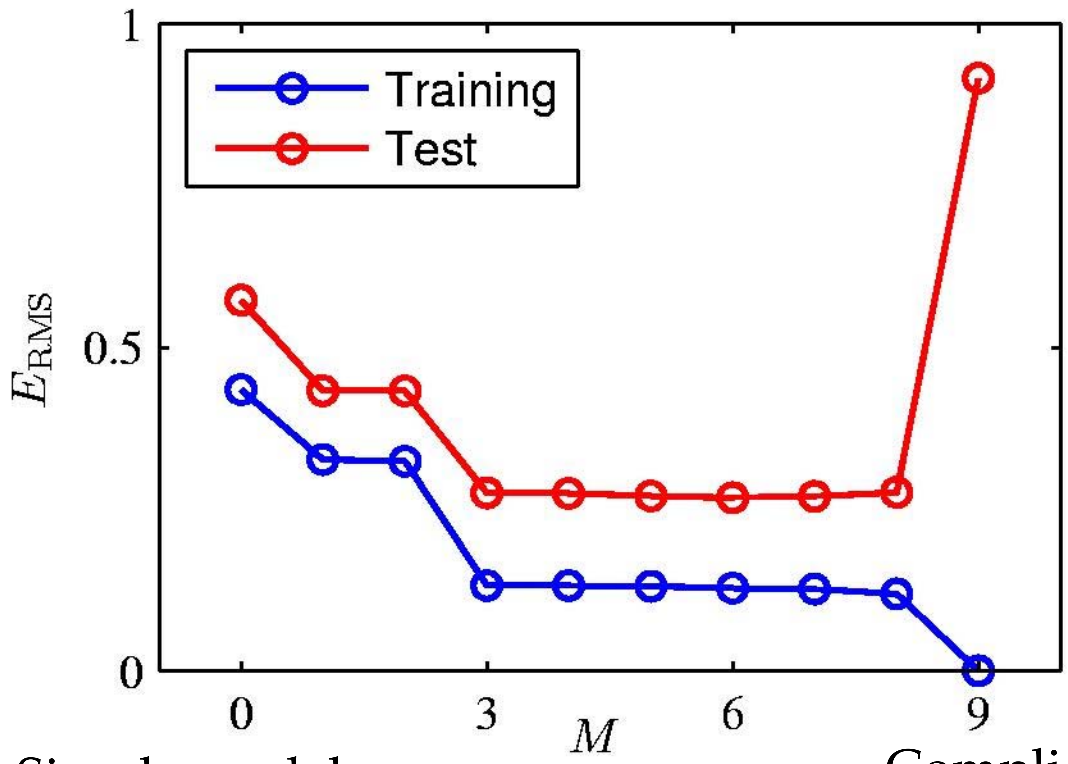
3^{rd} Order Polynomial



9^{th} Order Polynomial



Over-fitting



Simple model

Complicated model

Model complexity

Issues with MSE Criterion

$$\boldsymbol{a} = \left(XX^T\right)^{-1} X\boldsymbol{y}$$

- The solution for **a** can be obtained uniquely if XX^T is non-singular.
- If XX^T is singular, overfitting
- Coefficients

	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	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

- Why?

Ridge Regression

- How to control the size of the coefficients in Regression?

$$\boldsymbol{a}^* = \operatorname{argmin} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^T \boldsymbol{a})^2 + \lambda \sum_{j=1}^{p} a_j^2$$

- Equivalent formulation

$$\boldsymbol{a}^* = \operatorname{argmin} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^T \boldsymbol{a})^2$$

Subject to $\sum_{j=1}^{p} a_j^2 \leq t$

Lagrange multipliers

Ridge Regression

$$\boldsymbol{a}^* = \operatorname{argmin} \sum_{i=1}^{n} \left(y_i - \boldsymbol{x}_i^T \boldsymbol{a}\right)^2 + \lambda \sum_{j=1}^{p} a_j^2$$

- Matrix notations:

$$\left(\boldsymbol{y} - X^T \boldsymbol{a}\right)^T \left(\boldsymbol{y} - X^T \boldsymbol{a}\right) + \lambda \boldsymbol{a}^T \boldsymbol{a}$$

- Computing the gradient gives:

$$-2X\left(\boldsymbol{y} - X^T \boldsymbol{a}\right) + 2\lambda \boldsymbol{a}$$

- Setting the gradient to zero,

$$\left(XX^T + \lambda I\right)\boldsymbol{a} = X\boldsymbol{y}$$

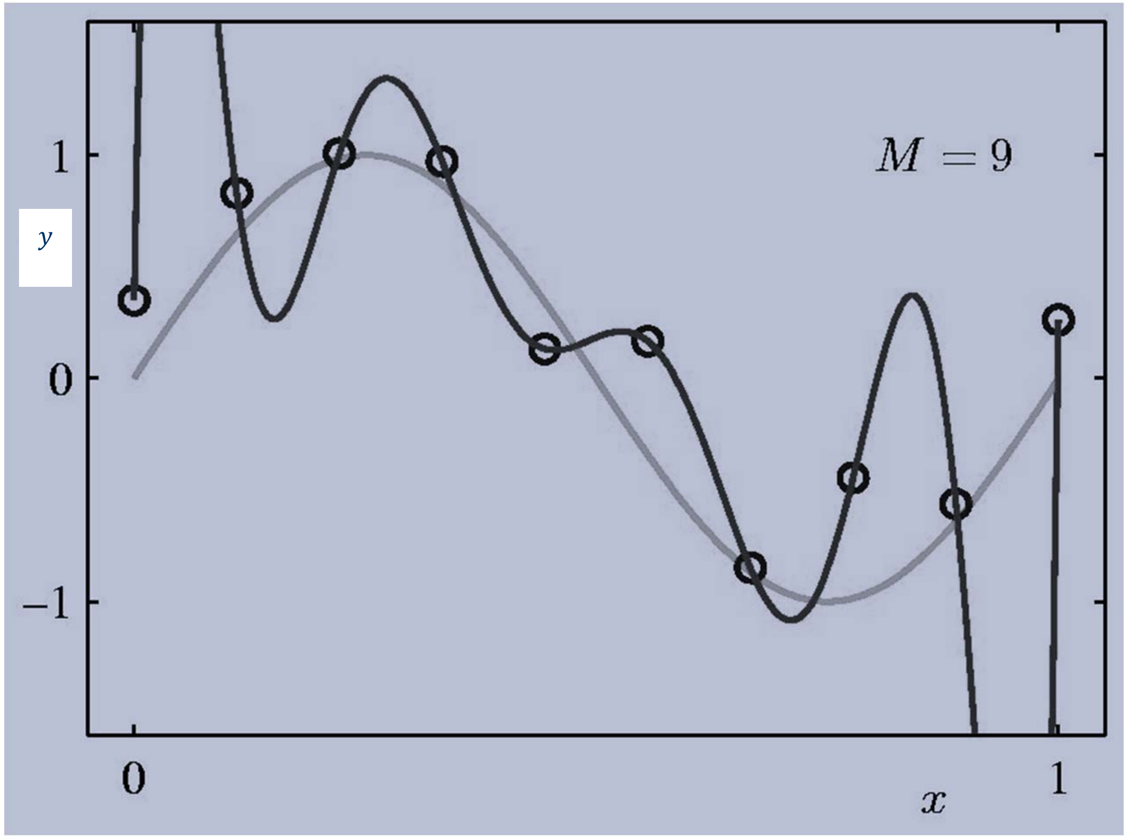
- The unique solution:

$$\boldsymbol{a}^* = \left(XX^T + \lambda \boldsymbol{I}\right)^{-1} X\boldsymbol{y}$$

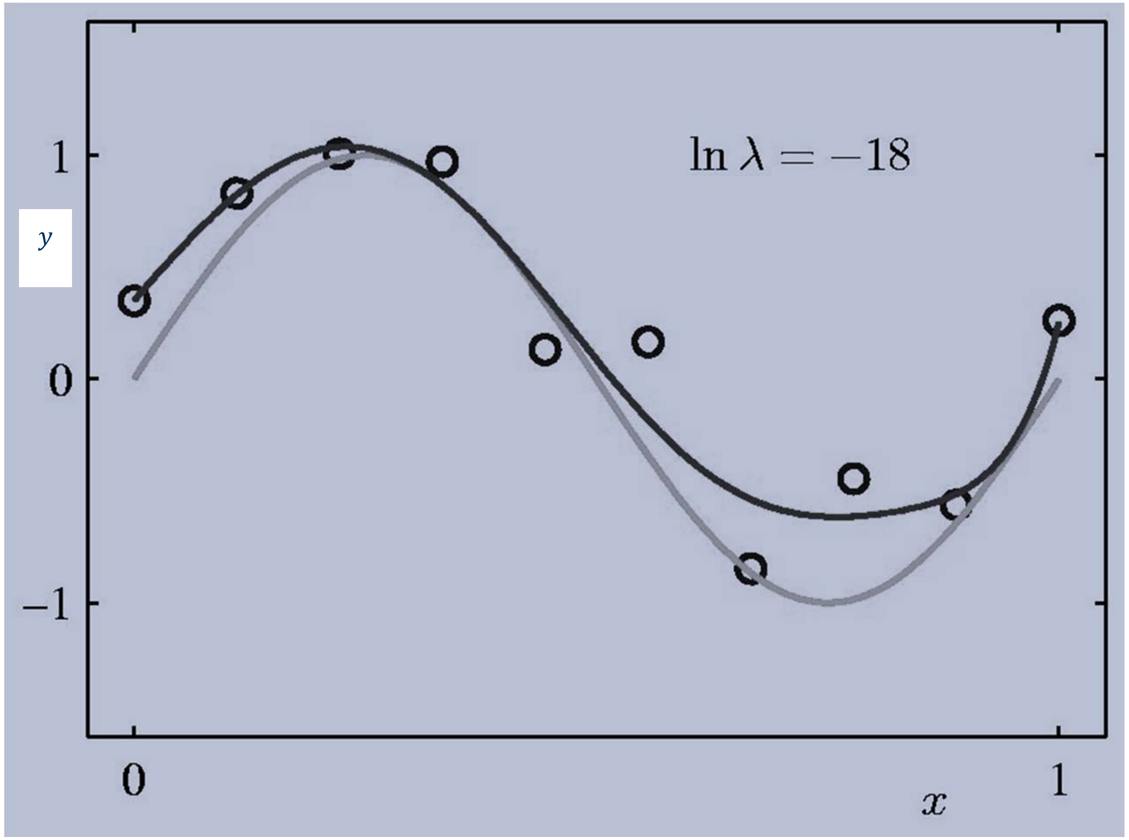
Polynomial Coefficients

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	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

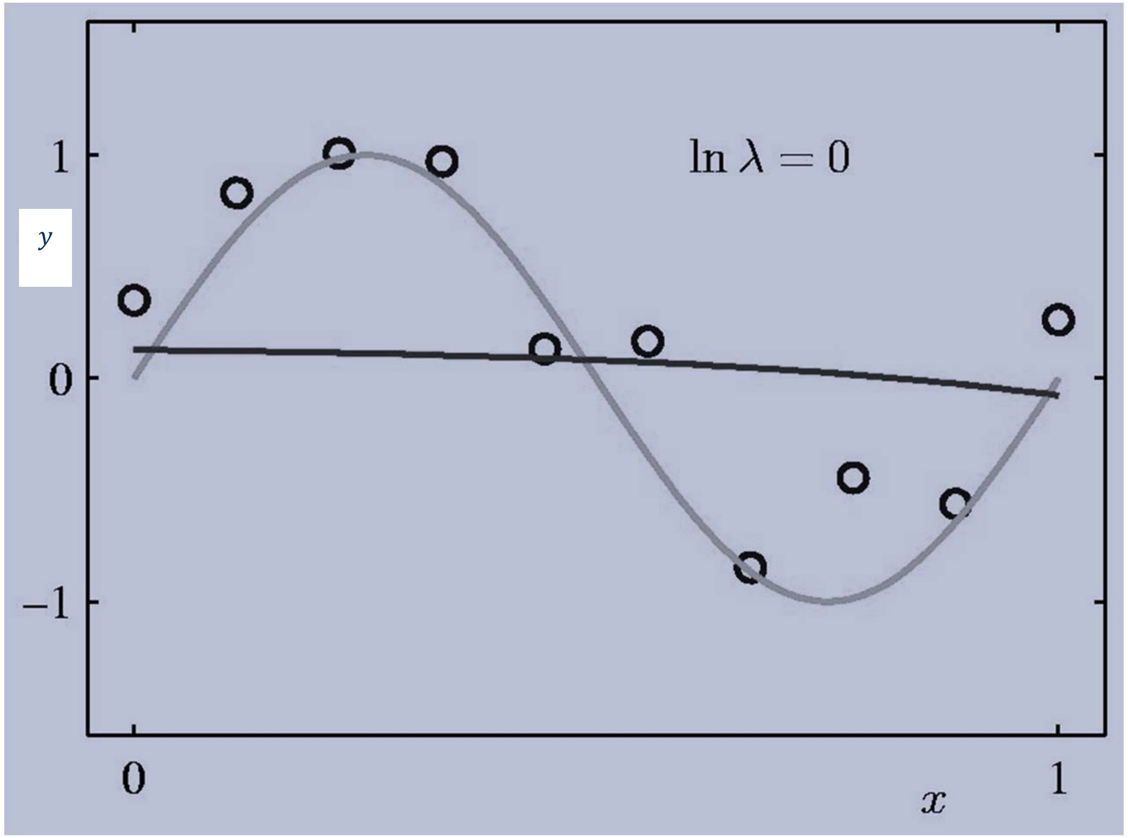
9^{th} Order Polynomial



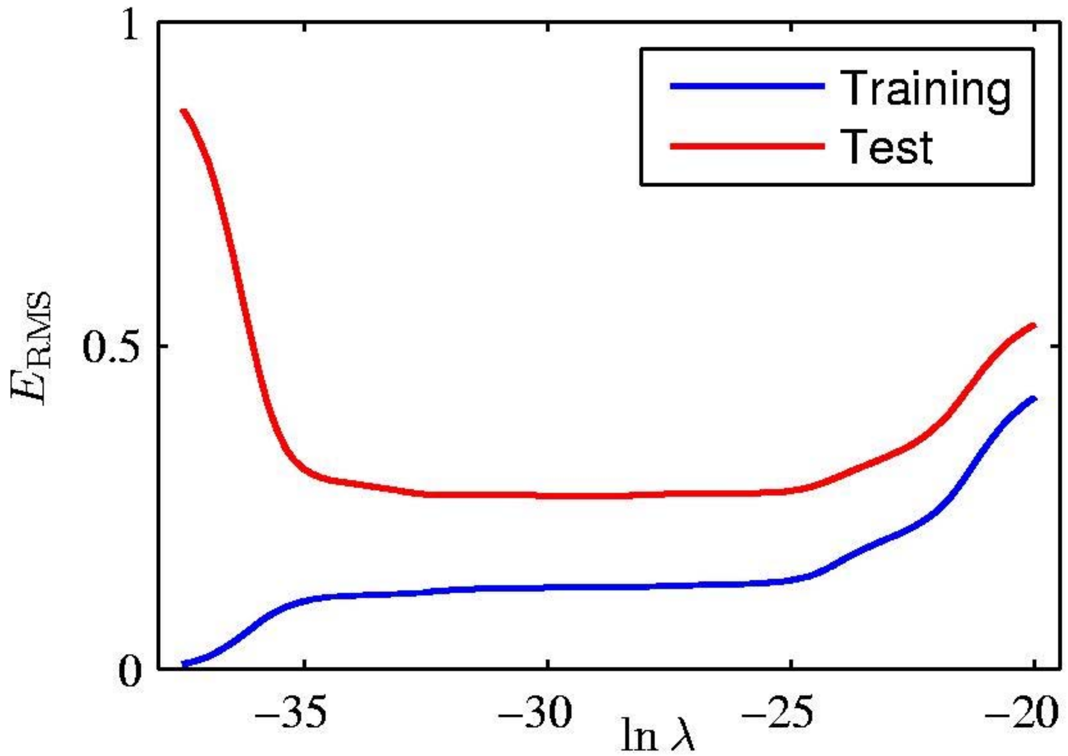
Regularization ($\ln\lambda = -18$)



Regularization ($\ln\lambda = 0$)



Regularization



Complicated model

Simple model

Model complexity

Maximum Likelihood Estimation

- A generative model: $y = f(\boldsymbol{x}, \boldsymbol{a}) + \epsilon$
- Assume: $\epsilon \sim N(0, \sigma^2)$
- $p(y|\boldsymbol{x}, \boldsymbol{a}, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(y - f(\boldsymbol{x},\boldsymbol{a}))^2}$

$$\boldsymbol{a}^* = \operatorname{argmax} L(\boldsymbol{D}, \boldsymbol{a}, \sigma) = \operatorname{argmax} \prod_{i=1}^{n} p(y_i|\boldsymbol{x}_i, \boldsymbol{a}, \sigma)$$

- $l(\boldsymbol{D}, \boldsymbol{a}, \sigma) = \log(L(\boldsymbol{D}, \boldsymbol{a}, \sigma)) = -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y - f(\boldsymbol{x}, \boldsymbol{a}))^2 + c(\sigma)$

Bayesian Linear Regression

- Bayes rule

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

$$P(\boldsymbol{a}|y, \boldsymbol{x}, \sigma) = \frac{P(y|\boldsymbol{a}, \boldsymbol{x}, \sigma)P(\boldsymbol{a}|\boldsymbol{x}, \sigma)}{P(y|\boldsymbol{x}, \sigma)}$$

$$\underbrace{\boxed{P(\boldsymbol{a}|\mathcal{D})}}_{\text{posterior}} = \frac{\overbrace{\boxed{P(\mathcal{D}|\boldsymbol{a})}}^{\text{likelihood}}\overbrace{\boxed{P(\boldsymbol{a})}}^{\text{prior}}}{P(\mathcal{D})}$$

Posterior $\propto$ likelihood $\times$ prior

Bayesian Linear Regression

Posterior $\propto$ likelihood $\times$ prior

- A common choice for the prior is

$$p(\boldsymbol{a}) = \mathcal{N}(\boldsymbol{a}|\mathbf{0}, \lambda^{-1}\boldsymbol{I})$$

$$= \frac{1}{(2\pi)^{\frac{q}{2}}|\lambda^{-1}\boldsymbol{I}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\boldsymbol{a}-\mathbf{0})^T(\lambda^{-1}\boldsymbol{I})^{-1}(\boldsymbol{a}-\mathbf{0})}$$

$$\ln(p(\boldsymbol{a})) = -\frac{\lambda}{2}\boldsymbol{a}^T\boldsymbol{a} + c$$

$$l(\boldsymbol{D}, \boldsymbol{a}, \sigma) = -\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y - f(\boldsymbol{x}, \boldsymbol{a})\right)^2 + c(\sigma)$$

LASSO

- Ridge Regression

$$\hat{\boldsymbol{\beta}} = \operatorname{argmin} \frac{1}{2n} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^T \boldsymbol{\beta})^2$$

subject to $\sum_{j=1}^{p} \beta_j^2 \le t$

$$\hat{\boldsymbol{\beta}} = \operatorname{argmin} \frac{1}{2n} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^T \boldsymbol{\beta})^2 + \lambda \|\boldsymbol{\beta}\|_2$$

- Least Absolute Selection and Shrinkage Operator

$$\hat{\boldsymbol{\beta}} = \operatorname{argmin} \frac{1}{2n} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^T \boldsymbol{\beta})^2$$

subject to $\sum_{j=1}^{p} |\beta_j| \le t$

$$\hat{\boldsymbol{\beta}} = \operatorname{argmin} \frac{1}{2n} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^T \boldsymbol{\beta})^2 + \lambda \|\boldsymbol{\beta}\|_1$$

- Sparse model

LASSO: Sparse Model

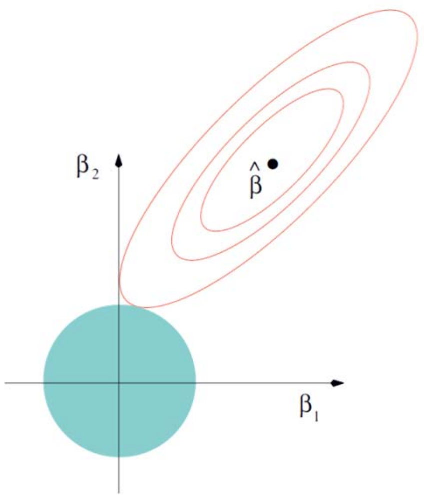
- Ridge regression VS. LASSO
  - Why LASSO → Sparse model ?

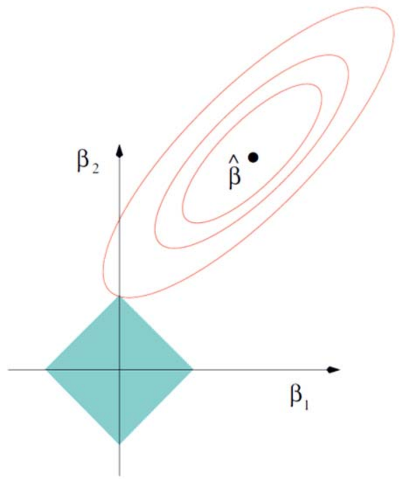
$$\hat{\boldsymbol{\beta}} = \operatorname{argmin} \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \boldsymbol{x}_i^T \boldsymbol{\beta}\right)^2$$

subject to $\sum_{j=1}^{p} \beta_j^2 \le t$

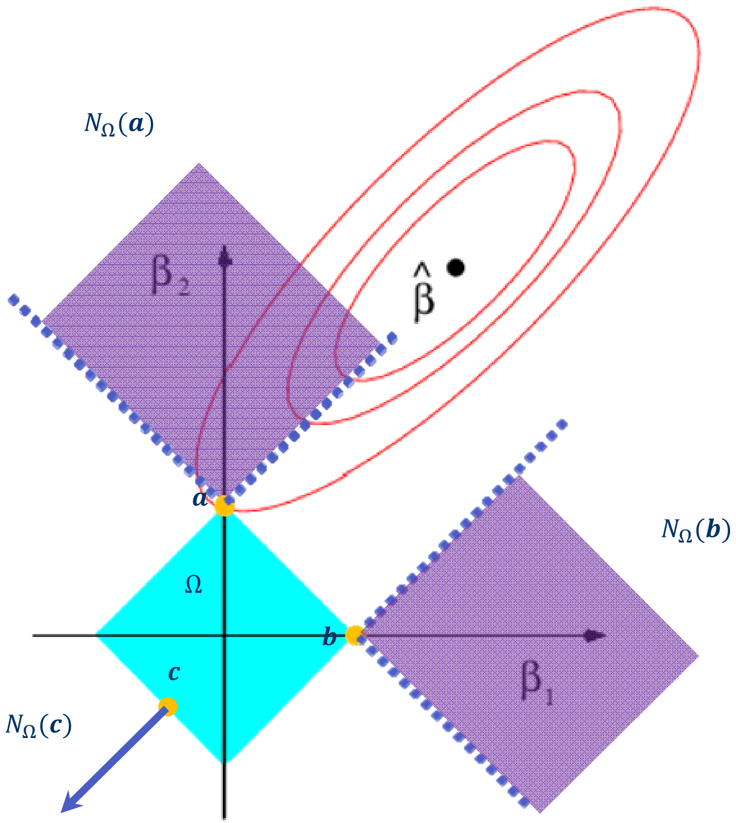
$$\hat{\boldsymbol{\beta}} = \operatorname{argmin} \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \boldsymbol{x}_i^T \boldsymbol{\beta}\right)^2$$

subject to $\sum_{j=1}^{p} |\beta_j| \le t$





LASSO: Sparse Model



- The normal cones to the feasible set at the corner points, such as $\boldsymbol{a}$ and $\boldsymbol{b}$, contain infinitely many rays, while they reduce to a singleton (only contain a single ray) at the other boundary points.
- The first order (necessary and sufficient) condition concludes that: a feasible point becomes the optimum if and only if the opposite gradient direction of the objective function falls inside the normal cone to the feasible set at that point.
- Thus, the optimum will more likely fall at the points with "larger" normal cones.
- This also explains why non-convex regularizers usually induce sparser models than the convex regularizers.

LASSO Solution

$\frac{1}{n}\sum_i y_i = 0,\ \frac{1}{n}\sum_i \boldsymbol{x}_i = \mathbf{0},\ \frac{1}{n}\sum_i x_{ij}^2 = 1$

$$\hat{\boldsymbol{\beta}} = \operatorname{argmin} \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - \boldsymbol{x}_i^T\boldsymbol{\beta}\right)^2 + \lambda\|\boldsymbol{\beta}\|_1$$

- Convex optimization
- Coordinate descent
- Single predictor (feature) setting

$$\hat{\beta} = \operatorname{argmin} \frac{1}{2n}\sum_{i=1}^{n}(y_i - z_i\beta)^2 + \lambda|\beta|$$

$\hat{\beta} = \frac{1}{n}\langle\mathbf{z},\mathbf{y}\rangle - \lambda$ If $\frac{1}{n}\langle\mathbf{z},\mathbf{y}\rangle > \lambda$

$\hat{\beta} = 0$ If $\left|\frac{1}{n}\langle\mathbf{z},\mathbf{y}\rangle\right| \le \lambda$

$\hat{\beta} = \frac{1}{n}\langle\mathbf{z},\mathbf{y}\rangle + \lambda$ If $\frac{1}{n}\langle\mathbf{z},\mathbf{y}\rangle < -\lambda$

LASSO Solution

- Single predictor (feature) setting

$$\hat{\beta} = \operatorname{argmin} \frac{1}{2n}\sum_{i=1}^{n}(y_i - z_i\beta)^2 + \lambda|\beta|$$

$$\frac{1}{n}\sum_i y_i = 0, \ \frac{1}{n}\sum_i z_i = 0, \ \frac{1}{n}\sum_i z_i^2 = 1$$

$$f(\beta) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - z_i\beta)^2 + \lambda|\beta| = \frac{1}{2n}(\boldsymbol{y} - \beta\boldsymbol{z})^T(\boldsymbol{y} - \beta\boldsymbol{z}) + \lambda|\beta|$$

$$f(\beta) = \frac{1}{2n}\boldsymbol{z}^T\boldsymbol{z}\beta^2 - \frac{1}{n}\langle\mathbf{z},\boldsymbol{y}\rangle\beta + \lambda|\beta| + \frac{1}{2n}\boldsymbol{y}^T\boldsymbol{y}$$

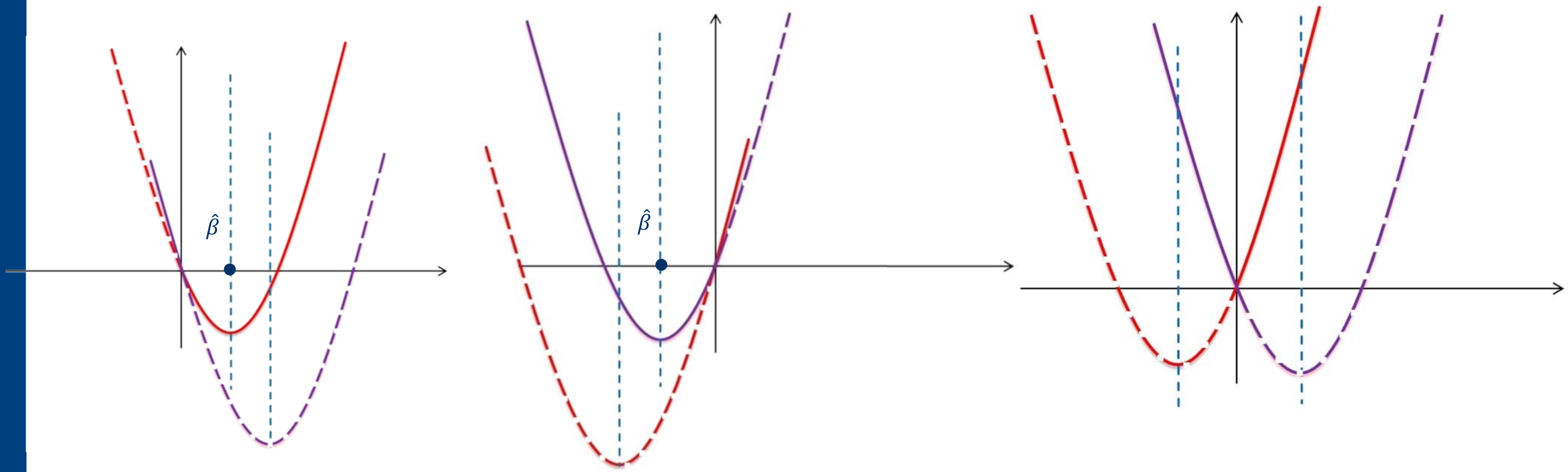
$$f(\beta) = \frac{1}{2}\beta^2 - \frac{1}{n}\langle\mathbf{z},\boldsymbol{y}\rangle\beta + \lambda|\beta|$$

$$f(\beta) = \begin{cases} \frac{1}{2}\beta^2 - \left(\frac{1}{n}\langle\mathbf{z},\mathbf{y}\rangle - \lambda\right)\beta, & \beta \geq 0 \\ \frac{1}{2}\beta^2 - \left(\frac{1}{n}\langle\mathbf{z},\mathbf{y}\rangle + \lambda\right)\beta, & \beta < 0 \end{cases}$$

LASSO Solution

- $$f(\beta) = \begin{cases} \frac{1}{2}\beta^2 - \left(\frac{1}{n}\langle \mathbf{z}, \mathbf{y} \rangle - \lambda\right)\beta, & \beta \geq 0 \\ \frac{1}{2}\beta^2 - \left(\frac{1}{n}\langle \mathbf{z}, \mathbf{y} \rangle + \lambda\right)\beta, & \beta < 0 \end{cases}$$

$$\frac{1}{n}\langle \mathbf{z}, \mathbf{y} \rangle > \lambda \qquad \frac{1}{n}\langle \mathbf{z}, \mathbf{y} \rangle < -\lambda \qquad \left|\frac{1}{n}\langle \mathbf{z}, \mathbf{y} \rangle\right| < \lambda$$



$$\hat{\beta} = \frac{1}{n}\langle \mathbf{z}, \mathbf{y} \rangle - \lambda \qquad \hat{\beta} = \frac{1}{n}\langle \mathbf{z}, \mathbf{y} \rangle + \lambda \qquad \hat{\beta} = 0$$

LASSO Solution

$$\hat{\boldsymbol{\beta}} = \operatorname{argmin} \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \boldsymbol{x}_i^T \boldsymbol{\beta}\right)^2 + \lambda \|\boldsymbol{\beta}\|_1$$

- Convex optimization
- Coordinate descent
- Single predictor (feature) setting
- Multiple predictors (features)
  - Cyclic Coordinate descent
- LARs

Bias & Variance Decomposition

The Supervised Learning Problem

- **Given** example pairs $[\boldsymbol{x_i}, y_i]$
- **Learn** a function $f(\boldsymbol{x})$, such as $f(\boldsymbol{x}) = y$
- Loss: $L\big(y, f(\boldsymbol{x})\big)$
- Expected Loss:

$$\mathrm{E}(L) = \iint L\big(y, f(\boldsymbol{x})\big)\, p(\boldsymbol{x}, y) d\boldsymbol{x} dy$$

- Squared loss: $L\big(y, f(\boldsymbol{x})\big) = (y - f(\boldsymbol{x}))^2$
- Expected Prediction Error:

$$\mathrm{EPE}(f) = \boxed{\iint (y - f(\boldsymbol{x}))^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy}$$

$$f^*(\boldsymbol{x}) = ?$$

$$\text{EPE}(f) = \iint (y - f(\boldsymbol{x}))^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy$$

- Squared loss

$$\left(y - f(\boldsymbol{x})\right)^2 = \left(y - E(y|\boldsymbol{x}) + E(y|\boldsymbol{x}) - f(\boldsymbol{x})\right)^2$$

$$= \left(y - E(y|\boldsymbol{x})\right)^2 + \left(E(y|\boldsymbol{x}) - f(\boldsymbol{x})\right)^2 + 2\left(y - E(y|\boldsymbol{x})\right)\left(E(y|\boldsymbol{x}) - f(\boldsymbol{x})\right)$$

- Expected Prediction Error:

$$\text{EPE}(f) = \int \left(f(\boldsymbol{x}) - E(y|\boldsymbol{x})\right)^2 p(\boldsymbol{x}) d\boldsymbol{x} + \int \text{var}(y|\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}$$

- The first term:
  - $f^*(\boldsymbol{x}) = E(y|\boldsymbol{x})$
- The second term:

In Reality

- **Given** training set D, contains n example pairs $[\boldsymbol{x_i}, y_i]$
- **Learn** a function $f(\boldsymbol{x})$, such as $f(\boldsymbol{x})=y$
- Expected Prediction Error:

$$\mathrm{EPE}(f) = \iint (y - f(\boldsymbol{x}))^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy$$

$$f(\boldsymbol{x}) \rightarrow f(\boldsymbol{x}; D)$$

$$E_D\big(f(\boldsymbol{x}; D)\big)$$

In Reality

- Expected Prediction Error:

$$\mathrm{EPE}(f) = \int \left(f(\boldsymbol{x}) - E(y|\boldsymbol{x})\right)^2 p(\boldsymbol{x}) d\boldsymbol{x} + \int \mathrm{var}(y|\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}$$

$$= \int \boxed{\left(f(\boldsymbol{x}; D) - E(y|\boldsymbol{x})\right)^2} p(\boldsymbol{x}) d\boldsymbol{x} + \int \mathrm{var}(y|\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}$$

$$\boxed{\left[f(\boldsymbol{x}; D) - E_D\left(f(\boldsymbol{x}; D)\right) + E_D\left(f(\boldsymbol{x}; D)\right) - E(y|\boldsymbol{x})\right]^2}$$

$$\boxed{\left(f(\boldsymbol{x}; D) - E_D\left(f(\boldsymbol{x}; D)\right)\right)^2 + \left(E_D\left(f(\boldsymbol{x}; D)\right) - E(y|\boldsymbol{x})\right)^2 + 2\left(f(\boldsymbol{x}; D) - E_D\left(f(\boldsymbol{x}; D)\right)\right)\left(E_D\left(f(\boldsymbol{x}; D)\right) - E(y|\boldsymbol{x})\right)}$$

$$E_{\boldsymbol{D}}\left\{\left(f(\boldsymbol{x}; D) - E(y|\boldsymbol{x})\right)^2\right\}$$

$$\underbrace{E_D\left\{\left[f(\boldsymbol{x}; D) - E_D\left(f(\boldsymbol{x}; D)\right)\right]^2\right\}}_{\text{Variance}} + \underbrace{\left\{E_D\left(f(\boldsymbol{x}; D)\right) - E(y|\boldsymbol{x})\right\}^2}_{(\text{Bias})^2}$$

Bias-variance Decomposition

- $\text{EPE}(f) = \iint (y - f(\boldsymbol{x}))^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy$
- Expected prediction error (expected loss) =

  (bias)^2 + variance + noise

- (bias)^2:

$$\int \left\{E_D\big(f(\boldsymbol{x}; D)\big) - E(y|\boldsymbol{x})\right\}^2 p(\boldsymbol{x}) d\boldsymbol{x}$$

- variance:

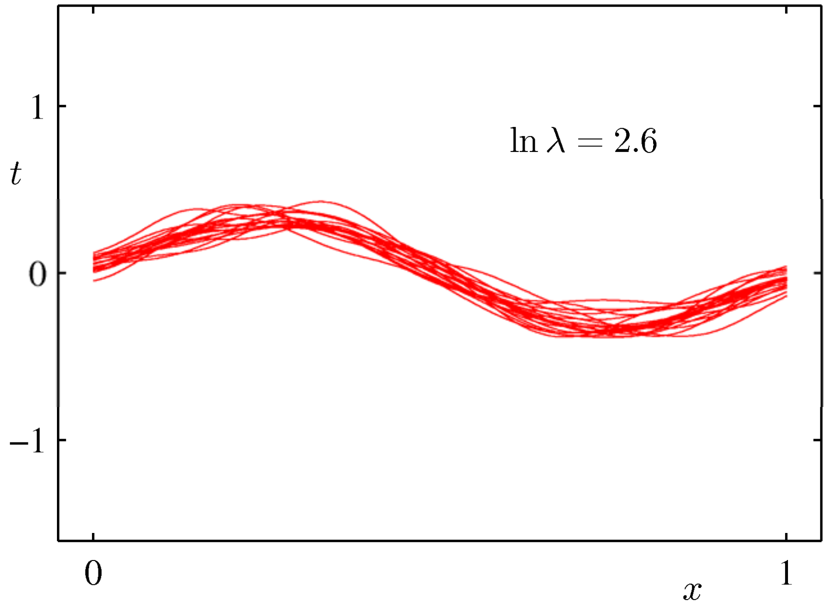
$$\int E_D\left\{\left[f(\boldsymbol{x}; D) - E_D\big(f(\boldsymbol{x}; D)\big)\right]^2\right\} p(\boldsymbol{x}) d\boldsymbol{x}$$

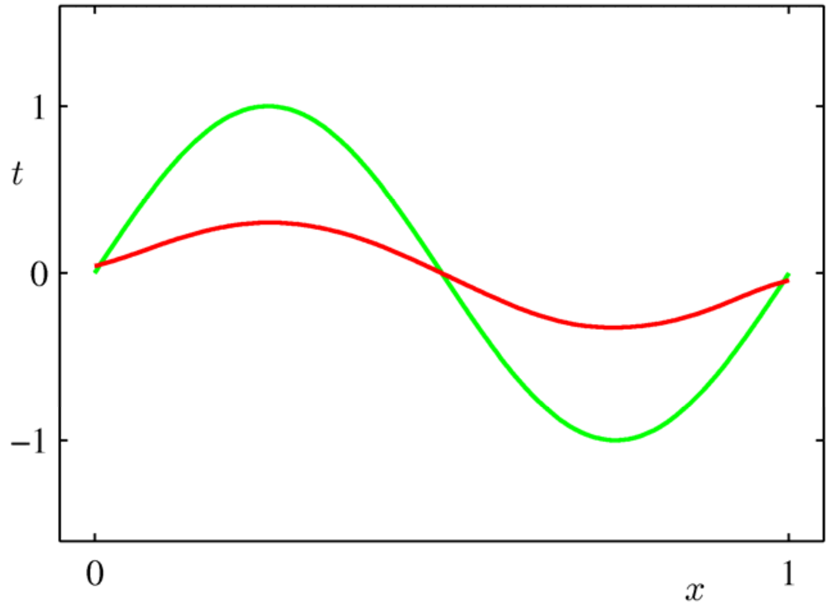
- noise:

$$\int \text{var}(y|\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}$$

The Bias-Variance Decomposition

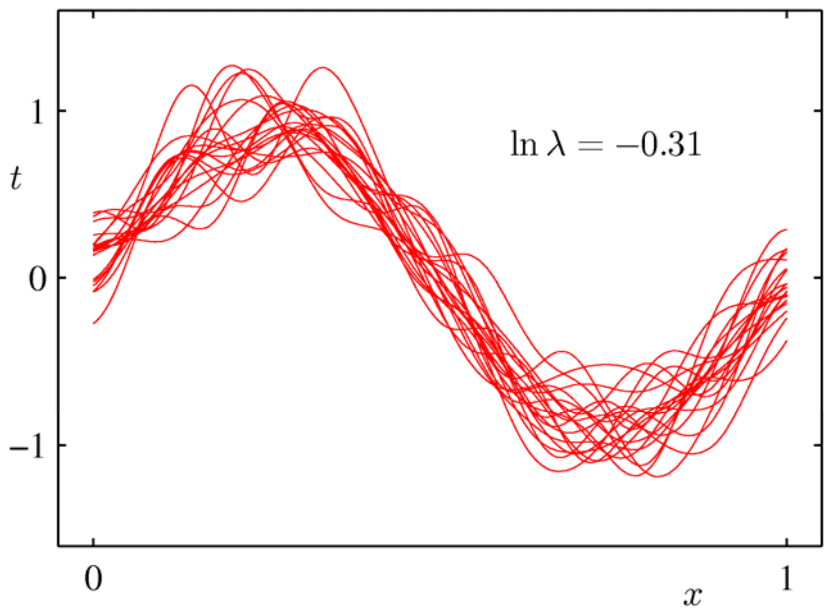
- Example: 25 data sets from the sinusoidal, varying the degree of regularization, λ.

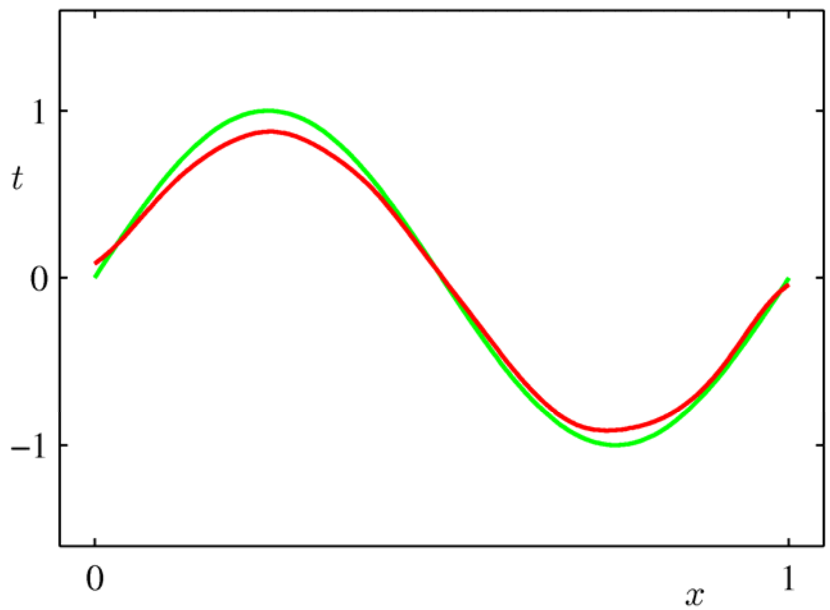




The Bias-Variance Decomposition

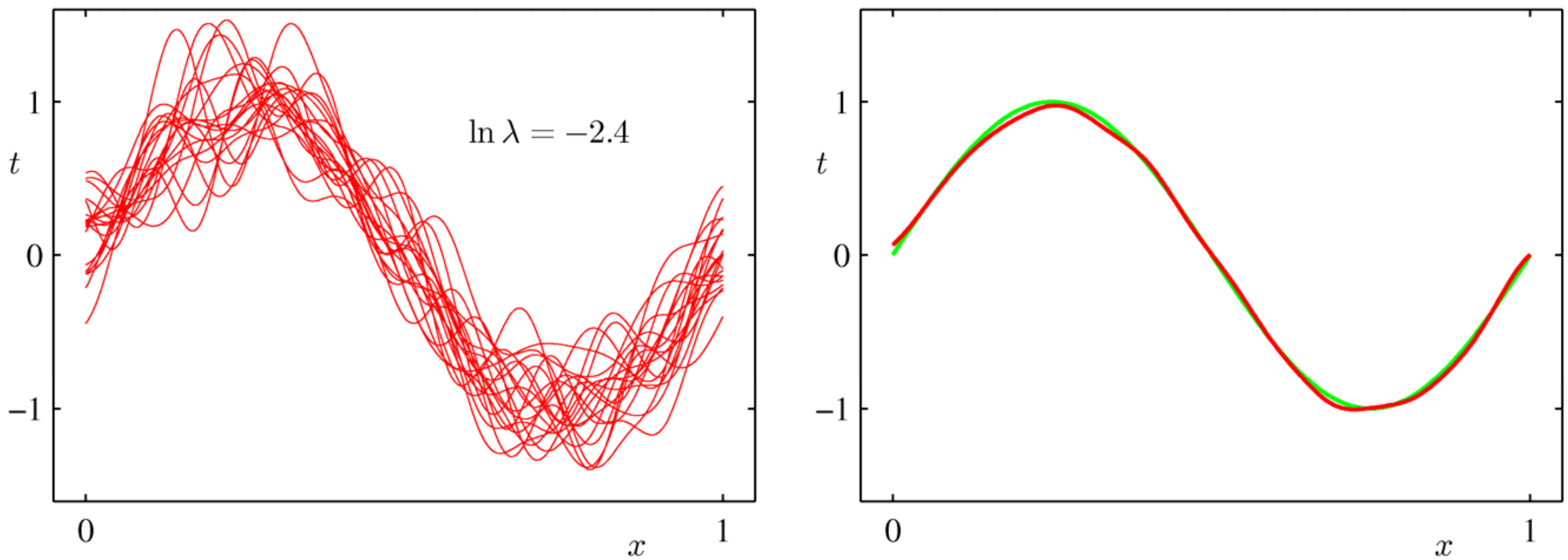
- Example: 25 data sets from the sinusoidal, varying the degree of regularization, λ.





The Bias-Variance Decomposition

- Example: 25 data sets from the sinusoidal, varying the degree of regularization, λ.



The Bias-Variance Trade-off

- Over-regularized model (large λ) →high bias
- Under-regularized model (small λ) → high variance.