

Machine Learning

### Linear Regression with multiple variables

### Multiple features

#### Multiple features (variables).

Size (feet²)	Price (\$1000)		
$\rightarrow x$	y <b>~</b>		
2104	460		
1416	232		
1534	315		
852	178		

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

#### Multiple features (variables).

Size (feet²)	Number of bedrooms	Number of floors	Age of home (years)	Price (\$1000)
<b>×</b> <sub>1</sub>	×z	<b>×3</b>	*4	9
2104	5	1	45	460
<del>&gt;</del> 1416	3	2	40	232 M= 47
1534	3	2	30	315
852	2	1	36	178
 Notation:	 <b>★</b>	 *	 1	] / [1416]
$\rightarrow n$ = number of features $n = 4$ $\rightarrow x^{(i)}$ = input (features) of $i^{th}$ training example.			$\frac{\chi^{(2)}}{2} = \begin{bmatrix} 1416 \\ \frac{3}{2} \\ 40 \end{bmatrix} \in$	
$\Rightarrow x_j^{(i)}$ = value of feature $j$ in $i^{th}$ training example.				

#### Hypothesis:

Previously: 
$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

For convenience of notation, define 
$$x_0 = 1$$
. [So  $\theta_1 = 1$ ]

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^{m_1} \qquad 0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_1 \end{bmatrix}$$

Multivariate linear regression.



Machine Learning

### Linear Regression with multiple variables

Gradient descent for multiple variables

Hypothesis: 
$$h_{\theta}(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

Parameters: 
$$\theta_0, \theta_1, \dots, \theta_n$$

#### Cost function:

$$J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

#### Gradient descent:

Repeat 
$$\{$$
  $\Rightarrow \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \dots, \theta_n)$ . **5(e)**  $\}$  (simultaneously update for every  $j=0,\dots,n$ )

#### **Gradient Descent**

Previously (n=1):

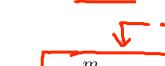
$$t = \theta_0 - o \left[ \frac{1}{m} \sum_{i=1}^{m} (h_{\theta}(x^{(i)}) - y^{(i)}) \right]$$

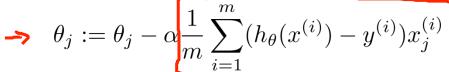
$$\left[ rac{\partial}{\partial heta_0} J( heta) 
ight]$$

$$i=1$$
(simultaneously undate  $\hat{H}_0$ ,  $\hat{H}_1$ )

(simultaneously update  $\theta_0, \theta_1$ )

New algorithm  $(n \ge 1)$ :





neously update 
$$\theta_i$$
 for

(simultaneously update 
$$\theta_j$$
 for  $j=0,\ldots,n$ )

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) \underline{x^{(i)}}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{\substack{i=1\\m}} (h_{\theta}(x^{(i)}) - y^{(i)}) x_1^{(i)}$$



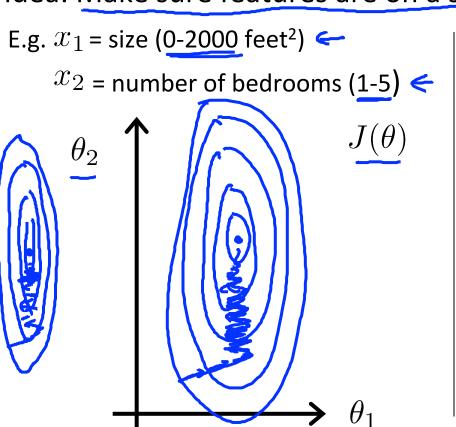
Machine Learning

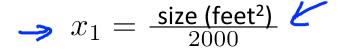
# Linear Regression with multiple variables

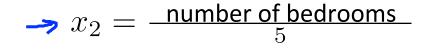
Gradient descent in practice I: Feature Scaling

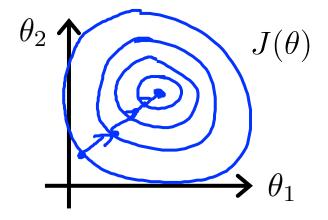
#### **Feature Scaling**

Idea: Make sure features are on a similar scale.









#### **Feature Scaling**

Get every feature into approximately a

#### Mean normalization

Replace  $\underline{x}_i$  with  $\underline{x}_i - \mu_i$  to make features have approximately zero mean (Do not apply to  $\underline{x}_0 = 1$ ).



Machine Learning

# Linear Regression with multiple variables

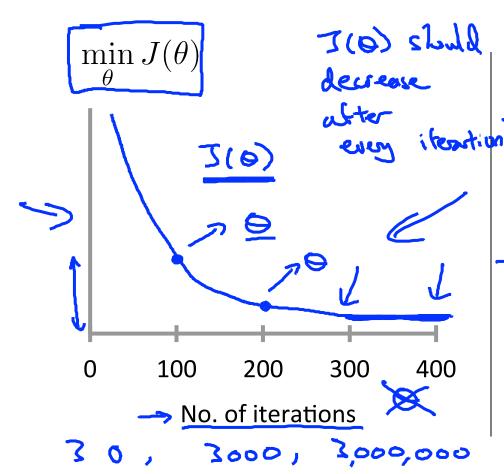
Gradient descent in practice II: Learning rate

#### **Gradient descent**

$$\rightarrow \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

- "Debugging": How to make sure gradient descent is working correctly.
- How to choose learning rate  $\alpha$ .

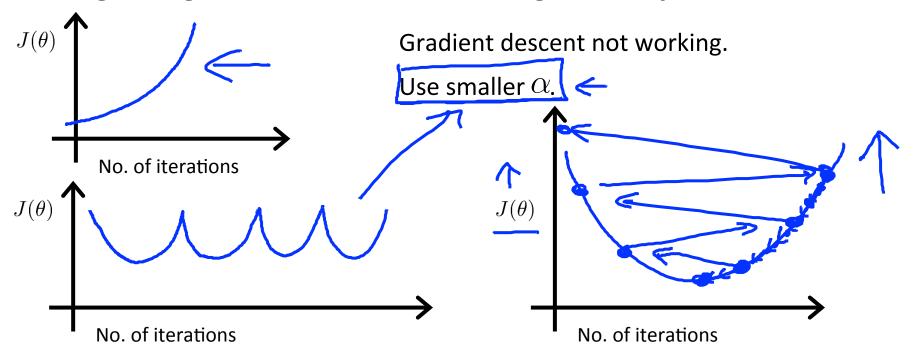
#### Making sure gradient descent is working correctly.



Example automatic convergence test:

 $\rightarrow$  Declare convergence if  $J(\theta)$  decreases by less than  $10^{-3}$  in one iteration.

#### Making sure gradient descent is working correctly.



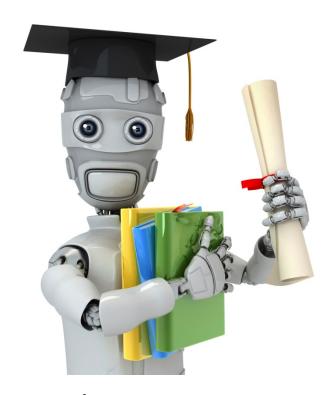
- For sufficiently small lpha, J( heta) should decrease on every iteration.
- But if lpha is too small, gradient descent can be slow to converge.

#### **Summary:**

- If  $\alpha$  is too small: slow convergence.
- If  $\alpha$  is too large:  $J(\theta)$  may not decrease on every iteration; may not converge. (Slow converge)

To choose  $\alpha$ , try

$$\dots, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, \dots$$



Machine Learning

# Linear Regression with multiple variables

Features and polynomial regression

#### Housing prices prediction

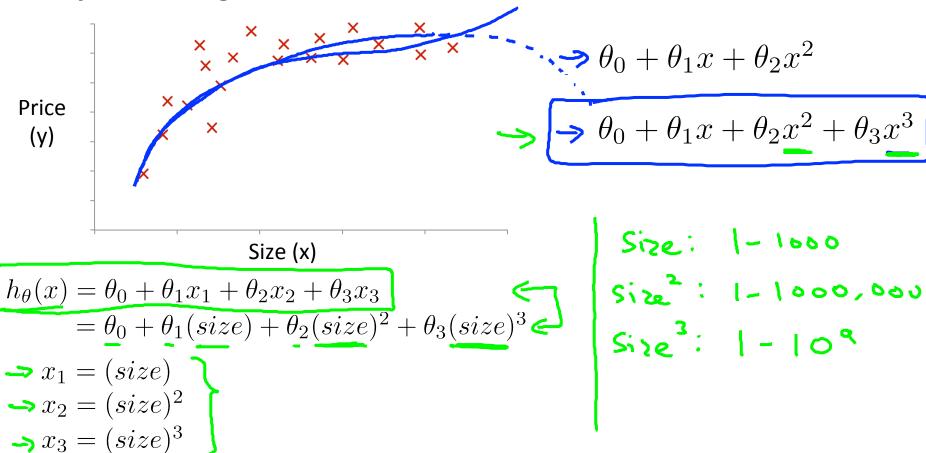
$$h_{\theta}(x) = \theta_0 + \theta_1 \times frontage + \theta_2 \times depth$$

Area

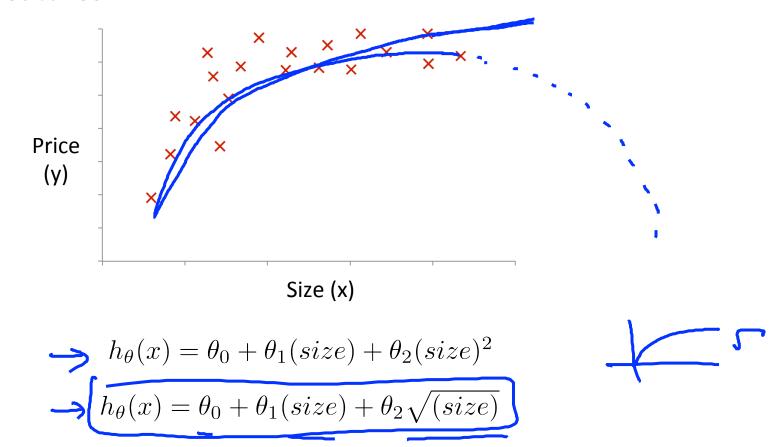
 $\times = frontage \times depth$ 
 $h_{\theta}(x) = \Theta_0 + \Theta_1 \times depth$ 

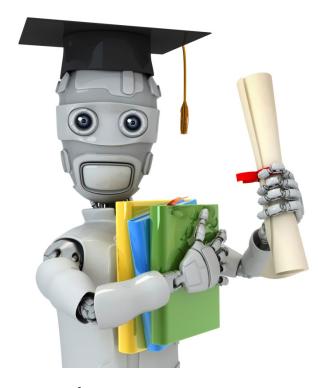


#### **Polynomial regression**



#### **Choice of features**



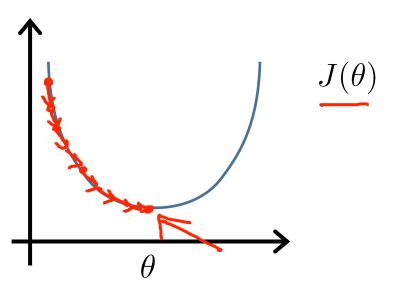


Machine Learning

# Linear Regression with multiple variables

Normal equation

#### **Gradient Descent**

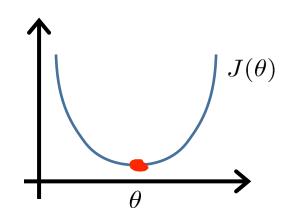


Normal equation: Method to solve for  $\theta$  analytically.

Intuition: If 1D  $(\theta \in \mathbb{R})$ 

$$J(\theta) = a\theta^2 + b\theta + c$$

$$\frac{\partial}{\partial \phi} J(\phi) = \dots \qquad \frac{\text{Set}}{\partial \phi} O$$
Solve for  $\phi$ 



$$\underline{\theta \in \mathbb{R}^{n+1}} \qquad \underline{J(\theta_0, \theta_1, \dots, \theta_m)} = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

$$\underline{\frac{\partial}{\partial \theta_i} J(\theta)} = \cdots \stackrel{\boldsymbol{\leq}}{=} 0 \qquad \text{(for every } j\text{)}$$

Solve for  $\theta_0, \theta_1, \ldots, \theta_n$ 

#### Examples: $\underline{m} = 4$ .

J	Size (feet²)	Number of bedrooms	Number of floors	Age of home (years)	Price (\$1000	)
$\rightarrow x_0$	$x_1$	$x_2$	$x_3$	$x_4$	y	
1	2104	5	1	45	460	7
1	1416	3	2	40	232	
1	1534	3	2	30	315	
1,	852	2	_1	<b>3</b> 6	178	7
	$X = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$	$2104   5   1$ $416   3   2$ $1534   3   2$ $852   2   1$ $M   \times                                  $	$\begin{bmatrix} 2 & 30 \\ 36 \end{bmatrix}$	$y = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	460 232 315 178	1est or

### <u>m</u> examples $(x^{(1)}, y^{(1)}), \ldots, (\underline{x^{(m)}, y^{(m)}})$ ; <u>n</u> features.

$$\underline{x^{(i)}} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$(\operatorname{des}_{\mathsf{sign}} \\ \operatorname{nock}_{\mathsf{n}})$$

$$(\operatorname{h}_{\mathsf{x}} (\operatorname{h}_{\mathsf{i}}))^{\mathsf{T}}$$

Andrew Ng

$$\underbrace{\theta = \underbrace{(X^T X)^{-1} X^T y}}_{(\mathbf{Y}^T \mathbf{Y})^{-1} \cdot \mathbf{1} \cdot \mathbf{1}} \boldsymbol{\angle}$$

 $(X^TX)^{-1}$  is inverse of matrix  $(X^TX)$ .

$$\frac{(x^{7}x)^{-1}}{(x^{7}x)^{-1}} = A^{-1}$$

$$\frac{\text{pino}(X^T + X) + X^T + y}{0 \leq X_1 \leq 1}$$

$$0 = 6 \left(X^T \times \right)^{-1} \times Ty \quad \text{min } J(6) \quad 0 \leq X_2 \leq 1000$$

$$0 \leq X_1 \leq 10^{-5} \times 10^{-5} \text{ min } J(6)$$

#### m training examples, $\underline{n}$ features.

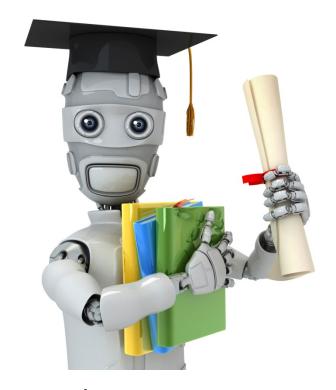
#### **Gradient Descent**

- $\rightarrow$  Need to choose  $\alpha$ .
- → Needs many iterations.
  - Works well even when n is large.



#### **Normal Equation**

- $\rightarrow$  No need to choose  $\alpha$ .
- Don't need to iterate.
  - Need to compute
- $(X^T X)^{-1} \quad \underset{\mathsf{n} \times \mathsf{n}}{\overset{\mathsf{n} \times \mathsf{n}}{\longrightarrow}} \quad O(\mathsf{n}^3)$ 
  - Slow if n is very large.



Machine Learning

# Linear Regression with multiple variables

Normal equation and non-invertibility (optional)

#### Normal equation

$$\theta = (X^T X)^{-1} X^T y$$



- What if  $X^TX$  is non-invertible? (singular/degenerate)
- Octave: pinv(X'\*X)\*X'\*y



### What if $X^TX$ s non-invertible?

Redundant features (linearly dependent).

E.g. 
$$x_1 = \text{size in feet}^2$$
 $x_2 = \text{size in m}^2$ 
 $x_1 = (3.18)^2 \times 2$ 

Too many features (e.g.  $m \le n$ ).

- Delete some features, or use regularization.

