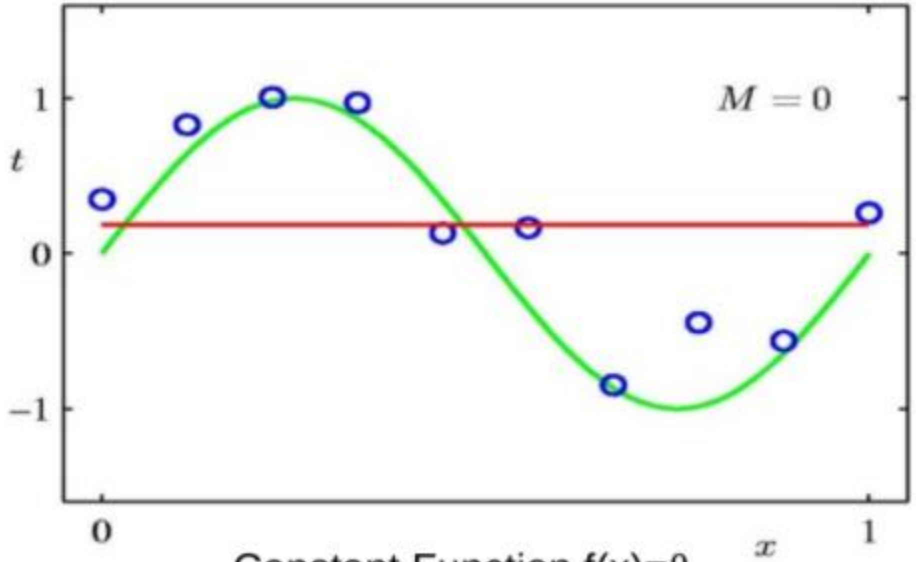
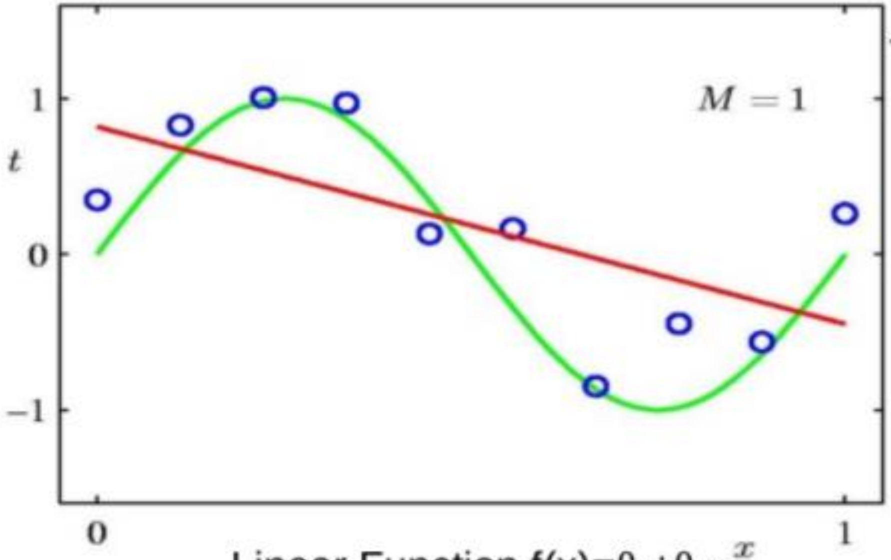
Linear Regression

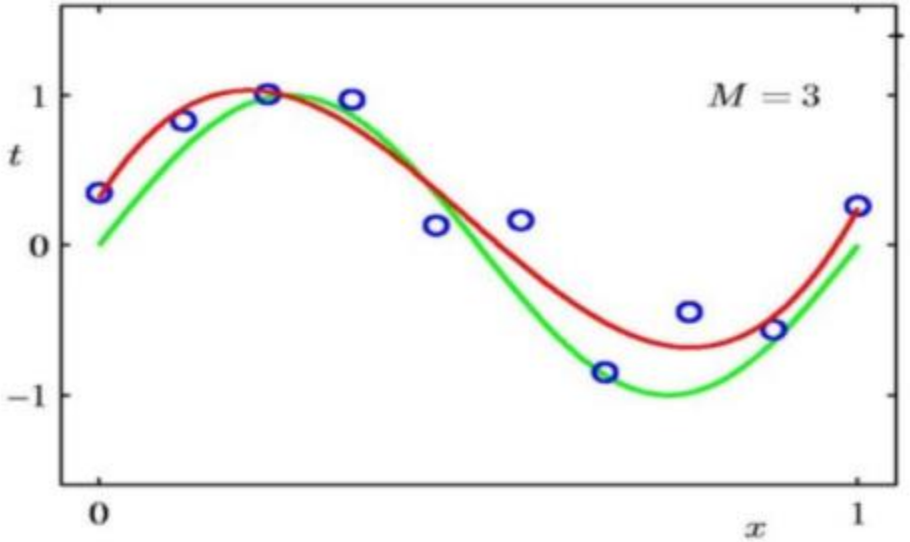


Constant Function $f(x)=\theta_0$

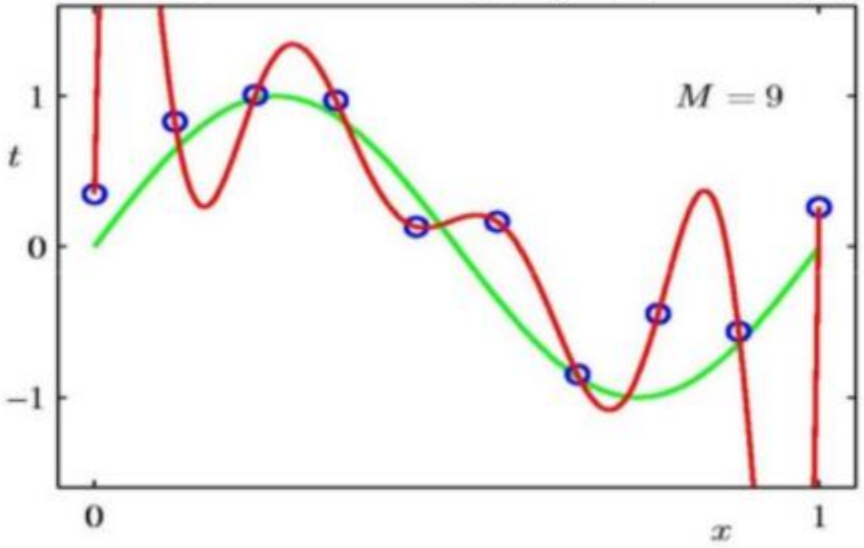


Linear Function $f(x)=\theta_0+\theta_1 x_1$

Cubic Function $f(x)=\theta_0+\theta_1 x_1+\theta_2 x_1^2+\theta_3 x_1^3$



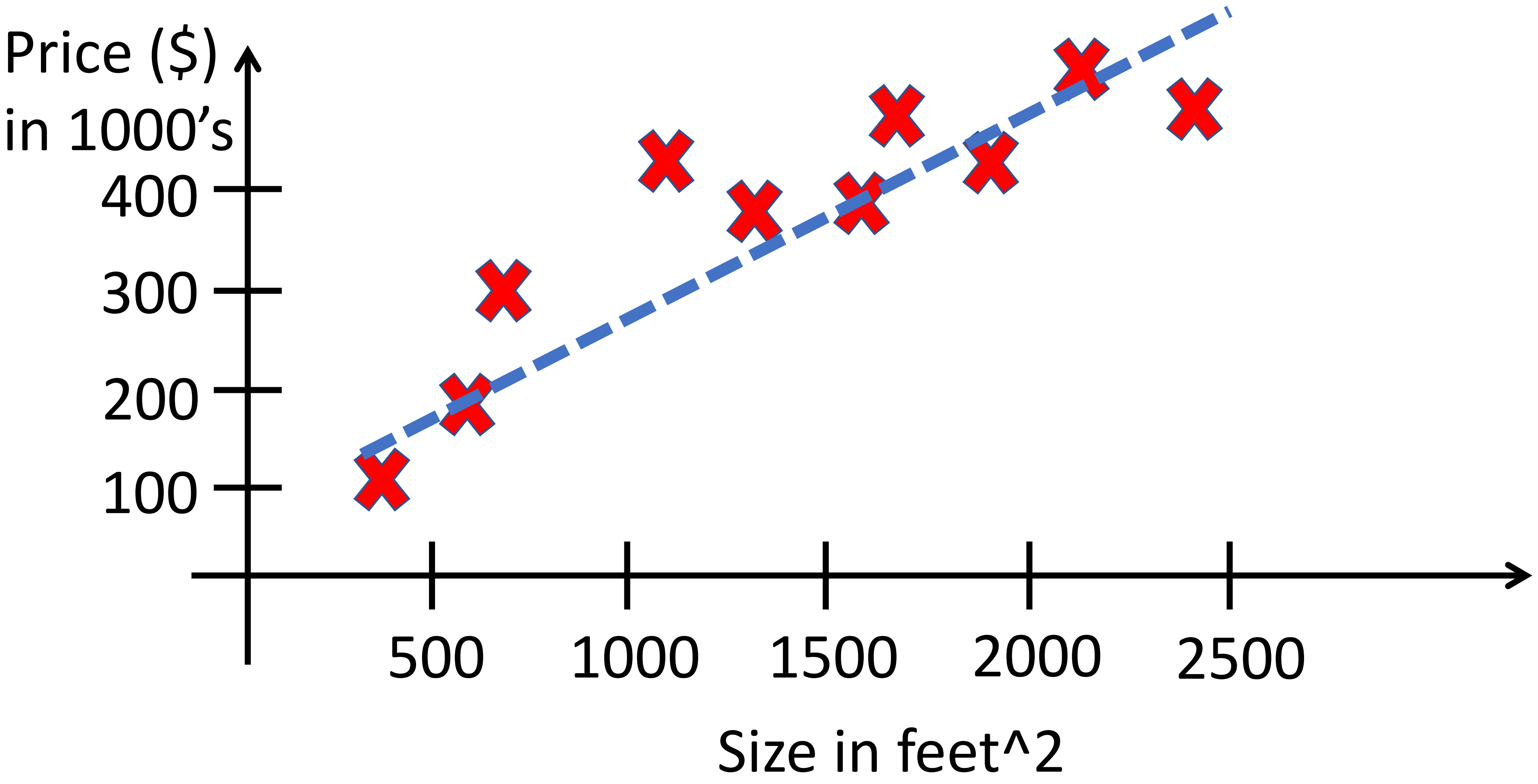
9th Degree Function $f(x)=\theta_0+\theta_1 x_1+ \dots +\theta_9 x_1^9$



Recap: Machine learning algorithms

	Supervised Learning	**Unsupervised Learning**
Discrete	Classification	Clustering
Continuous	Regression	Dimensionality reduction

House pricing prediction



Training set

Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

m = 47

- Notation:
  - m = Number of training examples
  - x = Input variable / features
  - y = Output variable / target variable
  - (x, y) = One training example
  - $(x^{(i)}, y^{(i)})$ = i^{th} training example

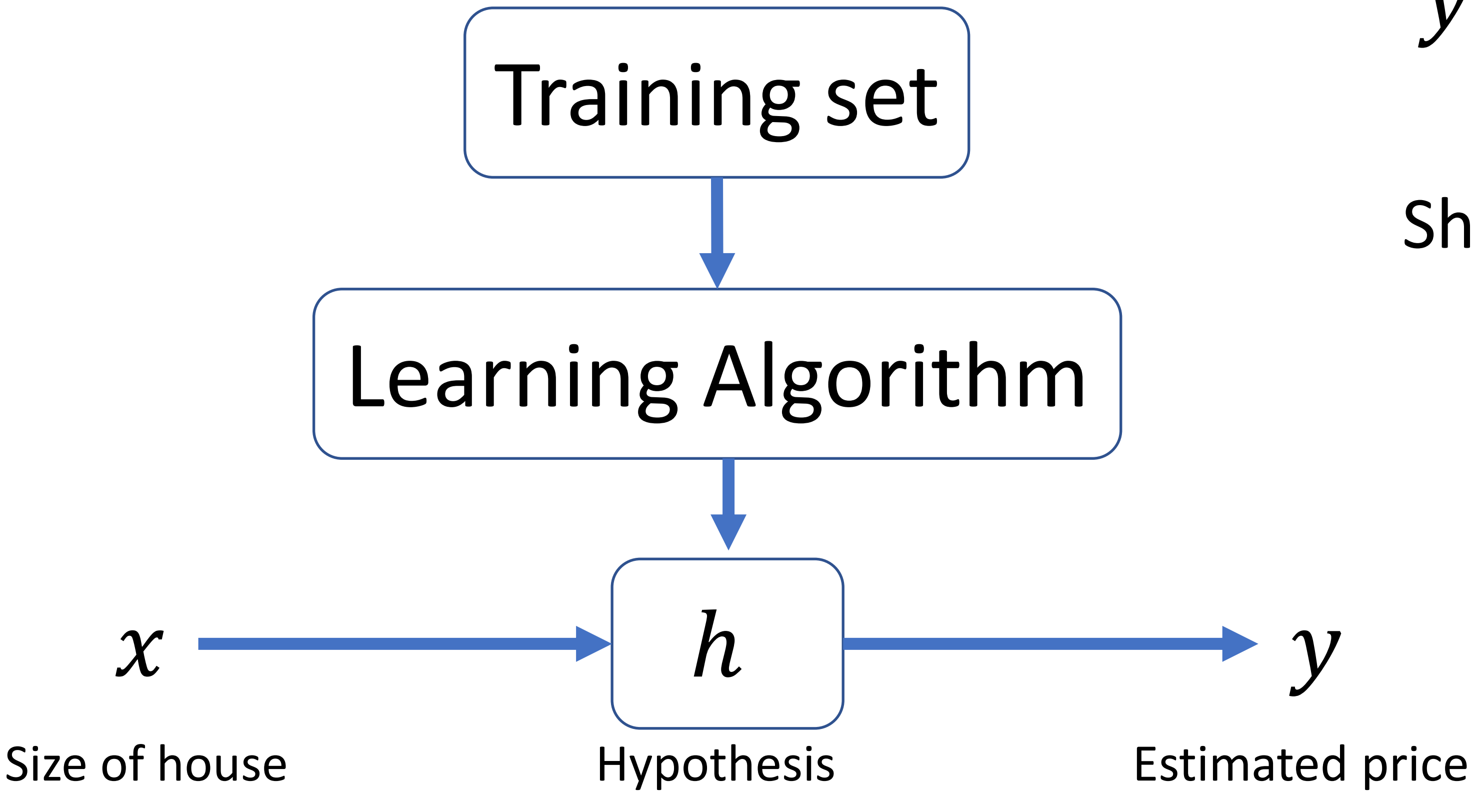
Examples:

$x^{(1)} = 2104$

$x^{(2)} = 1416$

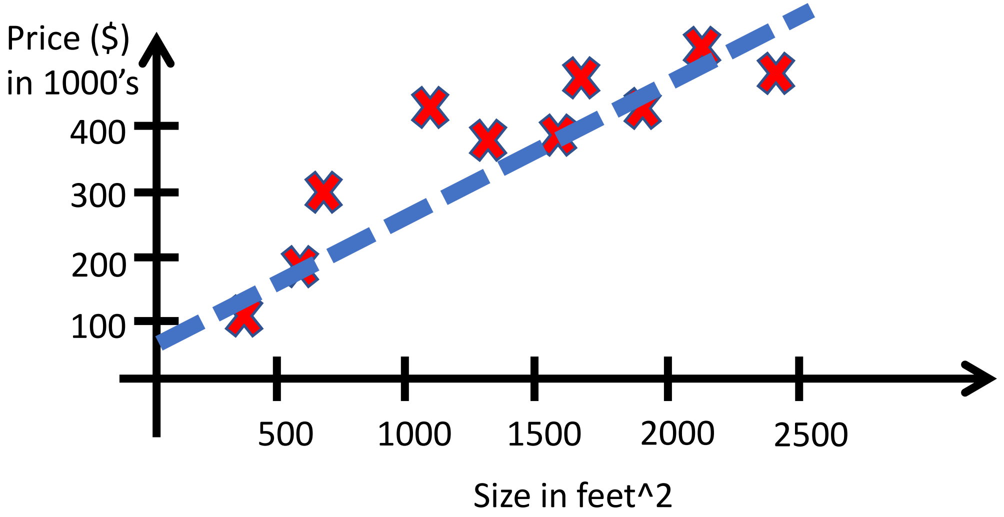
$y^{(1)} = \ 460$

Model representation



$$y = h_\theta(x) = \theta_0 + \theta_1 x$$

Shorthand $h(x)$



Univariate linear regression

Training set

Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

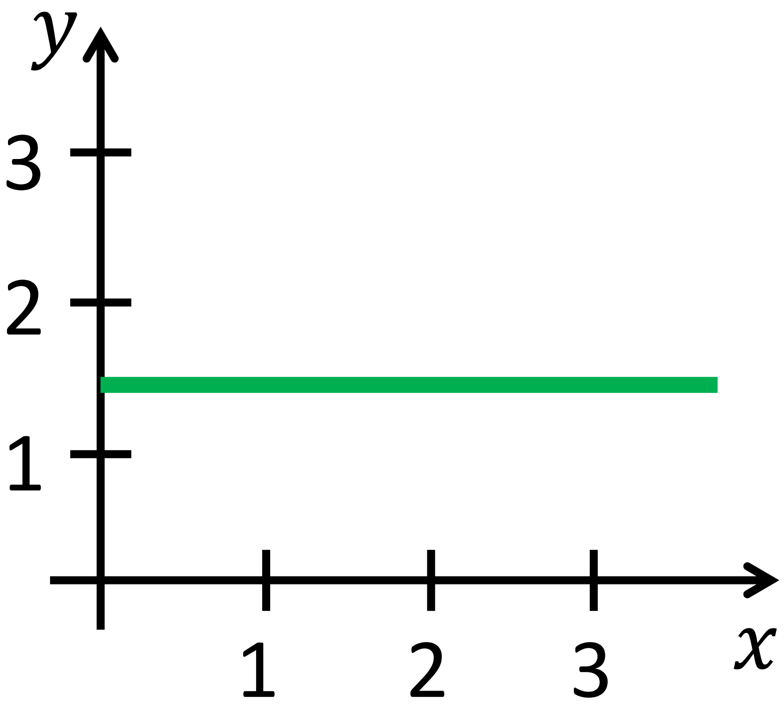
$m = 47$

- Hypothesis $h_\theta(x) = \theta_0 + \theta_1 x$

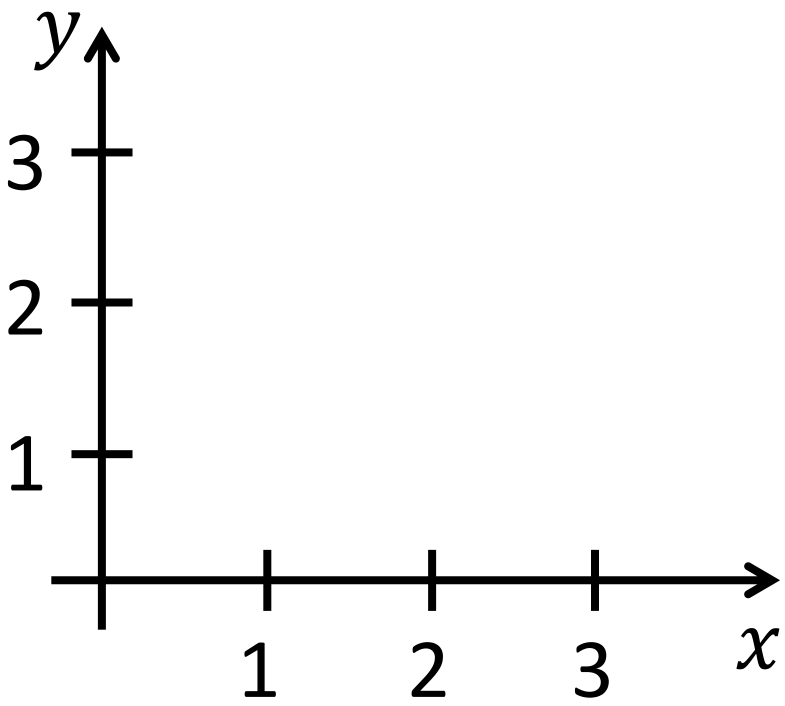
θ_0, θ_1: parameters/weights

How to choose θ_i's?

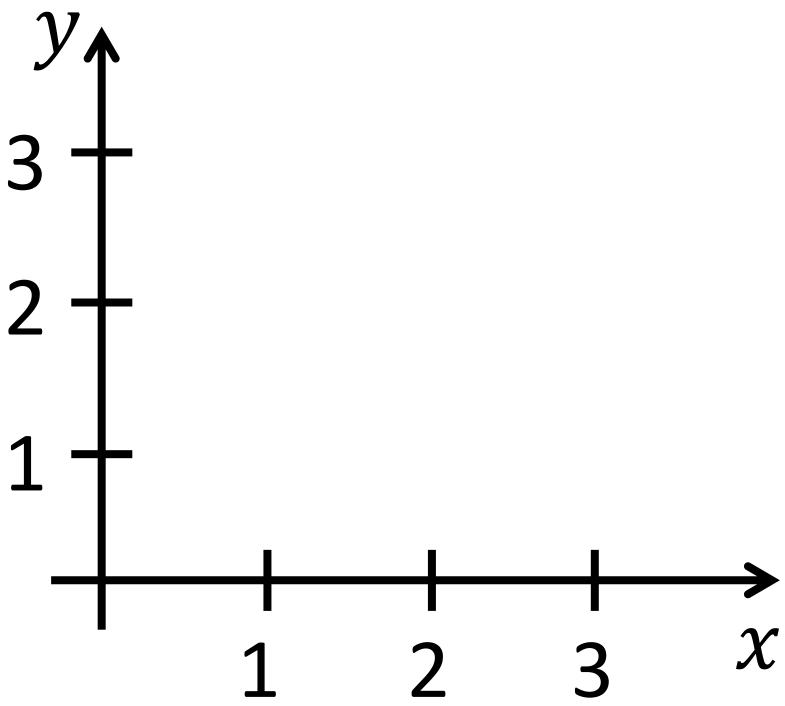
$$h_\theta(x) = \theta_0 + \theta_1 x$$



$\theta_0 = 1.5$
$\theta_1 = 0$

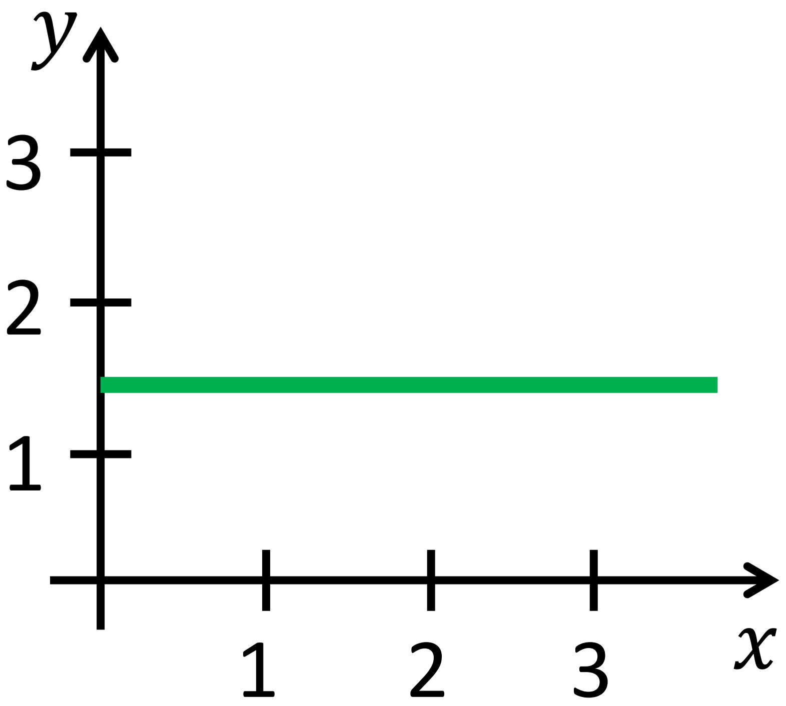


$\theta_0 = 0$
$\theta_1 = 0.5$

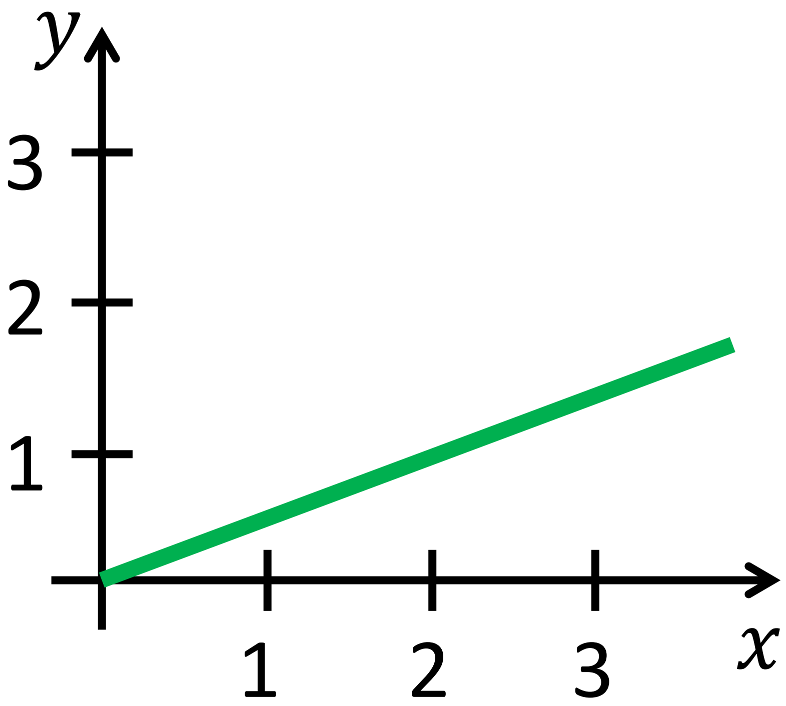


$\theta_0 = 1$
$\theta_1 = 0.5$

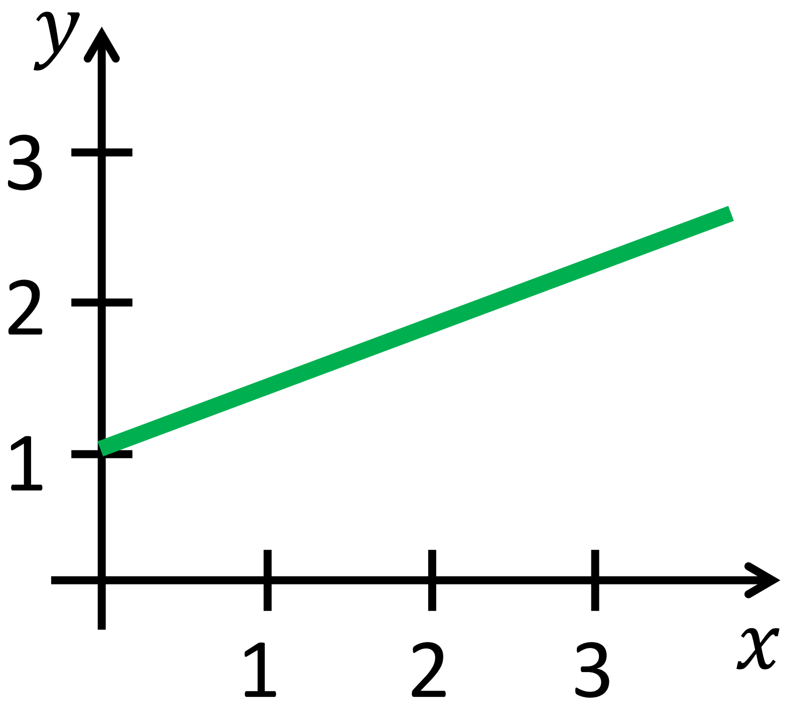
$$h_\theta(x) = \theta_0 + \theta_1 x$$



$\theta_0 = 1.5$
$\theta_1 = 0$



$\theta_0 = 0$
$\theta_1 = 0.5$



$\theta_0 = 1$
$\theta_1 = 0.5$

Cost function

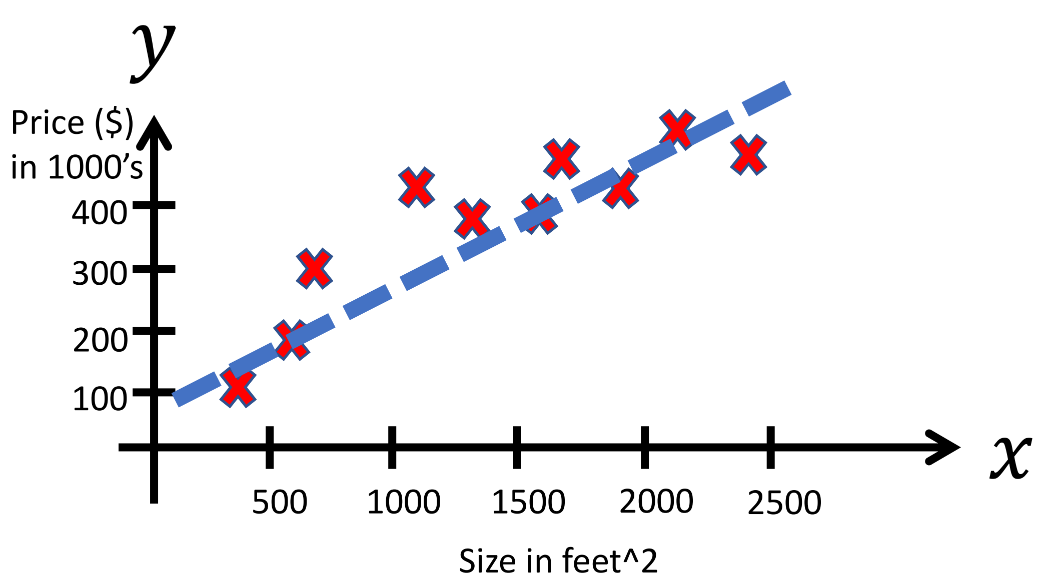
- **Idea:**
  Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training example (x, y)

$$\underset{\theta_0, \theta_1}{\text{minimize}} \ \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$$h_\theta\left(x^{(i)}\right) = \theta_0 + \theta_1 x^{(i)}$$

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

$$\underset{\theta_0, \theta_1}{\text{minimize}} \ \boxed{J(\theta_0, \theta_1)}$$ **Cost function**



Simplified

- **Hypothesis:**

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

- **Parameters:**

$$\theta_0, \theta_1$$

- **Cost function:**

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

- **Goal:**

$$\underset{\theta_0, \theta_1}{\text{minimize}} \; J(\theta_0, \theta_1)$$

- **Hypothesis:**

$$h_\theta(x) = \theta_1 x \qquad \theta_0 = 0$$

- **Parameters:**

$$\theta_1$$

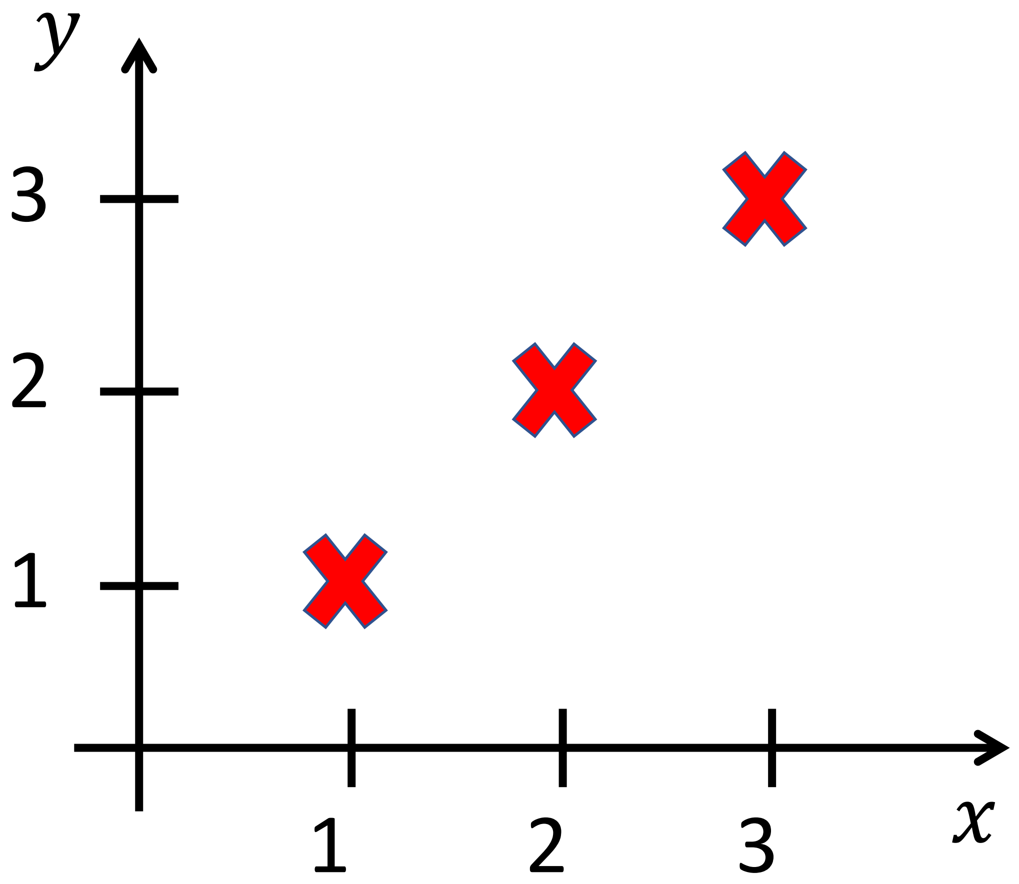
- **Cost function:**

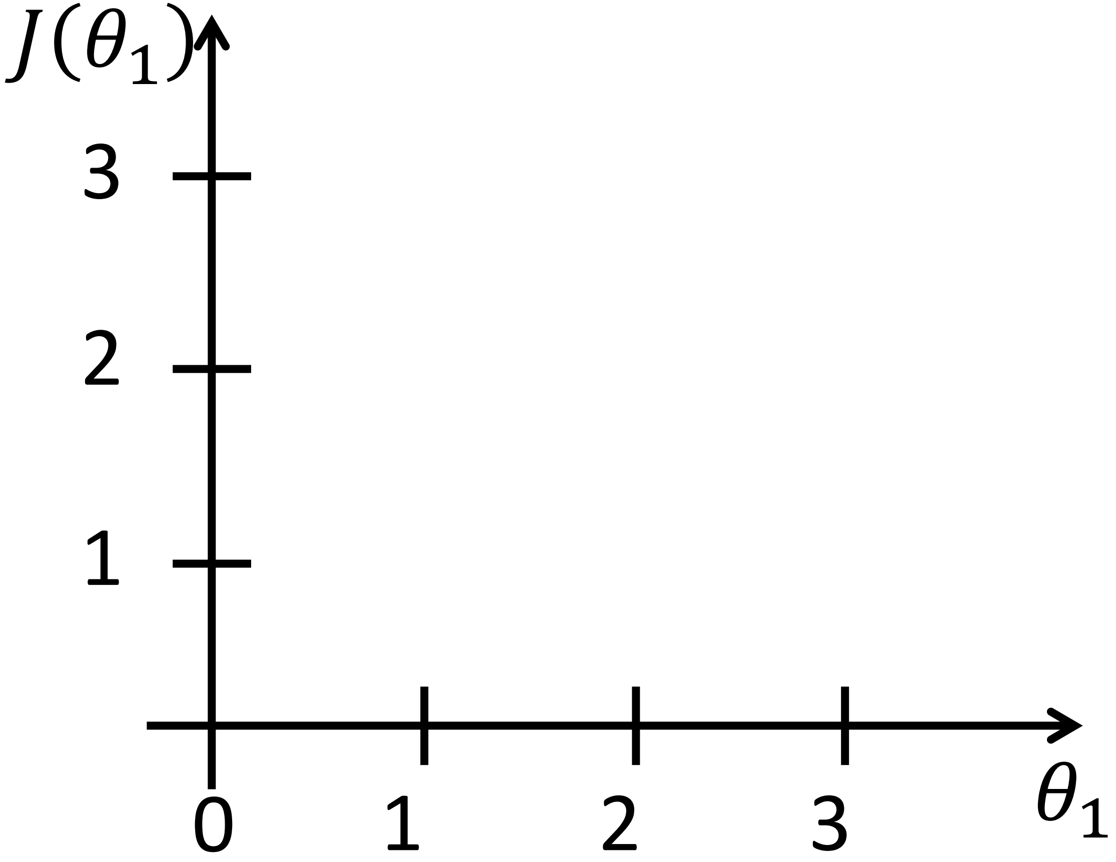
$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

- **Goal:**

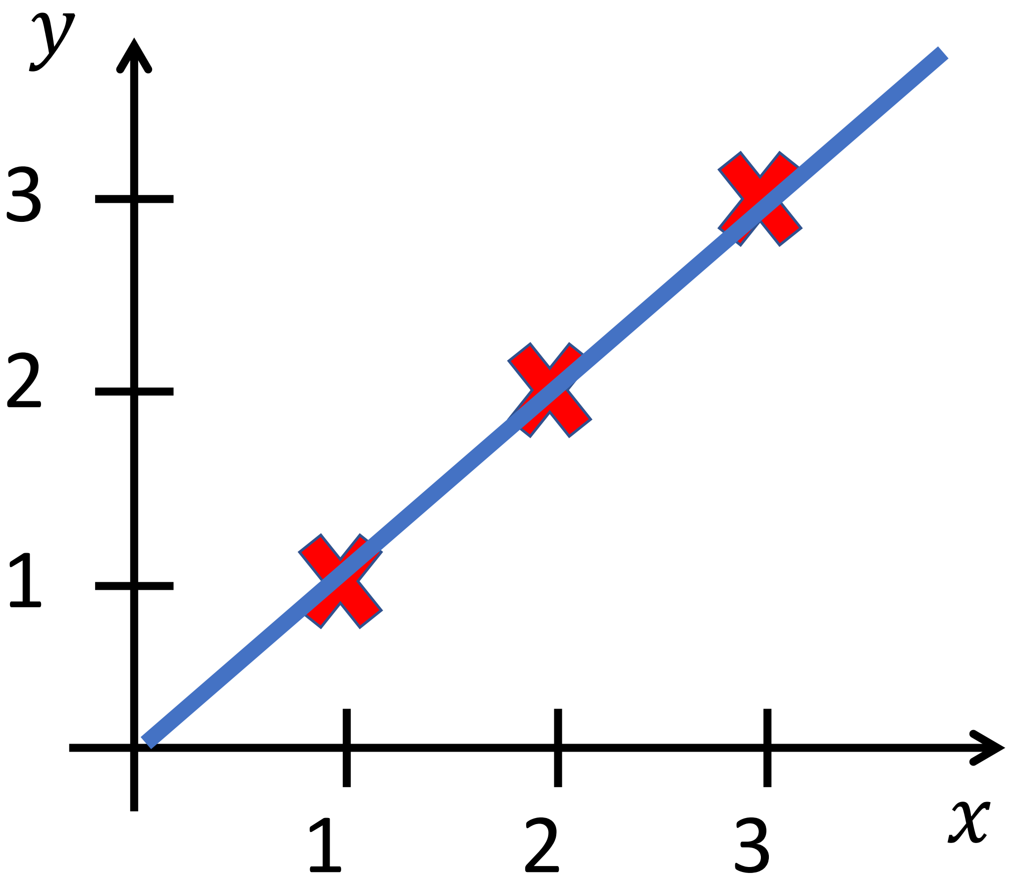
$$\underset{\theta_0, \theta_1}{\text{minimize}} \; J(\theta_1)$$

$h_\theta(x)$, function of x $J(\theta_1)$, function of θ_1

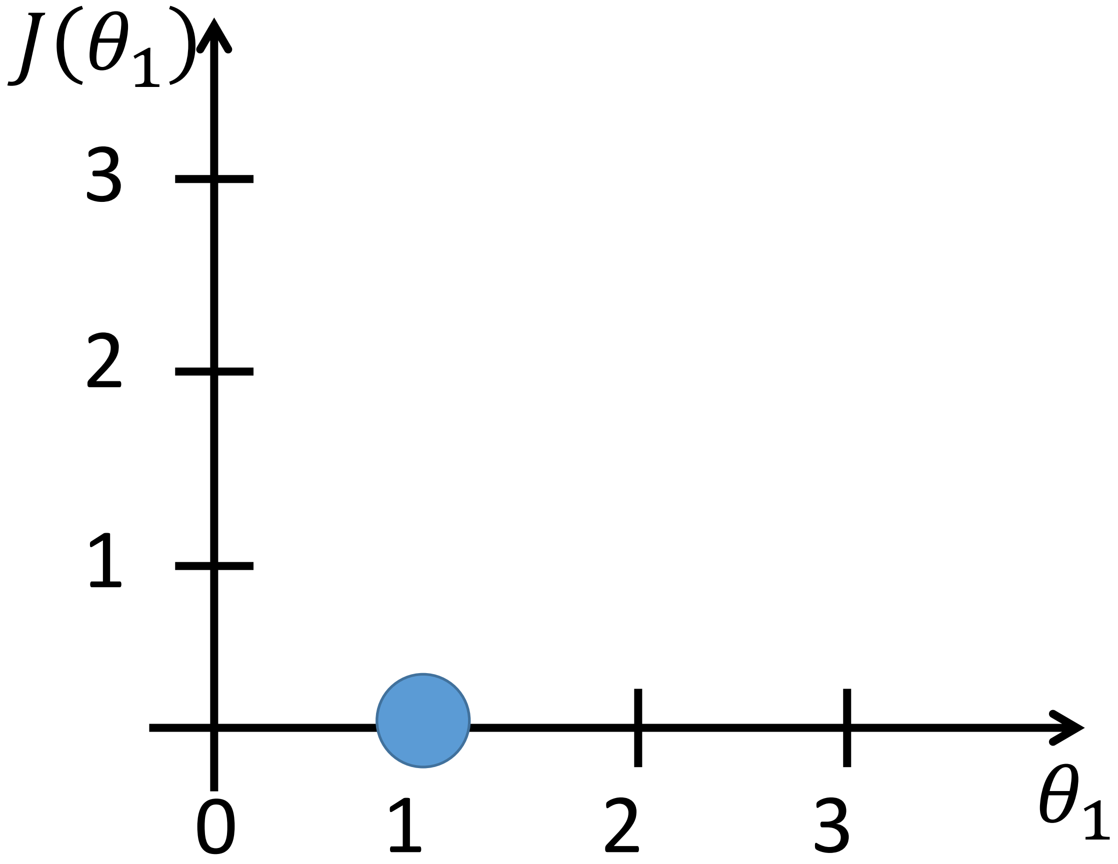




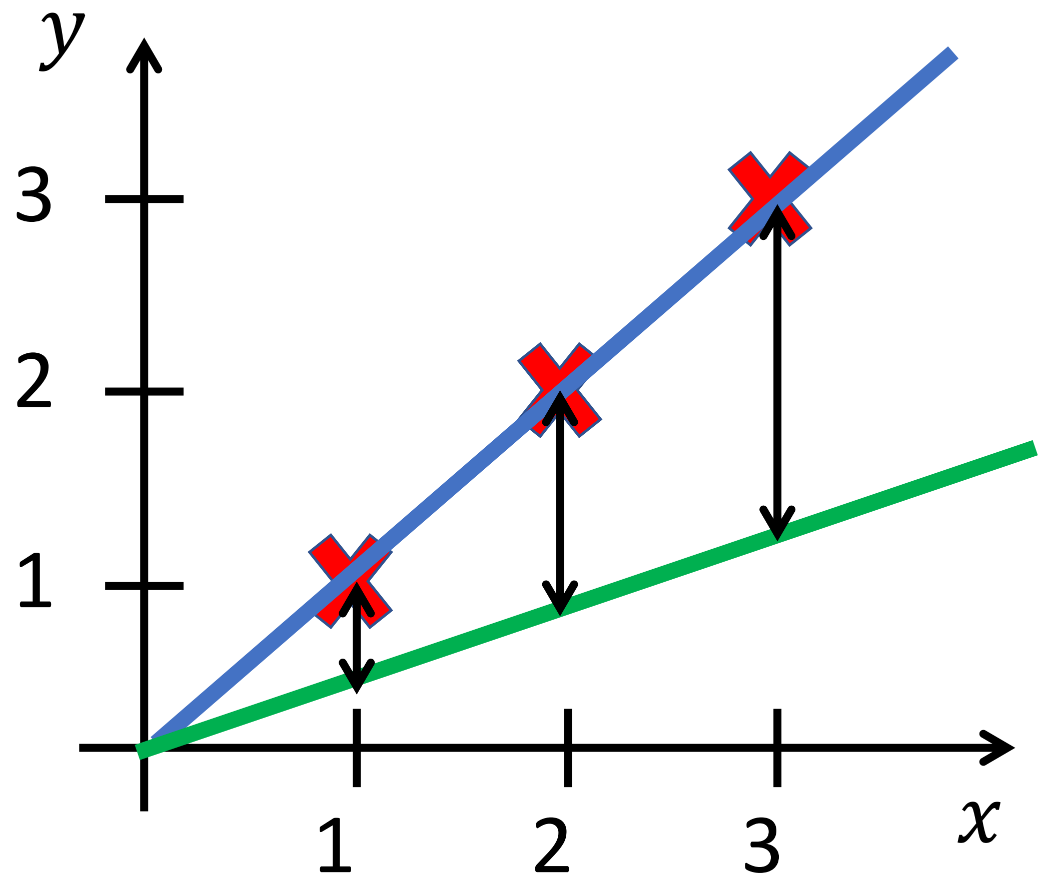
$h_\theta(x)$, function of x



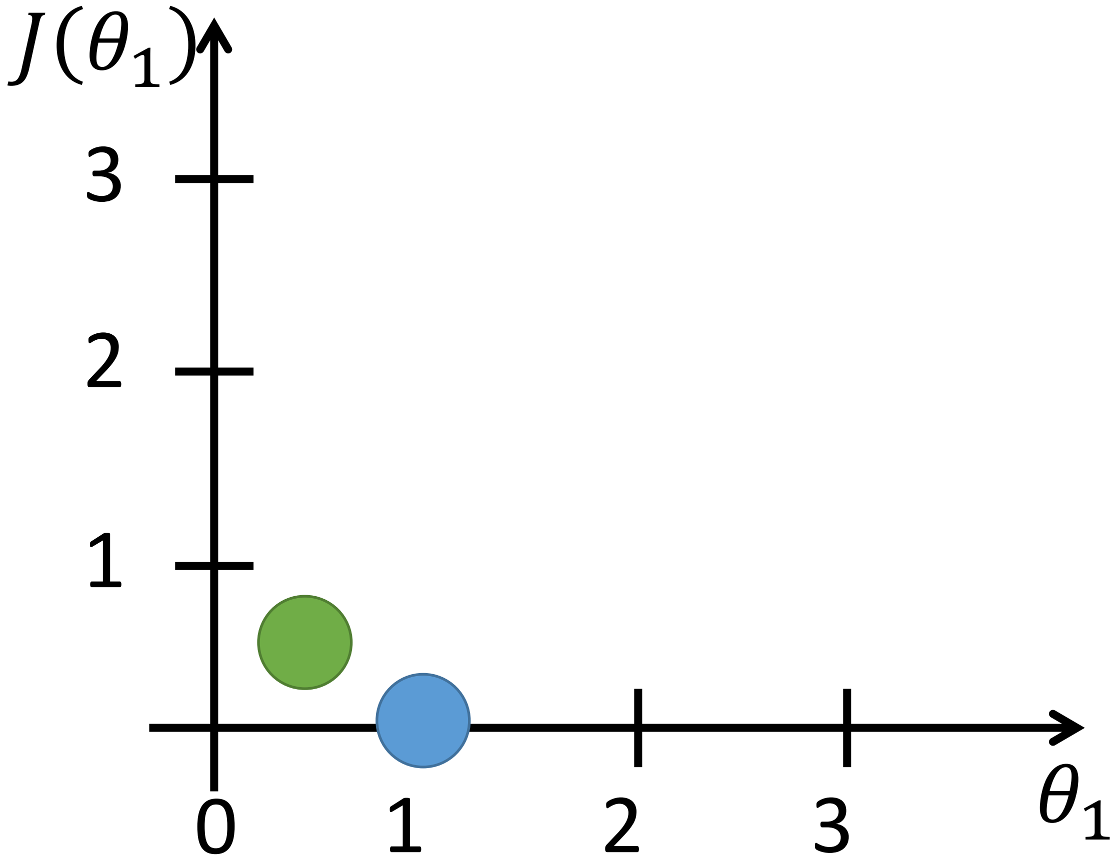
$J(\theta_1)$, function of θ_1



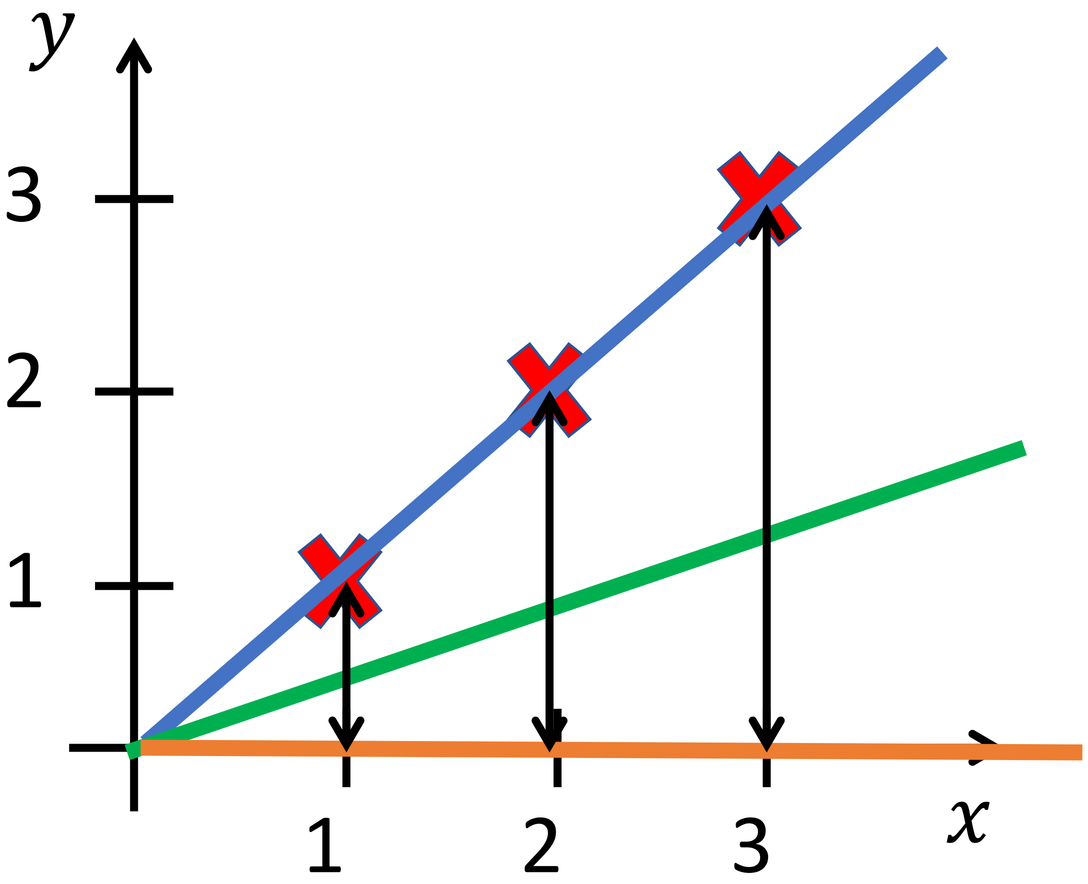
$h_\theta(x)$, function of x



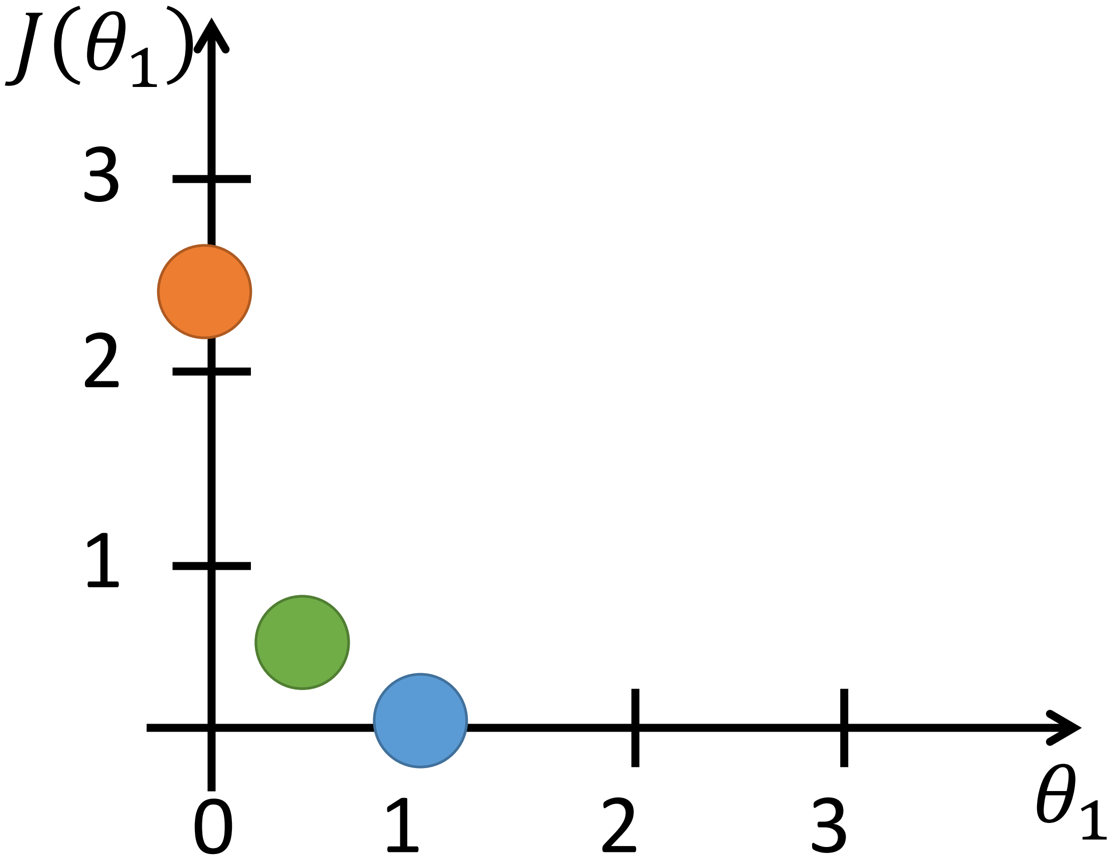
$J(\theta_1)$, function of θ_1



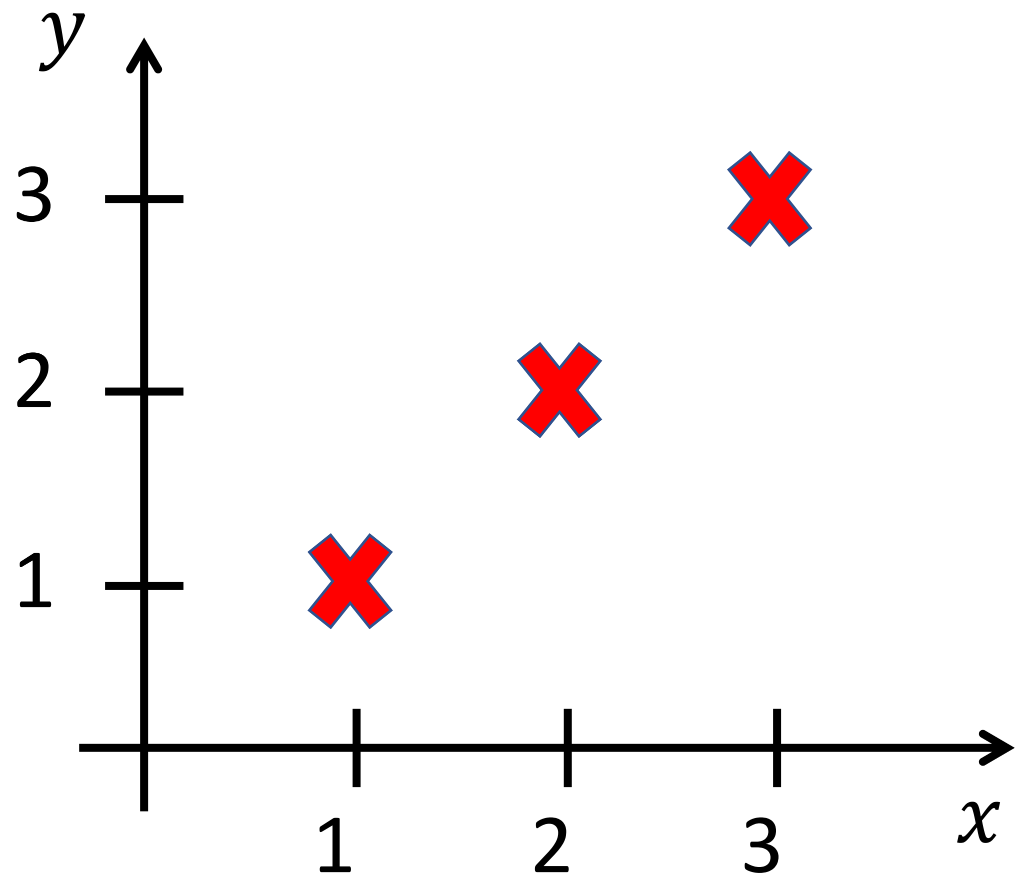
$h_\theta(x)$, function of x



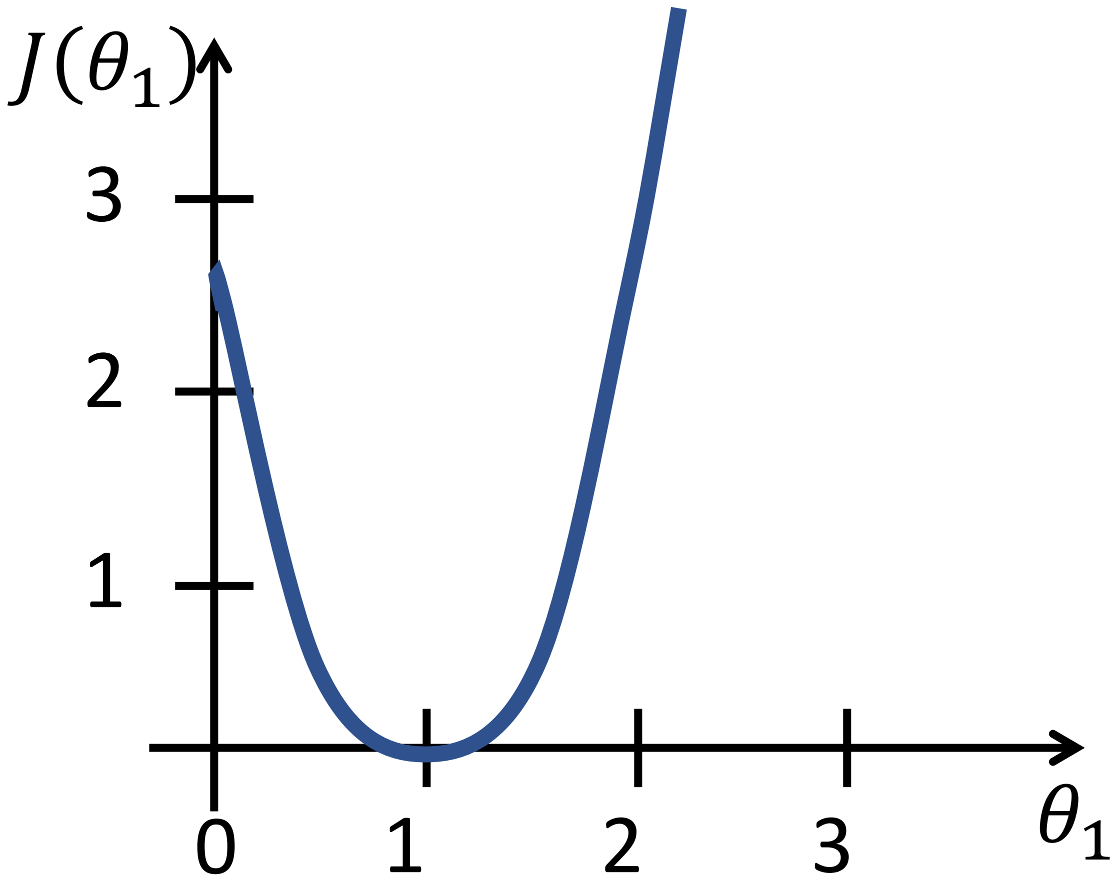
$J(\theta_1)$, function of θ_1



$h_\theta(x)$, function of x



$J(\theta_1)$, function of θ_1



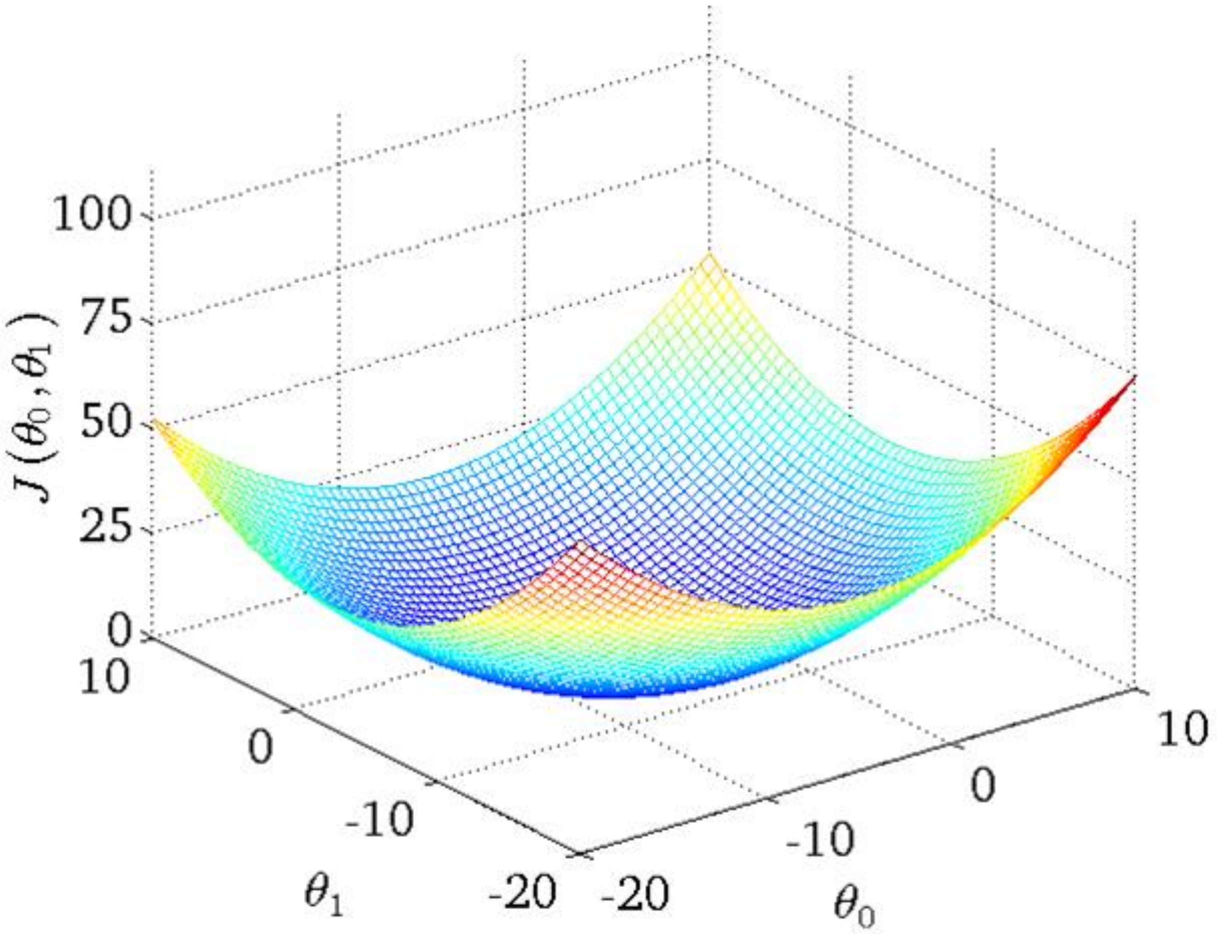
- **Hypothesis:** $h_\theta(x) = \theta_0 + \theta_1 x$

- **Parameters:** θ_0, θ_1

- **Cost function:** $J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$

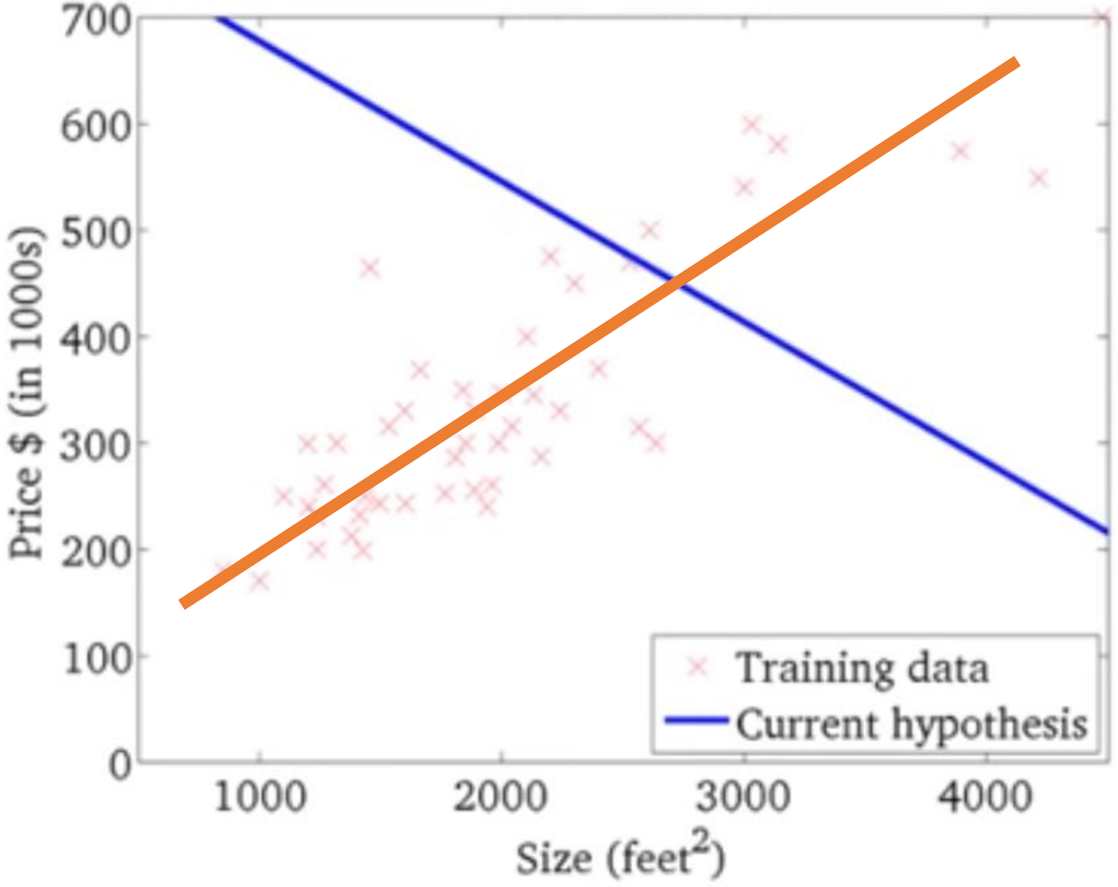
- **Goal**: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Cost function



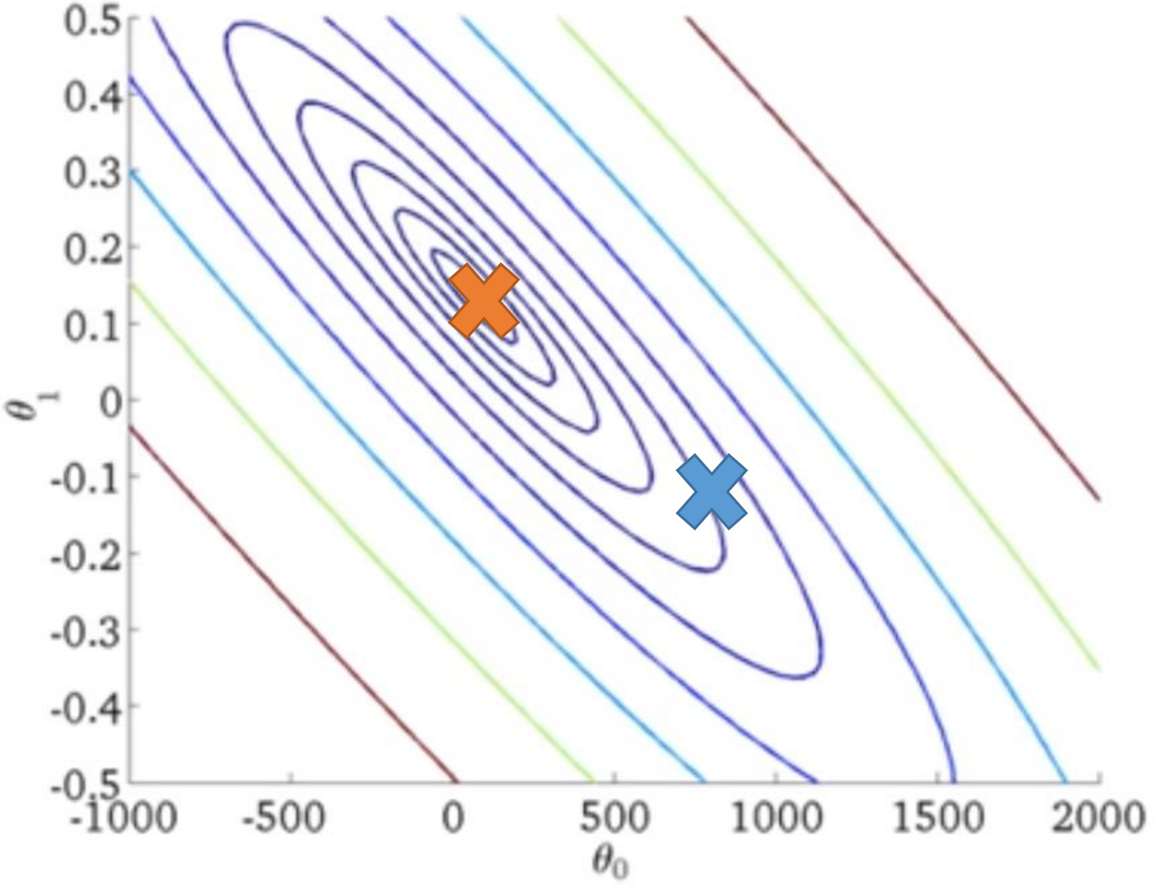
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

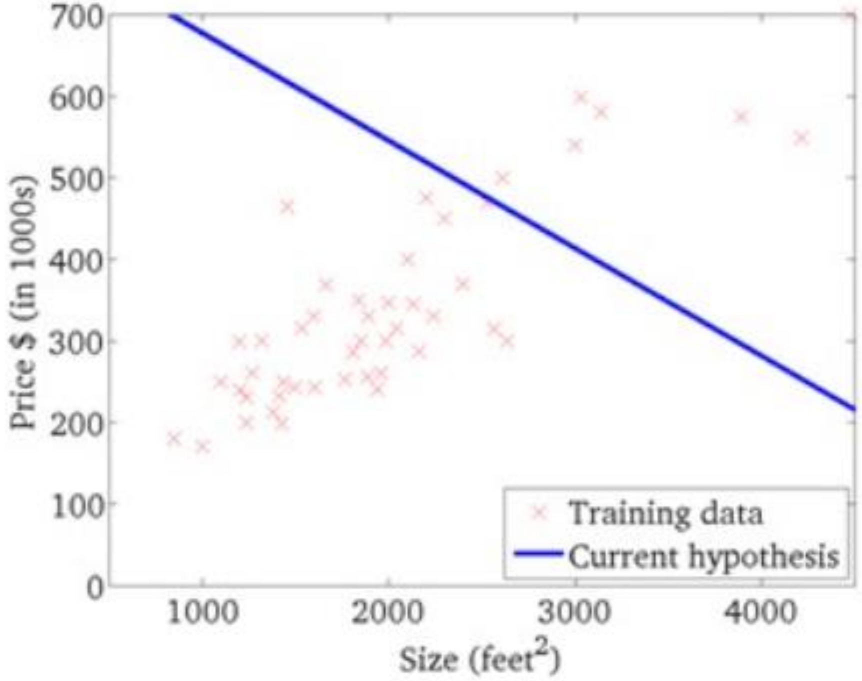
(function of the parameters θ_0, θ_1)



How do we find good θ_0, θ_1 that minimize $J(\theta_0, \theta_1)$?

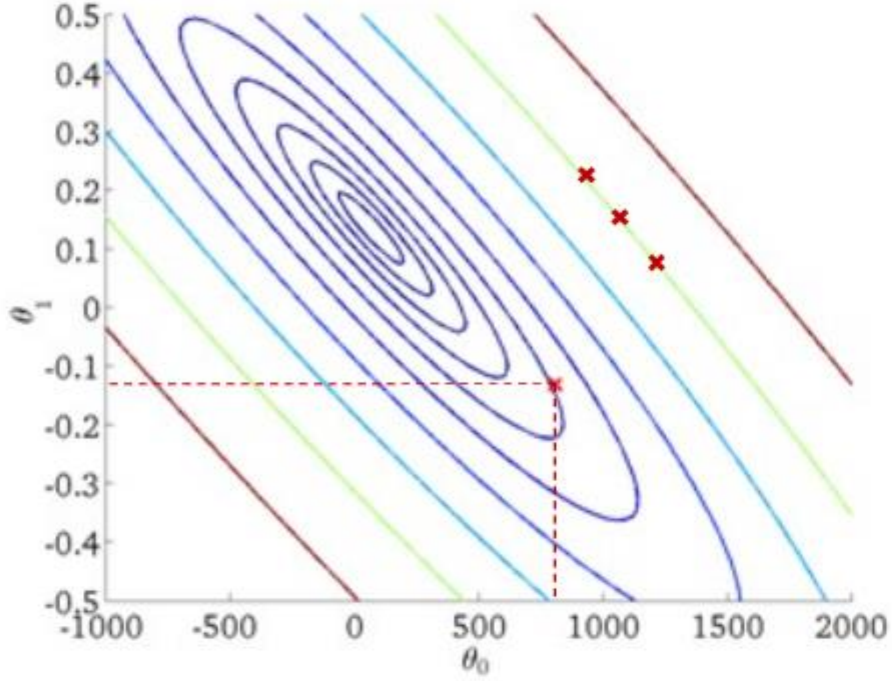
$$h_\theta(x)$$

(for fixed θ_0, θ_1, this is a function of x)



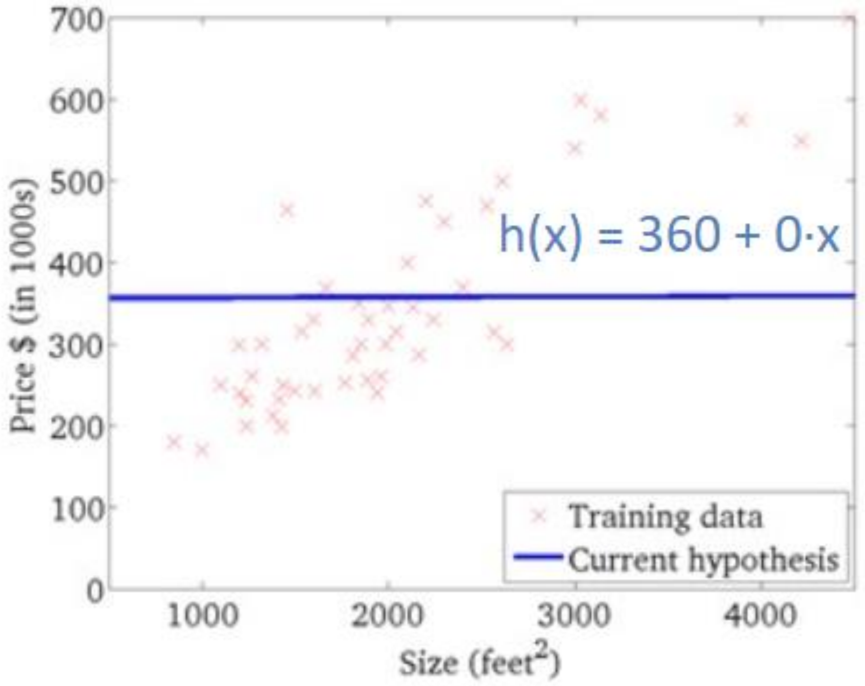
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



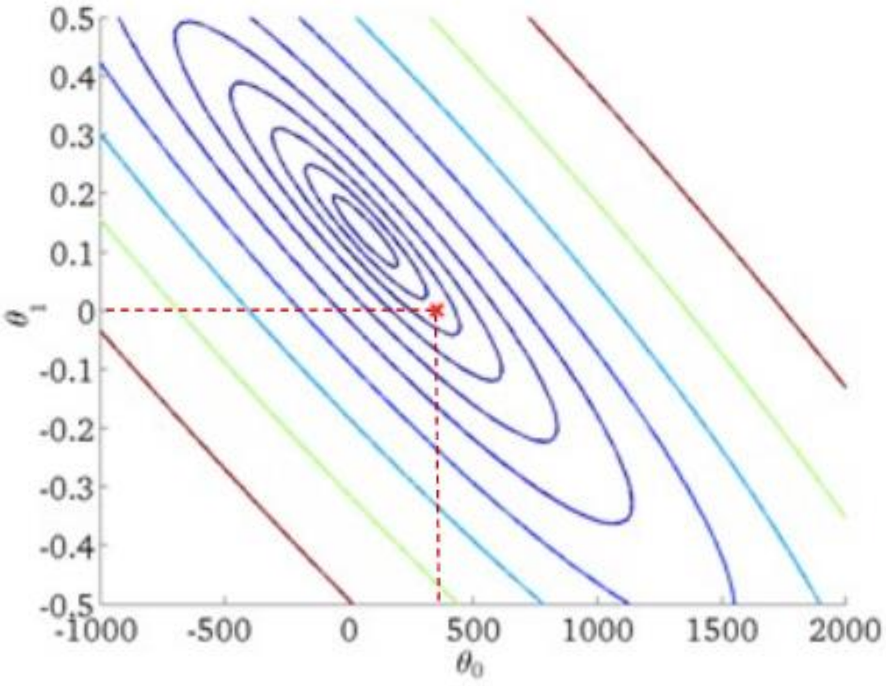
$$h_\theta(x)$$

(for fixed θ_0, θ_1, this is a function of x)



$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)

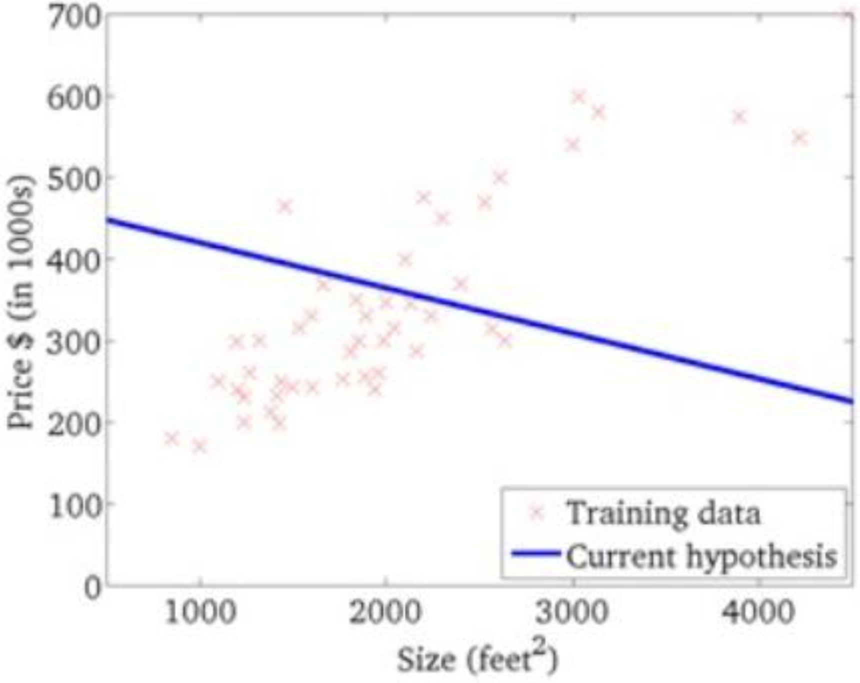


$\Theta_0 = 360$

$\Theta_1 = 0$

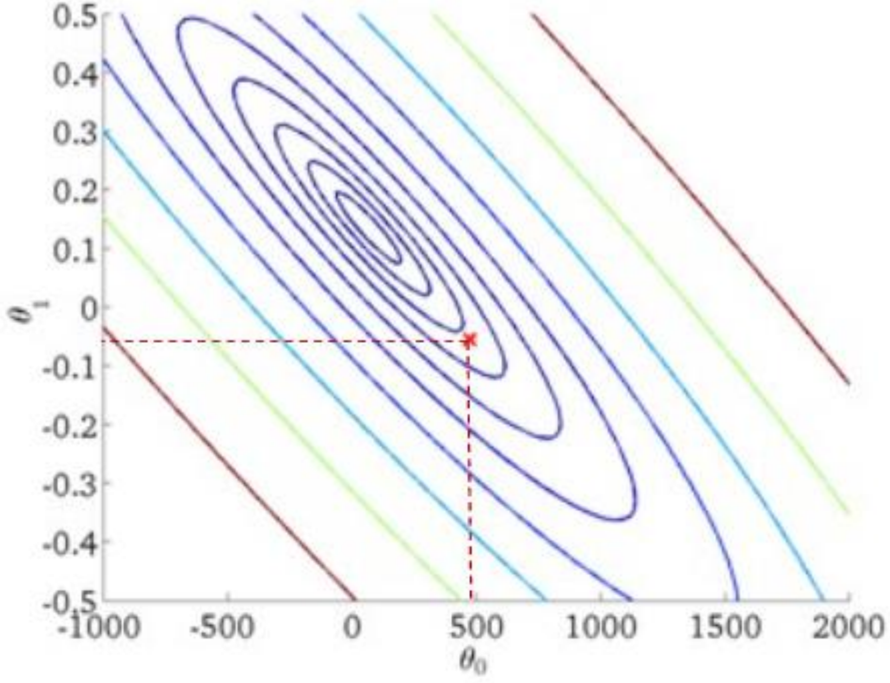
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



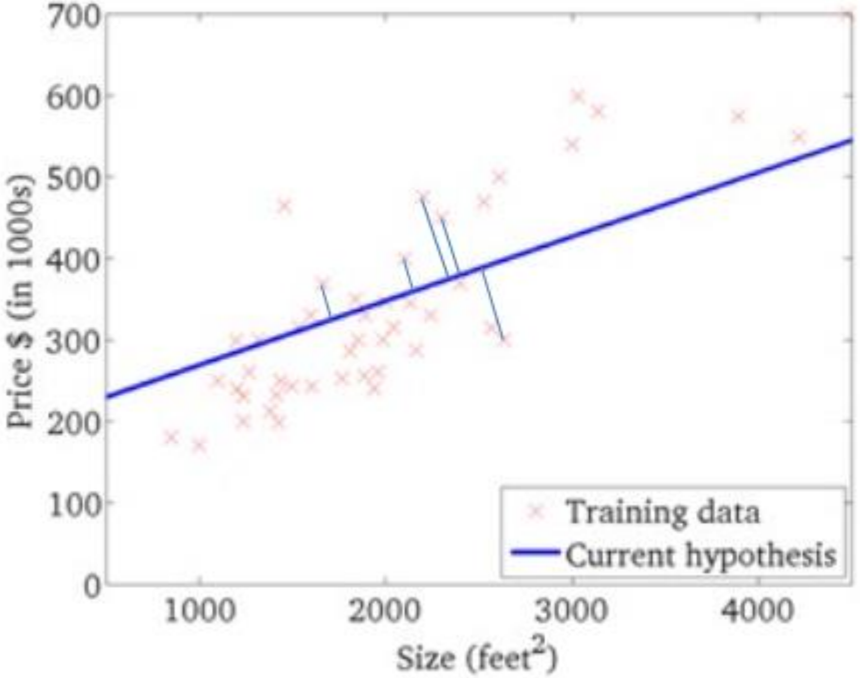
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



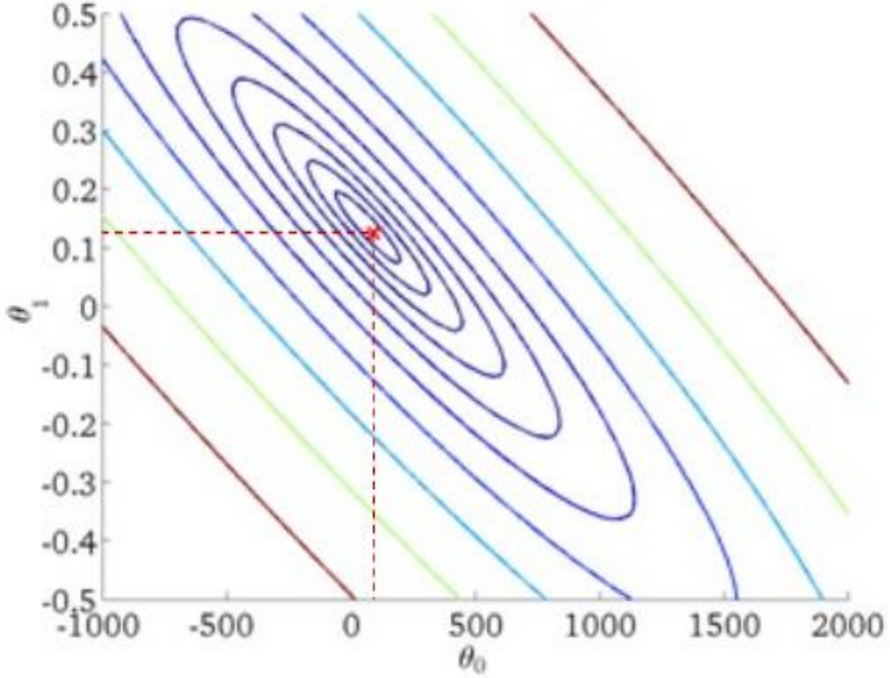
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



Linear Regression

- Model representation
- Cost function
- **Gradient descent**
- Features and polynomial regression
- Normal equation

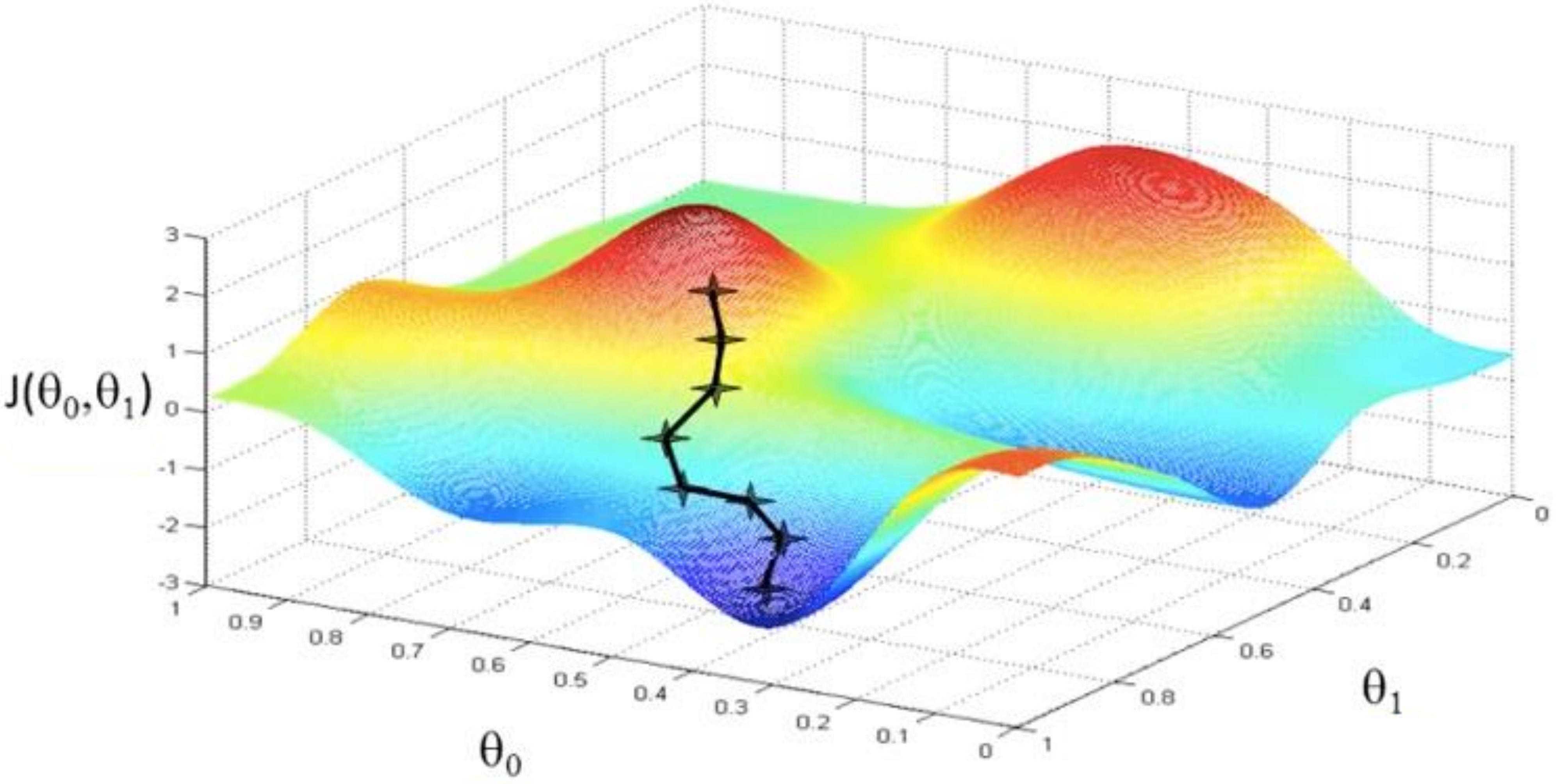
Gradient descent

Have some function $J(\theta_0, \theta_1)$

Want $\underset{\theta_0, \theta_1}{\operatorname{argmin}} \; J(\theta_0, \theta_1)$

Outline:

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at minimum



$J(\theta_0,\theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1

Gradient descent

Repeat until convergence{

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad \text{(for } j = 0 \text{ and } j = 1\text{)}$$

}

α: Learning rate (step size)

$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$: derivative (rate of change)

Gradient descent

Correct: simultaneous update

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_0 := \text{temp0}$$

$$\theta_1 := \text{temp1}$$

Incorrect:

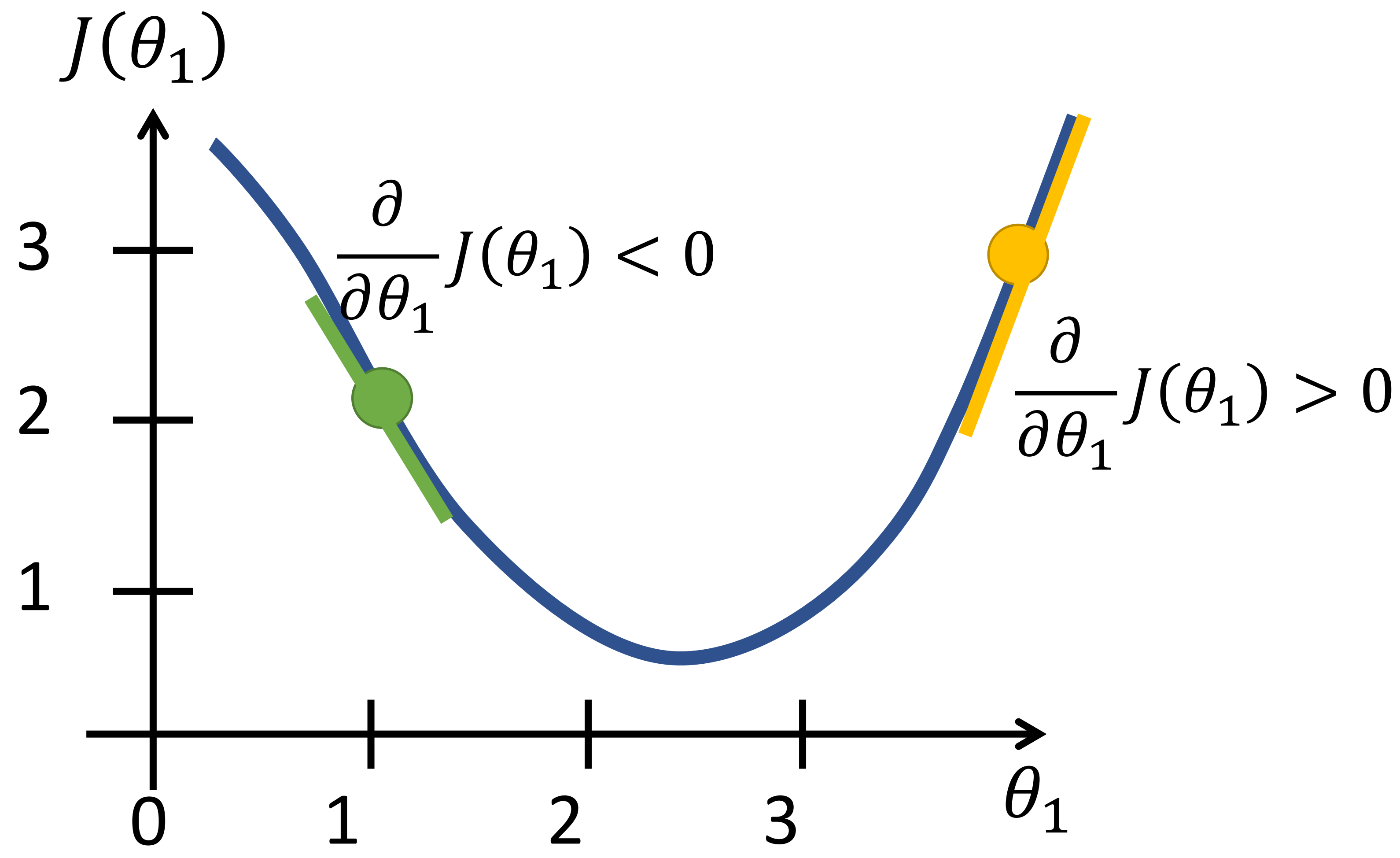
$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$\theta_0 := \text{temp0}$$

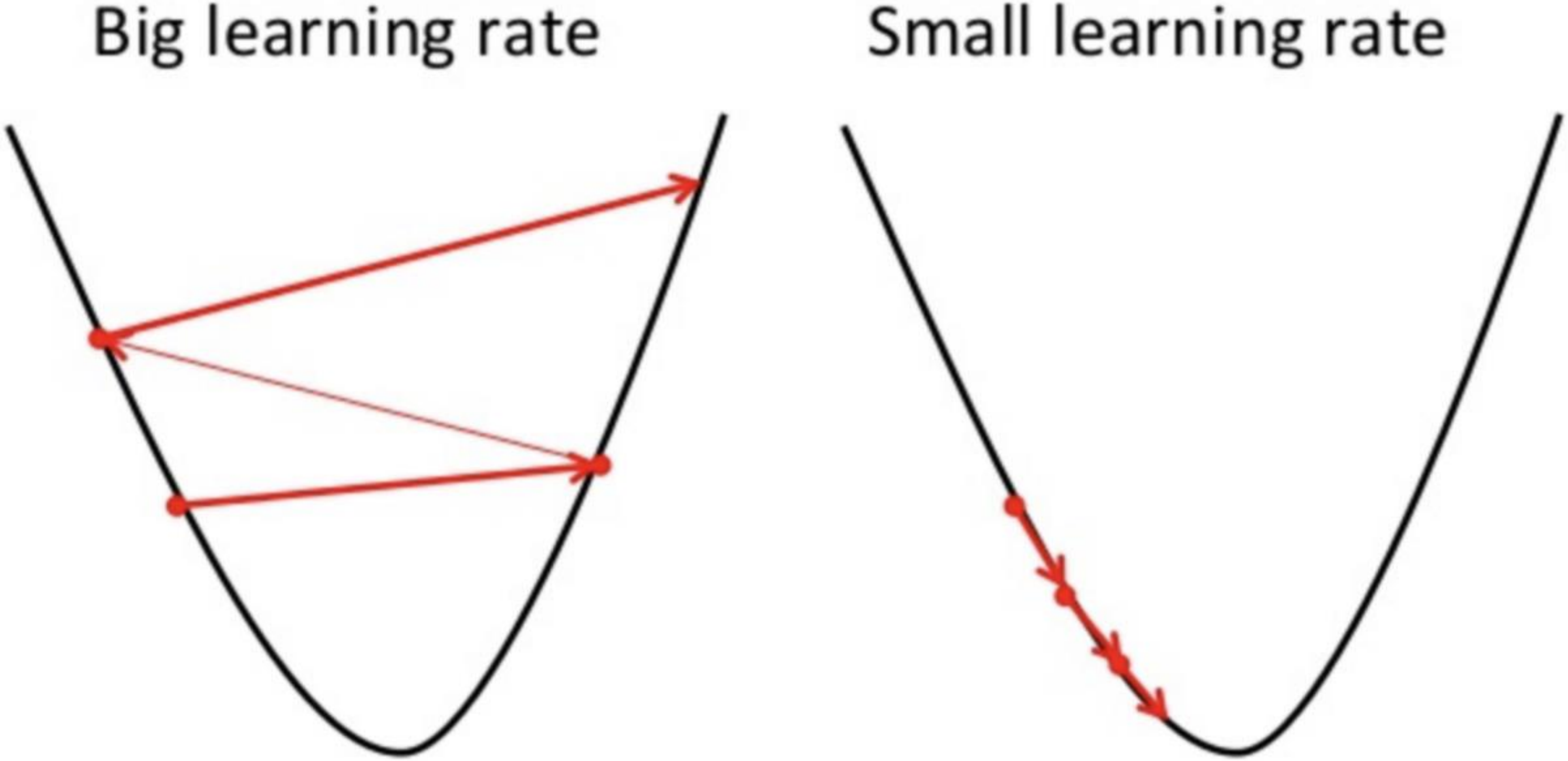
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_1 := \text{temp1}$$

$$\theta_1 \coloneqq \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$



Learning rate



Gradient descent for linear regression

Repeat until convergence{

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ (for $j = 0$ and $j = 1$)

}

- Linear regression model

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

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

Computing partial derivative

- $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$

  $= \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right)^2$

- $j = 0$: $\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)$
- $j = 1$: $\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x^{(i)}$

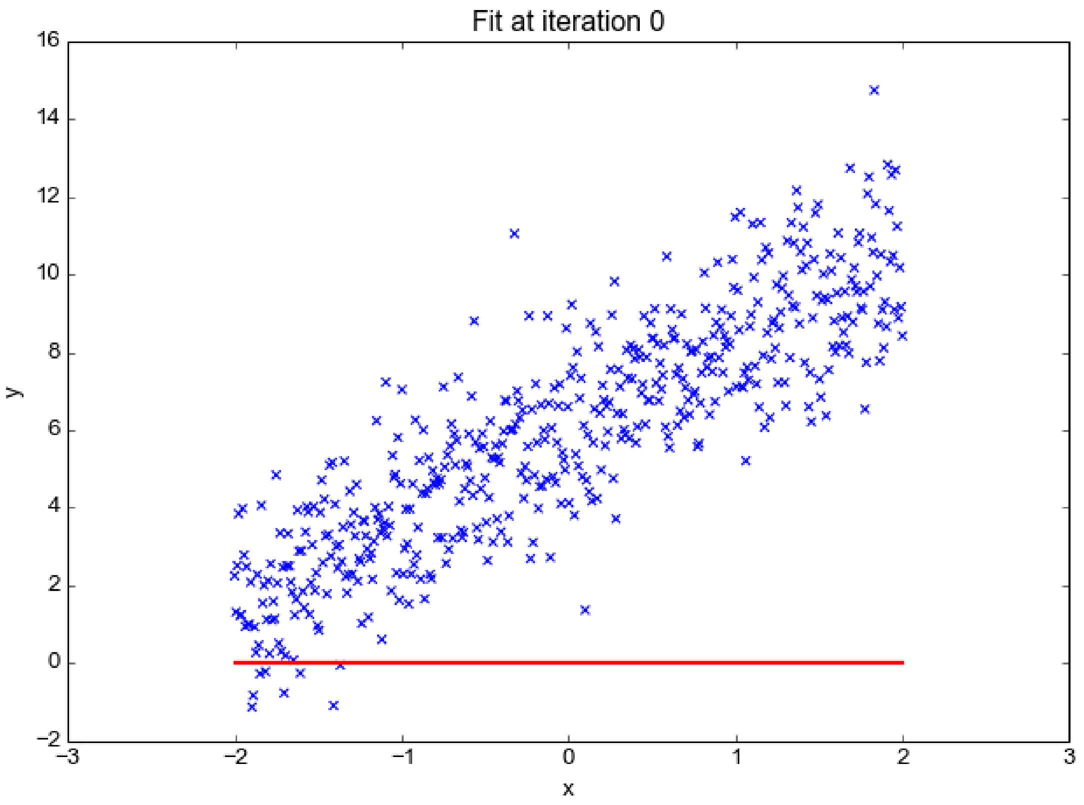
Gradient descent for linear regression

Repeat until convergence{

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)$$

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

}



Update θ_0 and θ_1 simultaneously

Batch gradient descent

- "Batch": Each step of gradient descent uses all the training examples

Repeat until convergence{

m: Number of training examples

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{\boxed{m}} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)$$

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

}

Training dataset

Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

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

Multiple features (input variables)

Size in feet^2 (x_1)	Number of bedrooms (x_2)	Number of floors (x_3)	Age of home (years) (x_4)	Price (\$) in 1000's (y)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…				…

Notation:

n = Number of features

$x^{(i)}$= Input features of i^{th} training example

$x_j^{(i)}$= Value of feature j in i^{th} training example

$x_3^{(2)} = ?$

$x_3^{(4)} = ?$

Hypothesis

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

Now: $$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

- For convenience of notation, define $x_0 = 1$ ($x_0^{(i)} = 1$ for all examples)

- $\boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in R^{n+1}$ $\quad\quad\quad$ $\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in R^{n+1}$

- $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$
  $\quad\quad\quad\;\; = \boldsymbol{\theta}^\top \boldsymbol{x}$

Gradient descent

- Previously ($n = 1$)

Repeat until convergence{

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)$$

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

}

- New algorithm ($n \geq 1$)

Repeat until convergence{

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) x_j^{(i)}$$

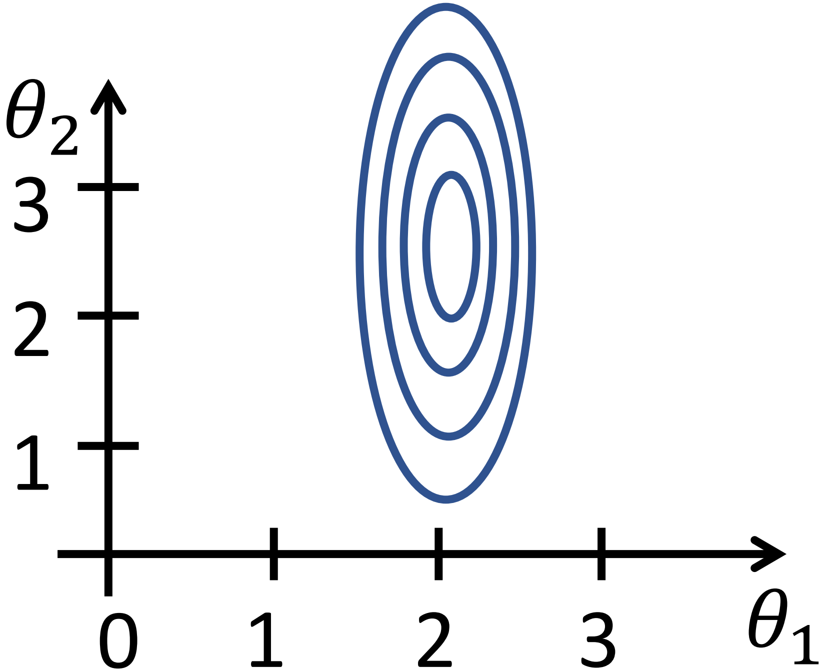
}

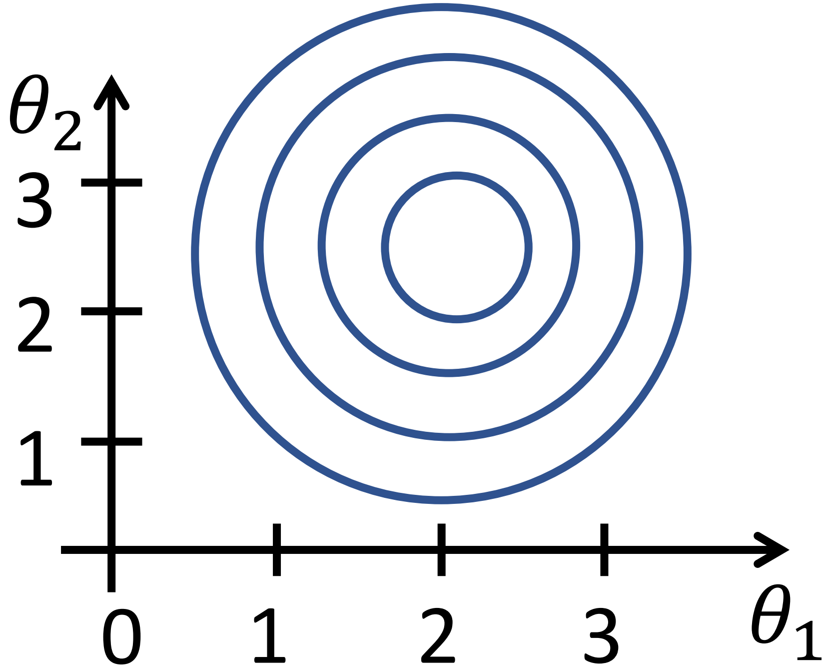
Simultaneously update
$\theta_j, \text{for } j = 0, 1, \cdots, n$

Gradient descent in practice: Feature scaling

- Idea: Make sure features are on a similar scale (e.g,. $-1 \le x_i \le 1$)
- E.g. x_1 = size (0-2000 feat^2)

  x_2 = number of bedrooms (1-5)





Question

Midterm Exam	(midterm exam)^2	Final Exam
89	7921	96
72	5184	74
94	8836	87
69	4761	78

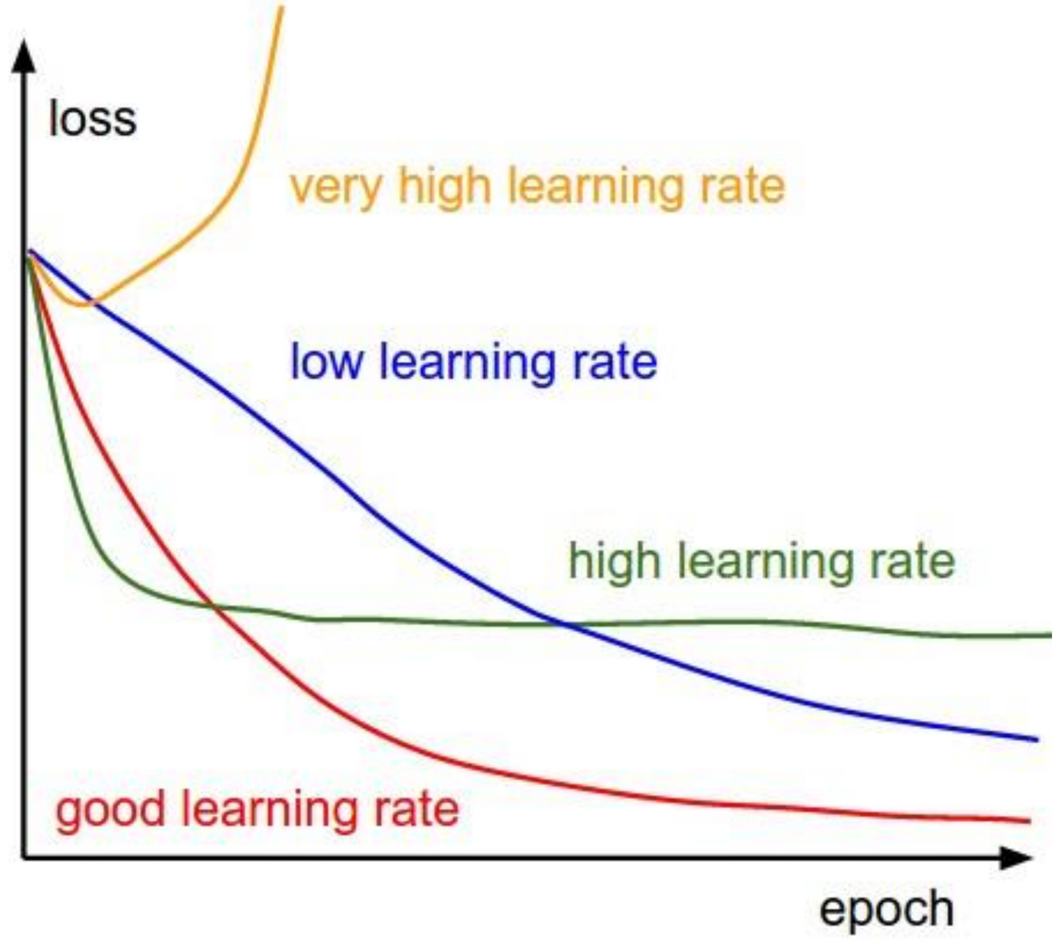
What is the normalized feature x2(4)?

Gradient descent in practice: Learning rate

- Automatic convergence test
- α too small: slow convergence
- α too large: may not converge

- To choose α, try

0.001, … 0.01, …, 0.1, … , 1

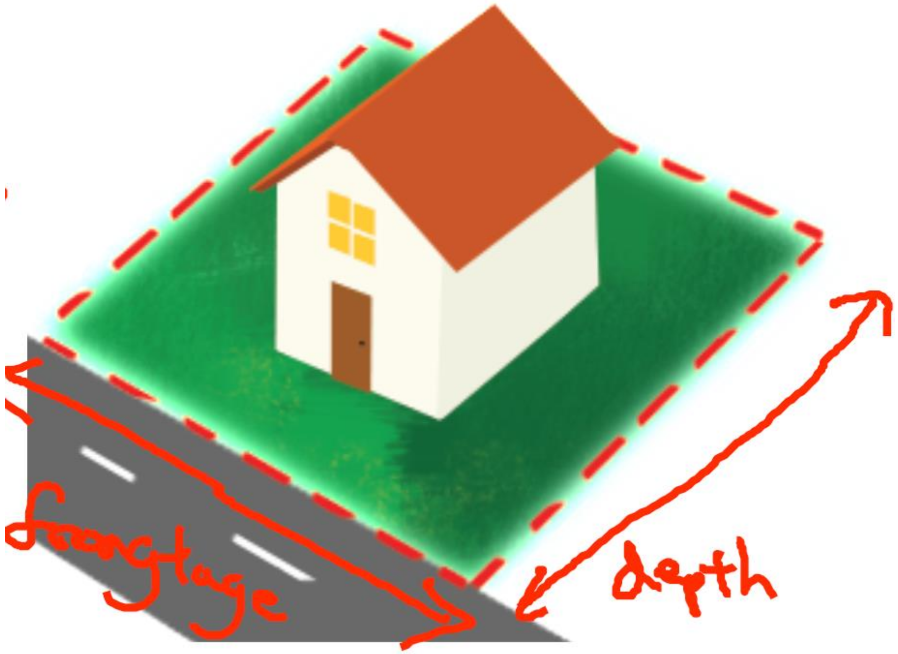


House prices prediction

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

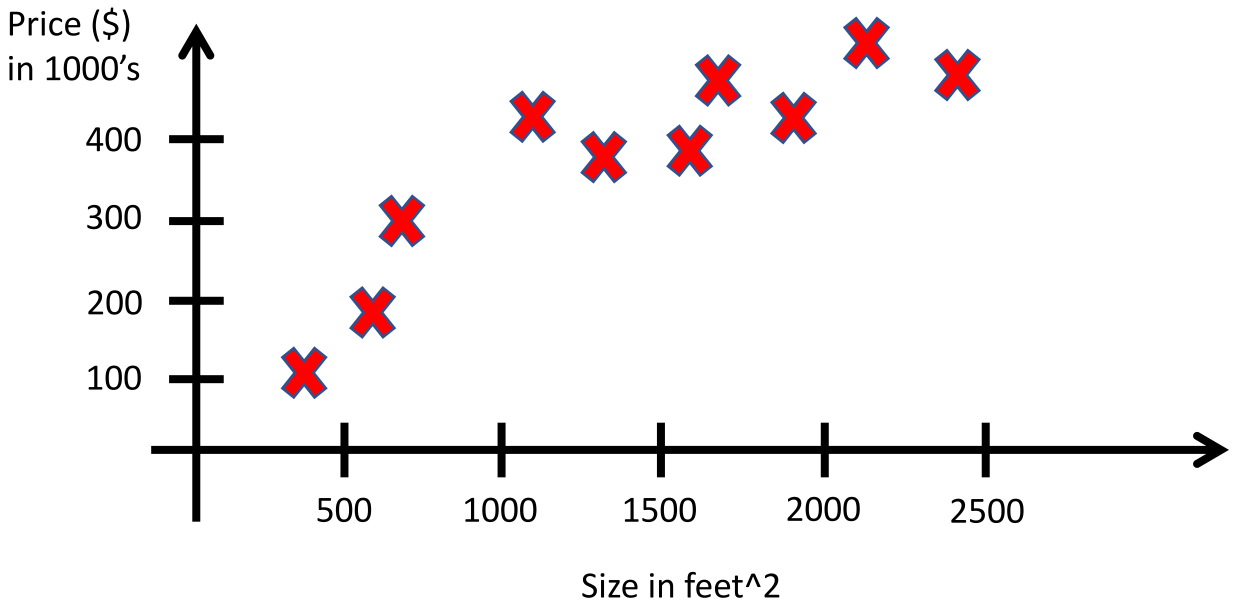
- Area

$$x = \text{frontage} \times \text{depth}$$



- $h_\theta(x) = \theta_0 + \theta_1 x$

Polynomial regression



x_1 = (size)
x_2 = (size)^2
x_3 = (size)^3

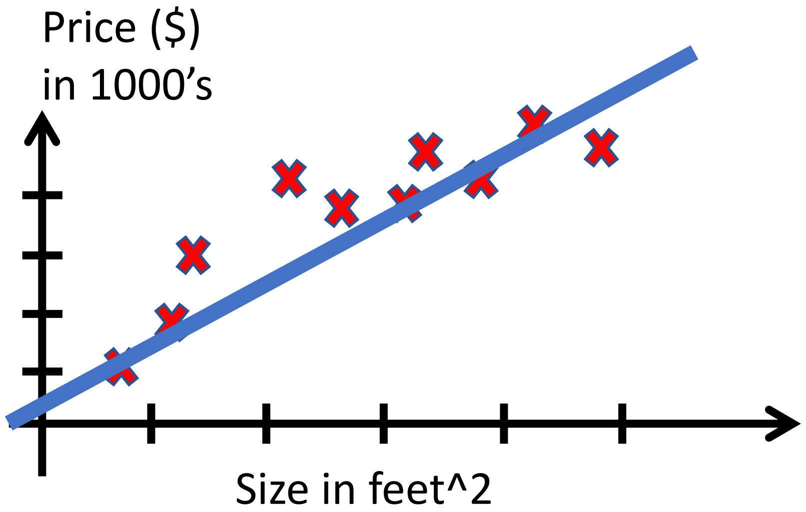
- $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$

$$= \theta_0 + \theta_1(size) + \theta_2(size)^2 + \theta_3(size)^3$$

Regularization

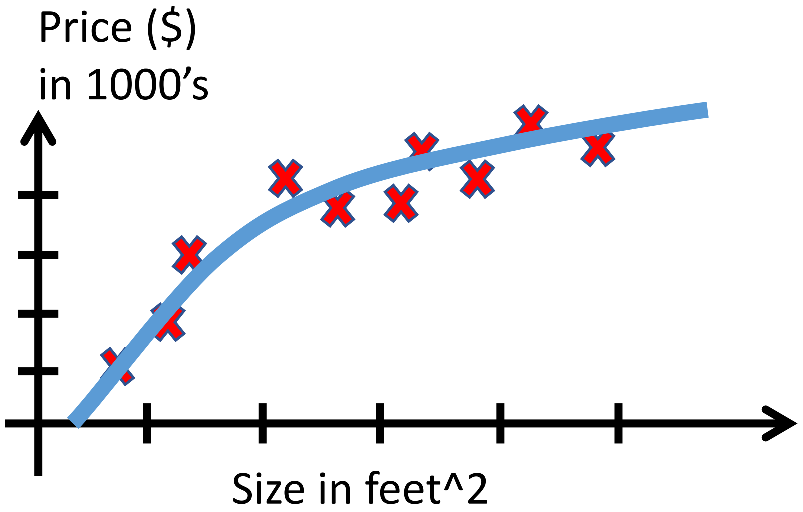
- **Overfitting**
- Cost function
- Regularized linear regression

Example: Linear regression



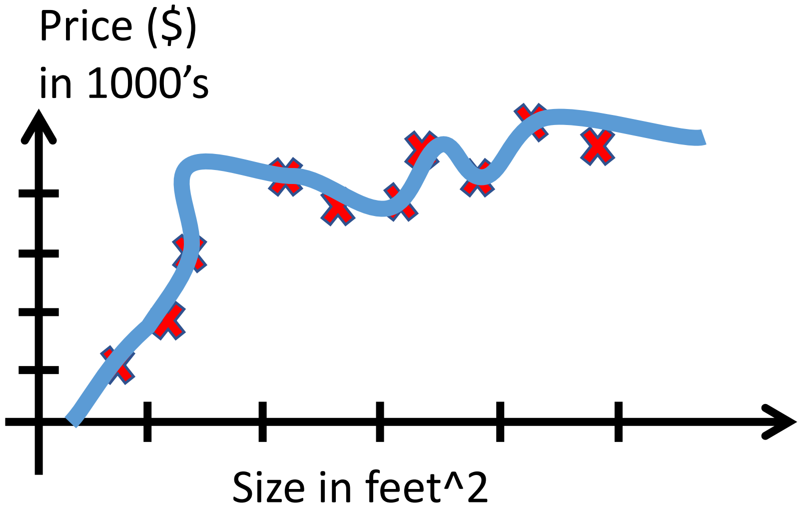
$$h_\theta(x) = \theta_0 + \theta_1 x$$

Underfitting



$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

Just right



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

Overfitting

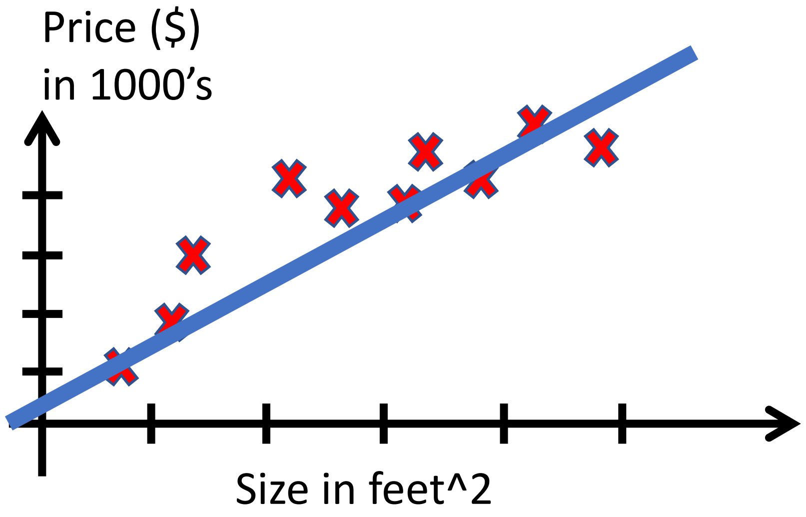
Overfitting

- If we have too many features (i.e. complex model), the learned hypothesis may fit the training set very well

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 \approx 0$$

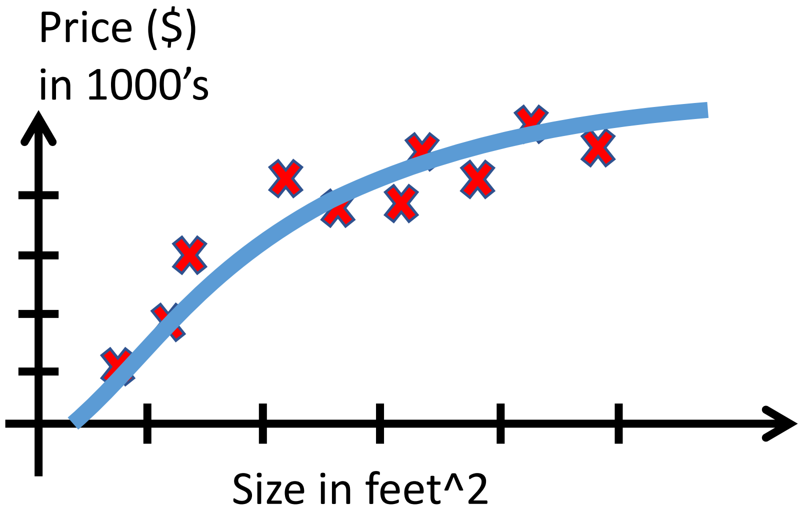
but fail to generalize to new examples (predict prices on new examples).

Example: Linear regression



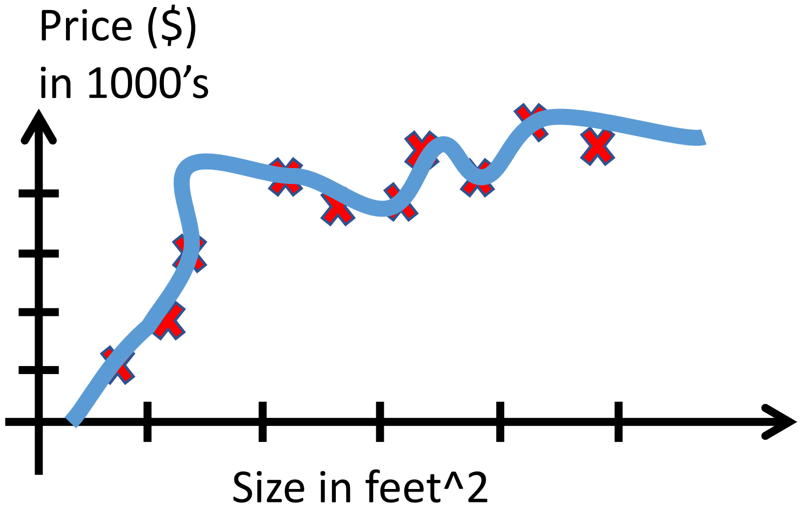
$$h_\theta(x) = \theta_0 + \theta_1 x$$

Underfitting



$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

Just right



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

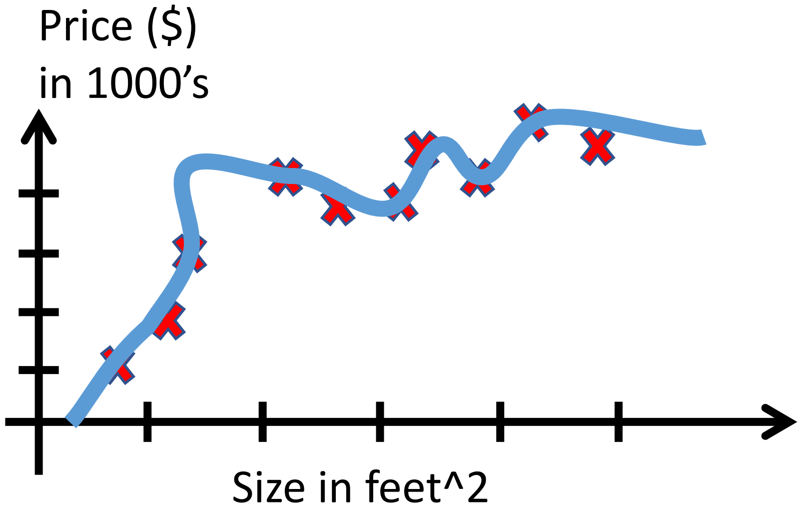
Overfitting

High bias

High variance

Addressing overfitting

- x_1 = size of house
- x_2 = no. of bedrooms
- x_3 = no. of floors
- x_4 = age of house
- x_5 = average income in neighborhood
- x_6 = kitchen size
- $\vdots$
- x_{100}



Addressing overfitting

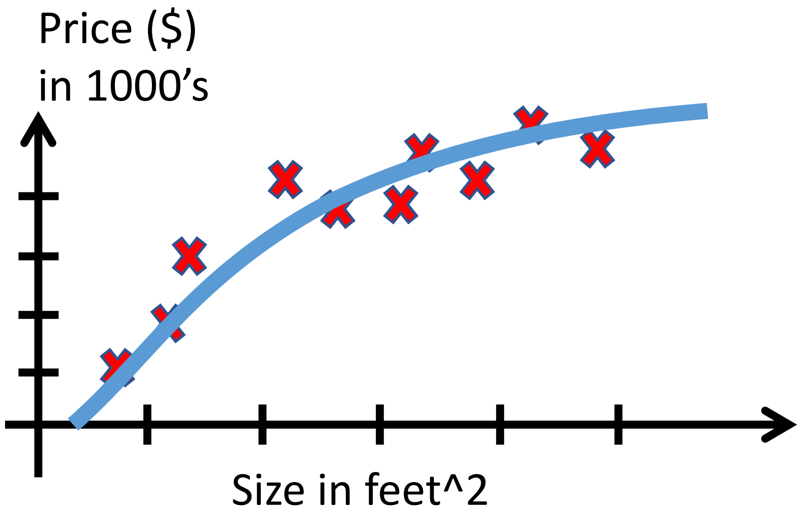
- **1. Reduce number of features.**
  - Manually select which features to keep.
  - Model selection algorithm.

- **2. Regularization.**
  - Keep all the features, but reduce magnitude/values of parameters θ_j.
  - Works well when we have a lot of features, each of which contributes a bit to predicting y.

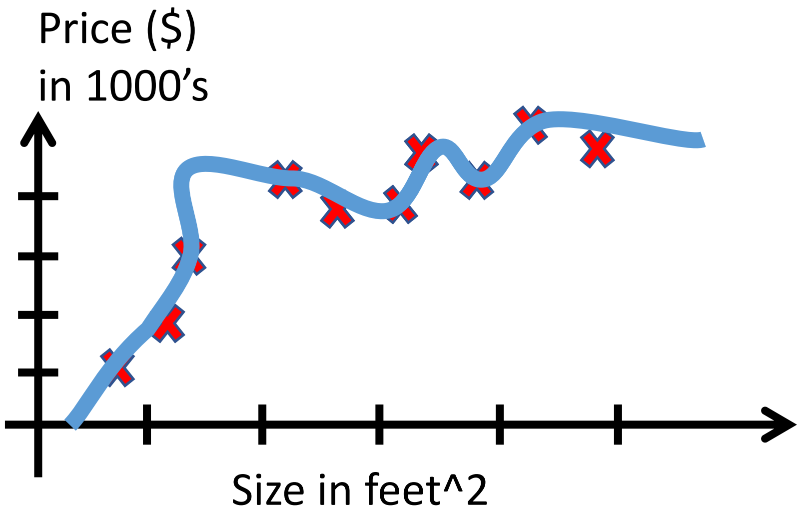
Regularization

- Overfitting
- **Cost function**
- Regularized linear regression

Intuition



$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$



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

- Suppose we penalize and make θ_3, θ_4 really small.

$$\min_\theta J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + 1000\,\theta_3^2 + 1000\,\theta_4^2$$

Regularization.

- Small values for parameters $\theta_1, \theta_2, \cdots, \theta_n$
  - "Simpler" hypothesis
  - Less prone to overfitting
- Housing:
  - Features: $x_1, x_2, \cdots, x_{100}$
  - Parameters: $\theta_0, \theta_1, \theta_2, \cdots, \theta_{100}$

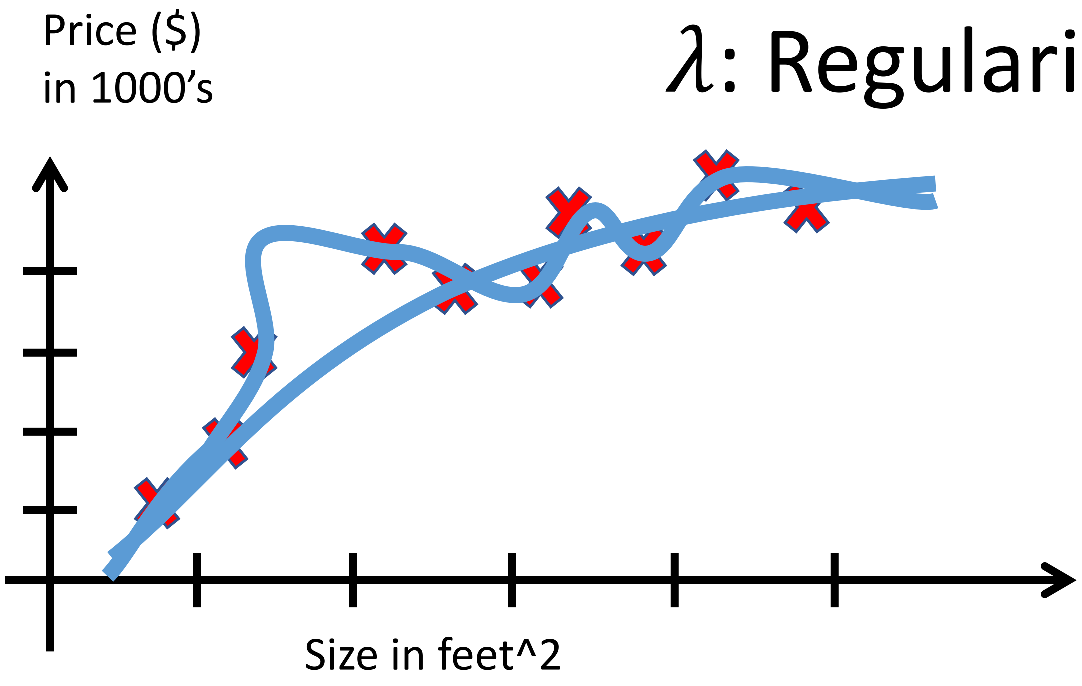
$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

Regularization

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right]$$

λ: Regularization parameter

$$\min_\theta J(\theta)$$



Question

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

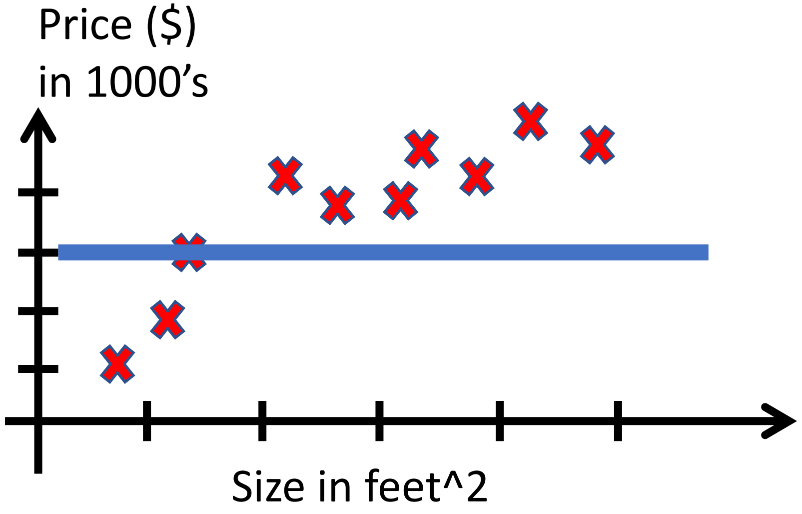
What if λ is set to an extremely large value (say $\lambda = 10^{10}$)?

1. Algorithm works fine; setting to be very large can't hurt it
2. Algorithm fails to eliminate overfitting.
3. Algorithm results in underfitting. (Fails to fit even training data well).
4. Gradient descent will fail to converge.

Question

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

What if λ is set to an extremely large value (say $\lambda = 10^{10}$)?



$$h_\theta(x) = \theta_0 + \cancel{\theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n} = \theta^\top x$$

Important points about λ:

- λ is the tuning parameter used in regularization that decides how much we want to penalize the flexibility of our model i.e, **controls the impact on bias and variance**.
- When **λ = 0**, the penalty term has no effect, the equation becomes the cost function of the linear regression model. Hence, for the minimum value of λ i.e, λ=0, the model will resemble the linear regression model. So, the estimates produced by ridge regression will be equal to least squares.
- However, as **λ→∞** (tends to infinity), the impact of the shrinkage penalty increases, and the ridge regression coefficient estimates will approach zero.

Regularization

- Overfitting
- Cost function
- **Regularized linear regression**

Regularized linear regression

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{\boxed{j=1}}^{n} \theta_j^2\right]$$

$$\min_\theta J(\theta)$$

n: Number of features

θ_0 is not panelized

Gradient descent (Previously)

Repeat {

$$\theta_0 \coloneqq \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) \qquad (j = 0)$$

$$\theta_j \coloneqq \theta_j - \alpha \frac{1}{m} \left[\sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) x_j^{(i)} \right] \qquad (j = 1, 2, 3, \cdots, n)$$

}

Gradient descent (Regularized)

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)$$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \left[\sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) x_j^{(i)} + \lambda \theta_j \right]$$

}

$$\theta_j := \theta_j \left(1 - \alpha \frac{\lambda}{m}\right) - \alpha \frac{1}{m} \left[\sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) x_j^{(i)} \right]$$

Comparison

$1 - \alpha\frac{\lambda}{m} < 1$: Weight decay

Regularized linear regression

$$\theta_j := \theta_j\left(1 - \alpha\frac{\lambda}{m}\right) - \alpha\frac{1}{m}\left[\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}\right]$$

Un-regularized linear regression

$$\theta_j := \theta_j - \alpha\frac{1}{m}\left[\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}\right]$$

Linear Regression

- Model representation
- Cost function
- Gradient descent
- Features and polynomial regression

Question

Let f be some function so that $f(\theta_0,\theta_1)$ outputs a number. For this problem, f is some arbitrary/unknown smooth function (not necessarily the cost function of linear regression, so f may have local optima). Suppose we use gradient descent to try to minimize $f(\theta_0,\theta_1)$ as a function of θ_0 and θ_1. Which of the following statements are true? (Check all that apply.)

- Even if the learning rate α is very large, every iteration of gradient descent will decrease the value of $f(\theta_0,\theta_1)$.
- If the learning rate is too small, then gradient descent may take a very long time to converge.
- If θ_0 and θ_1 are initialized at a local minimum, then one iteration will not change their values.
- If θ_0 and θ_1 are initialized so that $\theta_0=\theta_1$, then by symmetry (because we do simultaneous updates to the two parameters), after one iteration of gradient descent, we will still have $\theta_0=\theta_1$.

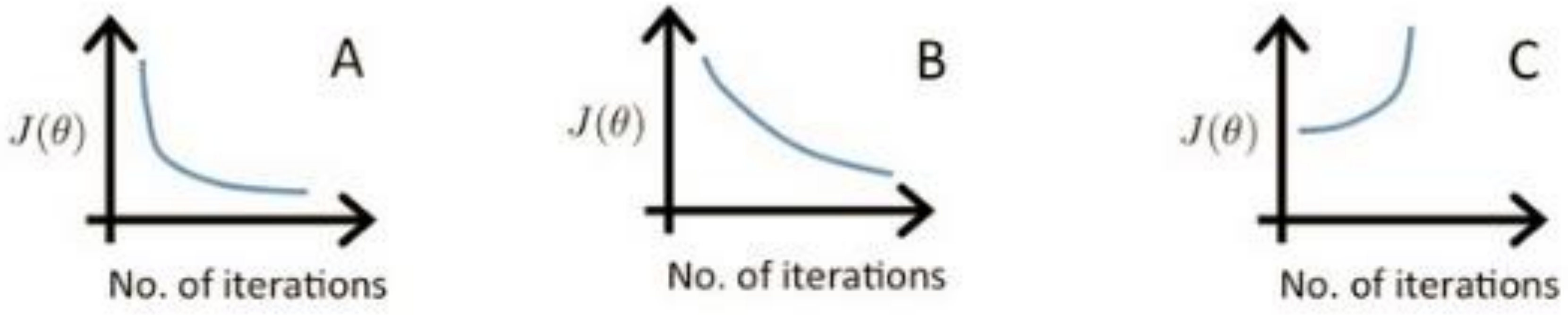
Question

You run gradient descent for 15 iterations with α=0.3 and compute $J(\theta)$ after each iteration. You find that the value of $J(\theta)$ decreases quickly then levels off. Based on this, which of the following conclusions seems most plausible?

- Rather than use the current value of α, it'd be more promising to try a larger value of α (say α=1.0).
- Rather than use the current value of α, it'd be more promising to try a smaller value of α (say α=0.1).
- α=0.3 is an effective choice of learning rate.

Question

- Which of the following is true about below graphs(A,B, C left to right) between the cost function and Number of iterations?



- A) l2 < l1 < l3
- B) l1 > l2 > l3
  C) l1 = l2 = l3
  D) None of the above

Reference

- https://www.analyticsvidhya.com/blog/2021/05/complete-guide-to-regularization-techniques-in-machine-learning/
- Machine Learning by Andrew Ng
- https://filebox.ece.vt.edu/~jbhuang/teaching/ECE5424-CS5824/sp19
- https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a
- https://towardsdatascience.com/feature-selection-in-machine-learning-using-lasso-regression-7809c7c2771a