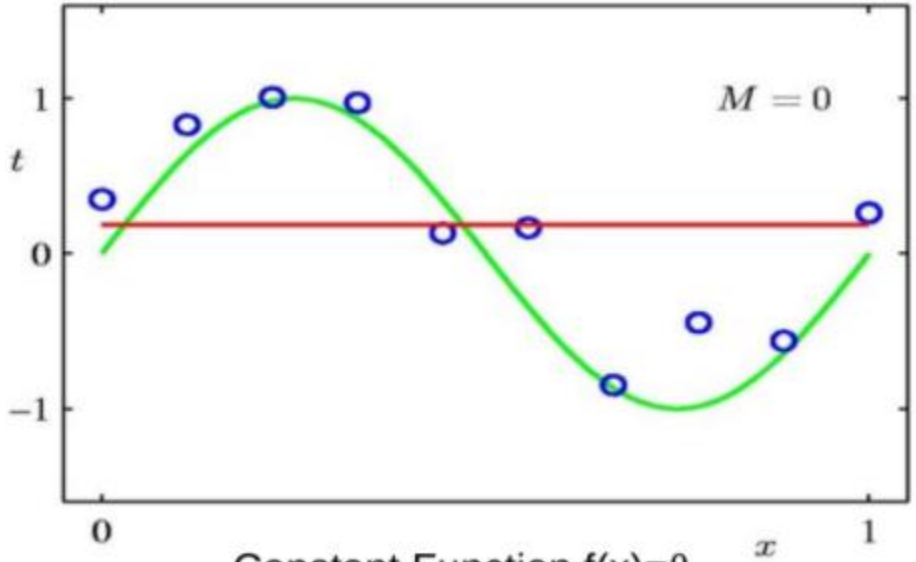
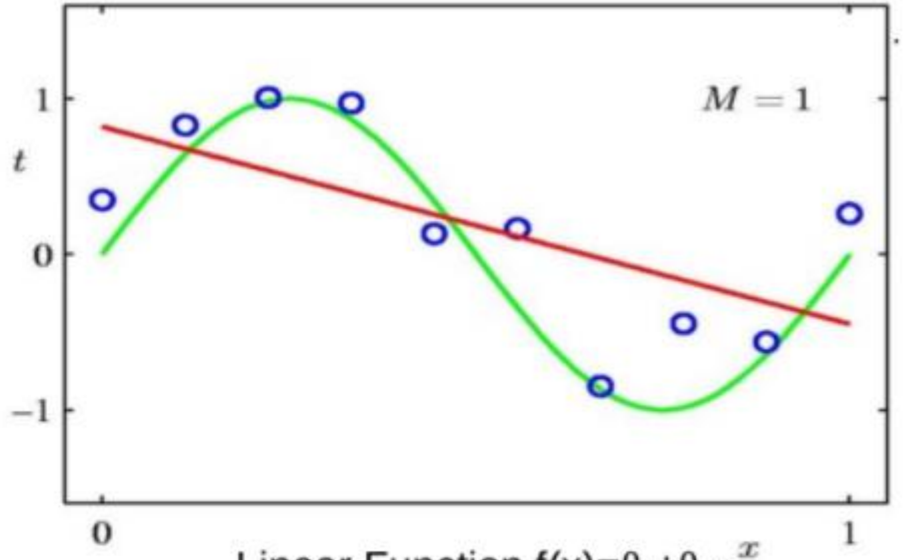
Linear Regression

Data Distribution and the function learned.

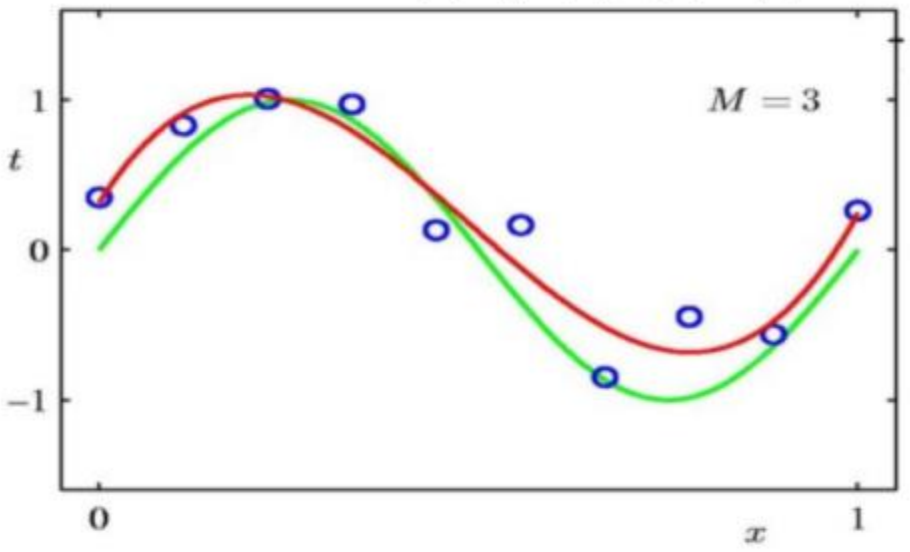


Constant Function $f(x)=\theta_0$

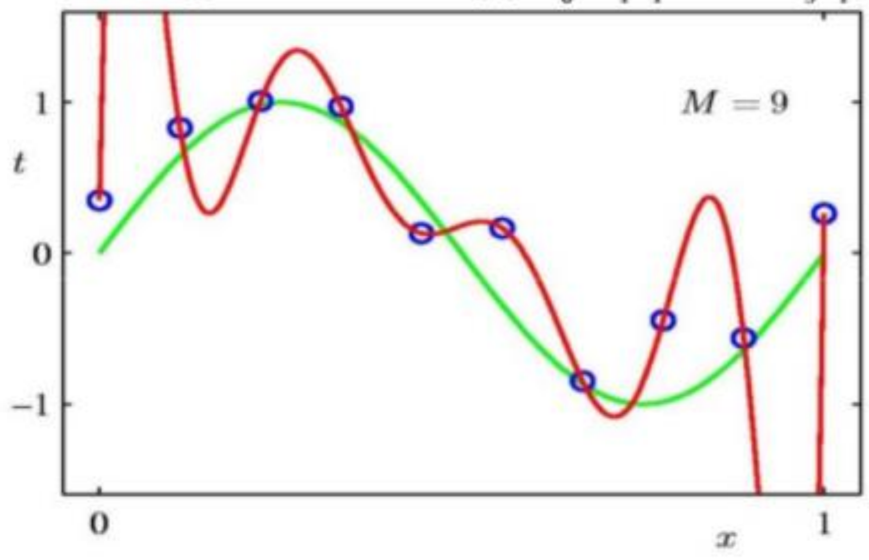


Linear Function $f(x)=\theta_0+\theta_1 x_1$

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



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

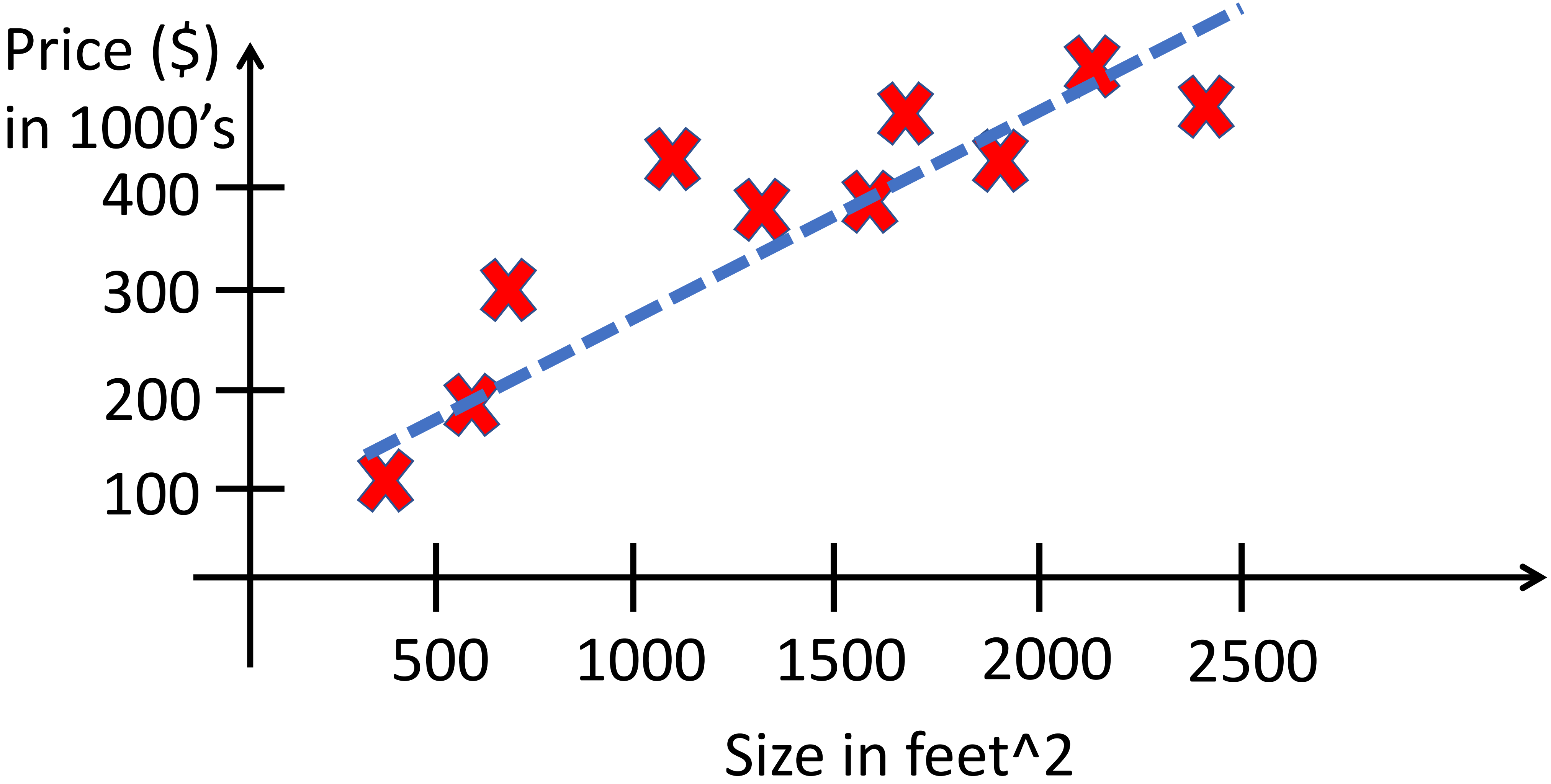


Recap: Machine learning algorithms

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

House pricing prediction

→ Regression Based problem example.



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:

$x^{(1)} = 2104$

$x^{(2)} = 1416$

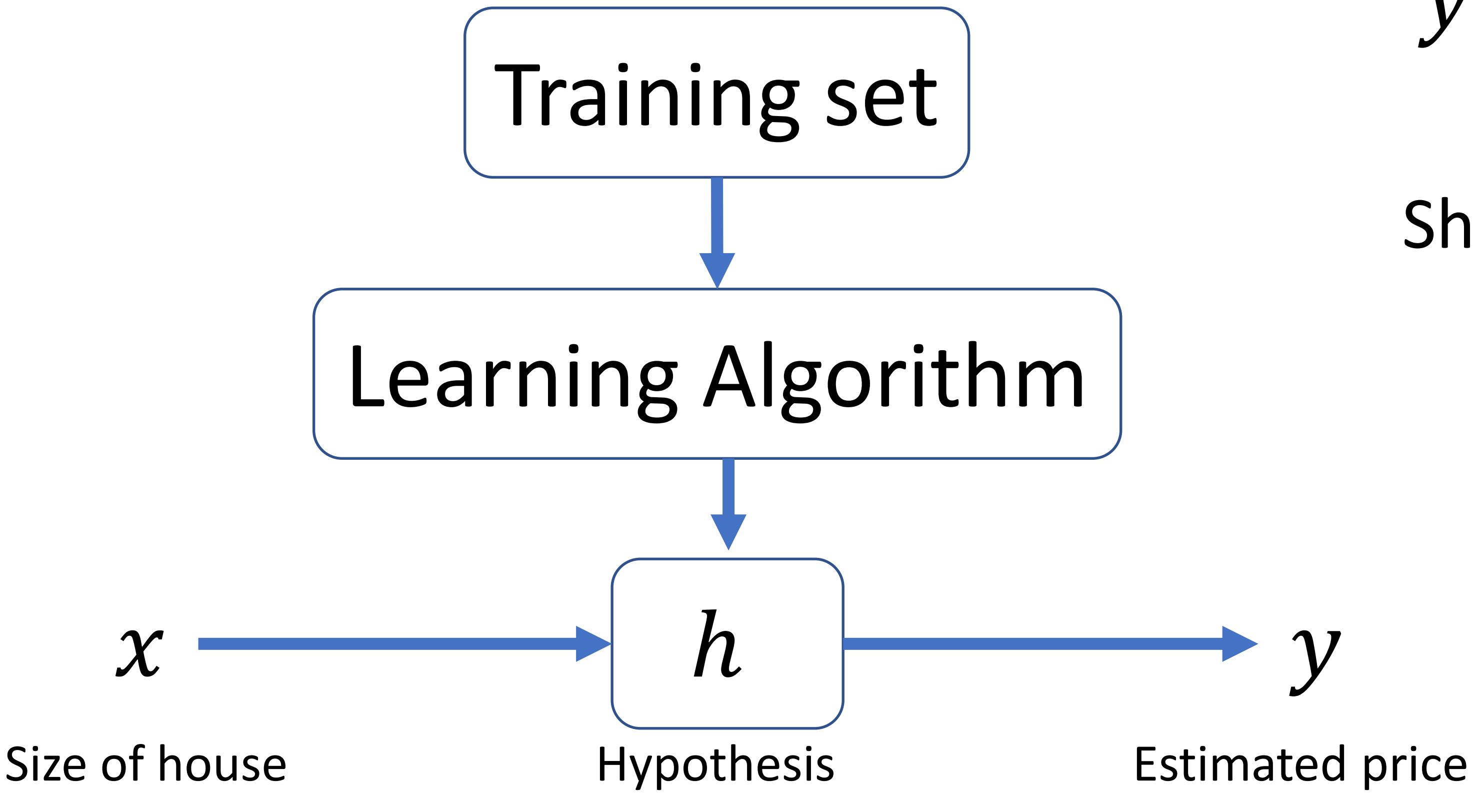
$y^{(1)} = 460$

$\Rightarrow$ Univariate Linear Regression

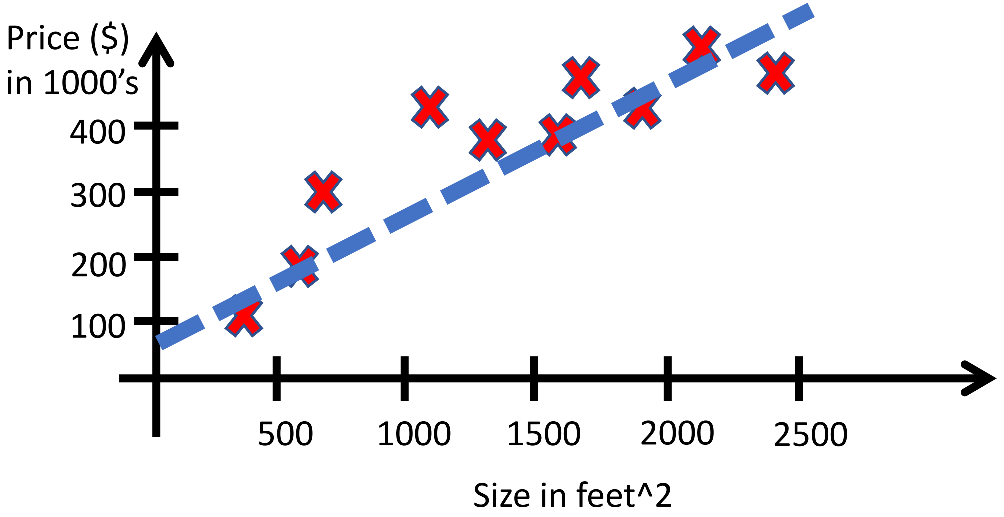
↳ Only one feature.

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
…	…

$m = 47$

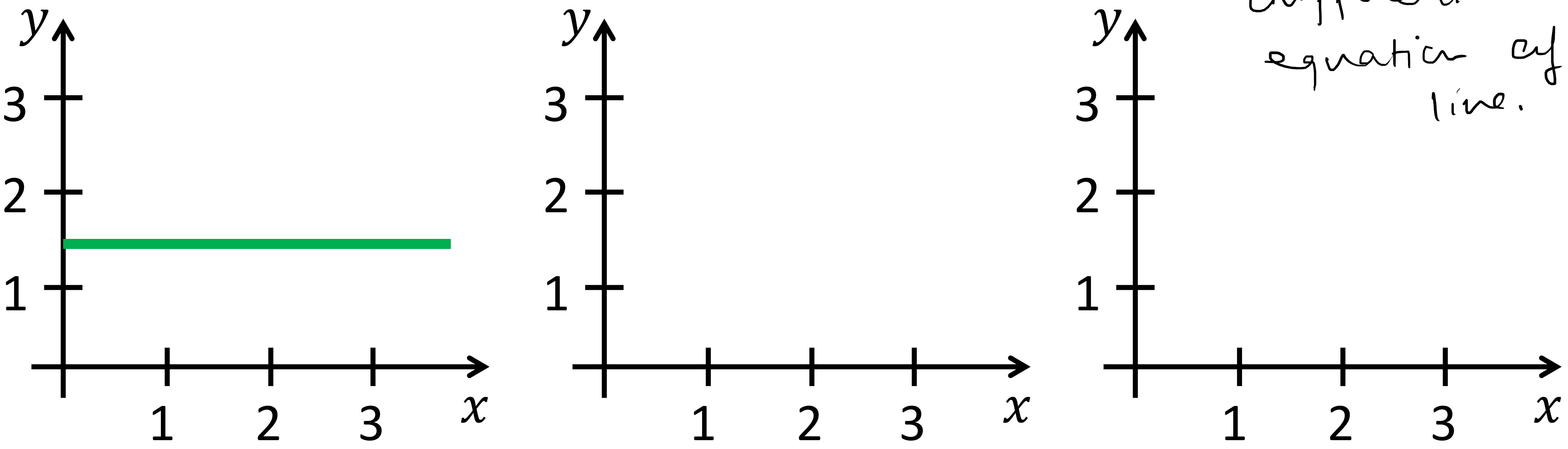
- Hypothesis $h_\theta(x) = \theta_0 + \theta_1 x$

θ_0, θ_1: parameters/weights

How to choose θ_i's?

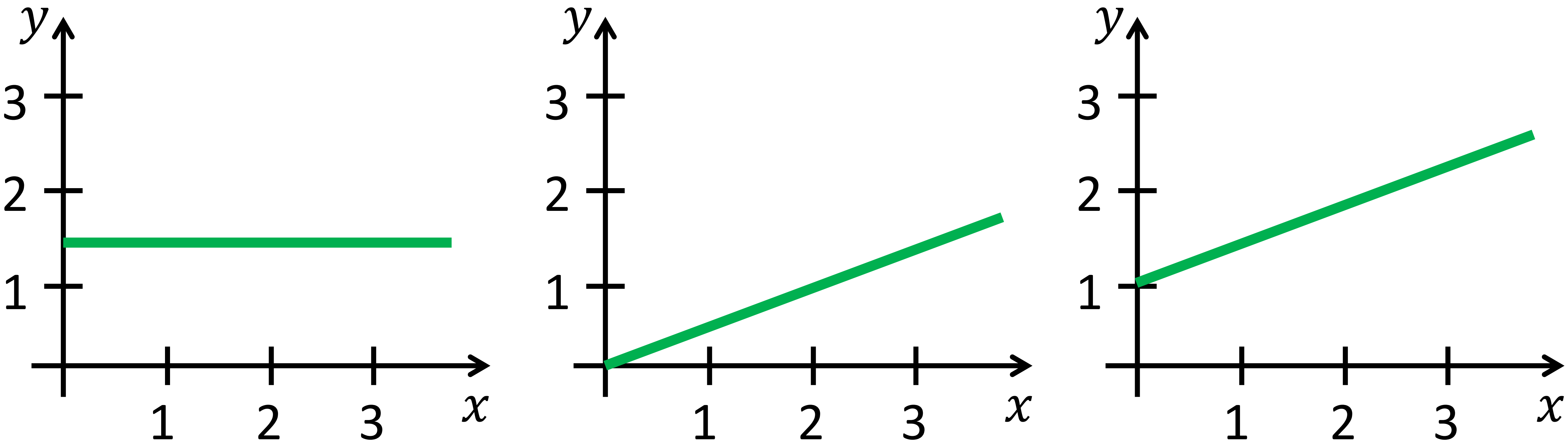
$$h_\theta(x) = \theta_0 + \theta_1 x$$

Different values of θ's give different equation of line.



$\theta_0 = 1.5$	$\theta_0 = 0$	$\theta_0 = 1$
$\theta_1 = 0$	$\theta_1 = 0.5$	$\theta_1 = 0.5$

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



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

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

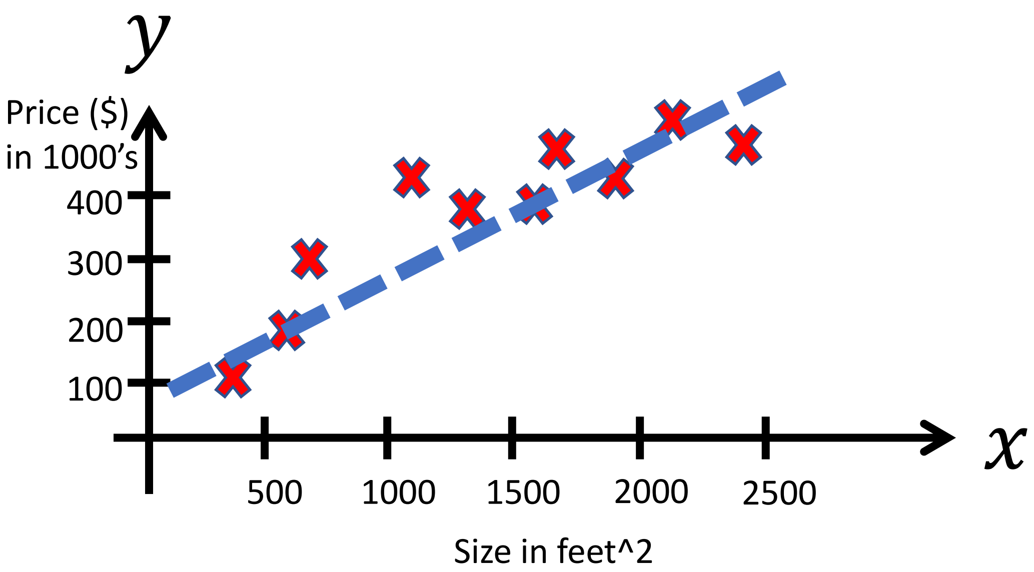
$\theta_0 = 1$
$\theta_1 = 0.5$

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:**

$$\theta_1$$

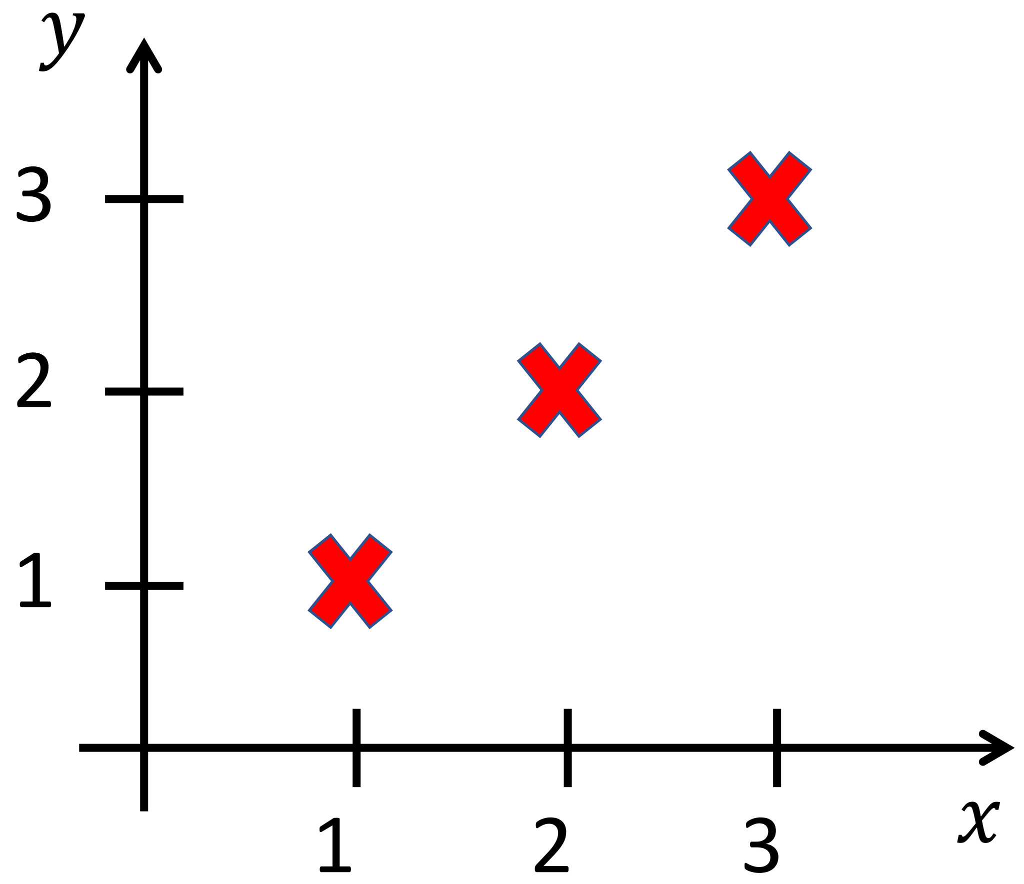
- **Cost function:**

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

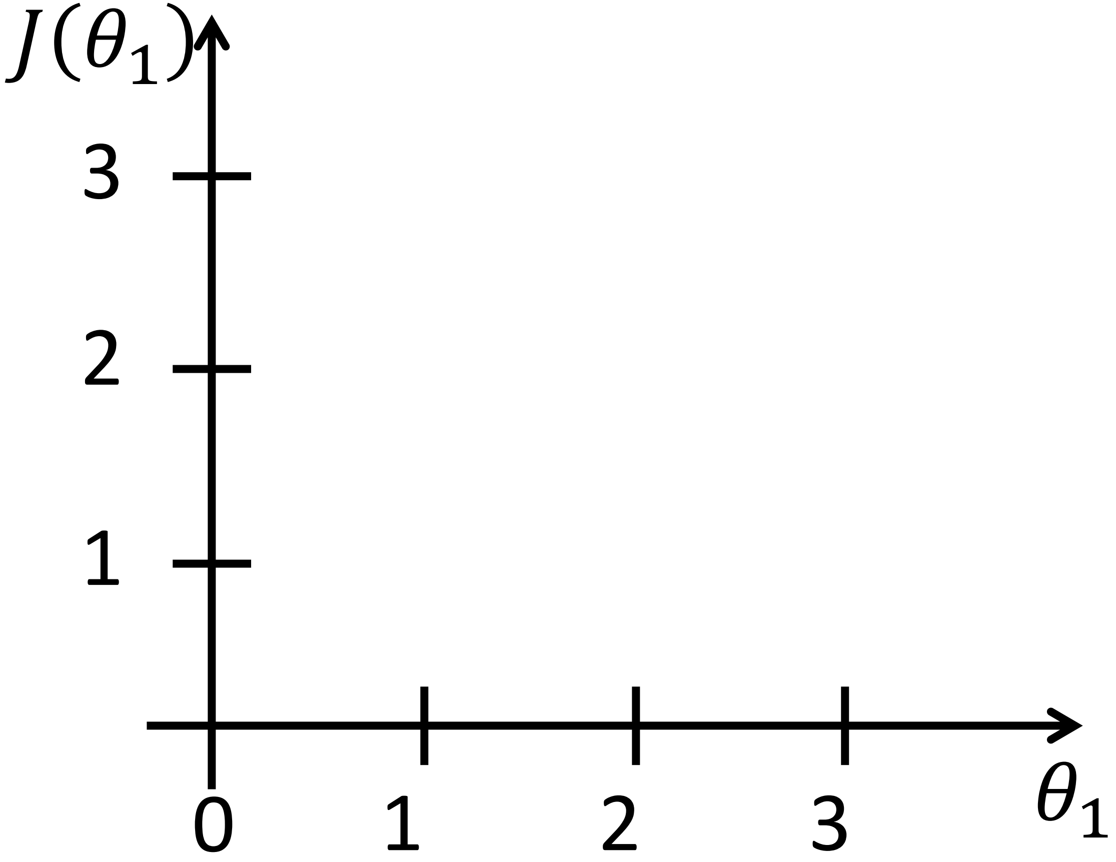
- **Goal:**

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

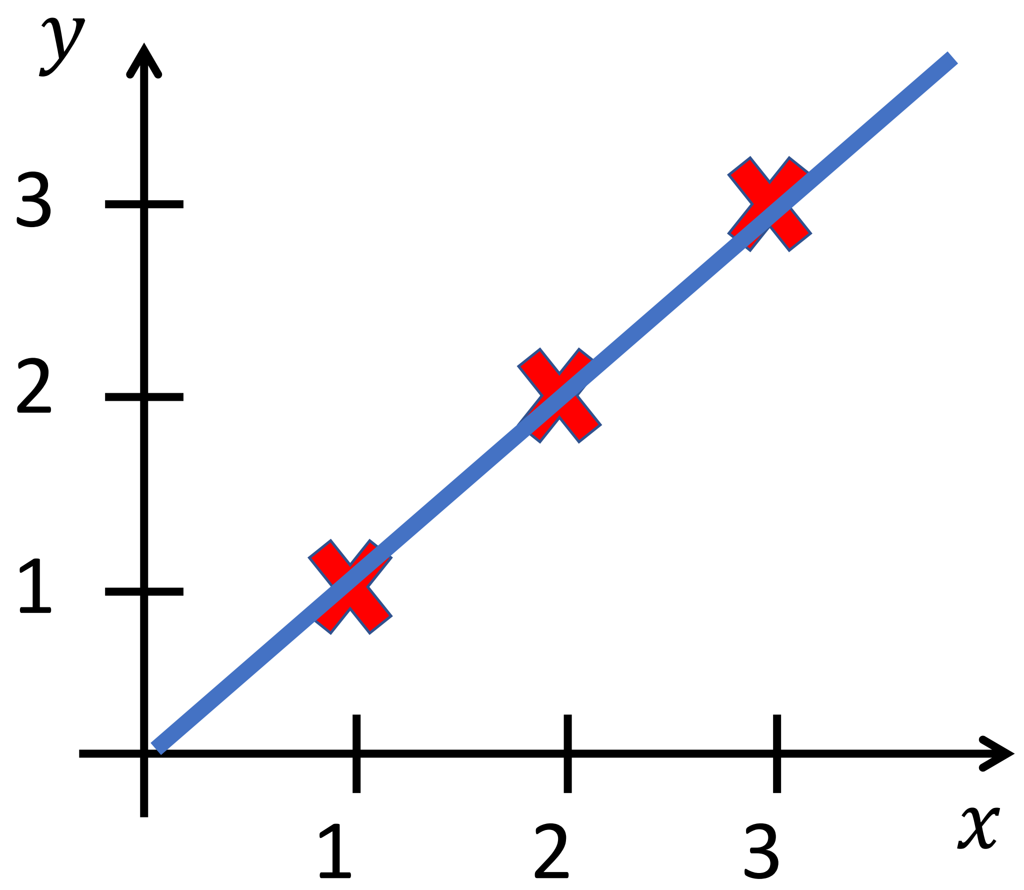
$h_\theta(x)$, function of x



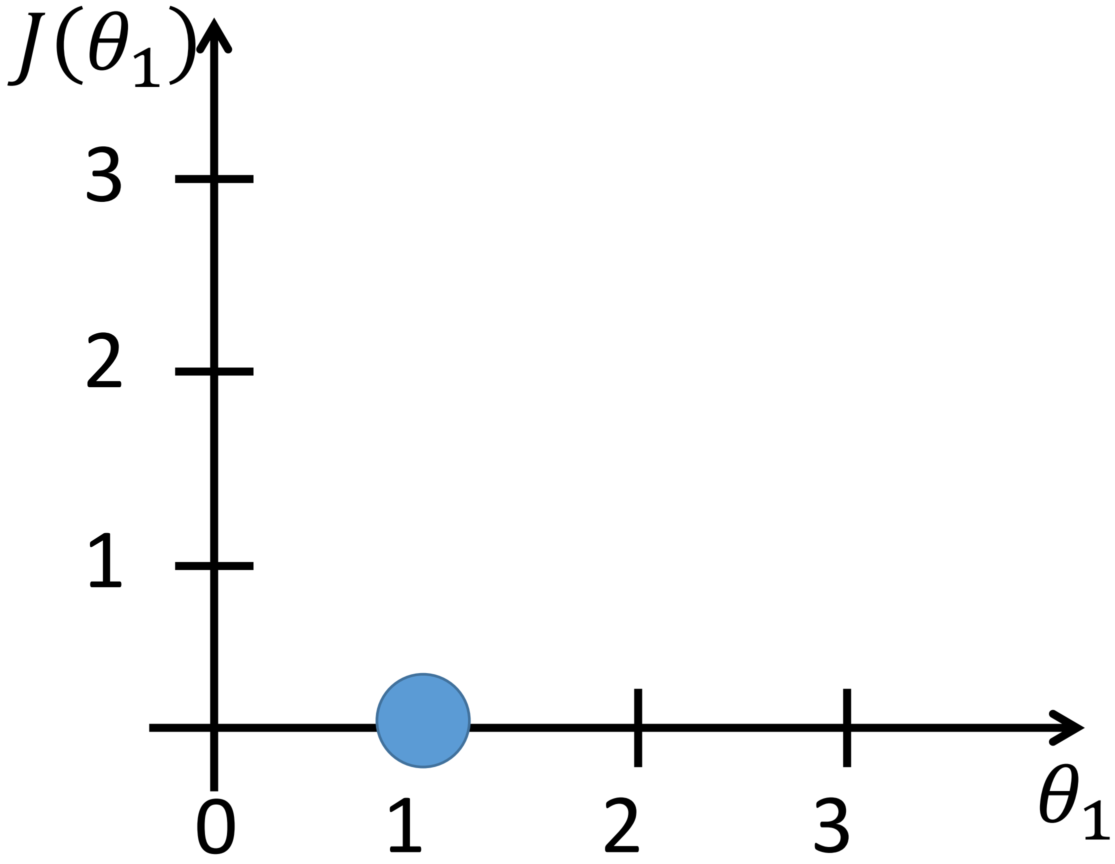
$J(\theta_1)$, function of θ_1



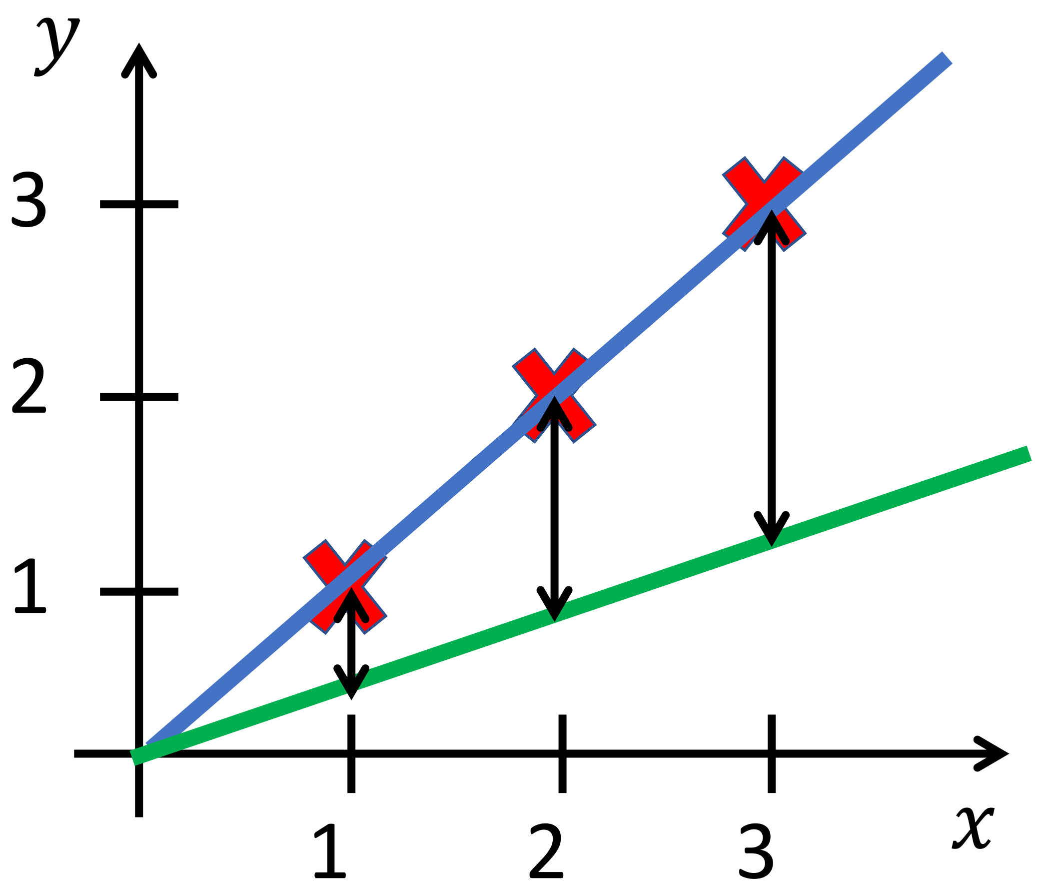
$h_\theta(x)$, function of x



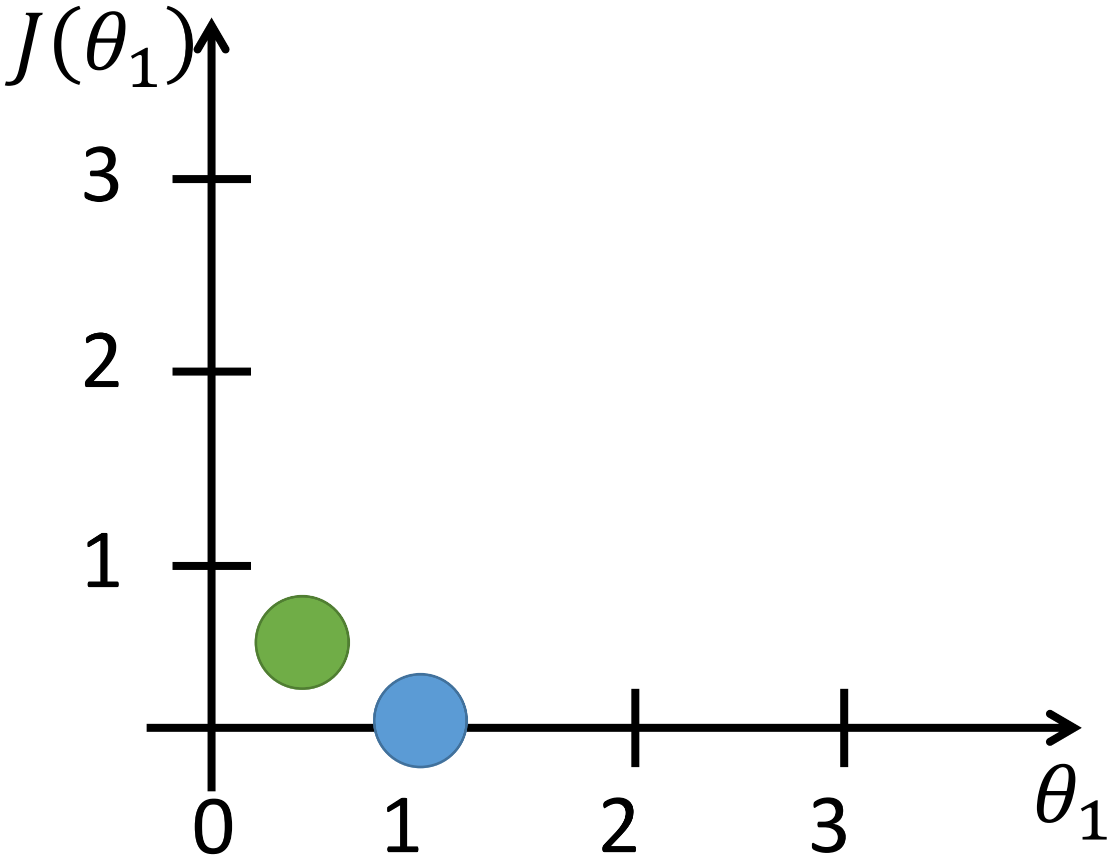
$J(\theta_1)$, function of θ_1



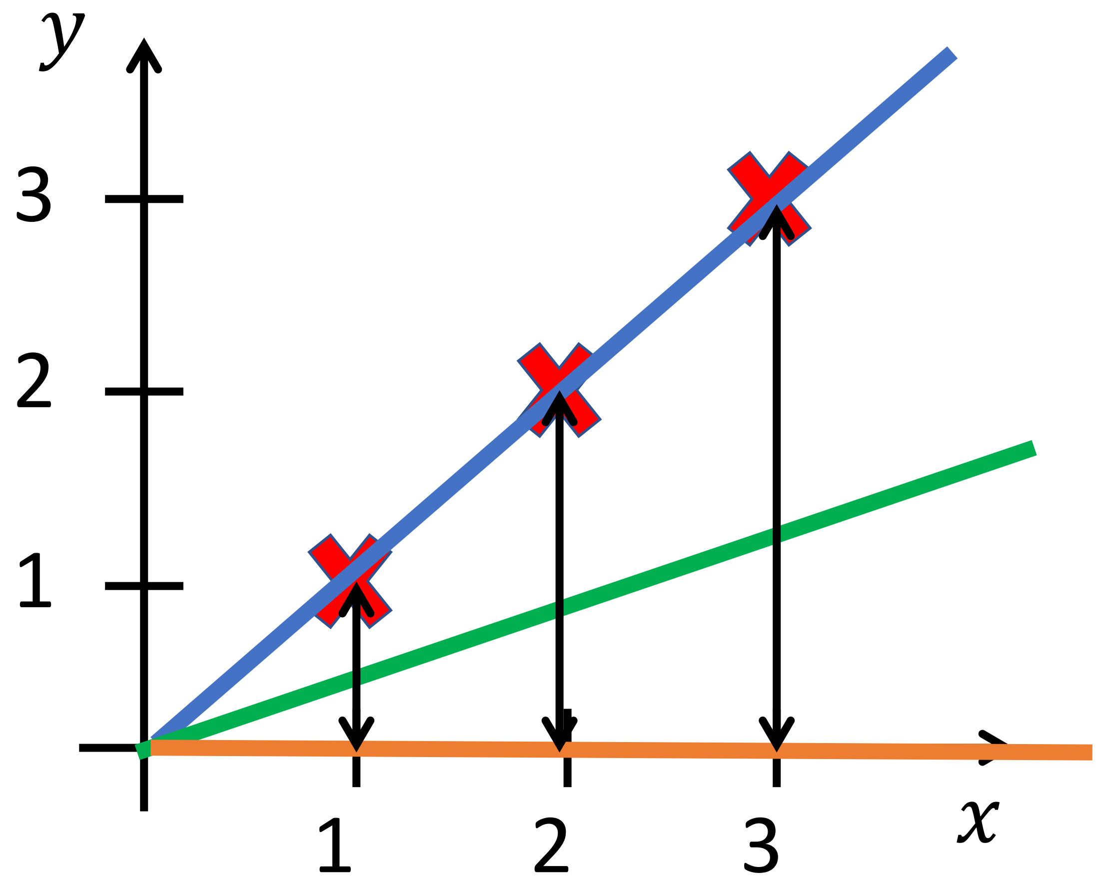
$h_\theta(x)$, function of x



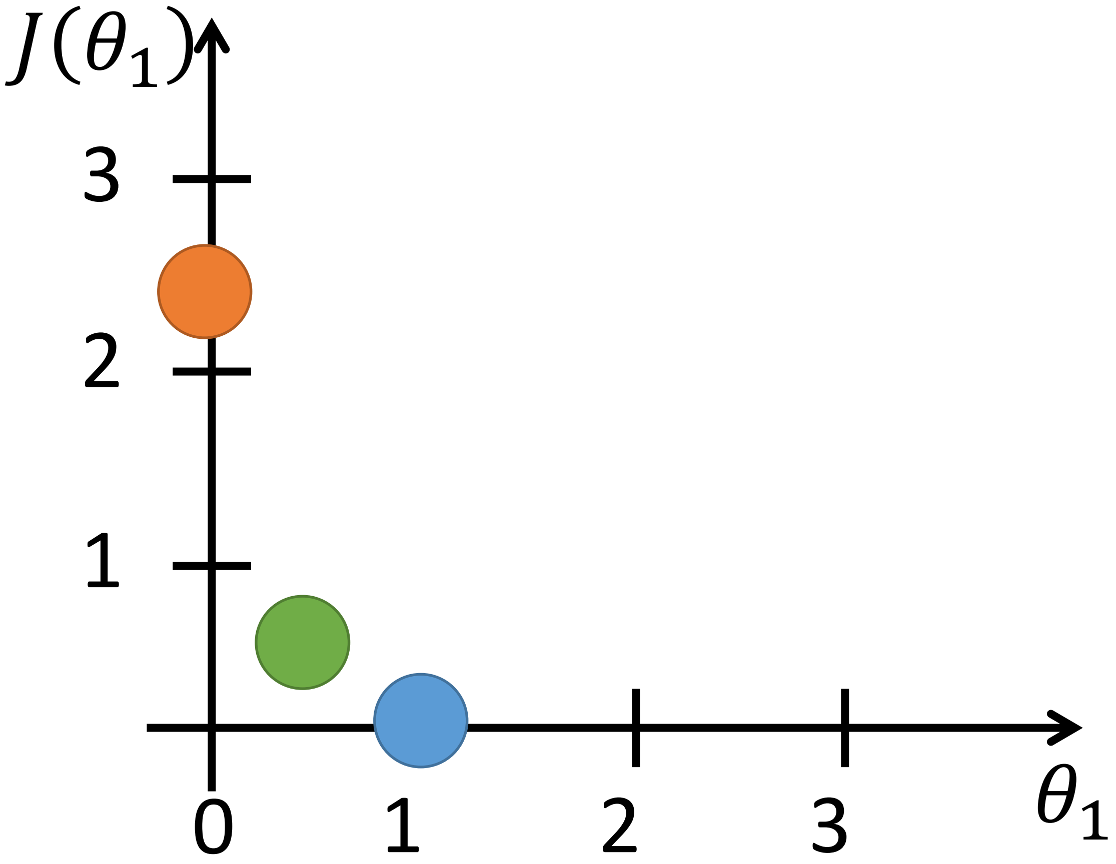
$J(\theta_1)$, function of θ_1



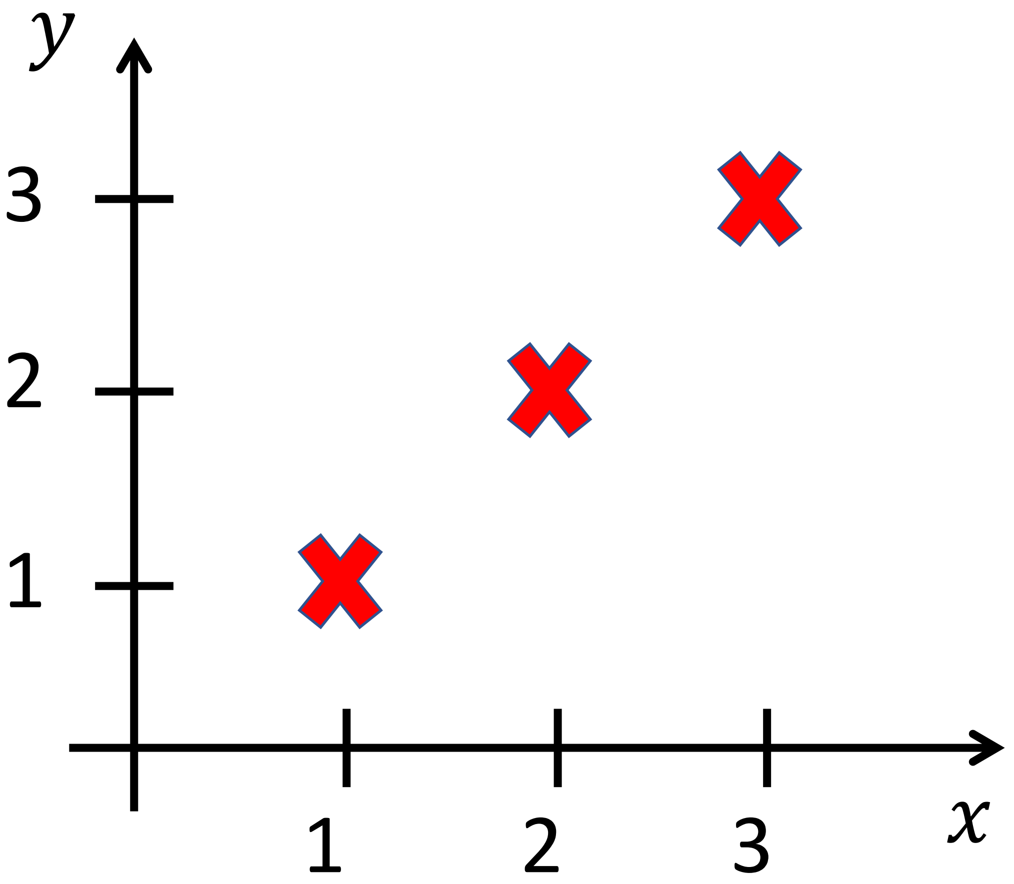
$h_\theta(x)$, function of x



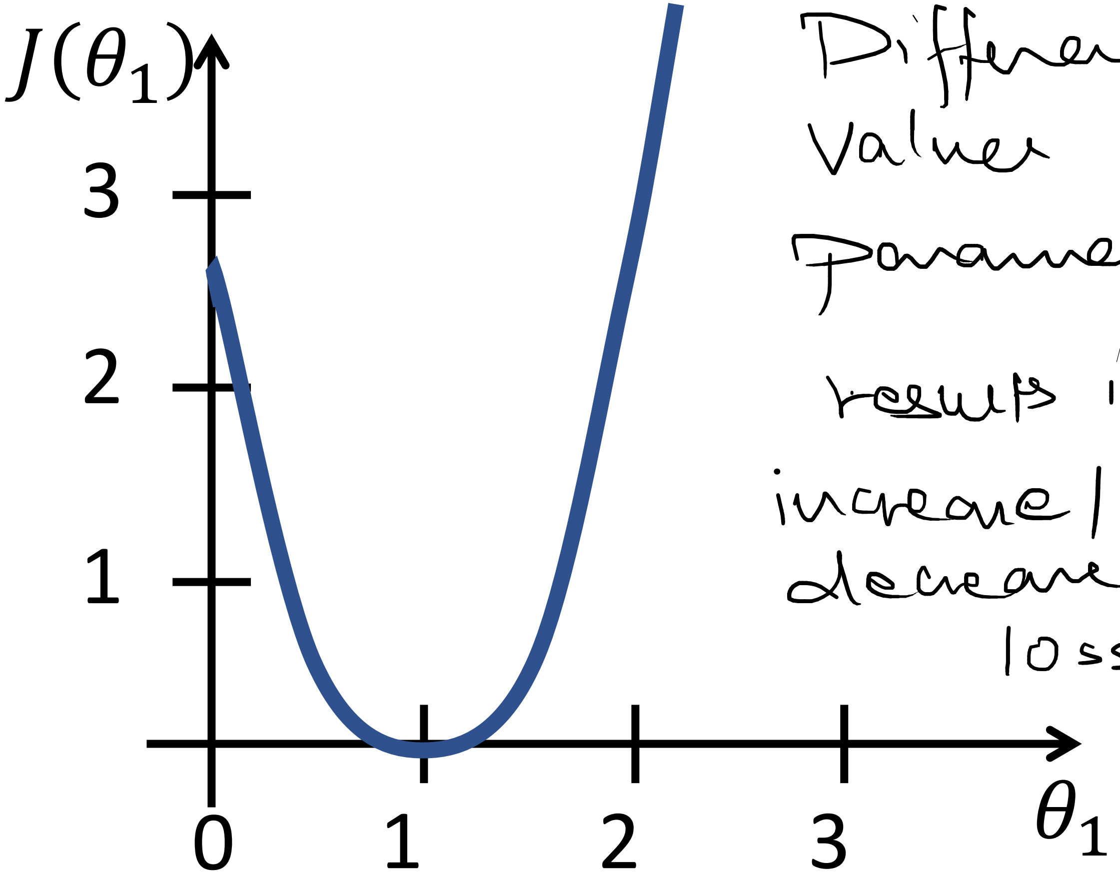
$J(\theta_1)$, function of θ_1



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



$J(\theta_1)$, function of θ_1



Different Values of Parameters results into increase/decrease of loss.

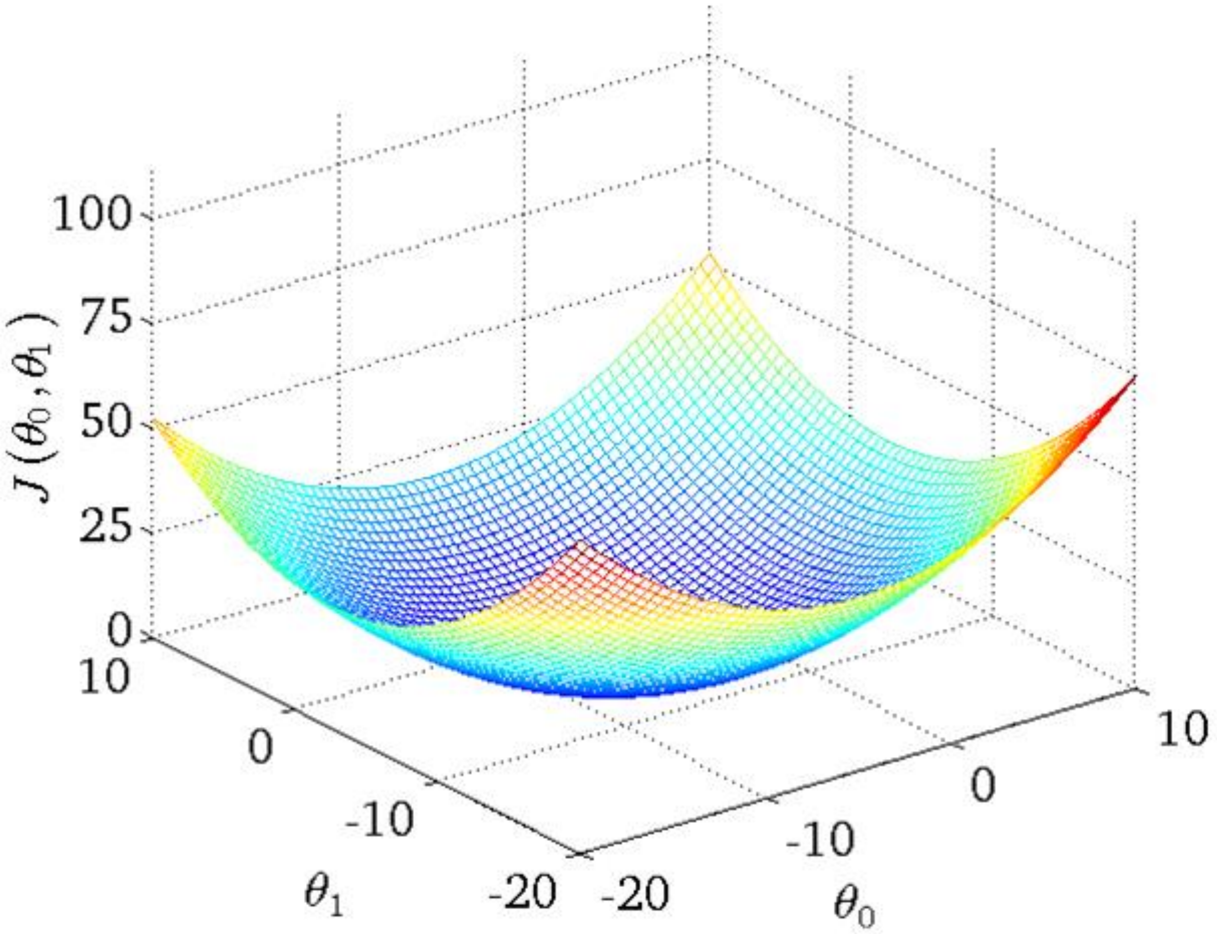
- **Hypothesis:** $h_\theta(x) = \theta_0 + \theta_1 x$

- **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

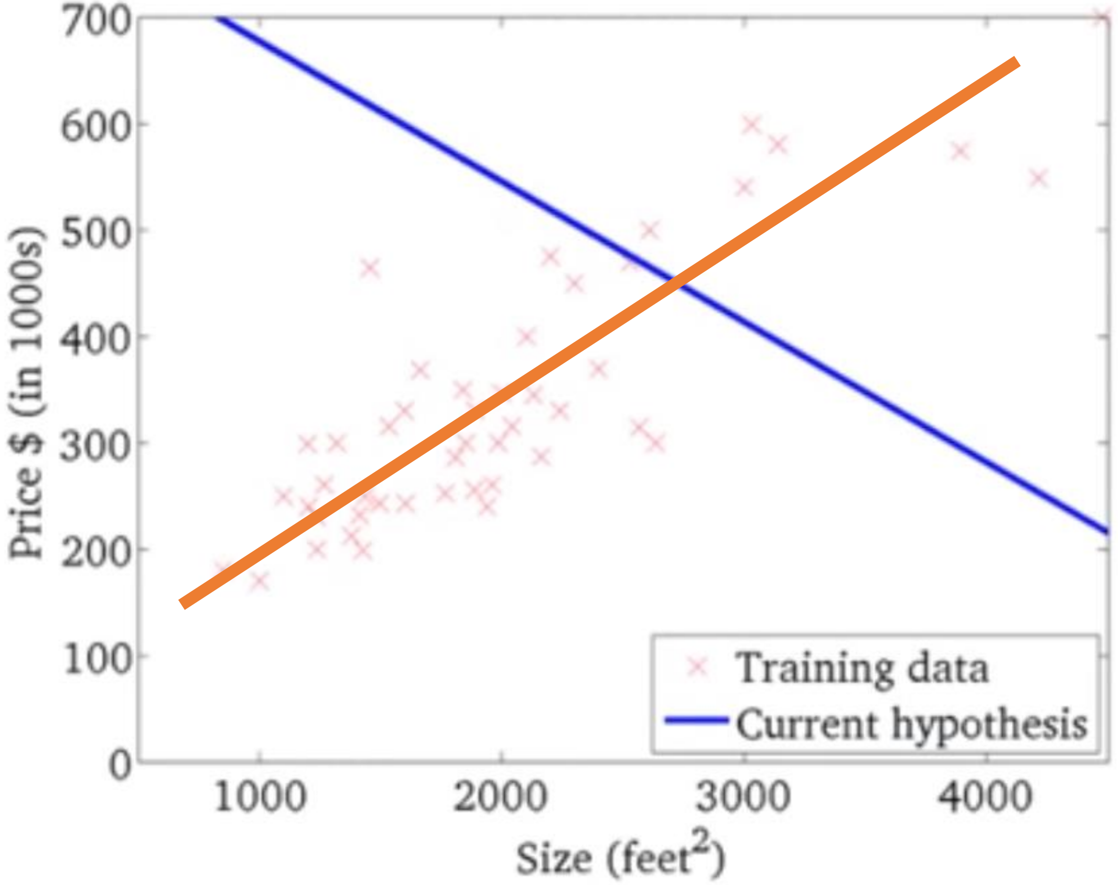


→ convex loss function.

→ w.r.t. to θ_0 & θ_1

$h_\theta(x)$

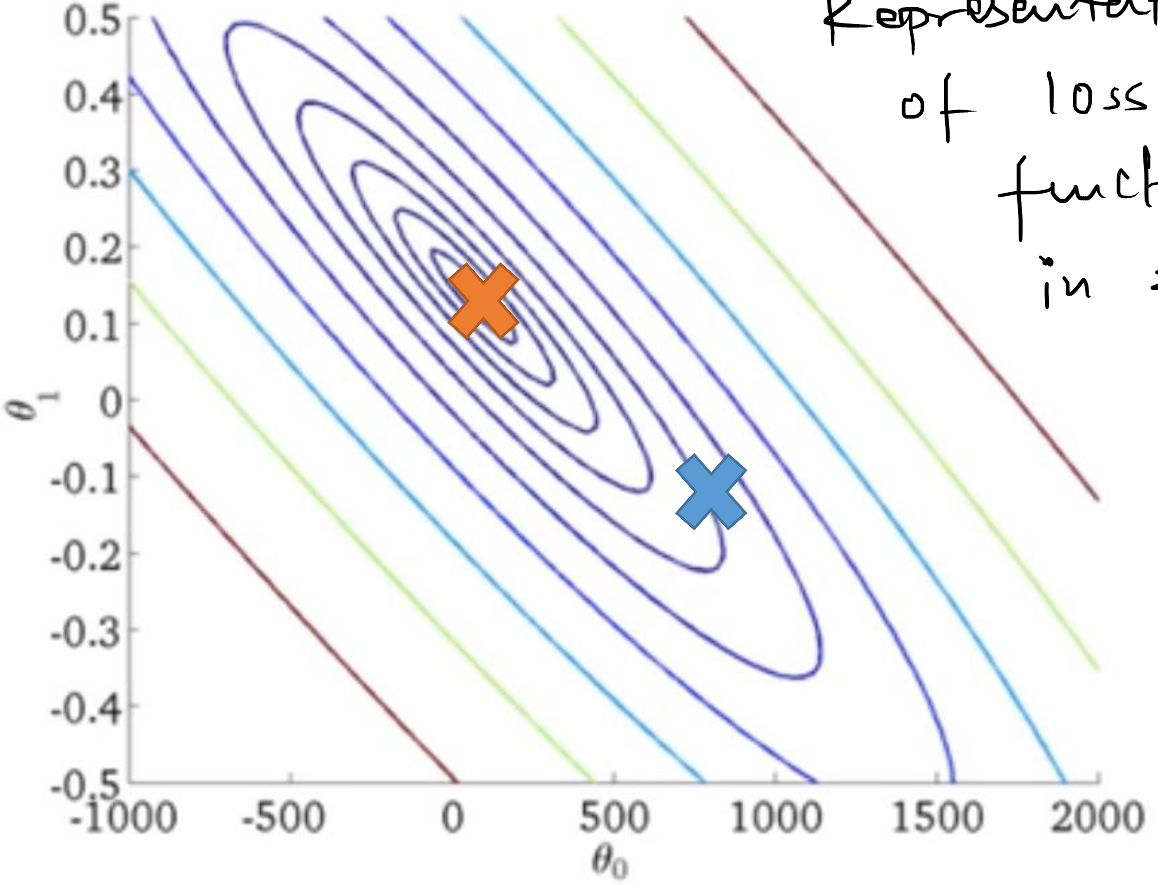
(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

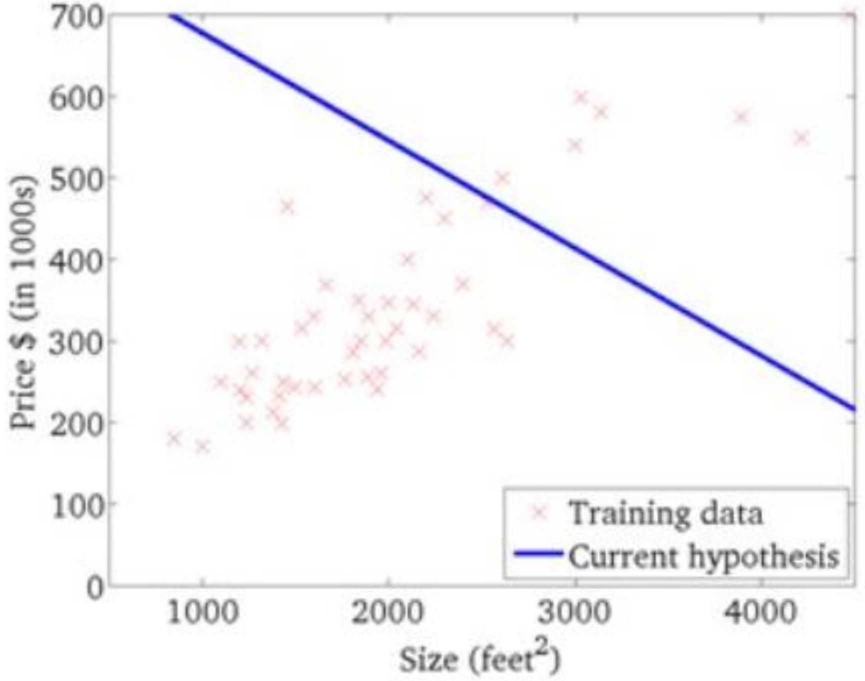
Representation of loss function in 2D.



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

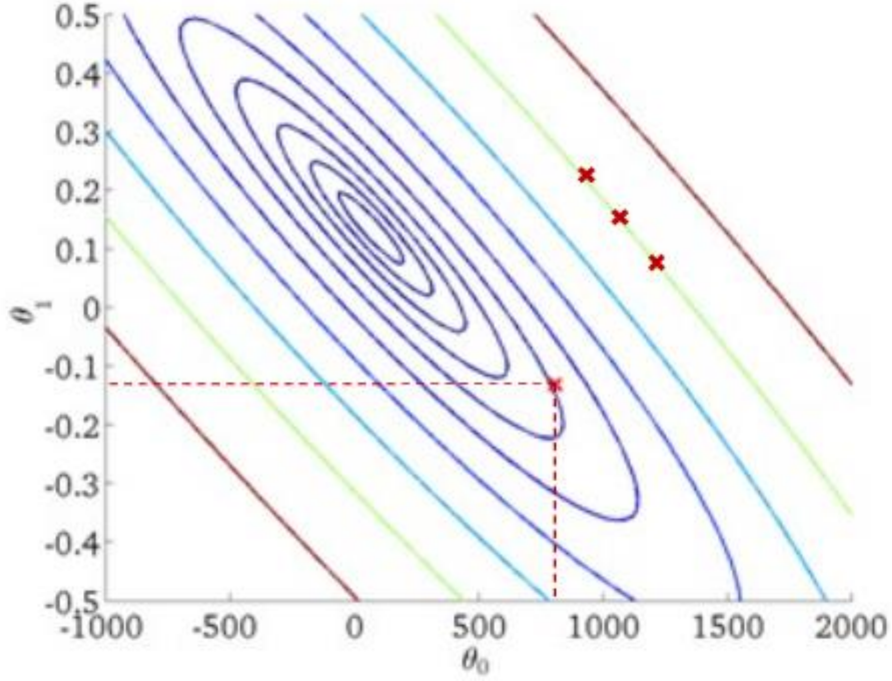
$$h_\theta(x)$$

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



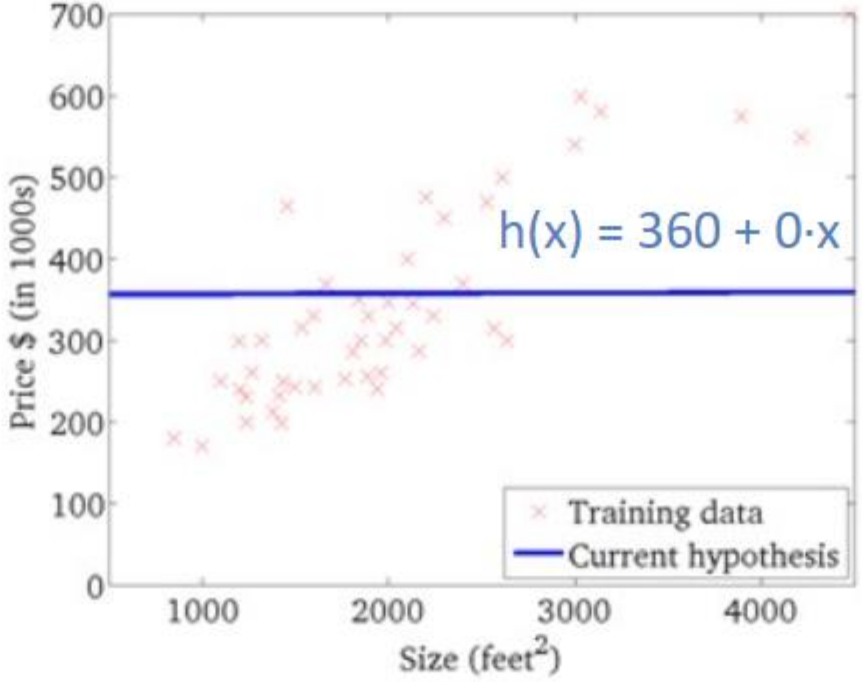
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



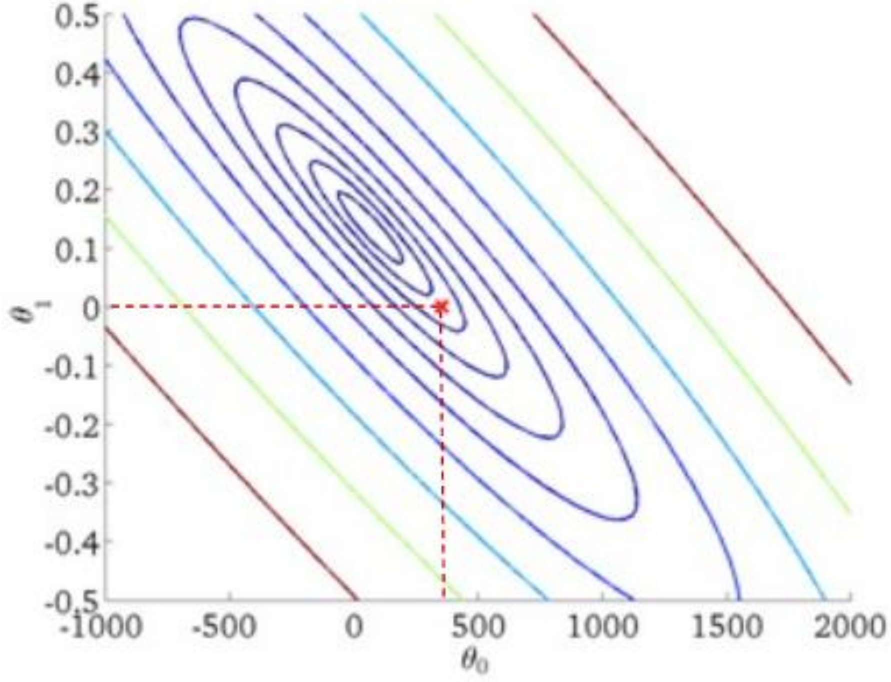
$h_\theta(x)$

(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)

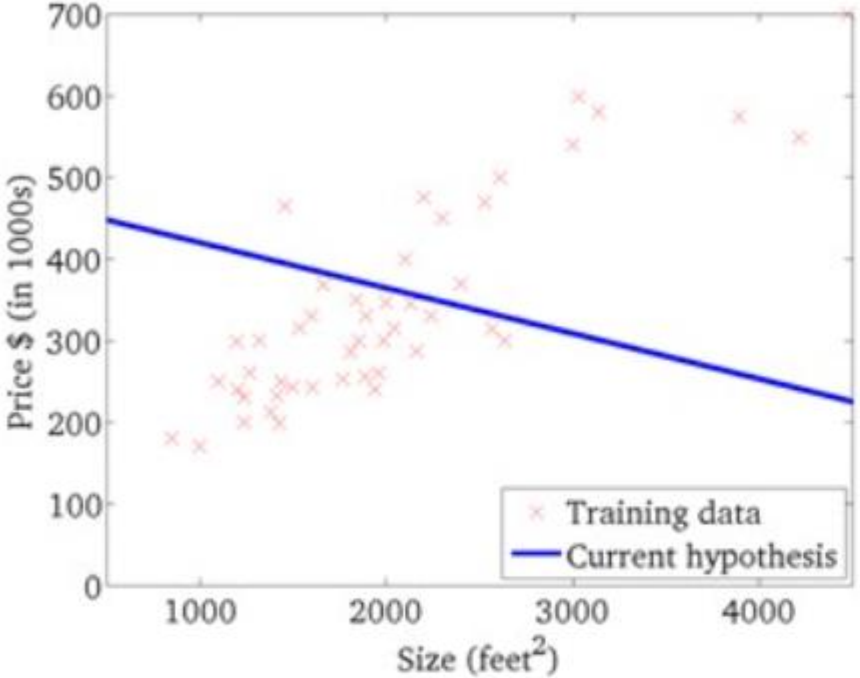


$\Theta_0 = 360$

$\Theta_1 = 0$

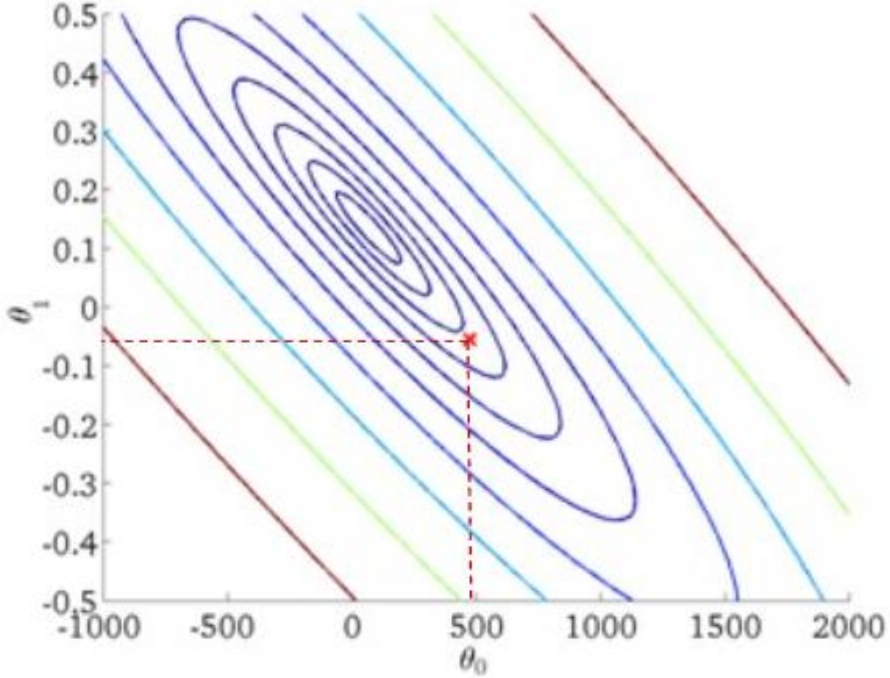
$$h_\theta(x)$$

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



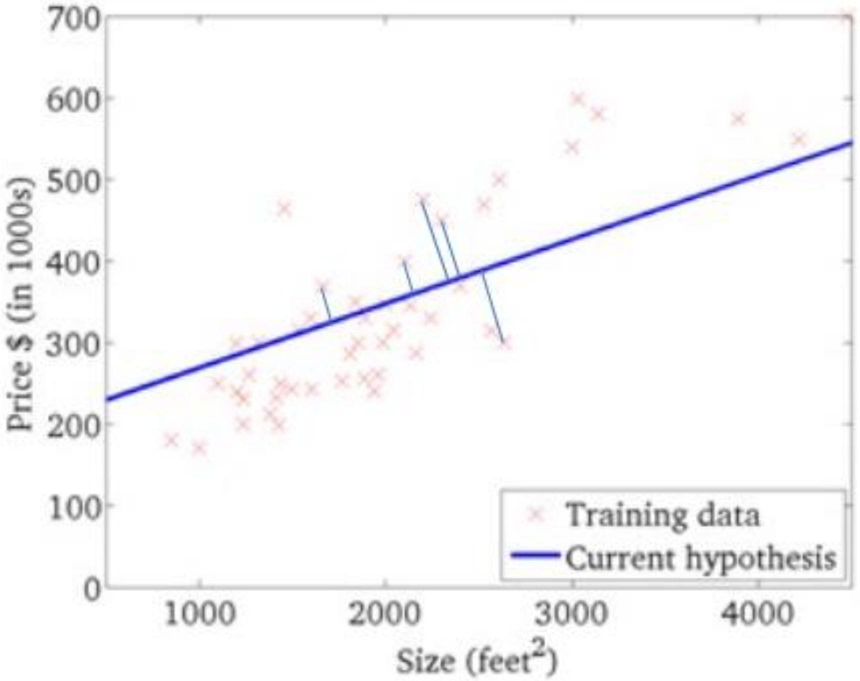
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



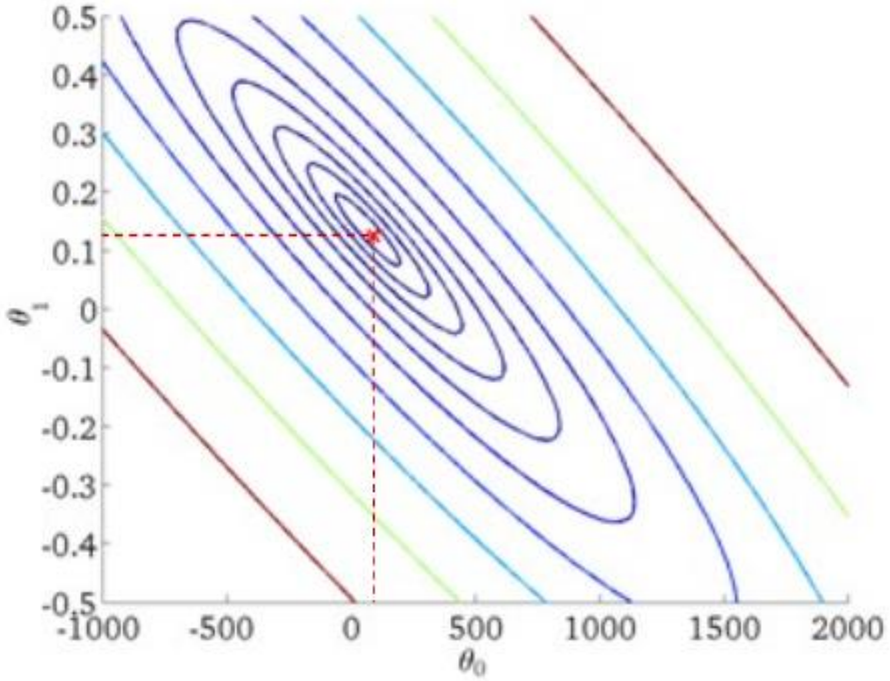
$h_\theta(x)$

(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)



$\Longrightarrow$ line fits the data perfectly

$\rightarrow$ Loss is minimum.

Linear Regression

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

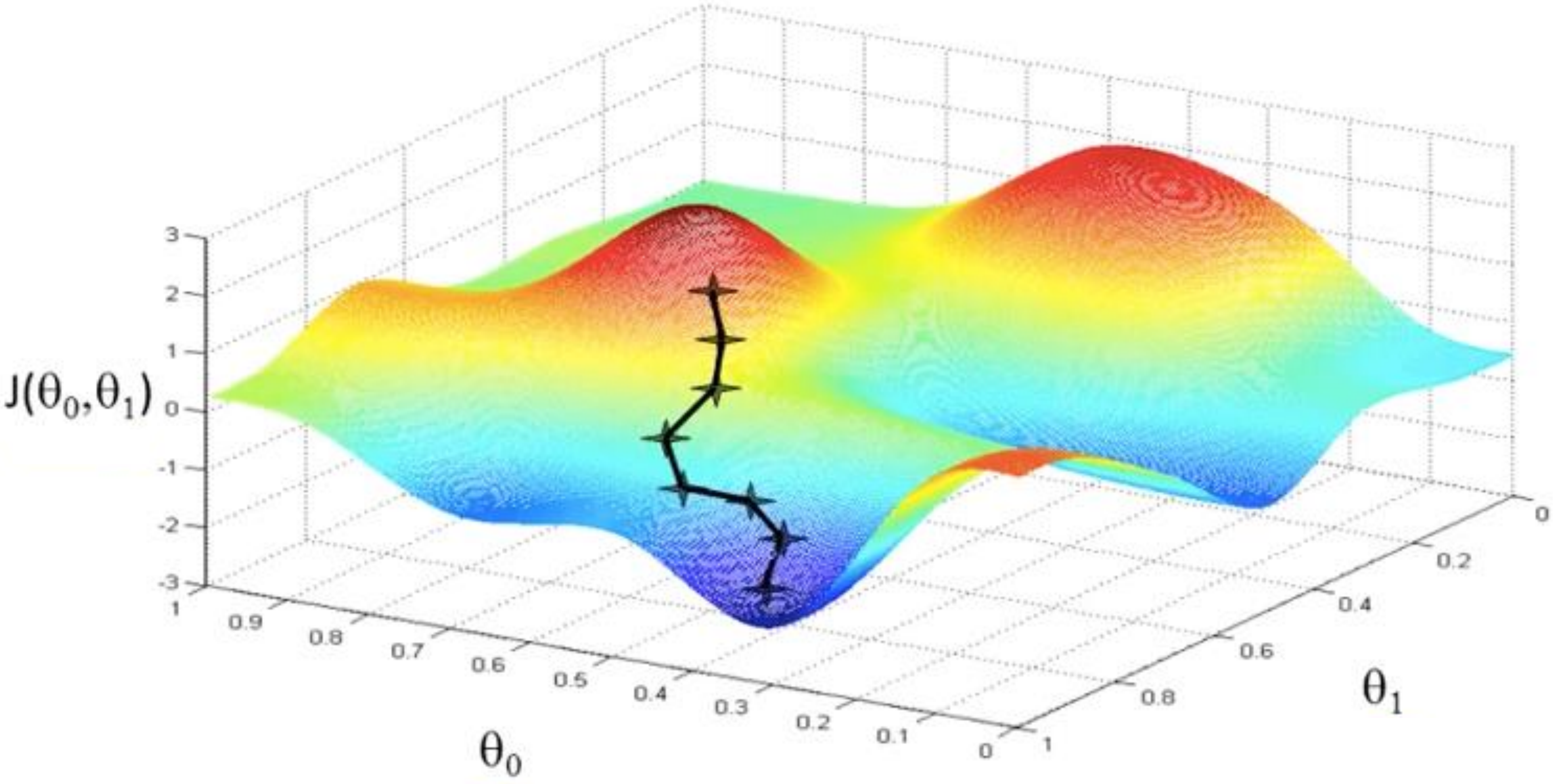
Gradient descent

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(θ0,θ1)
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
θ0
0
0.2
0.4
0.6
0.8
1
θ1

Gradient descent

Repeat until convergence{

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

}

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:

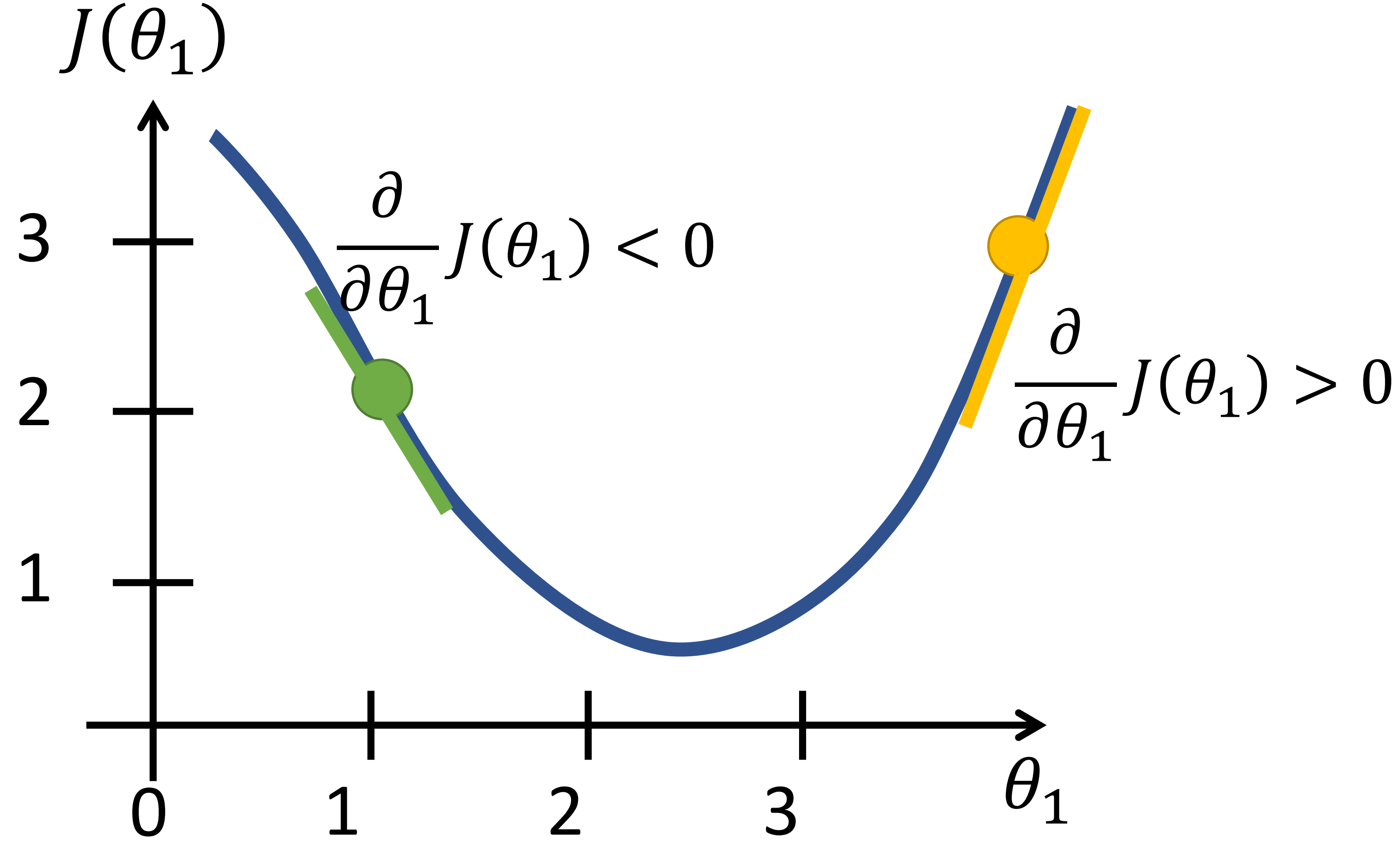
$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

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

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

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

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



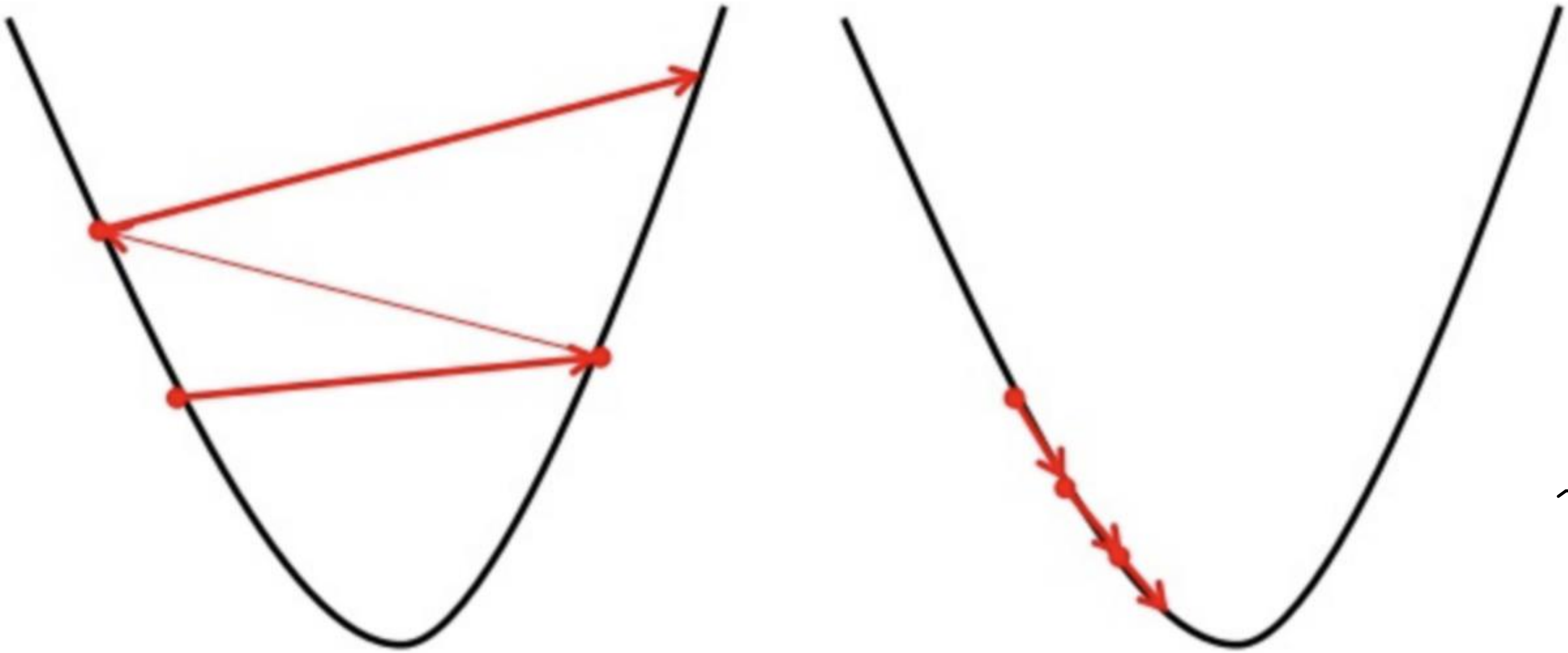
Learning rate

⇒ Very High learning rate
→ does not reach global optimum

→ Very low learning rate
↓
Time taken will be very large. to reach global optimum.

Big learning rate

Small 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

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

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

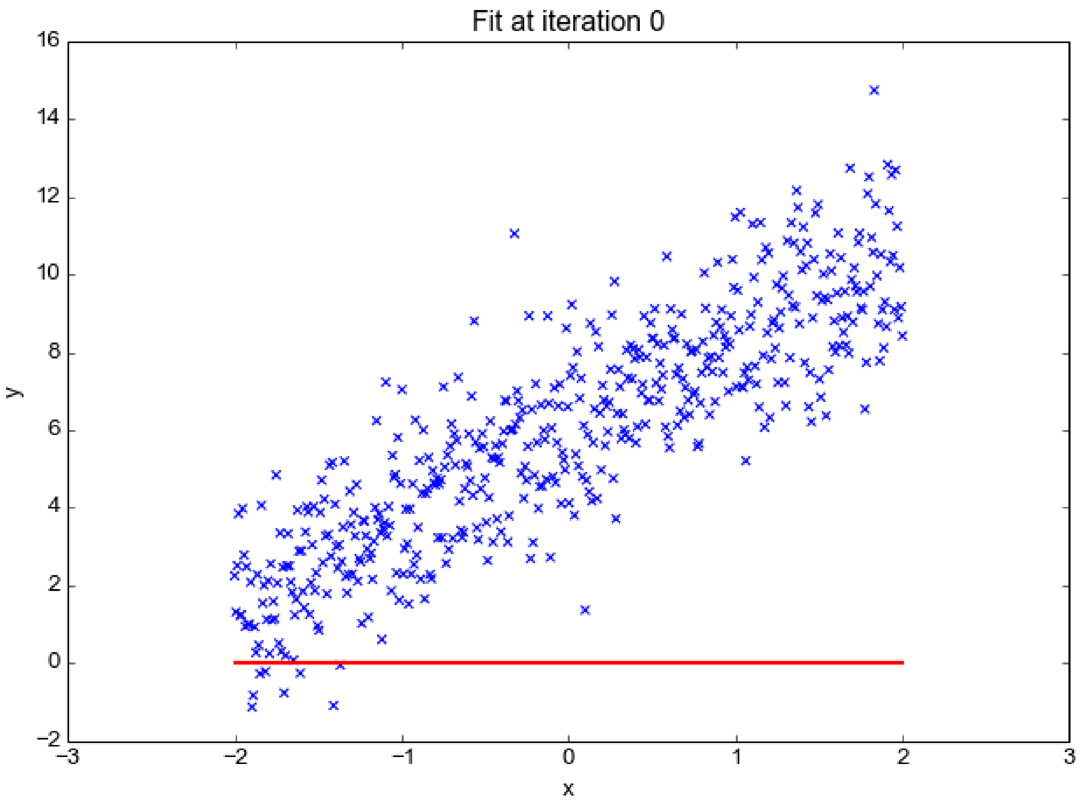
Gradient descent for linear regression

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)

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

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

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{

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

}

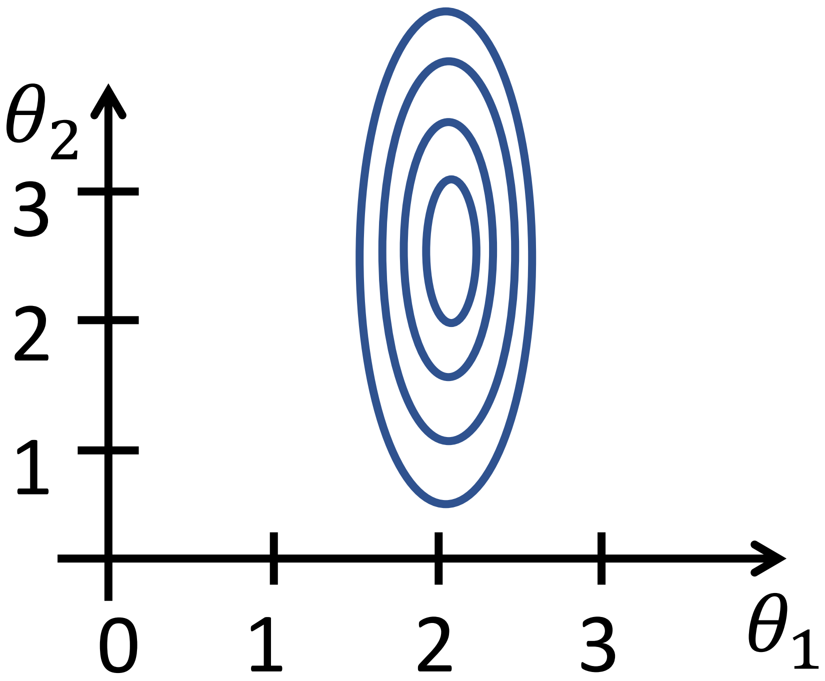
Simultaneously update
$\theta_j, \text{for } j = 0, 1, \cdots, n$

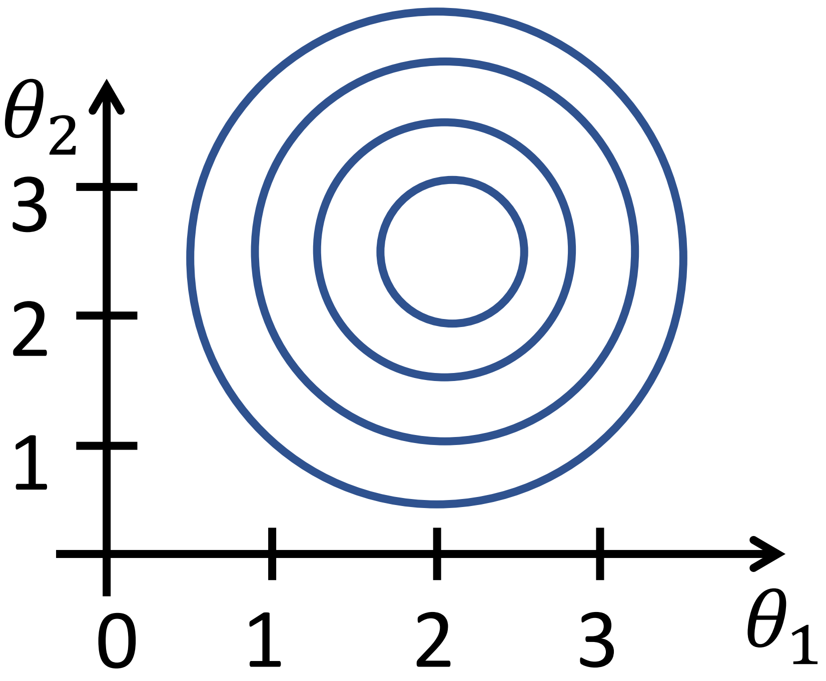
Gradient descent in practice: Feature scaling

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

  x_2 = number of bedrooms (1-5)

⇒ All features equally weighted in Gradient Descent due to feature scaling.





Question

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

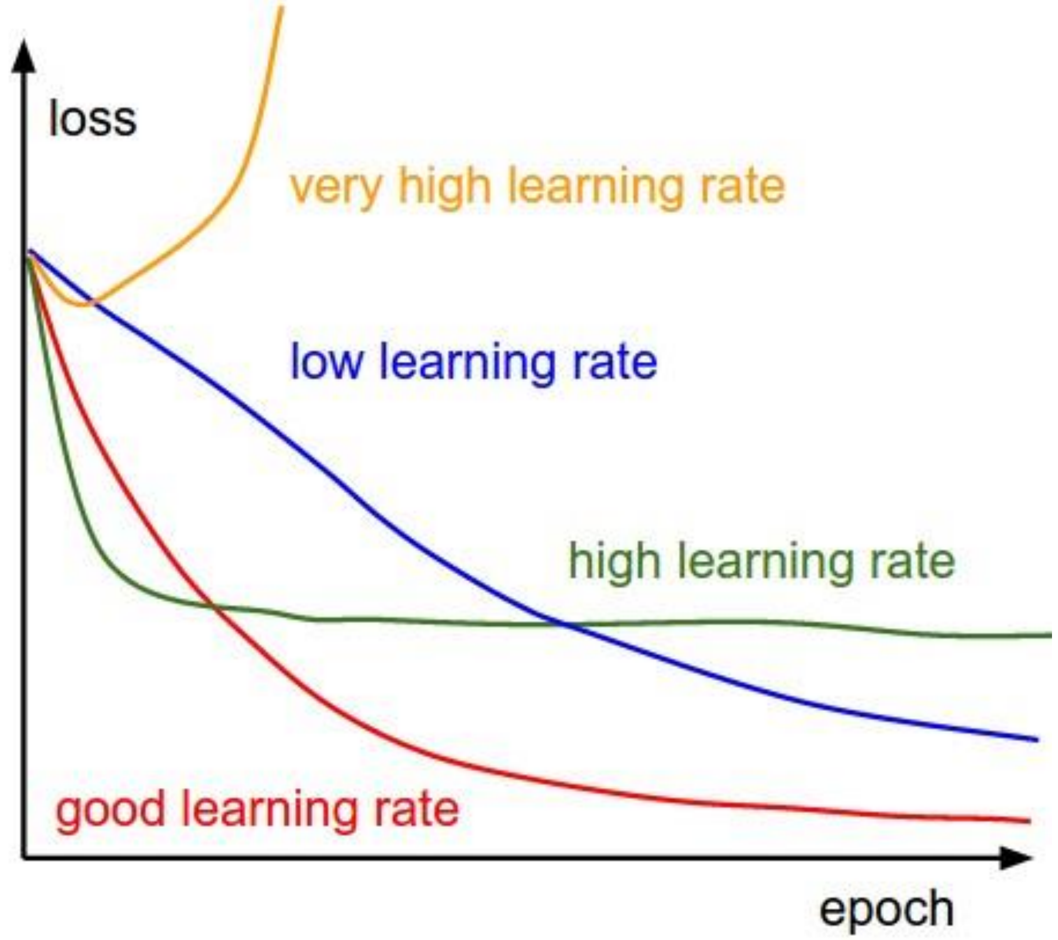
What is the normalized feature x2(4)? → Use Min-max Scalar.

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

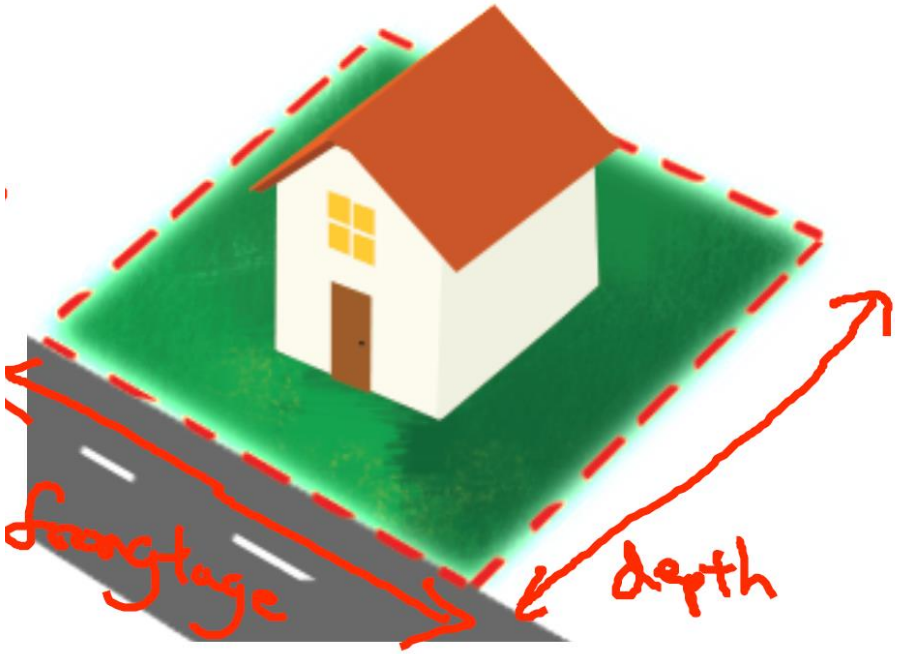


House prices prediction

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

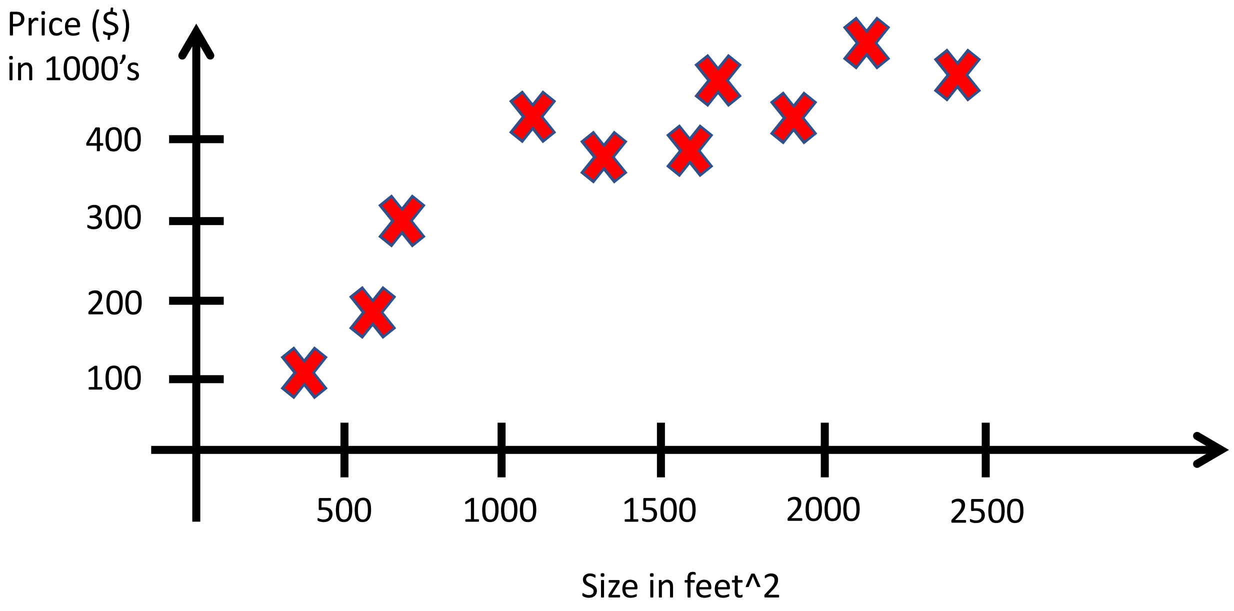
- Area

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



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

Polynomial regression



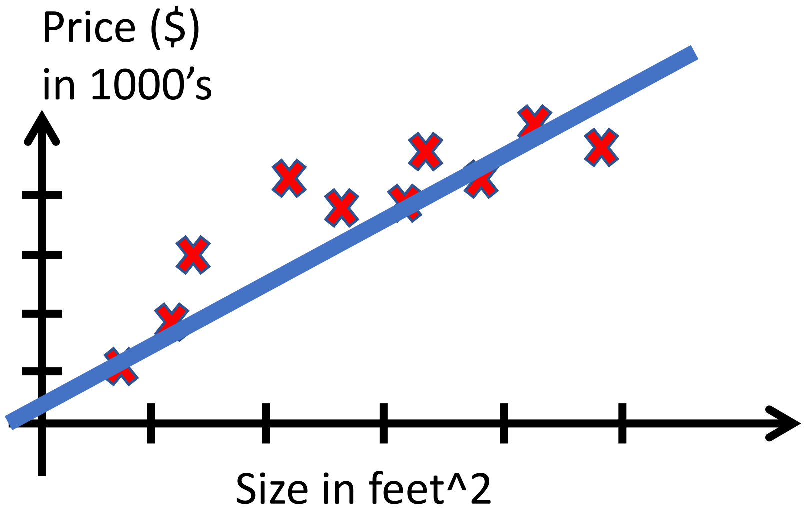
x_1 = (size)
x_2 = (size)^2
x_3 = (size)^3

- $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$
  $= \theta_0 + \theta_1(size) + \theta_2(size)^2 + \theta_3(size)^3$

Regularization

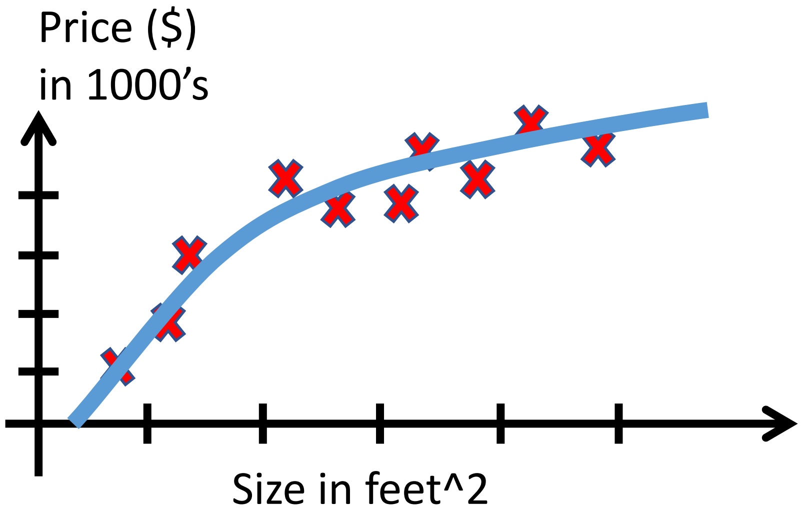
- **Overfitting**
- Cost function
- Regularized linear regression

Example: Linear regression



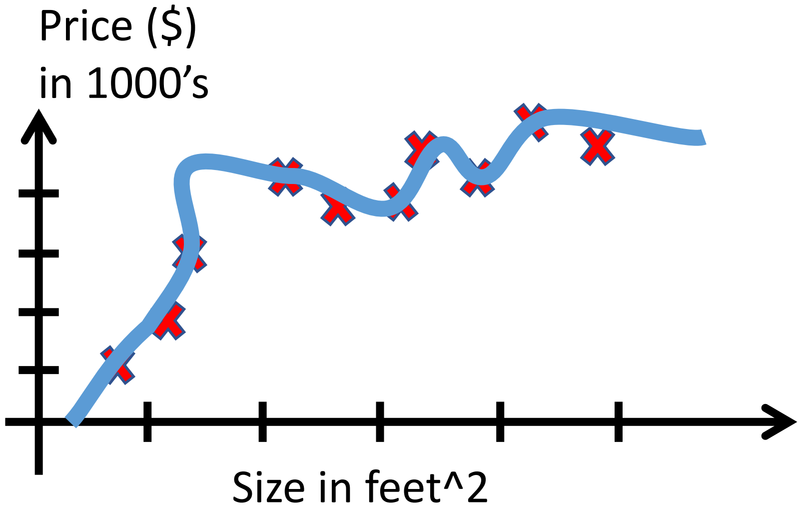
$$h_\theta(x) = \theta_0 + \theta_1 x$$

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

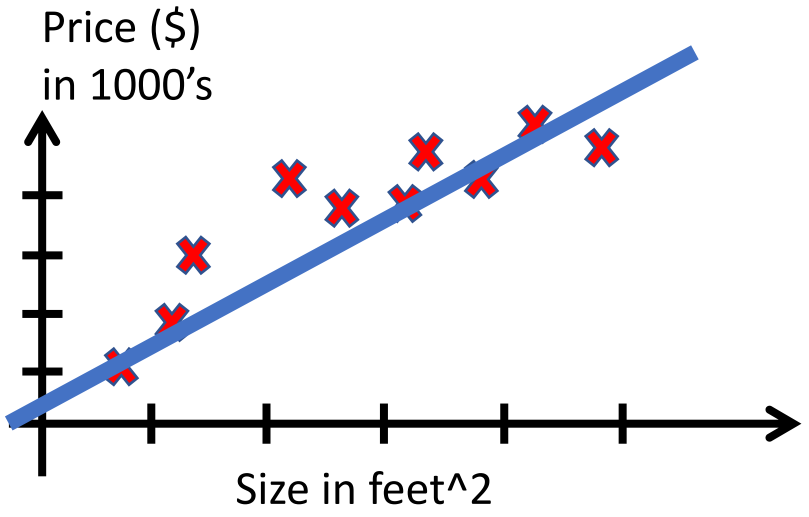
Overfitting

- 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).

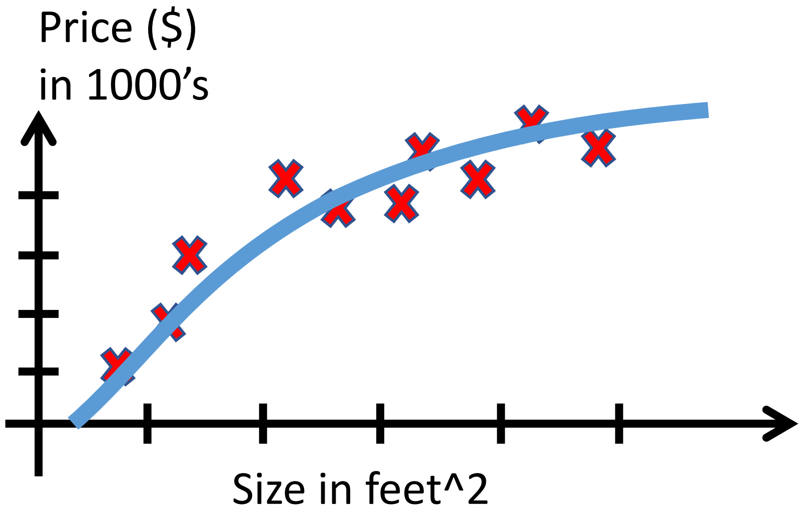
Example: Linear regression



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

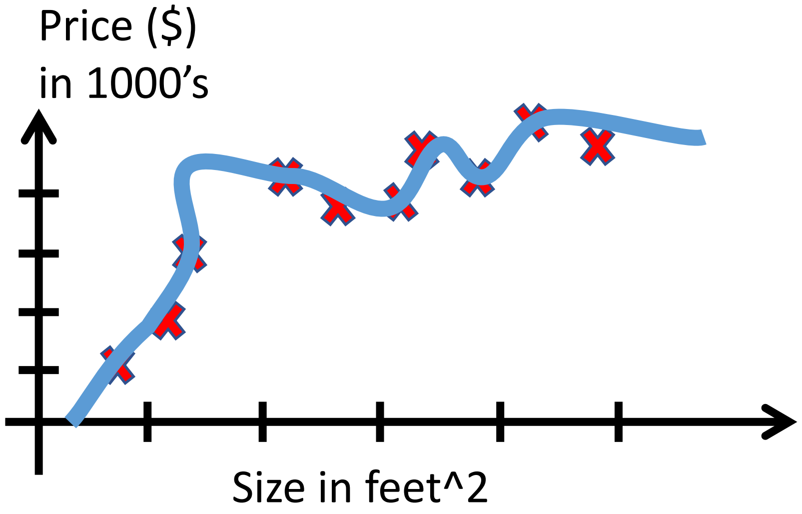
Underfitting

High bias



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

Just right



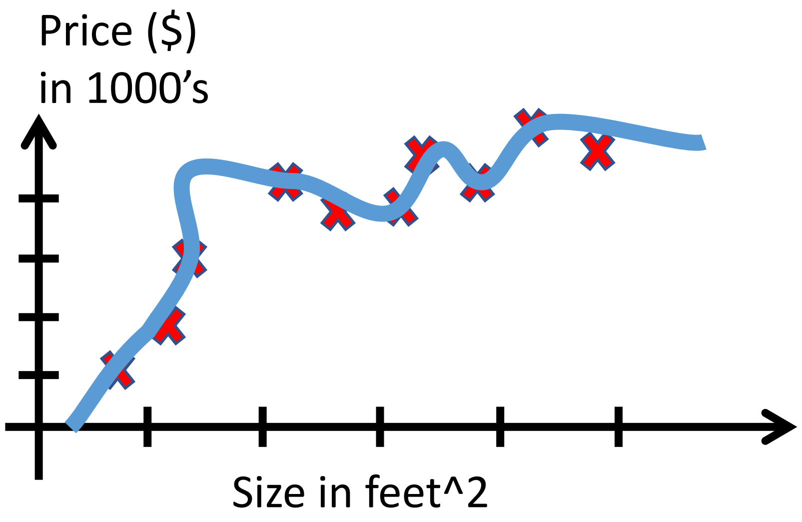
$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \cdots$$

Overfitting

High 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.

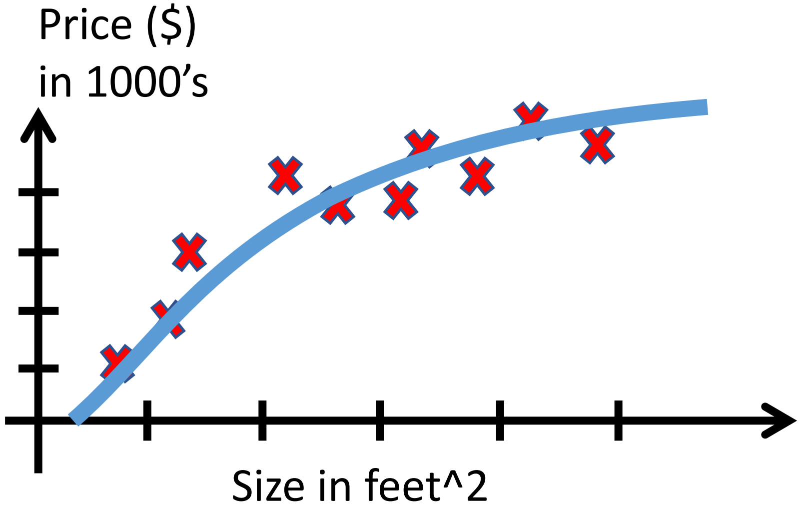
Regularization

- Overfitting

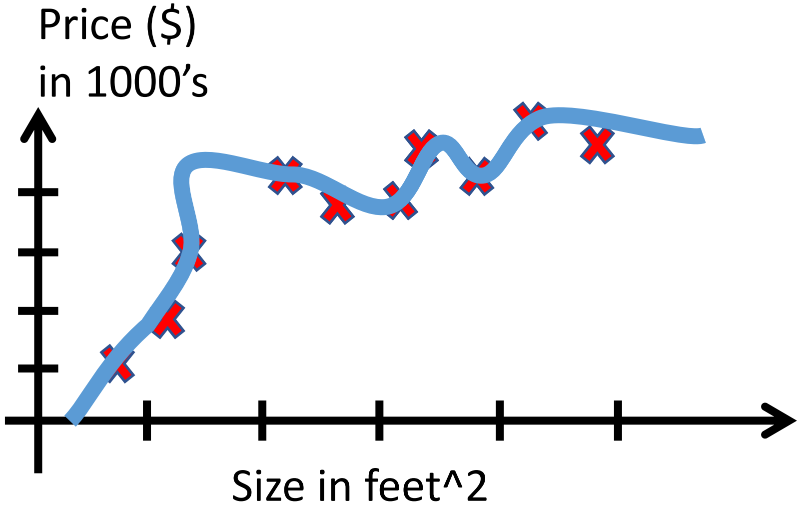
- **Cost function**

- 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: $\theta_0, \theta_1, \theta_2, \cdots, \theta_{100}$

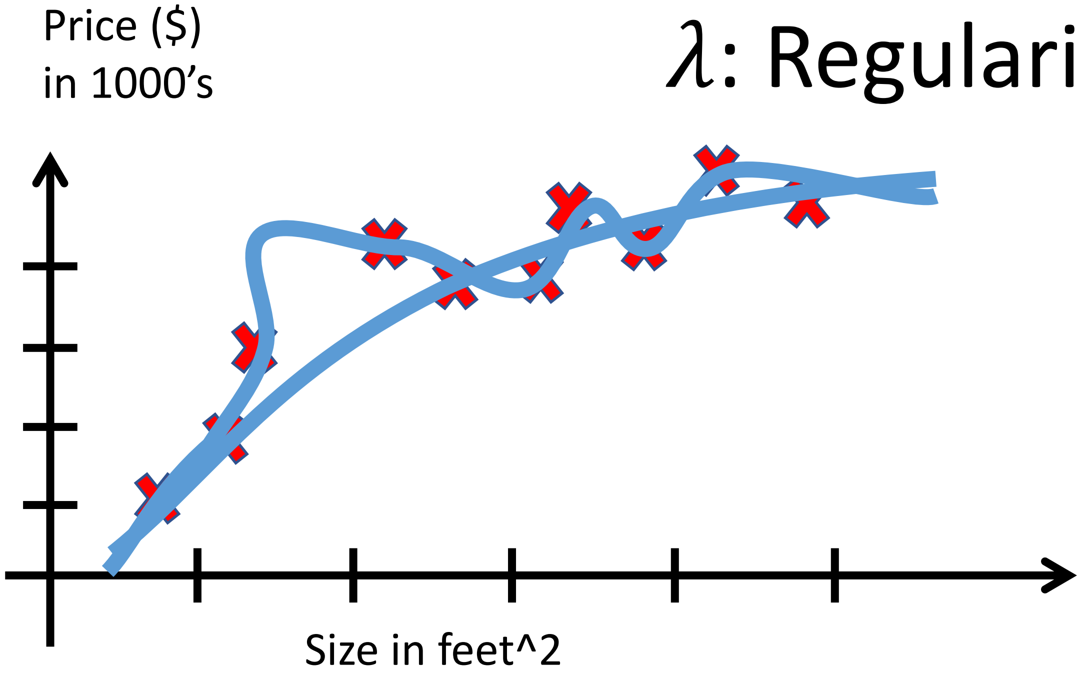
$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right]$$

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

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

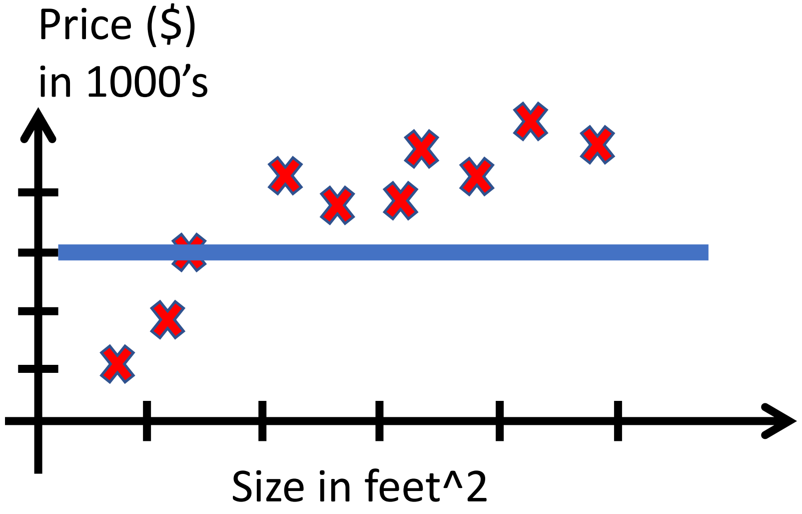
What if λ is set to an extremely large value (say $\lambda = 10^{10}$)?

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 {

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

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

}

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

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

Regularized linear regression

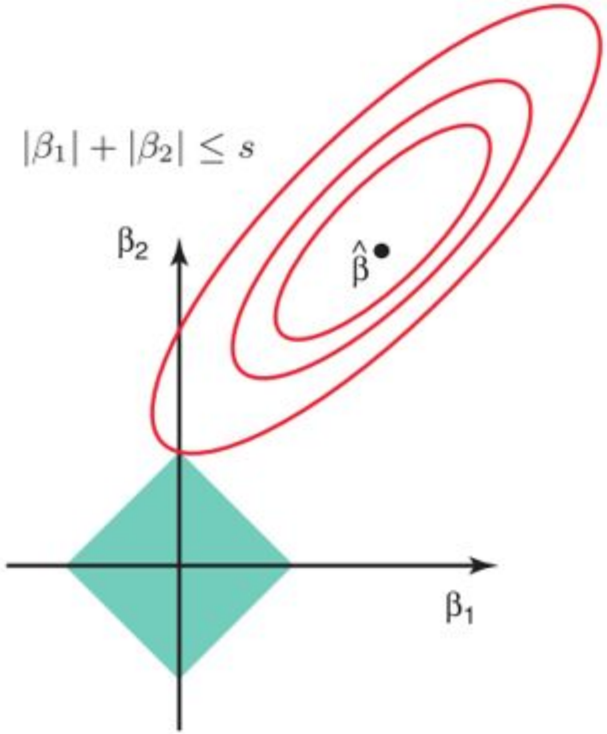
$$\theta_j := \theta_j\left(1 - \alpha\frac{\lambda}{m}\right) - \alpha\frac{1}{m}\left[\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}\right]$$

Un-regularized linear regression

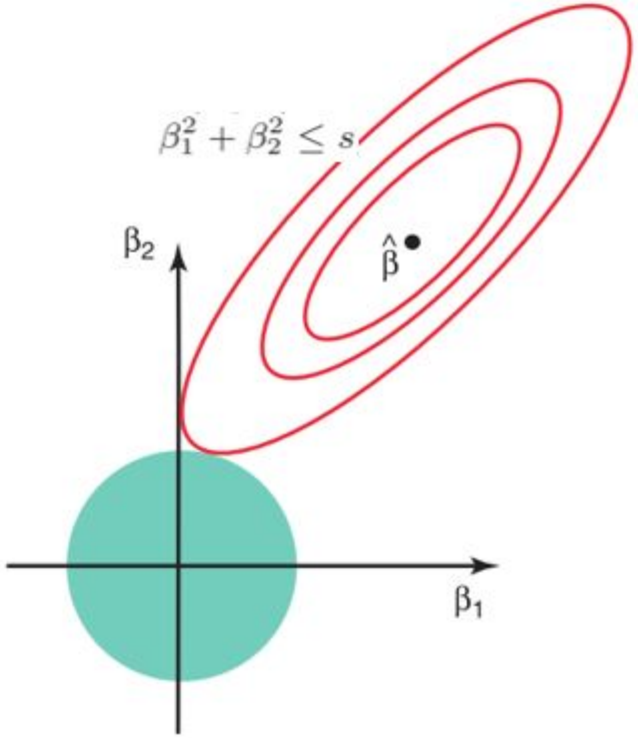
$$\theta_j := \theta_j - \alpha\frac{1}{m}\left[\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}\right]$$

Linear Regression

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



Lasso Regression



Ridge Regression

Particularly, regularization is implemented to avoid overfitting of the data, especially when there is a large variance between train and test set performances. With regularization, the number of features used in training is kept constant, yet the magnitude of the coefficients (w) is reduced.

$\Rightarrow$ Two Types of Regularization.

1) Ridge Regularization

$$L = \frac{1}{2m} \sum_{i=1}^{m} (h(x_i) - y_i)^2 + \lambda \sum_{i=1}^{j} \theta_j^2$$

2) Lasso Regularization

$$L = \frac{1}{2m} \sum_{i=1}^{m} (h(x_i) - y_i)^2 + \lambda \sum_{i=1}^{j} |\theta_j|$$

⇒ Here,

Both the Regularization are used to reduce the issue of overfitting

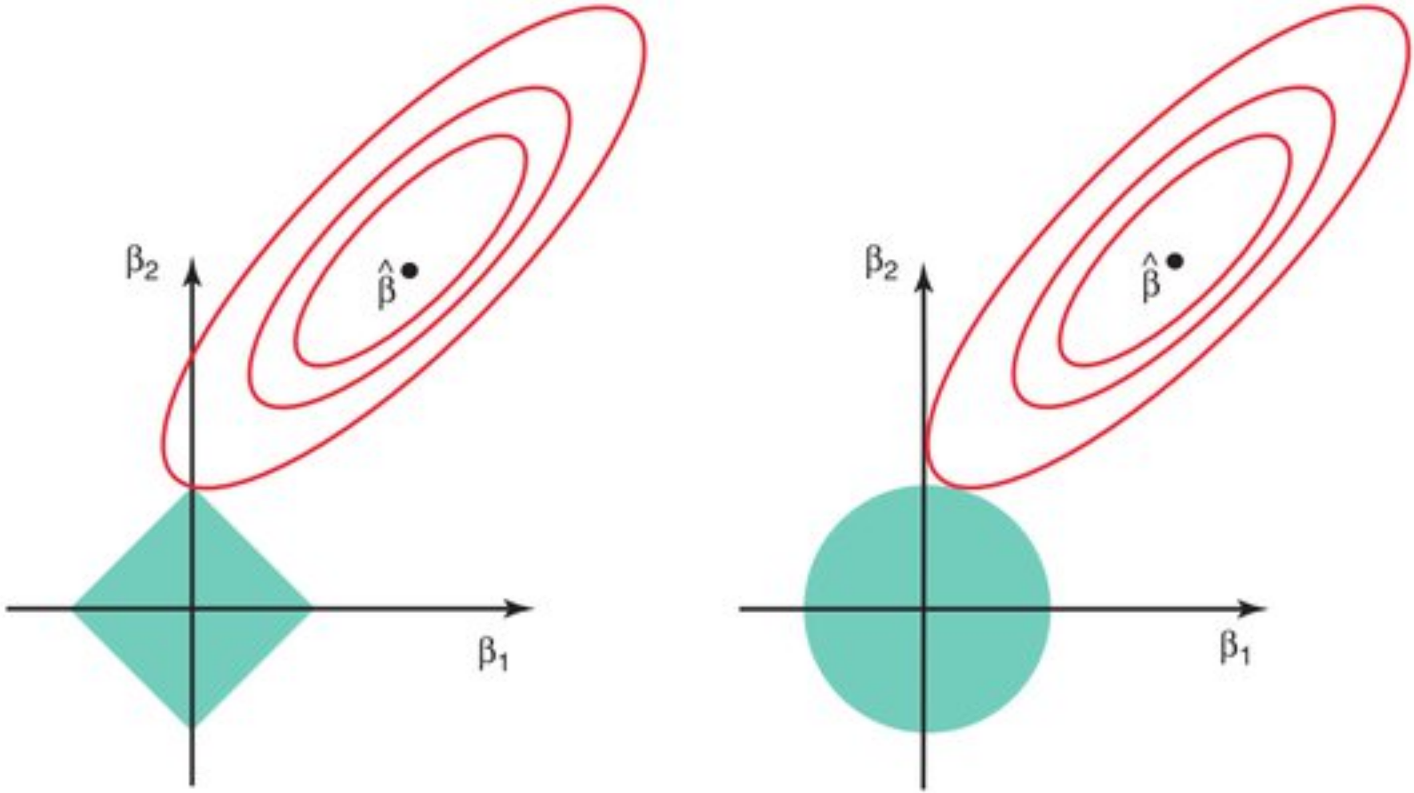


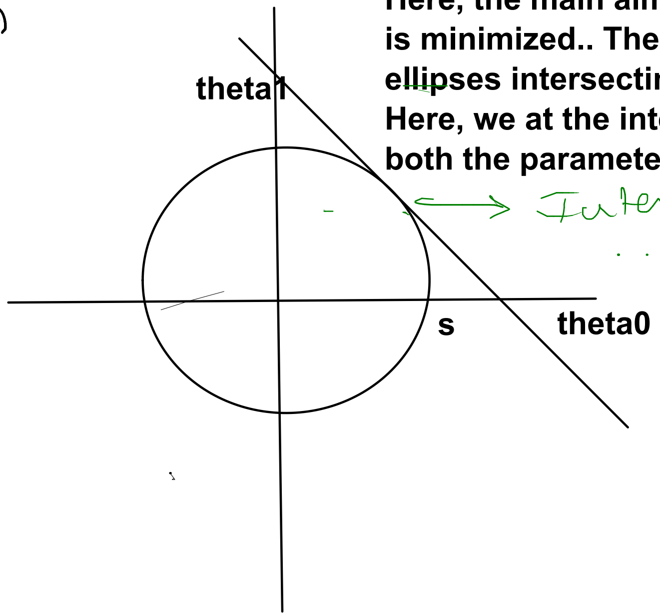
FIGURE 6.7. *Contours of the error and constraint functions for the lasso* (left) *and ridge regression* (right). *The solid blue areas are the constraint regions, $|\beta_1| + |\beta_2| \leq s$ and $\beta_1^2 + \beta_2^2 \leq s$, while the red ellipses are the contours of the RSS.*

⇒ The above graph, we are trying to find values of parameters such that loss is minimized and overfitting is removed. We have to find values of parameters where both the contours intersect each other.

⇒ Lasso → will find sparse solutions.

→ So, it can be used to ~~per~~ perform feature selection.

$\Rightarrow$

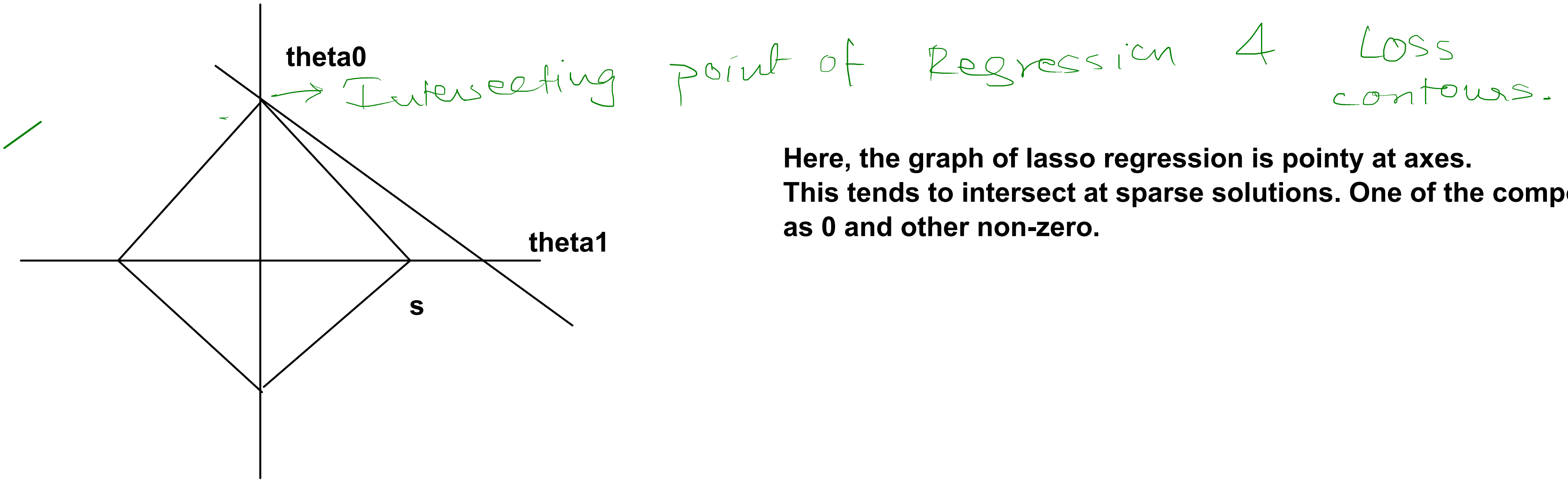


Here, the main aim is to find the value of parameters theta0 and theta1 such that the loss is minimized.. The elliptical contours can be considered as line when they are large ellipses intersecting the equation of Ridge Regression graph.
Here, we at the interesection point we have non-zero value for both the parameters.

Intersection point of Ridge Regression & Loss contours.

Considering the geometry of both the lasso (left) and ridge (right) models, the elliptical contours (red circles) are the cost functions for each.

However, both methods determine coefficients by finding the first point where the elliptical contours hit the region of constraints. Since lasso regression takes a diamond shape in the plot for the constrained region, each time the elliptical regions intersect with these corners, at least one of the coefficients becomes zero. This is impossible in the ridge regression model as it forms a circular shape and therefore values can be shrunk close to zero, but never equal to zero.



Here, the graph of lasso regression is pointy at axes.
This tends to intersect at sparse solutions. One of the component
as 0 and other non-zero.

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

References

- https://www.youtube.com/watch?v=76B5cMEZA4Y - Sparsity in Lasso
- https://www.datacamp.com/tutorial/tutorial-lasso-ridge-regression
- https://medium.com/@mukulranjan/how-does-lasso-regression-l1-encourage-zero-coefficients-but-not-the-l2-20e4893cba5d#:~:text=From%20both%20of%20the%20above,a%20method%20of%20variable%20selection.

- https://www.youtube.com/watch?v=3xXInc1Bf0w&list=PLKnIA16_Rmvat8luHBnlE68KvqSuurAR4