Regression: A Numerical Optimization Use Case

Alexander G. Ororbia II

COGS-621: Foundations of Scientific Computing

11/11/2025 and 11/13/2025

So...why might we want to use derivatives, gradient-based hill-climbing, and cost functions?

Linear regression

- **Model representation**
- Cost function
- Gradient descent
- Multivariate regression

Regression

real-valued output

Training set

Learning Algorithm

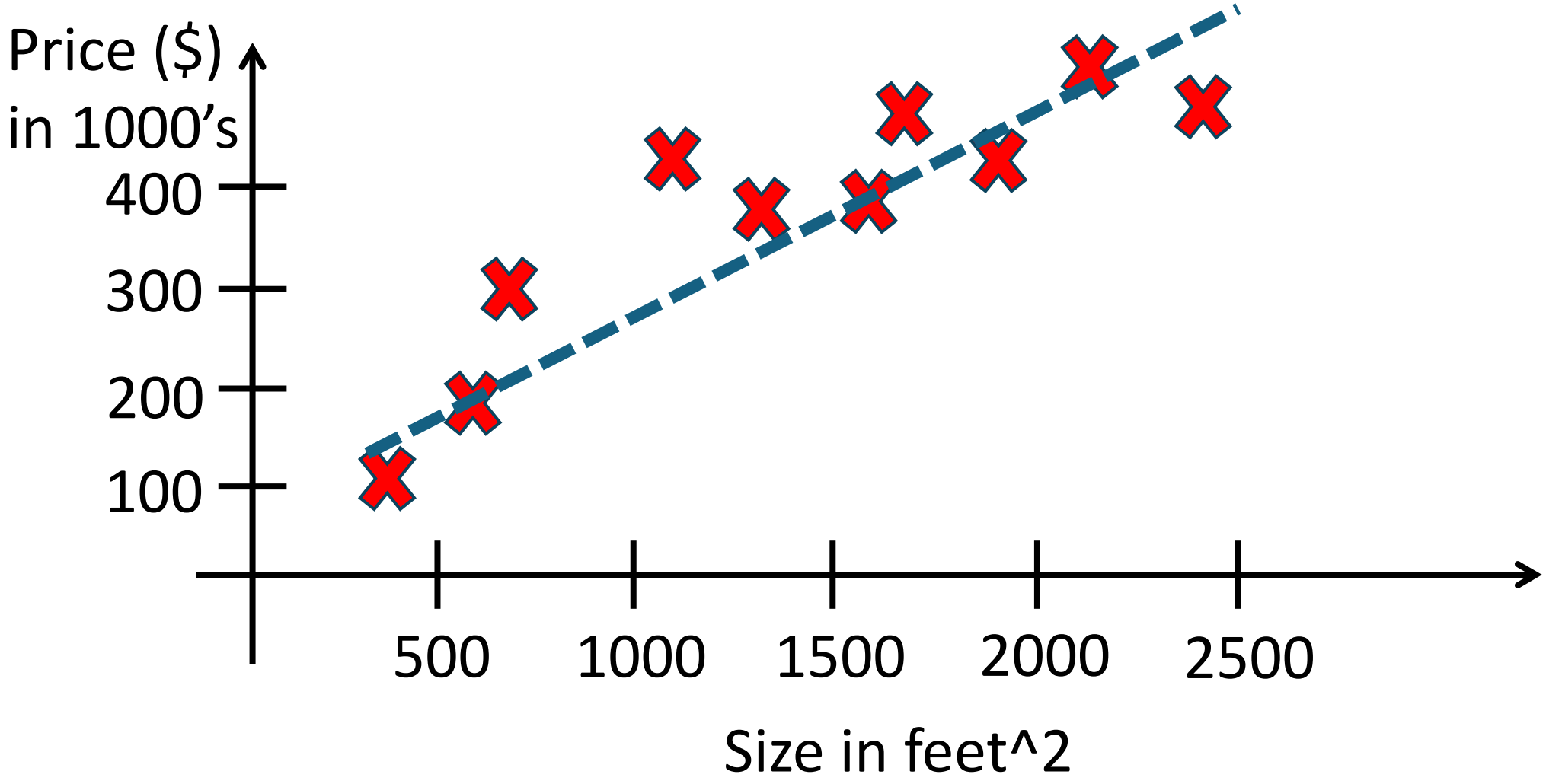
x → h → y

Size of house

Hypothesis

Estimated price

House pricing prediction



Training dataset

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:

$x^{(1)} = 2104$

$x^{(2)} = 1416$

$y^{(1)} = 460$

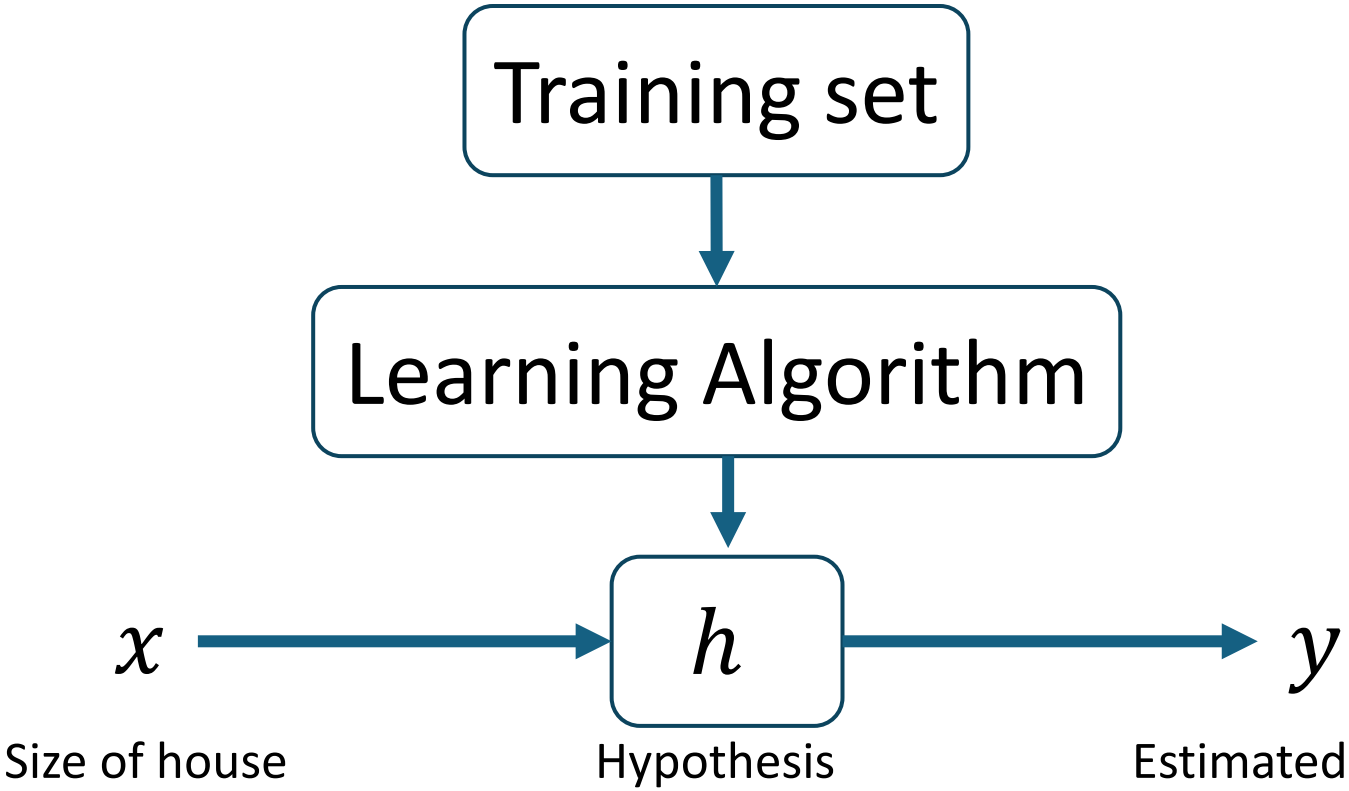
Model design / representation

Remember....

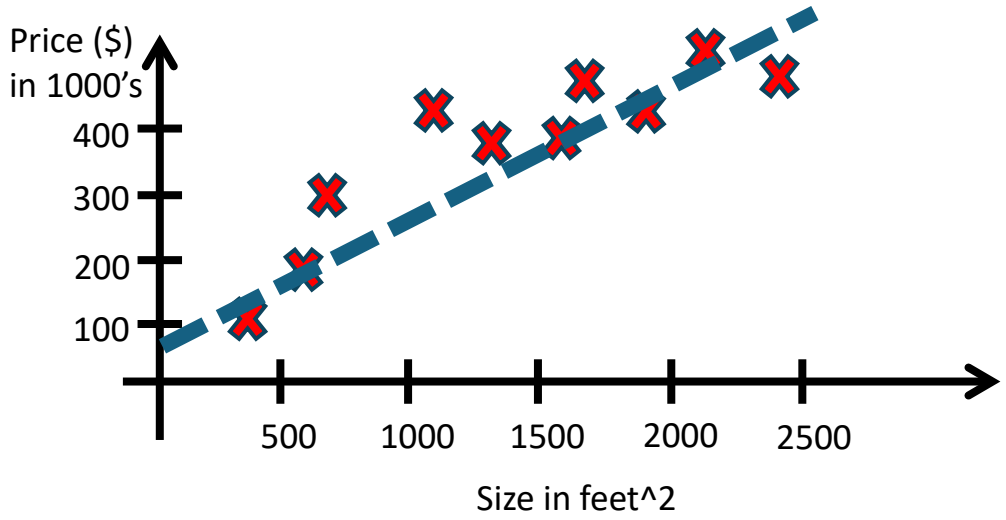
Homework #1 Model:

$$b + Wx$$

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



Shorthand $h(x)$



Univariate linear regression

Linear regression

- Model representation
- **Cost function**
- Gradient descent
- Multivariate regression

Training dataset

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

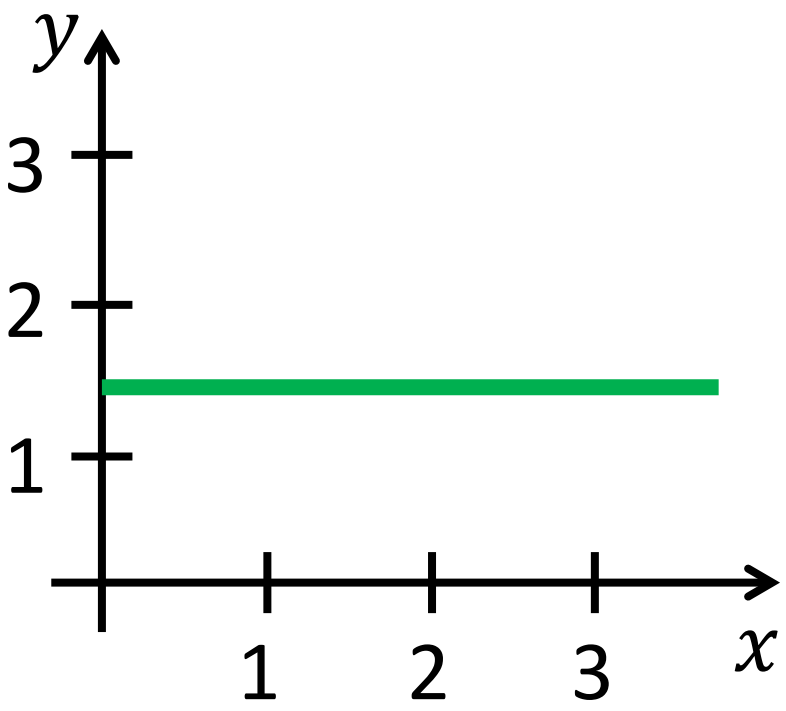
$m = 47$

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

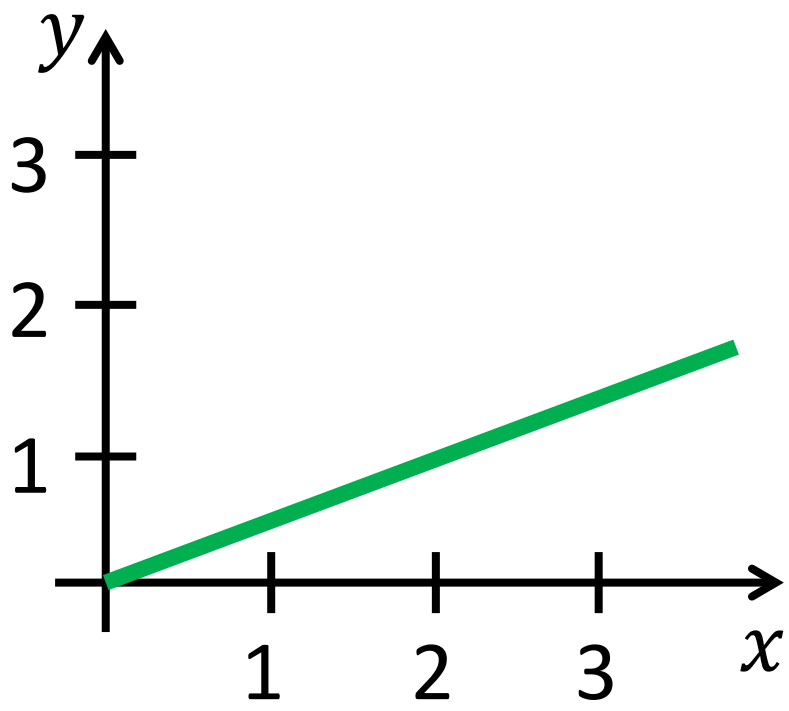
θ_0, θ_1: parameters/weights

How to choose θ_i's?

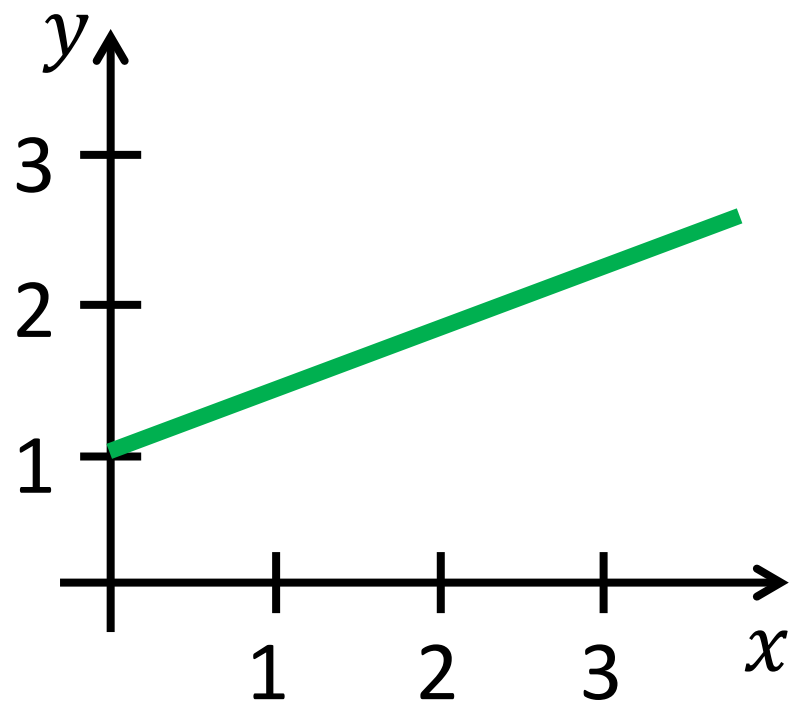
$$h_\theta(x) = \theta_0 + \theta_1 x$$



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



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



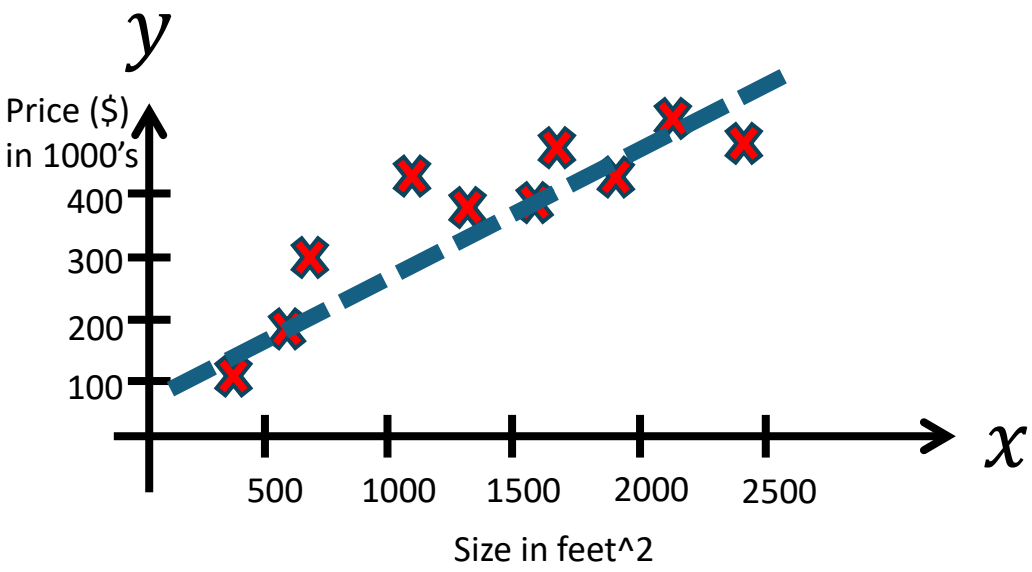
$\theta_0 = 1$
$\theta_1 = 0.5$

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:**

$$\theta_1$$

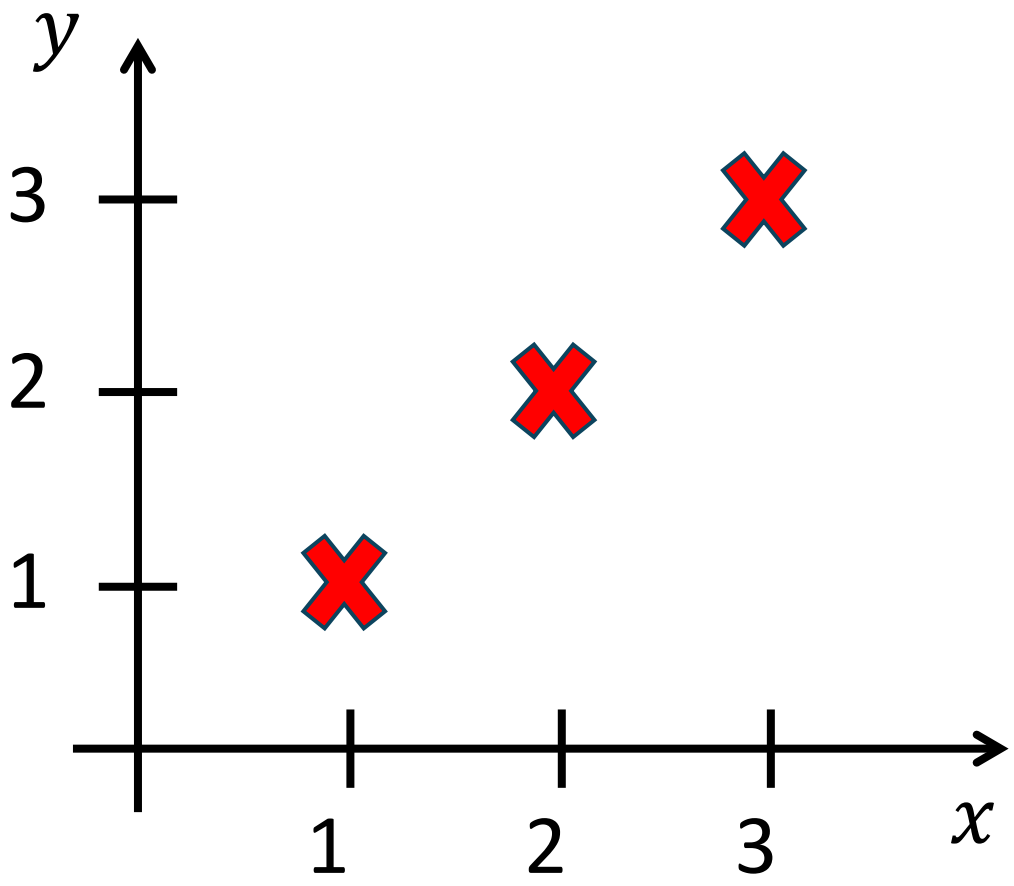
- **Cost function:**

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

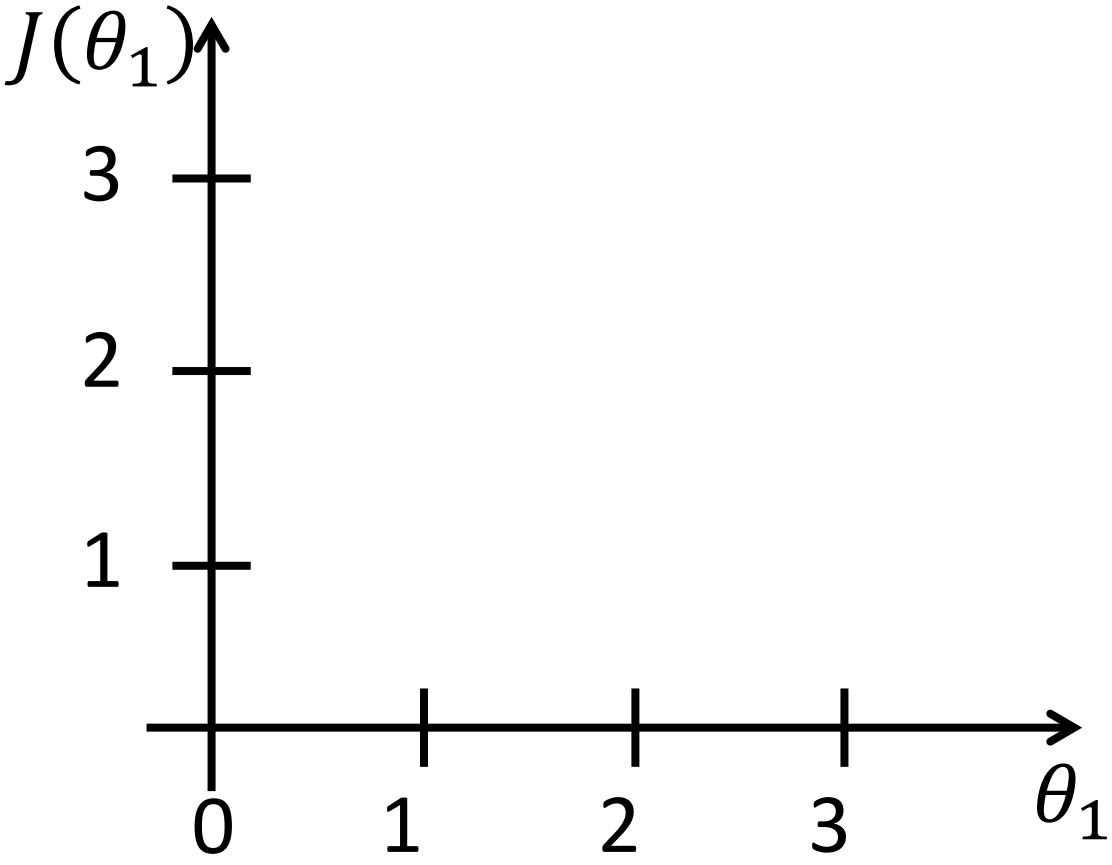
- **Goal**:

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

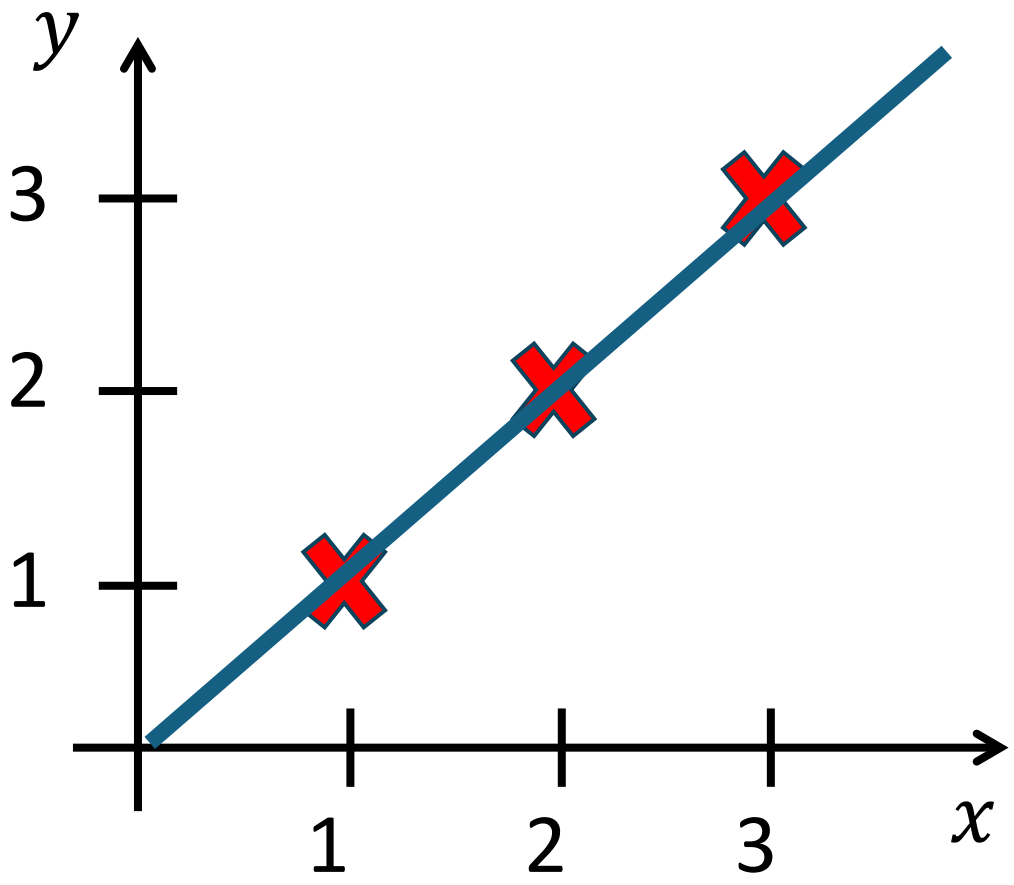
$h_\theta(x)$, function of x



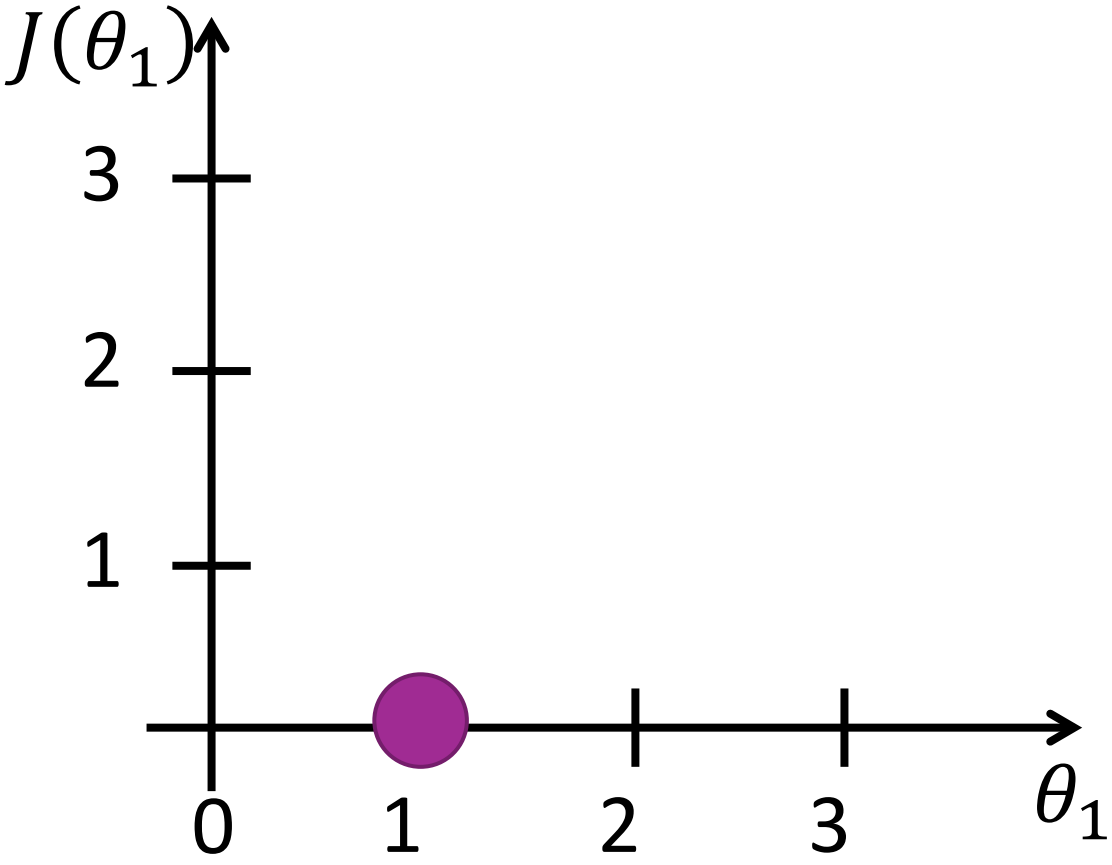
$J(\theta_1)$, function of θ_1



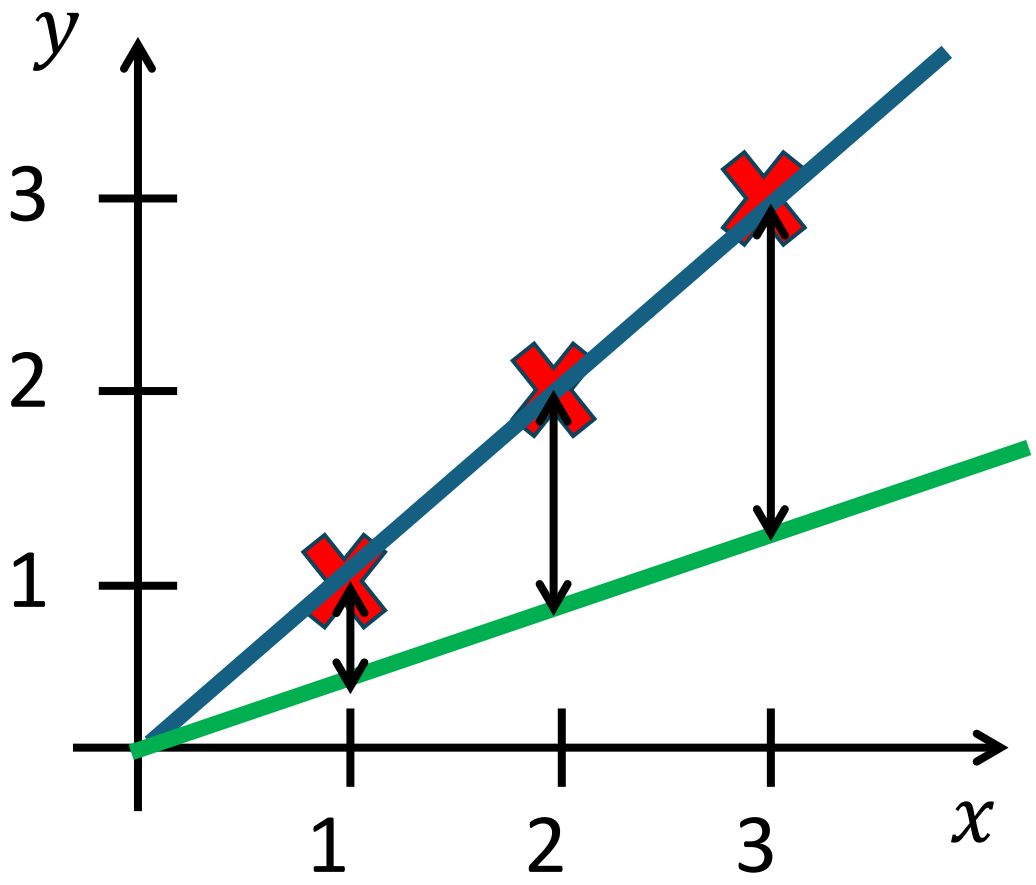
$h_\theta(x)$, function of x



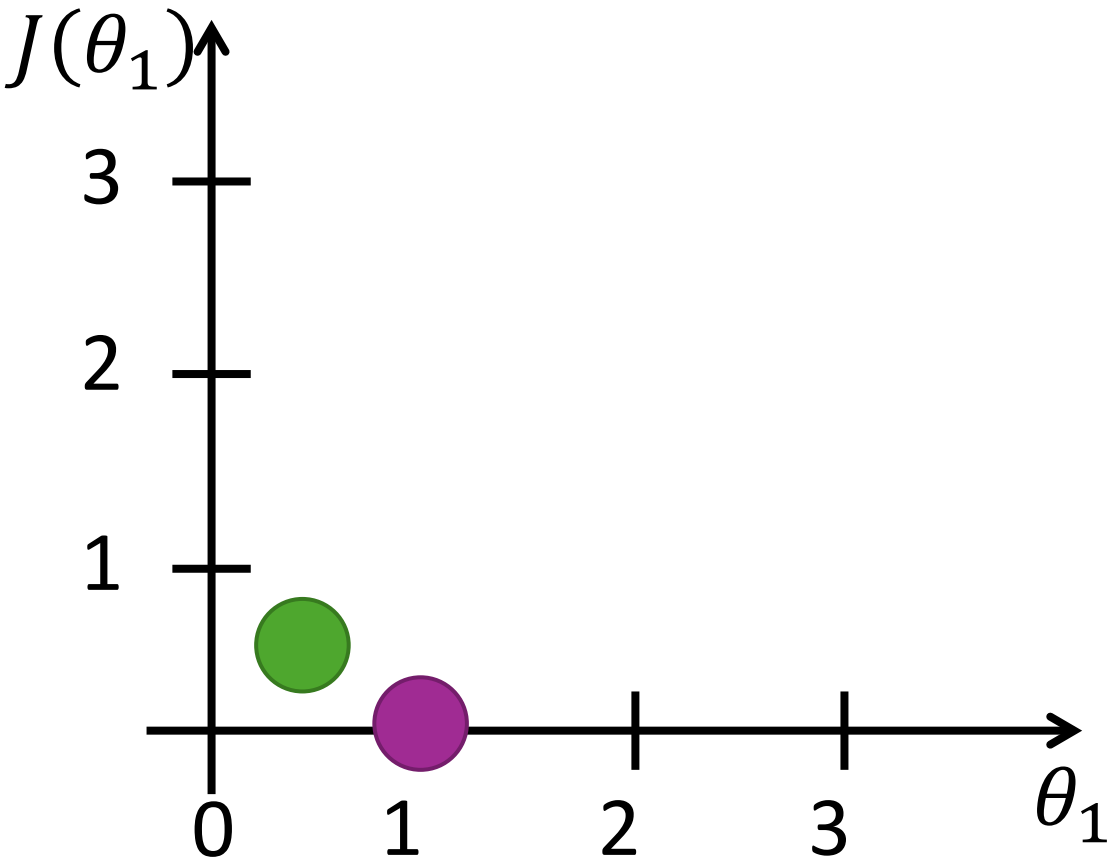
$J(\theta_1)$, function of θ_1



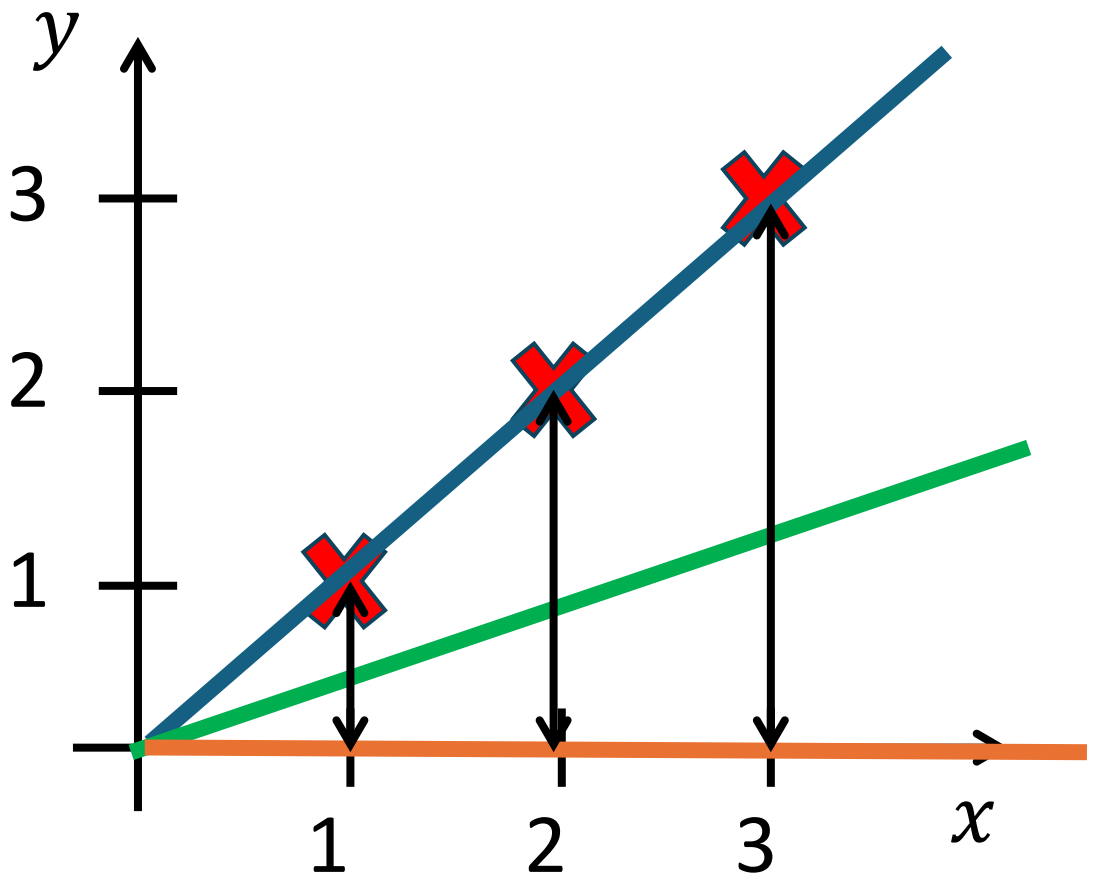
$h_\theta(x)$, function of x



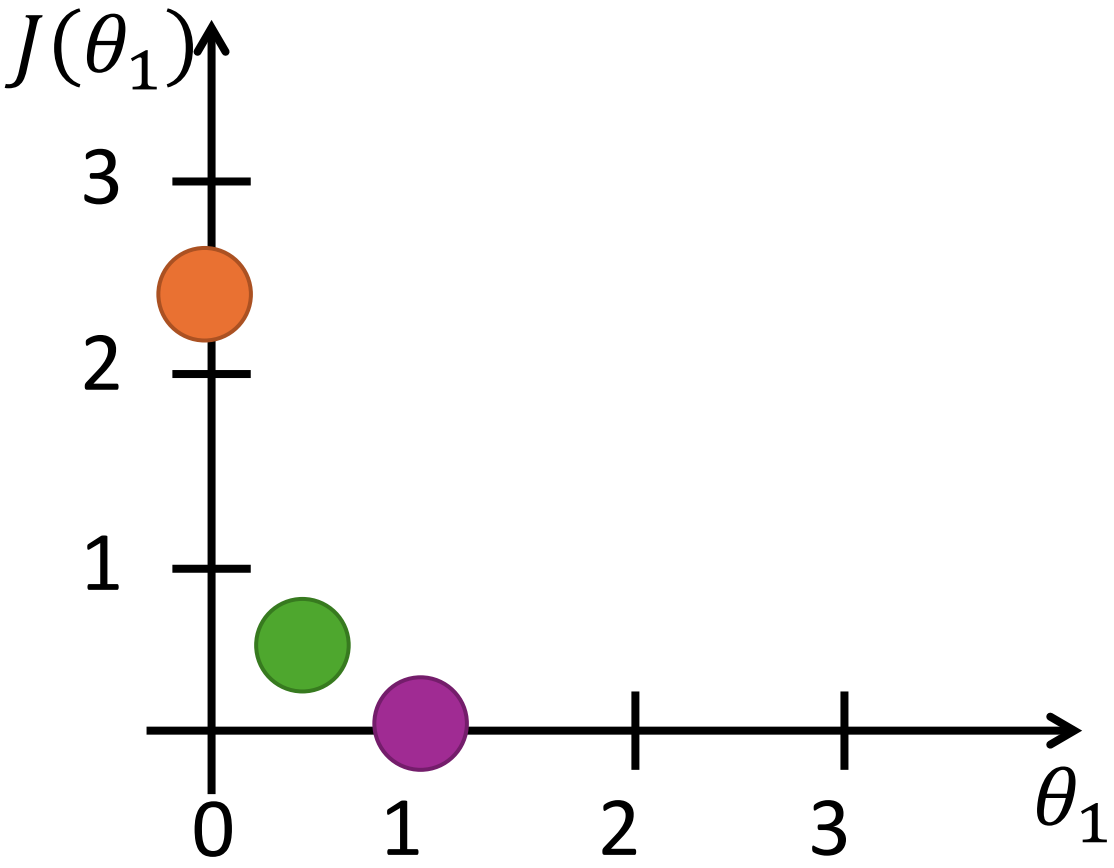
$J(\theta_1)$, function of θ_1



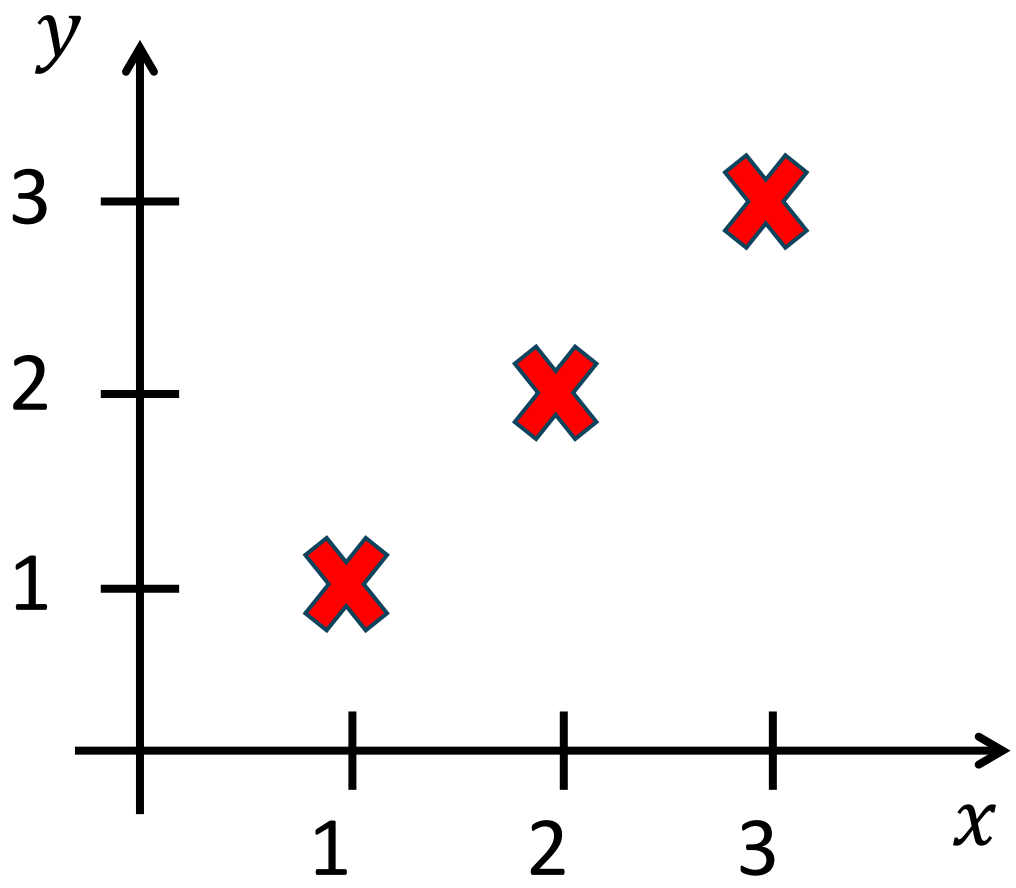
$h_\theta(x)$, function of x



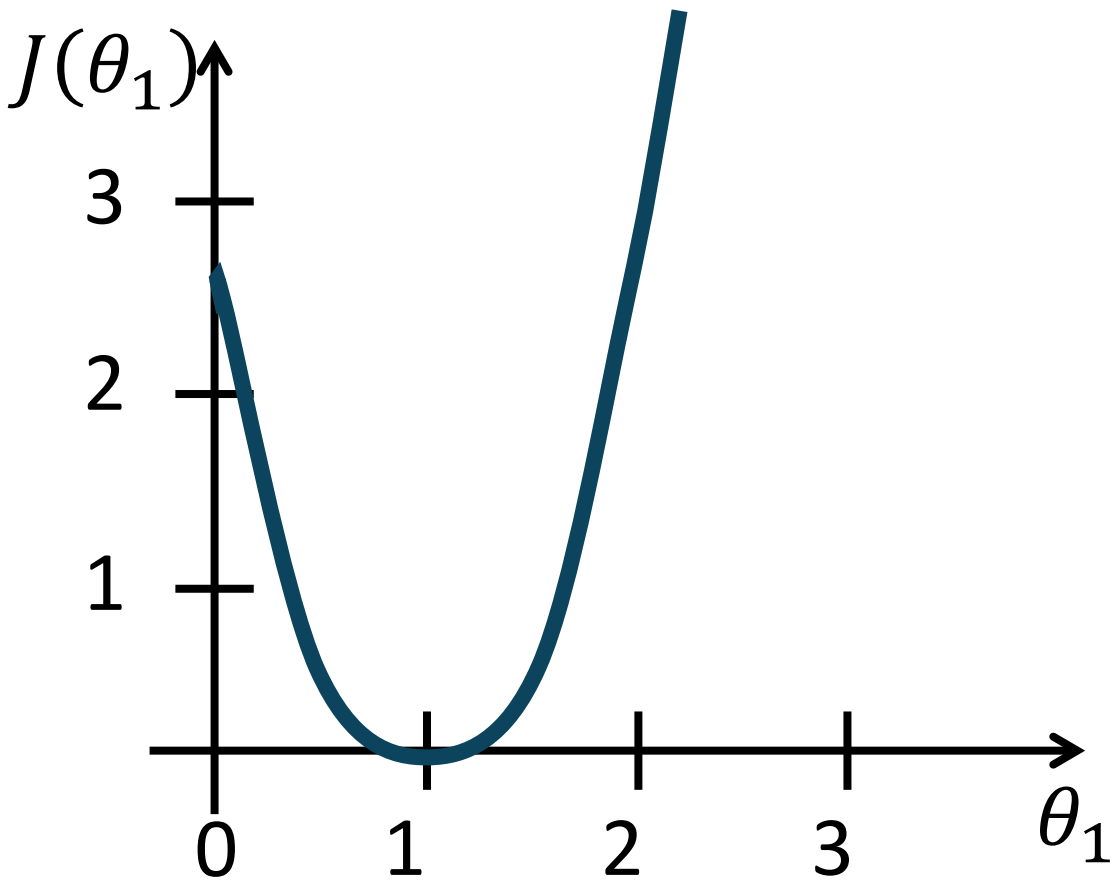
$J(\theta_1)$, function of θ_1



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

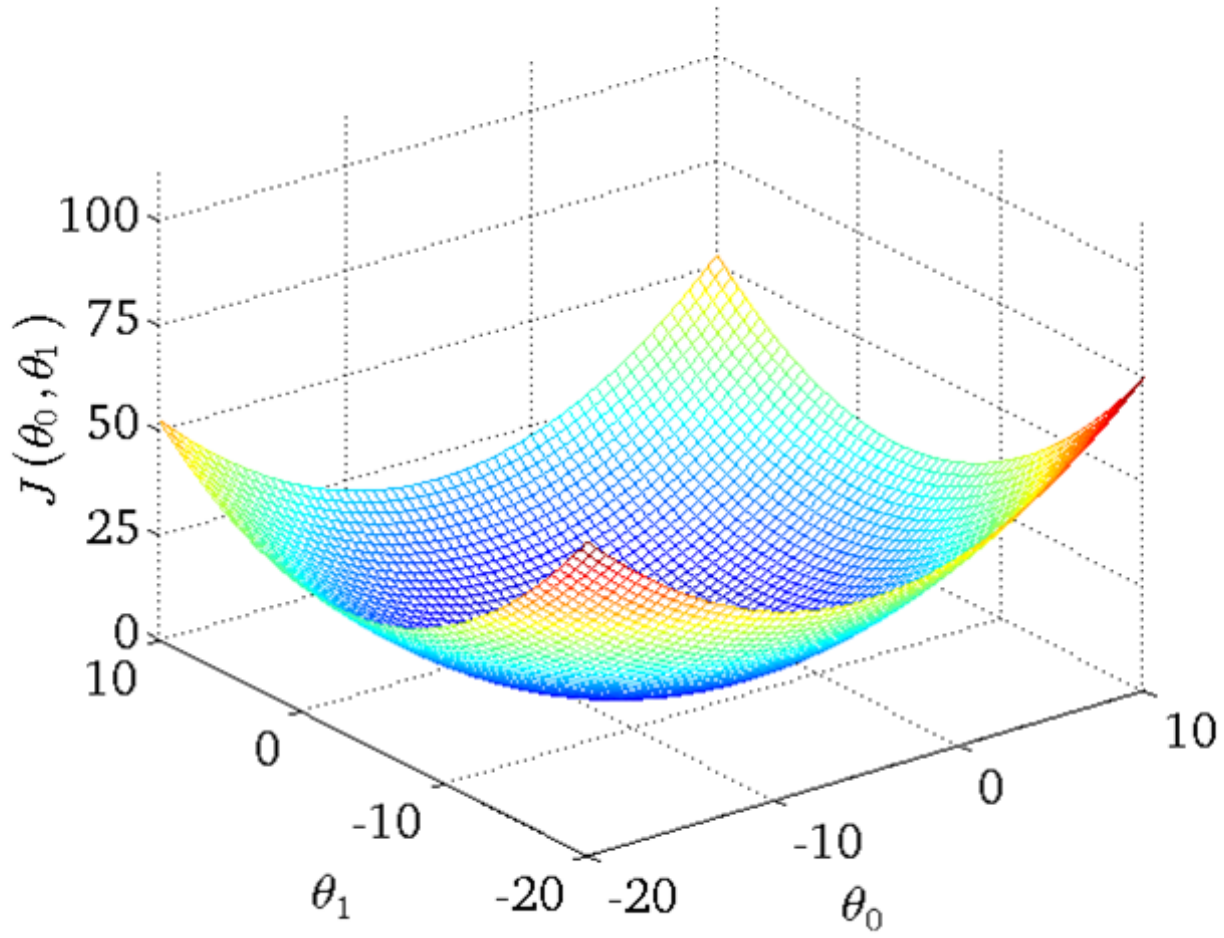


$J(\theta_1)$, function of θ_1



- **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 surface



Linear regression

- Model representation
- Cost function
- **Gradient descent**
- Multivariate regression

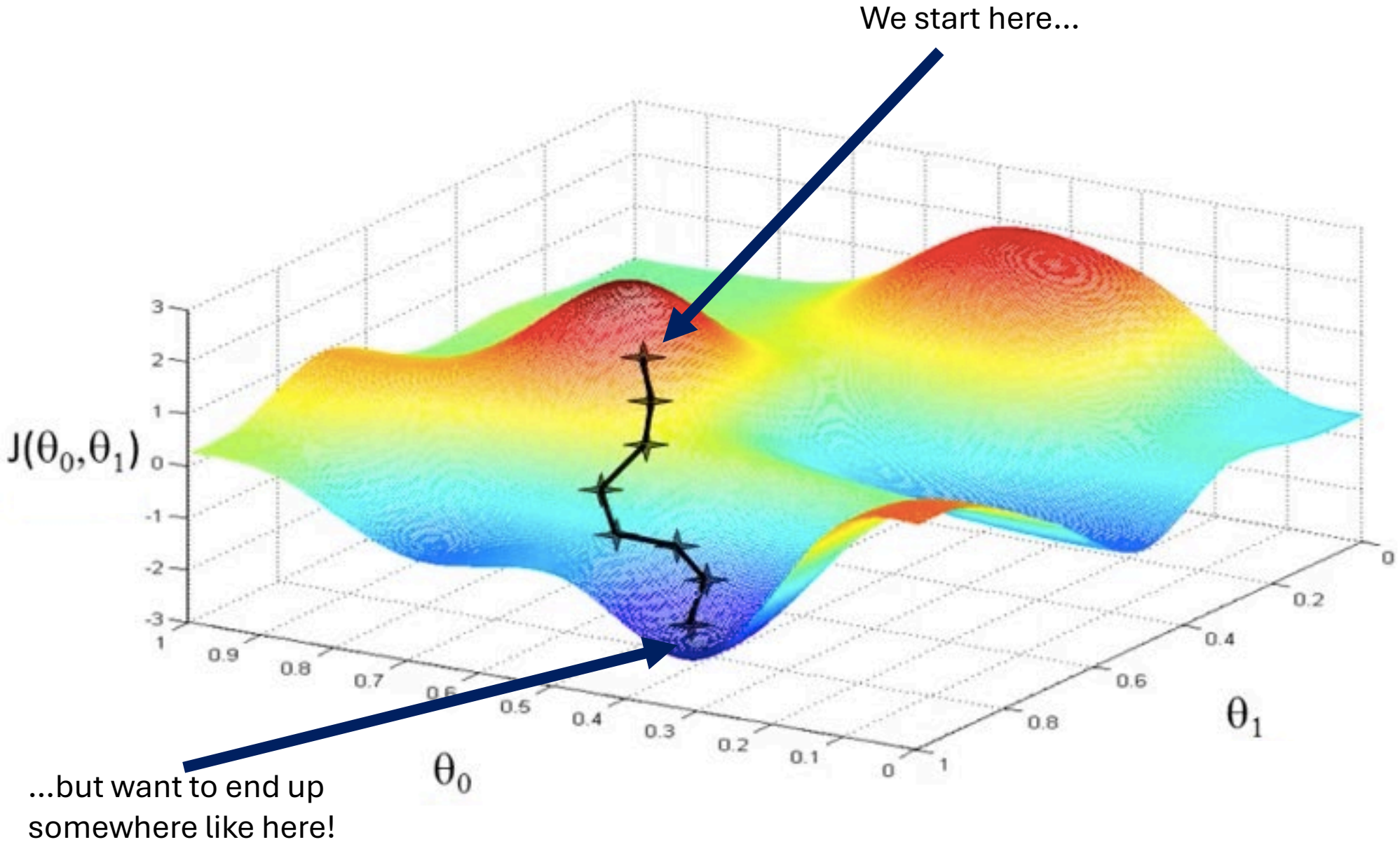
Gradient descent (hill-climbing with slopes)

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

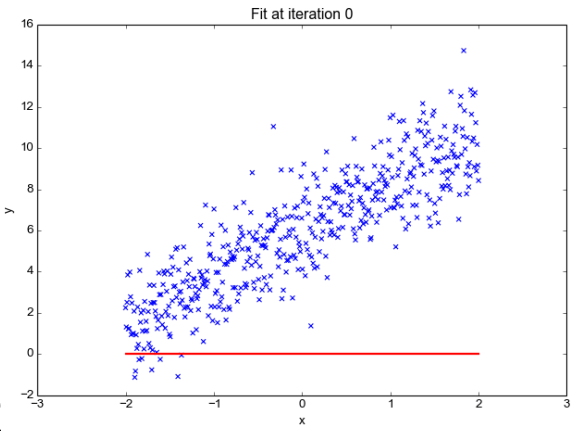
Want $\underset{\theta_0, \theta_1}{\text{argmin}}\ J(\theta_0, \theta_1)$

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


We start here…
…but want to end up somewhere like here!
$J(\theta_0,\theta_1)$
θ_0
θ_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
0
0.2
0.4
0.6
0.8
1

Computing partial derivative(s)



Taking Derivatives of Cost Functions:

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

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

Resulting Derivatives / Gradients:

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

Gradient descent recipe

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 (implementation note)

***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*:**

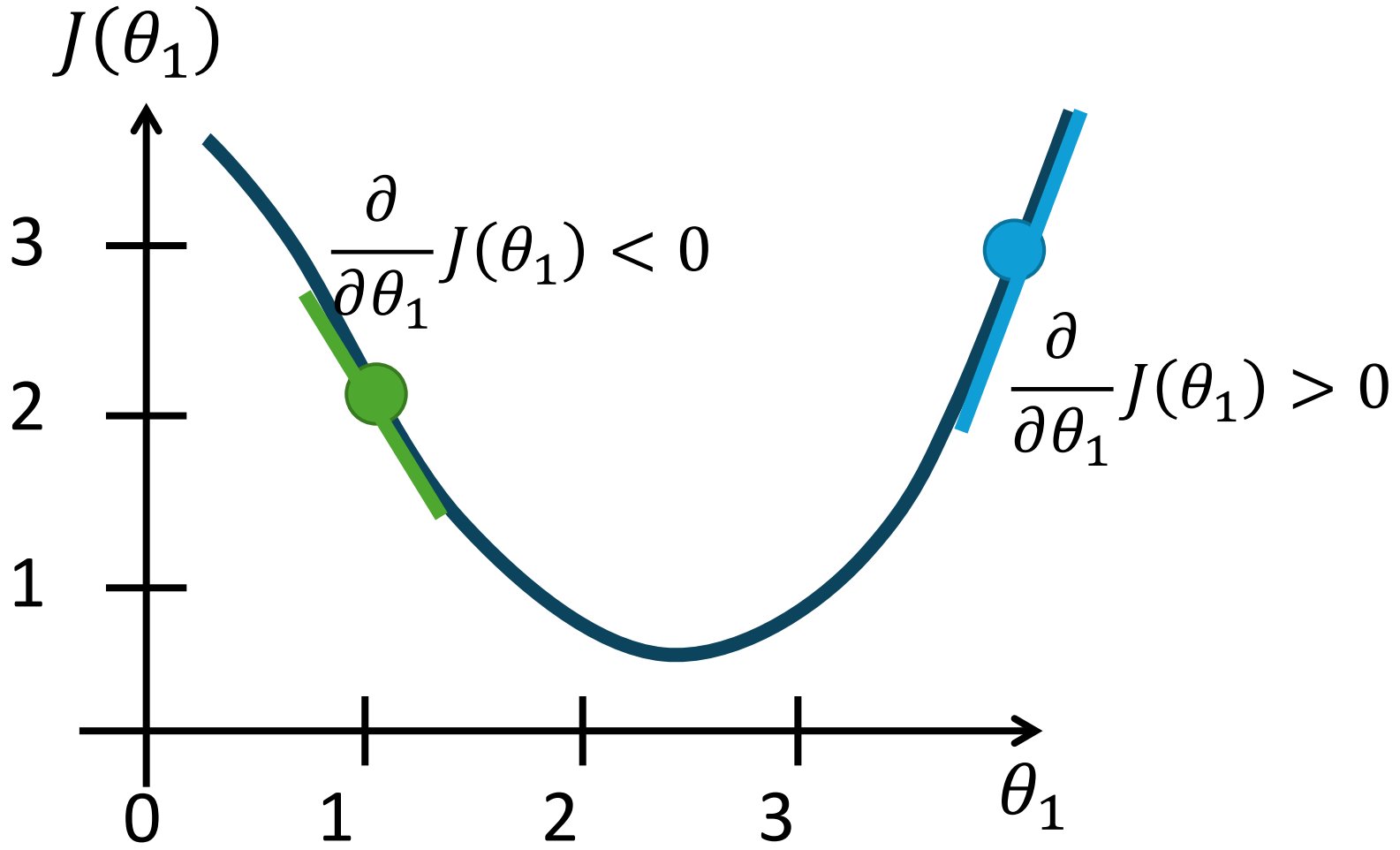
$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

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

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

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

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



Gradient descent for linear regression

Repeat until convergence {

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

}

- Linear regression model

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

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

Batch gradient descent

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

Repeat until convergence{

M: Number of training examples

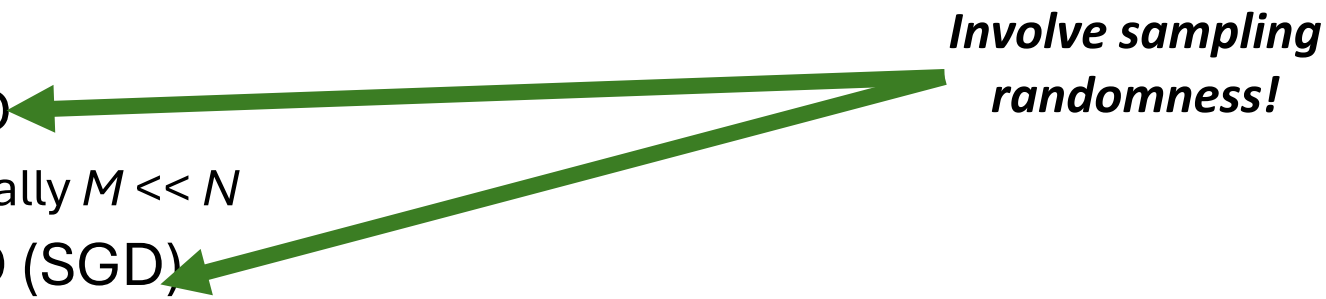
$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{M} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)$$

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

}

“Batching up” gradients

- *Another trade-off*: compute/simulation time versus gradient error
  - *N* patterns in a dataset, *M* used patterns at any time
  - Batch gradient descent (GD)
    - $M = N$
  - Mini-Batch GD
    - $M < N$, generally $M << N$
  - Stochastic GD (SGD)
    - $M = 1$



Multivariate Regression: Generalizing the Regressor

Alexander G. Ororbia II

COGS-621: Foundations of Scientific Computing

11/13/2025

Univariate Regression

- **Hypothesis:**

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

- **Parameters:**

$$\theta_0, \theta_1$$

- **Cost function:**

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

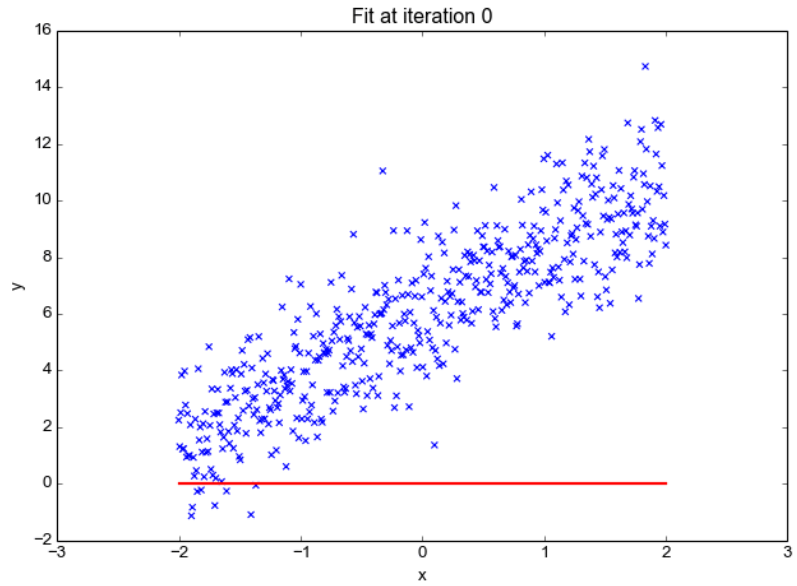
- **Goal**:

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

Gradients / (Partial) Derivatives:

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

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



Linear regression

- Model representation
- Cost function
- Gradient descent
- **Multivariate regression**

Training dataset

What if instead of this univariate dataset…

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

(Linear) Hypothesis

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

Dependent variable

Independent variable

Multiple features (input variables)

…we had this multivariate dataset instead?

	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)
1	2104	5	1	45	460
2	1416	3	2	40	232
3	1534	3	2	30	315
4	852	2	1	36	178
	…				…

Notation:

n = Number of features

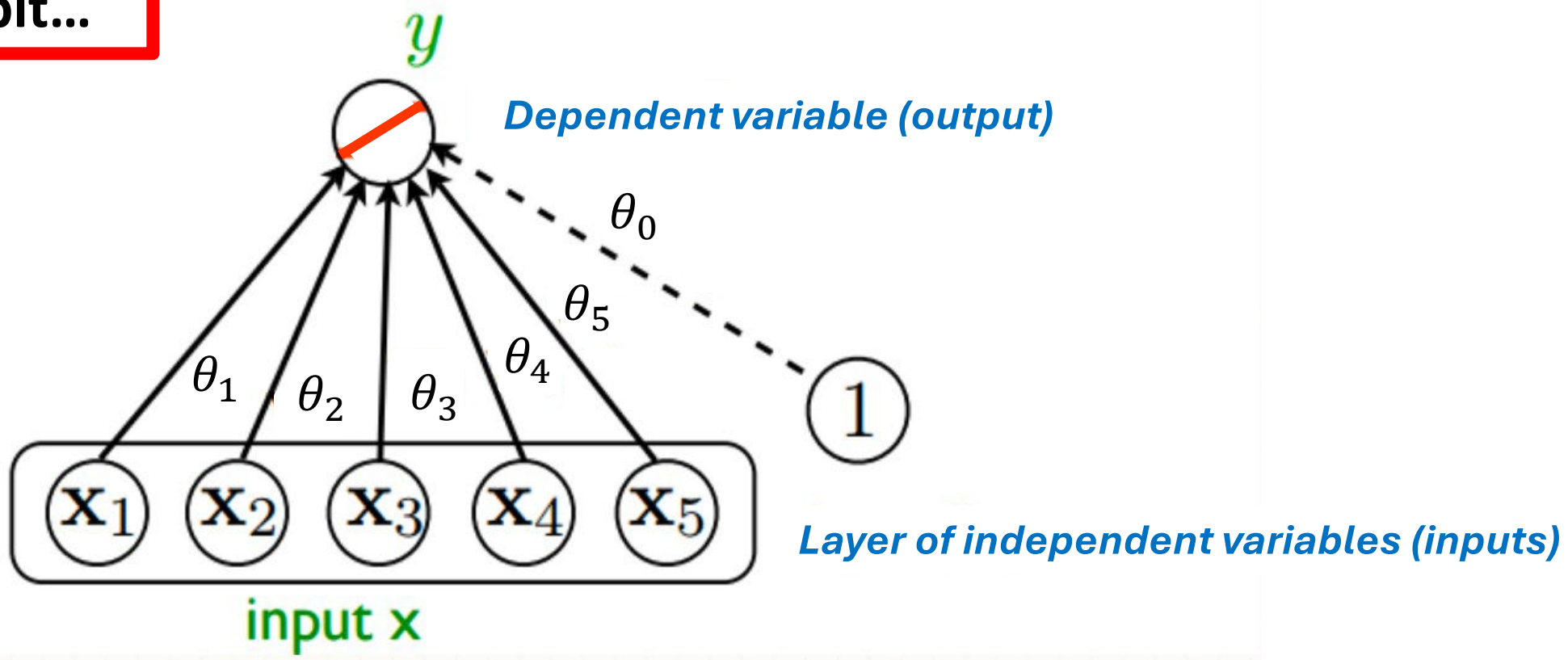
$x^{(i)}$= Input features of i^{th} training example

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

$x_3^{(2)} =?$

$x_3^{(4)} =?$

Looks like we're going to need to alter our regression model just a bit…



Hypothesis: Multivariate form

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)

It's like we are appending to each data record a redundant value of one

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

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

Notice: this is a dot product =)

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 ($n \geq 1$)

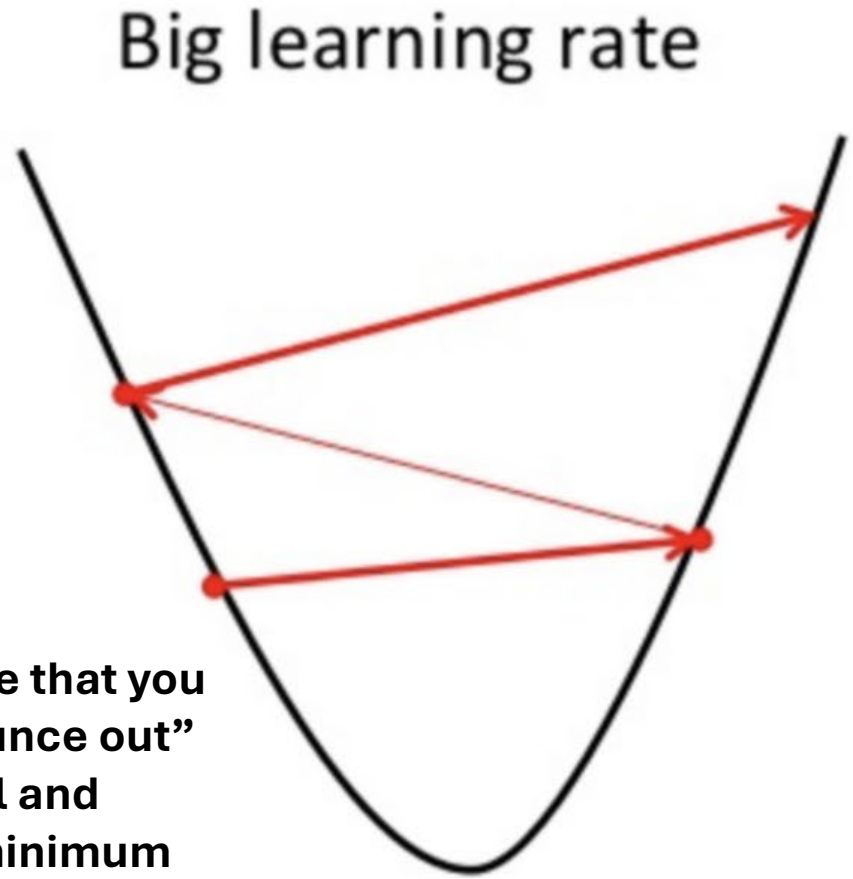
Repeat until convergence**{**

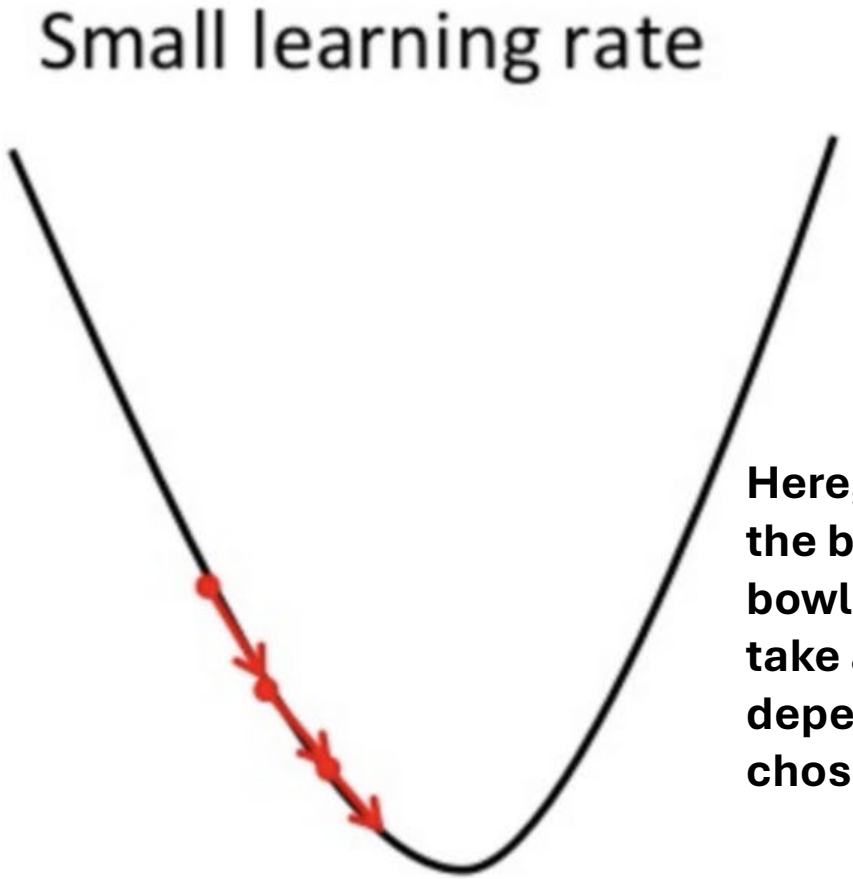
$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) x_j^{(i)}$$

}

Simultaneously update
θ_j, for $j = 0, 1, \cdots, n$

Gradient descent: learning rate (step size) size





Notice **here that you might "bounce out" of the bowl and miss the minimum**

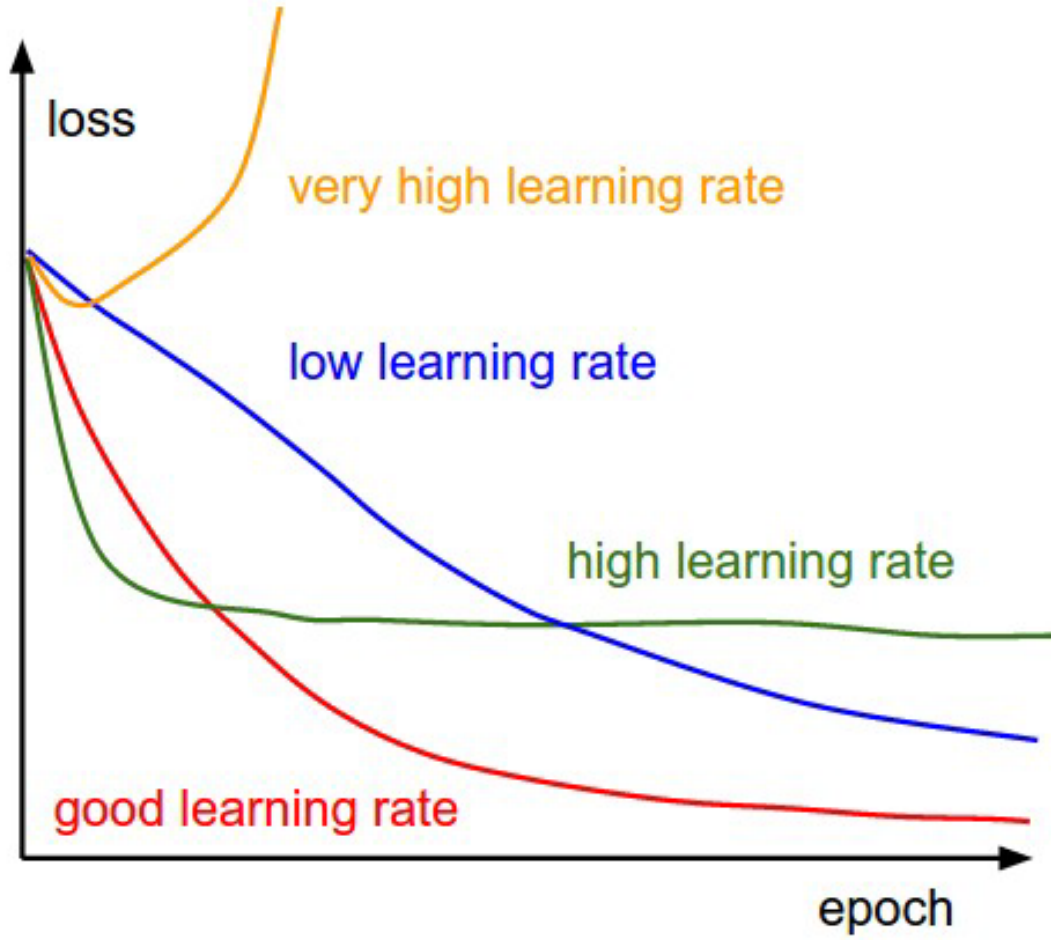
Here, you will reach the bottom of the bowl…but it could take a while depending on your chosen step-size

Gradient descent in practice: step size

- 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



Learning rate is also known as the "*step size*" in statistics/mathematics

Image credit: CS231n@Stanford

But what if we wanted some automatic capacity control? (wanted to down-weight some independent variables?)

(Answer = regularization/decay)

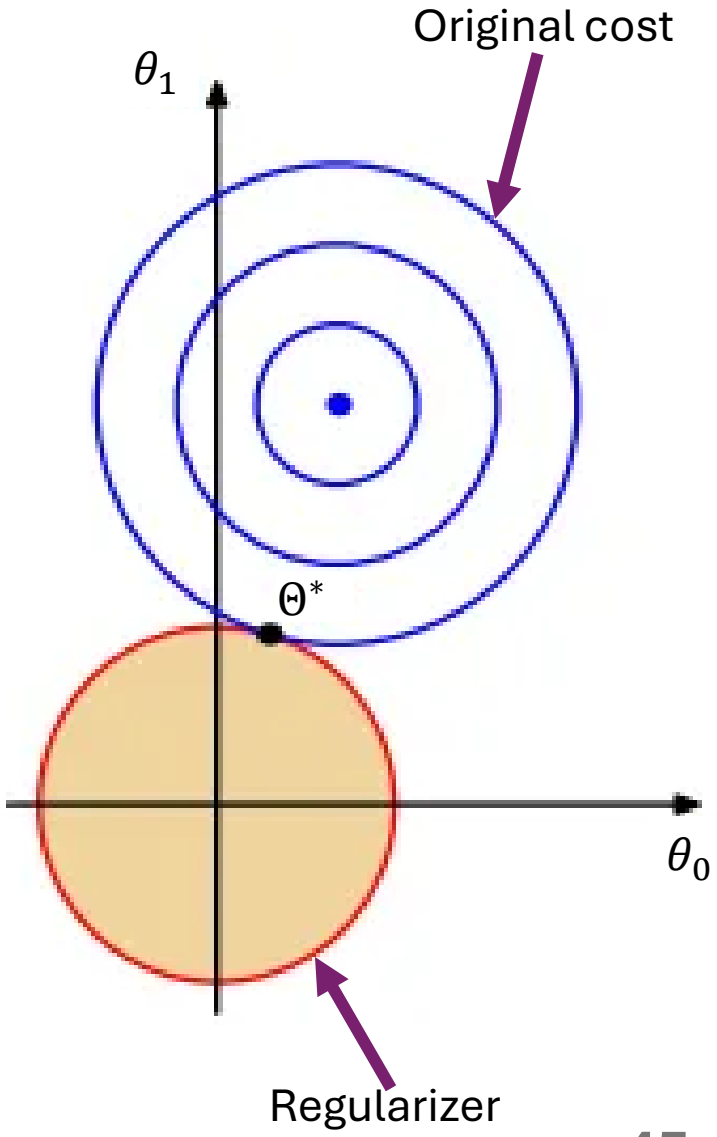
Capacity control: regularization

- We simply add another term to our cost function:

$$J(\Theta) = \frac{1}{2m}\Sigma_{i=1}^{\mathrm{m}}\left(f(\Theta) - y^{(i)}\right)^2 + \lambda\Omega(\Theta),$$

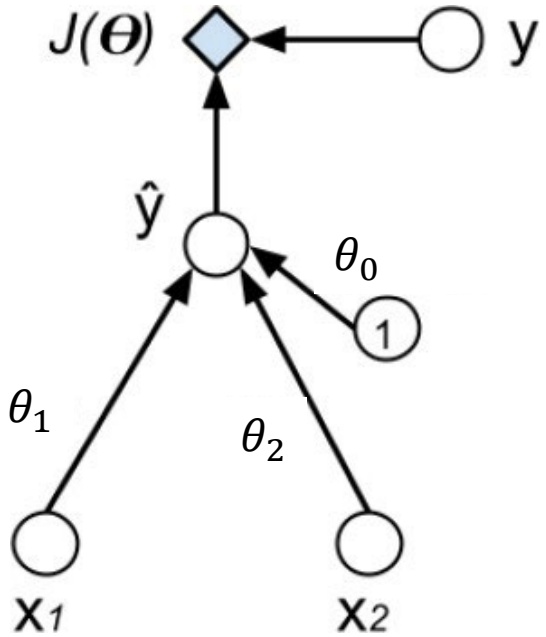
where $\Theta = \{\theta_0, \theta_1, \ldots, \theta_n\}$

- The extra term is referred to as a "soft constraint" or "regularizer":
  - $\Omega(\Theta) = \Sigma_{j=0}^{n}(\theta)^2$ <-- we just square each (j-th) piece of Θ and sum these together
  - λ is just a non-negative scalar used to control how much "influence" this soft constraint has on our gradient-based hill-climbing
    - It's like an attractive force towards zero (this keeps each θ_j from getting too big)



Multivariate regression architecture

$$\hat{y} = f_\Theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$ ← **Hypothesis!**



Cost:

$$\mathcal{J}(\Theta) = \frac{1}{2m}\sum_{i=1}^{m}(f_\Theta(x^i) - y^i)^2 + \frac{\beta}{2m}\sum_{j=1}^{n}\theta_j^2$$

Derivative/Update:

$$\frac{\partial \mathcal{J}(\Theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(f_\Theta(x^i) - y^i)x_j^i + \frac{\beta}{m}\theta_j$$

Optimizer:

$$\theta_j = \theta_j - \alpha\frac{\partial \mathcal{J}(\Theta)}{\partial \theta_j} = \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(f_\Theta(x^i) - y^i)x_j^i,\ j = 0, 1, 2, \cdots, n.$$