Linear Regression

Machine learning algorithms

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

Nearest neighbor classifier

- **Training data**

$$\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \cdots, \left(x^{(N)}, y^{(N)}\right)$$

- **Learning**

Do nothing.

- **Testing**

$$h(x) = y^{(k)}, \text{ where } k = \operatorname{argmin}_{\mathrm{i}} D\left(x, x^{(i)}\right)$$

Instance/Memory-based Learning

1. *A distance metric*
   - Continuous? Discrete? PDF? Gene data? Learn the metric?
2. *How many nearby neighbors to look at?*
   - *1? 3? 5? 15?*
3. *A weighting function (optional)*
   - Closer neighbors matter more
4. *How to fit with the local points?*
   - Kernel regression

Slide credit: Carlos Guestrin

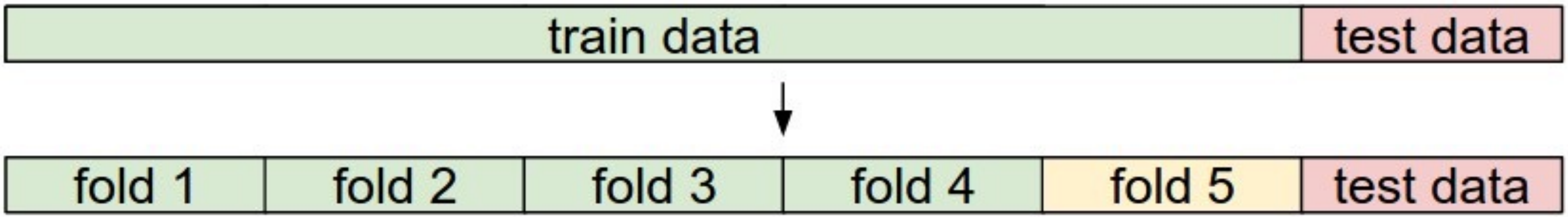
Validation set

- Spliting training set: A *fake* test set to tune hyper-parameters

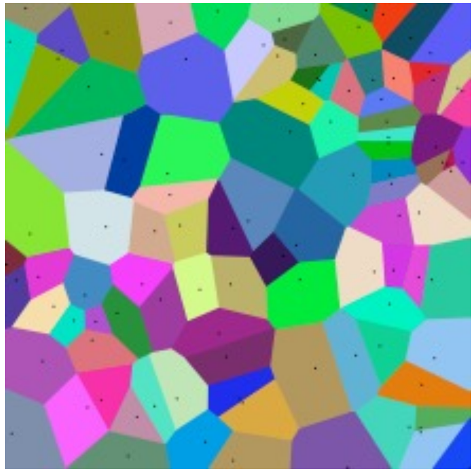
```
# assume we have Xtr_rows, Ytr, Xte_rows, Yte as before
# recall Xtr_rows is 50,000 x 3072 matrix
Xval_rows = Xtr_rows[:1000, :] # take first 1000 for validation
Yval = Ytr[:1000]
Xtr_rows = Xtr_rows[1000:, :] # keep last 49,000 for train
Ytr = Ytr[1000:]

# find hyperparameters that work best on the validation set
validation_accuracies = []
for k in [1, 3, 5, 10, 20, 50, 100]:

  # use a particular value of k and evaluation on validation data
  nn = NearestNeighbor()
  nn.train(Xtr_rows, Ytr)
  # here we assume a modified NearestNeighbor class that can take a k as input
  Yval_predict = nn.predict(Xval_rows, k = k)
  acc = np.mean(Yval_predict == Yval)
  print 'accuracy: %f' % (acc,)

  # keep track of what works on the validation set
  validation_accuracies.append((k, acc))
```

Cross-validation

- 5-fold cross-validation -> split the training data into 5 equal folds
- 4 of them for training and 1 for validation



Things to remember



- Supervised Learning
  - Training/testing data; classification/regression; Hypothesis
- k-NN
  - Simplest learning algorithm
  - With sufficient data, very hard to beat "strawman" approach
- Kernel regression/classification
  - Set k to n (number of data points) and chose kernel width
  - Smoother than k-NN
- Problems with k-NN
  - Curse of dimensionality
  - Not robust to irrelevant features
  - Slow NN search: must remember (very large) dataset for prediction

Today’s plan: Linear Regression

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

Linear Regression

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

<u>Regression</u>

real-valued output

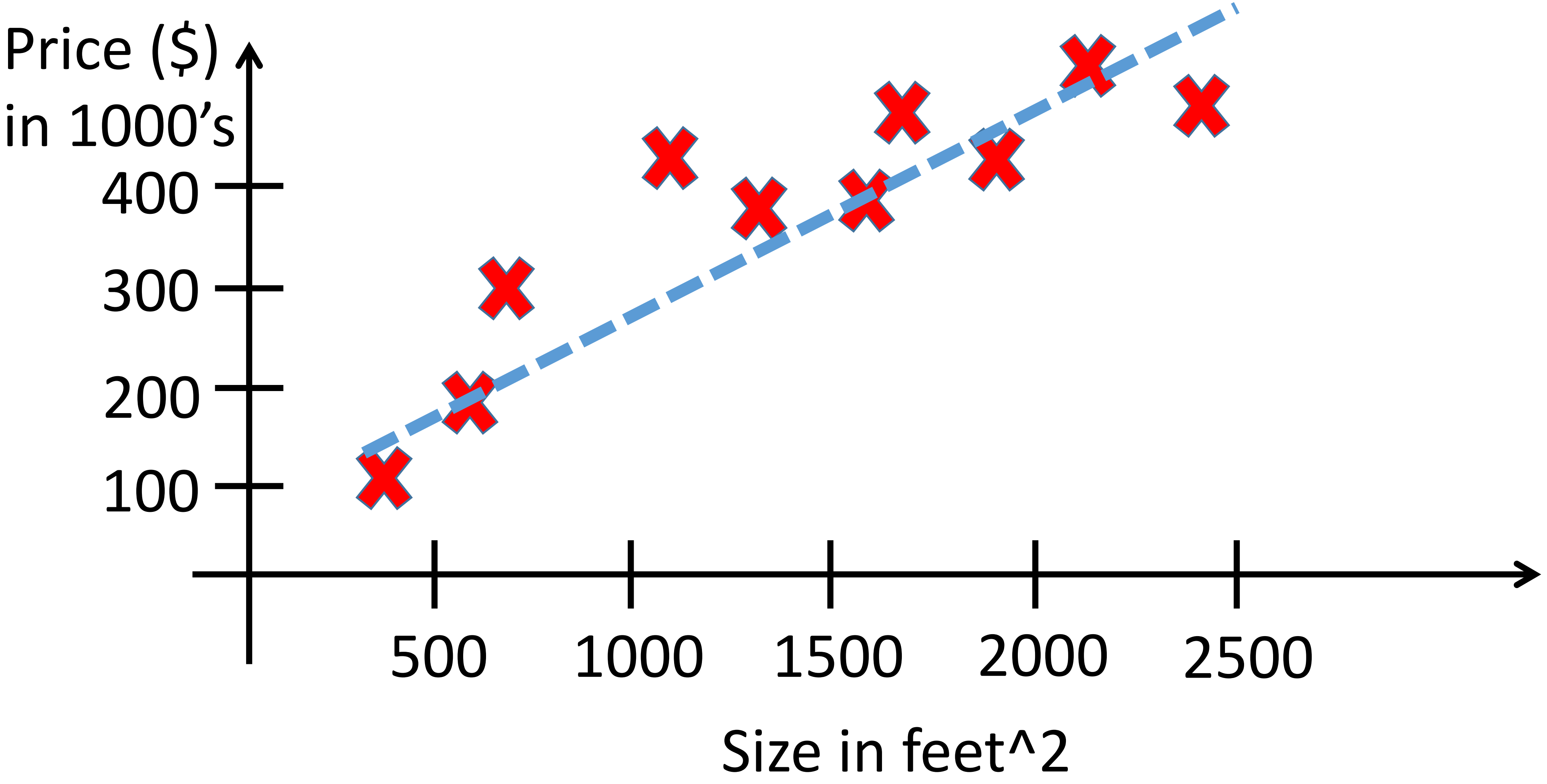
Training set

Learning Algorithm

x → h → y

Size of house | Hypothesis | Estimated price

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

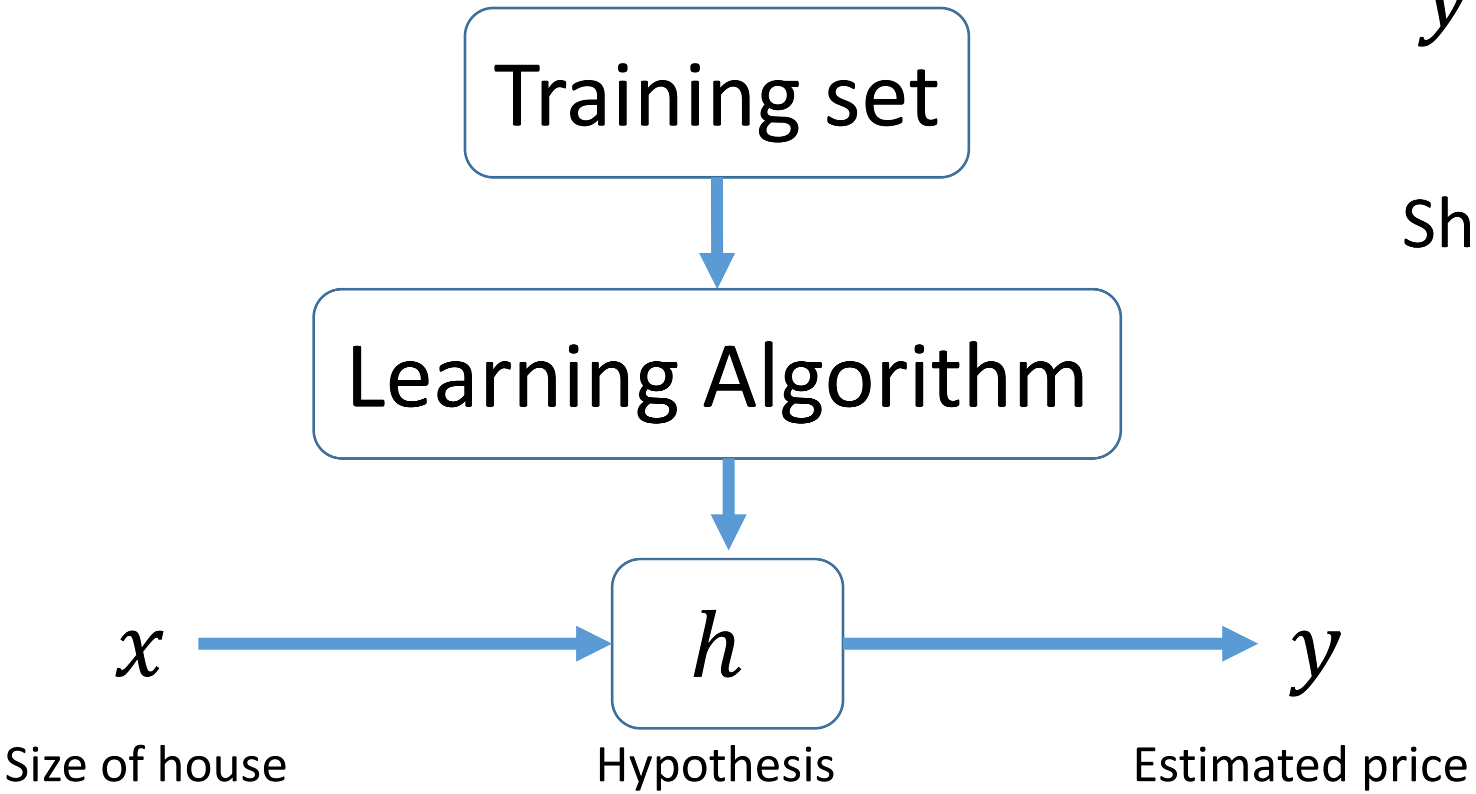
Examples:

$x^{(1)} = 2104$

$x^{(2)} = 1416$

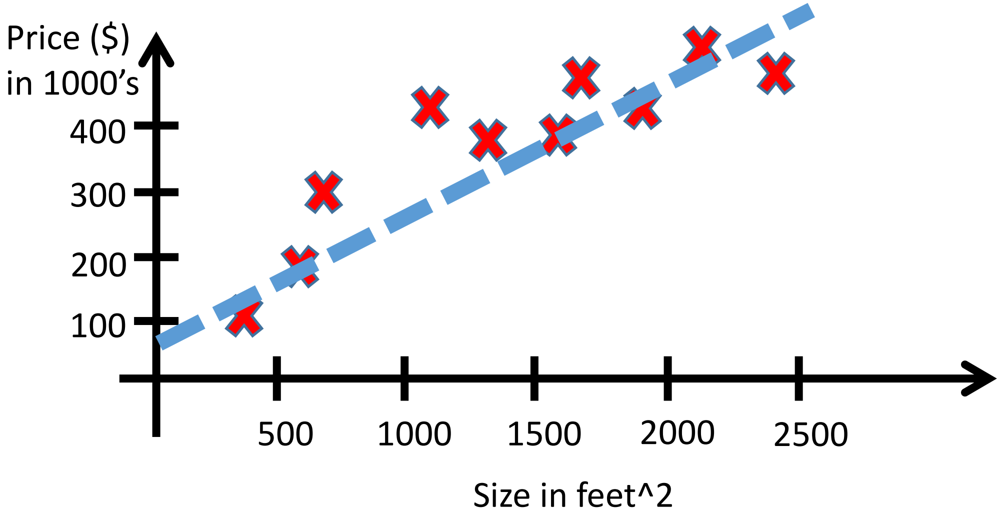
$y^{(1)} = \ 460$

Model representation



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

Shorthand $h(x)$



Univariate linear regression

Slide credit: Andrew Ng

Linear Regression

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

Training set

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

m = 47

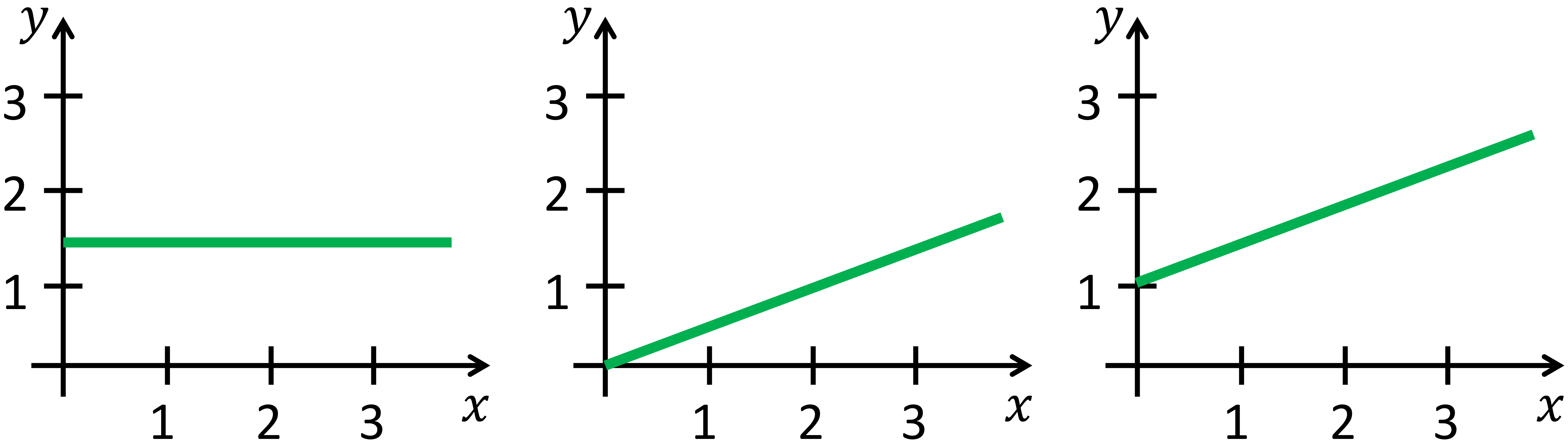
- Hypothesis $h_\theta(x) = \theta_0 + \theta_1 x$

θ_0, θ_1: parameters/weights

How to choose θ_i's?

Slide credit: Andrew Ng

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



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

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

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

Slide credit: Andrew Ng

Cost function

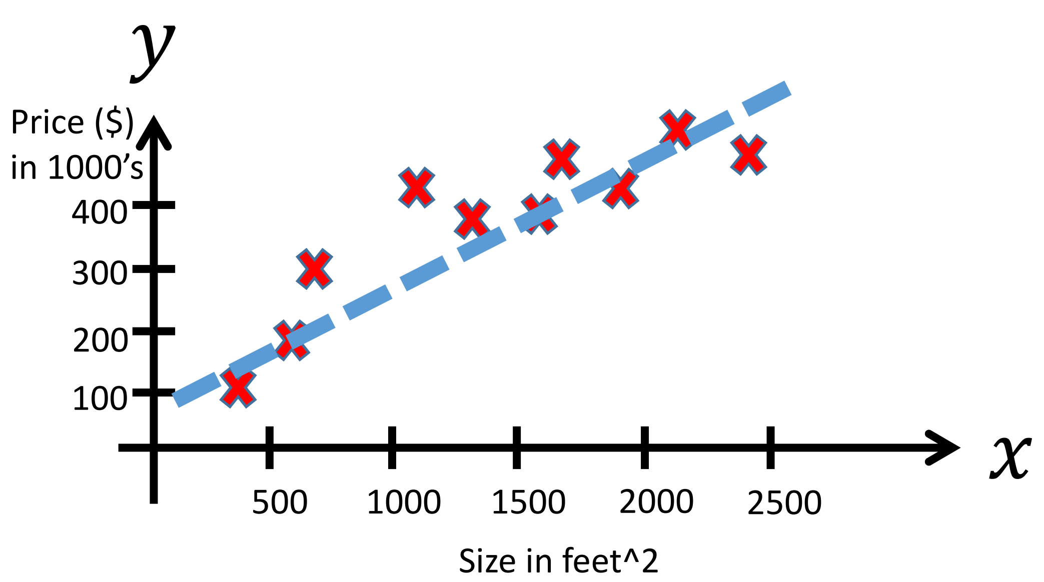
- Idea:
  Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training example (x, y)

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

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

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

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



Slide credit: Andrew Ng

Simplified

- **Hypothesis:**

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

- **Parameters:**

$$\theta_0, \theta_1 \longrightarrow \theta_1$$

- **Cost function:**

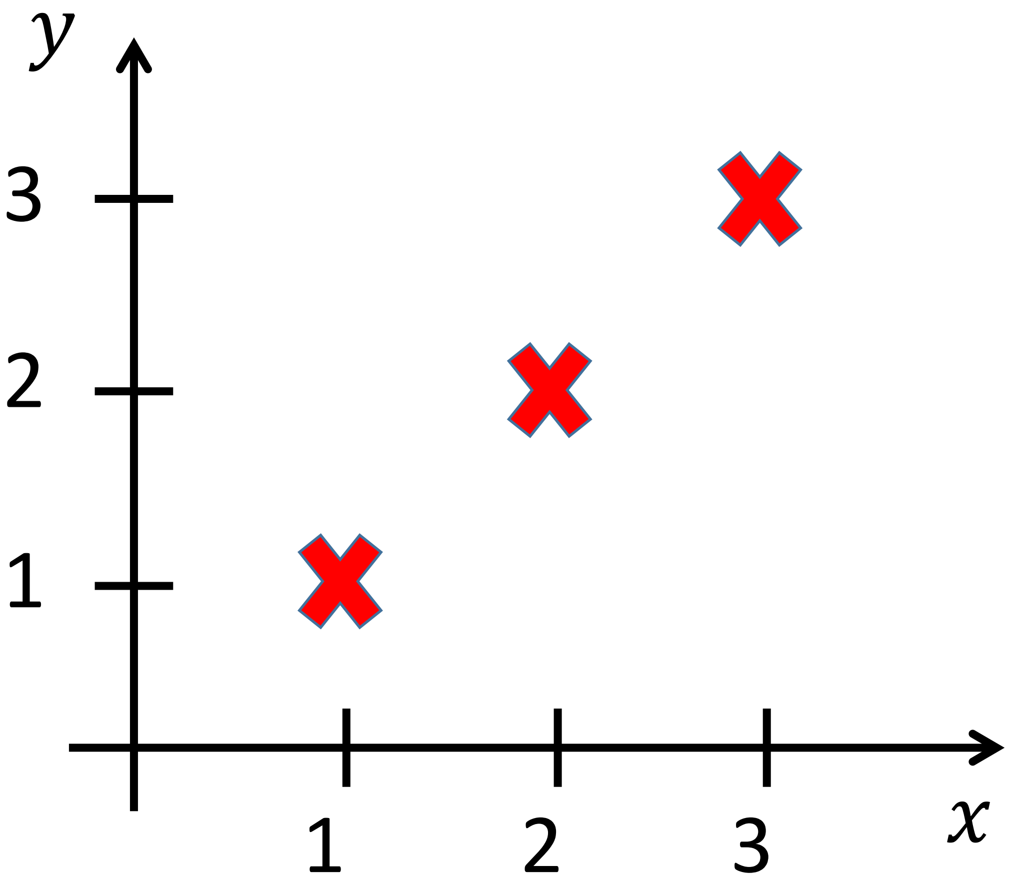
$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 \longrightarrow J(\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

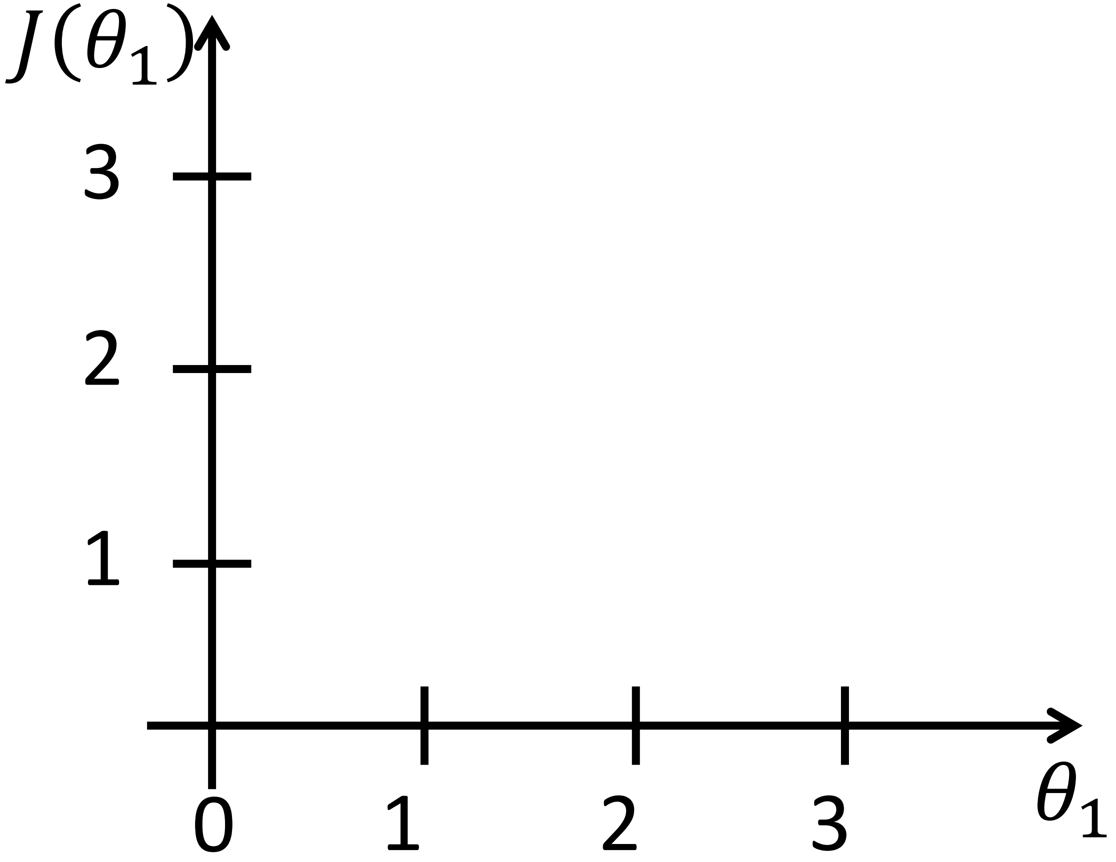
- **Goal:**

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

Slide credit: Andrew Ng

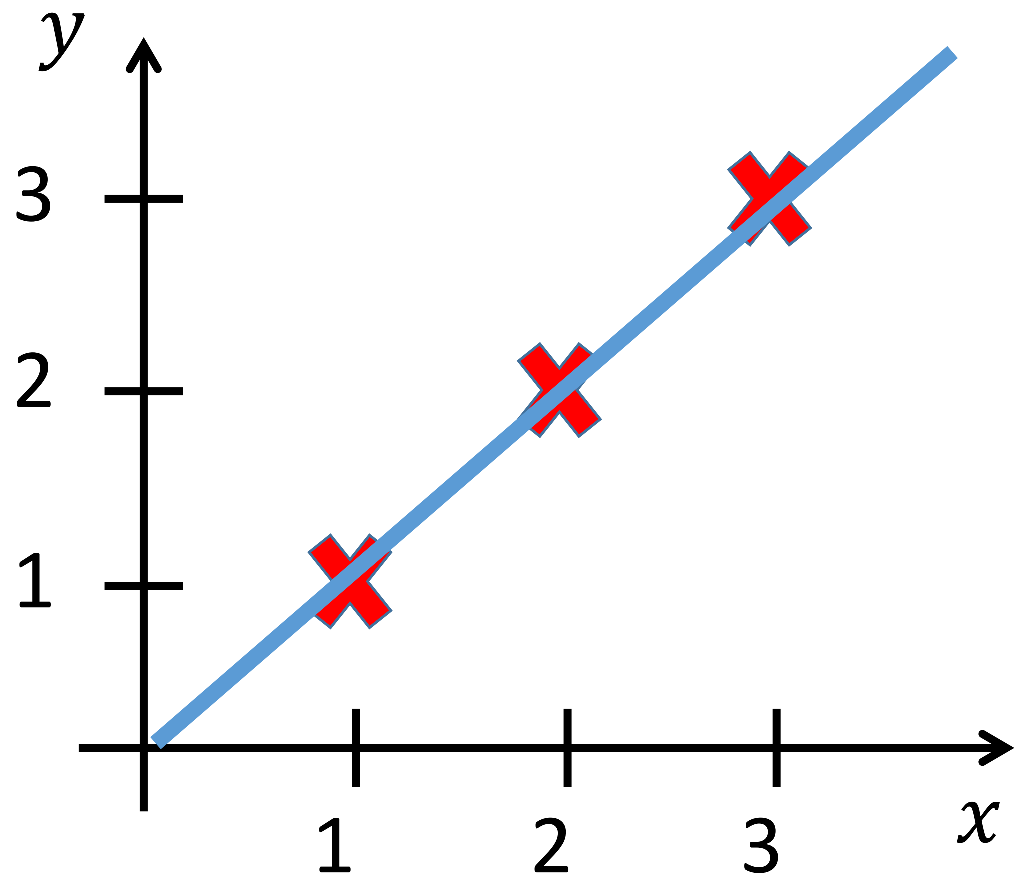
$h_\theta(x)$, function of x $J(\theta_1)$, function of θ_1



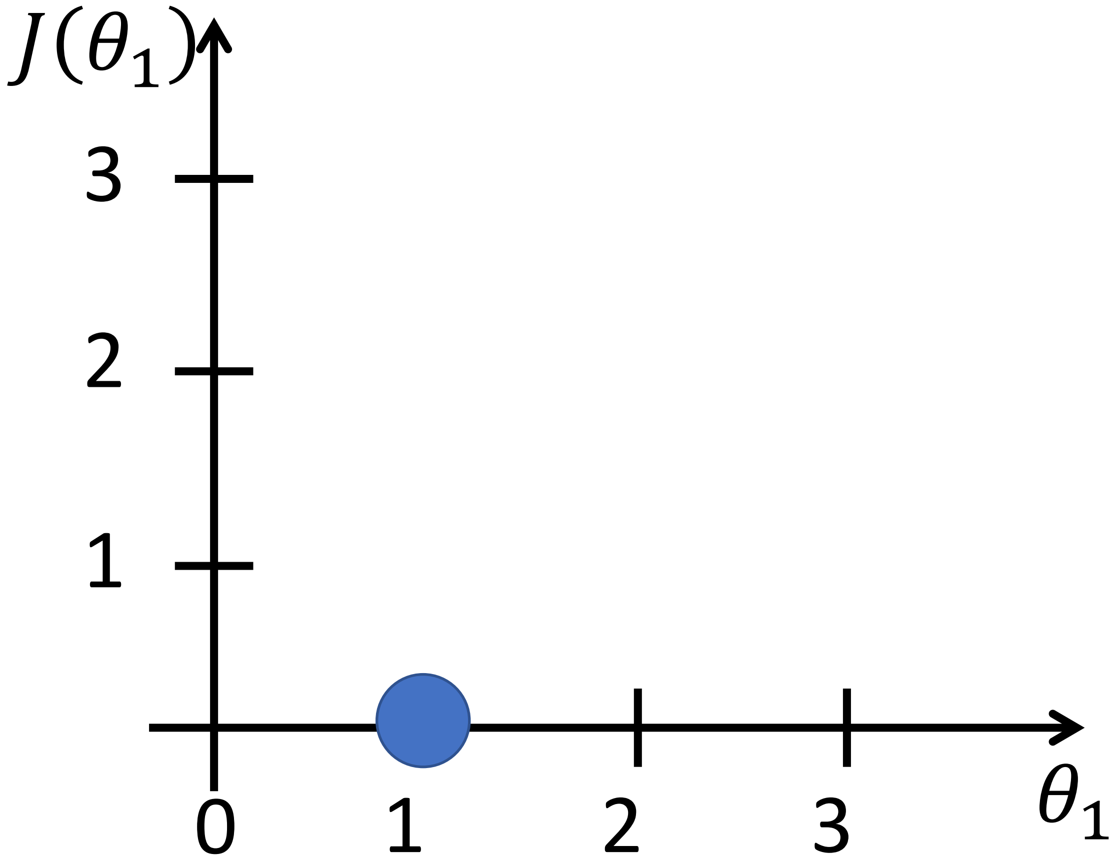


Slide credit: Andrew Ng

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

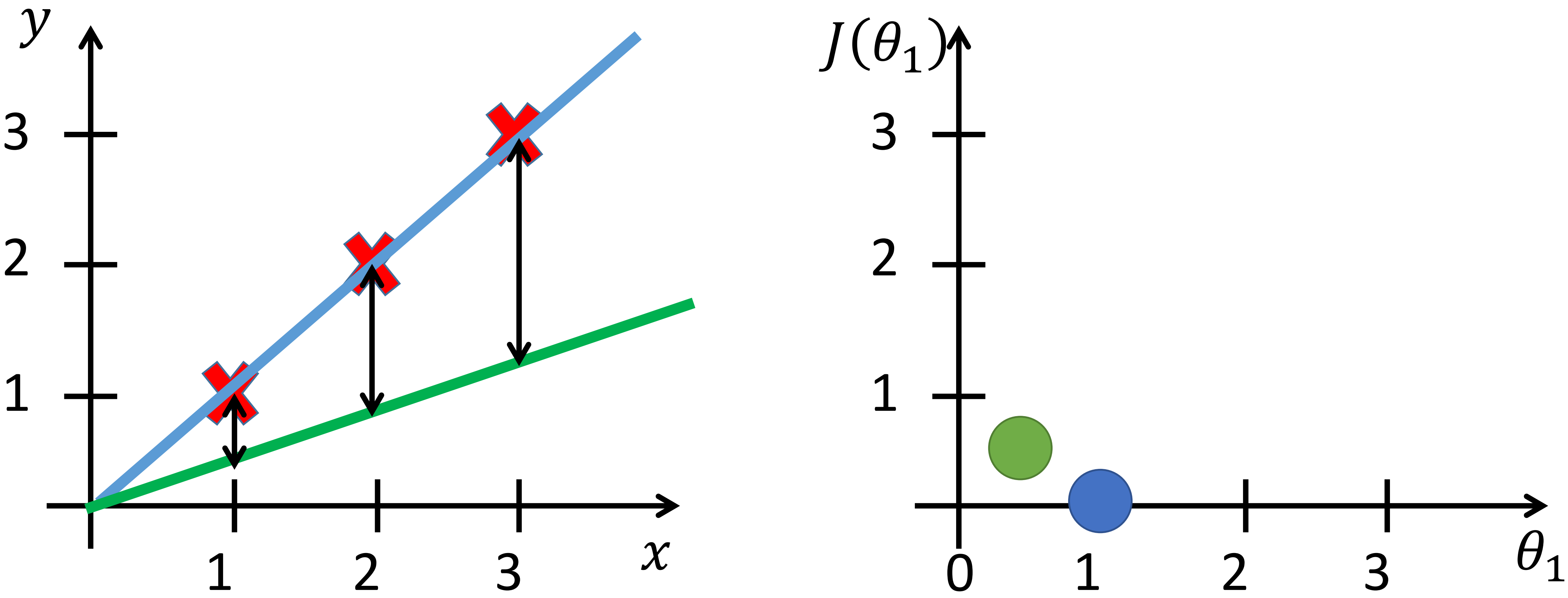


$J(\theta_1)$, function of θ_1



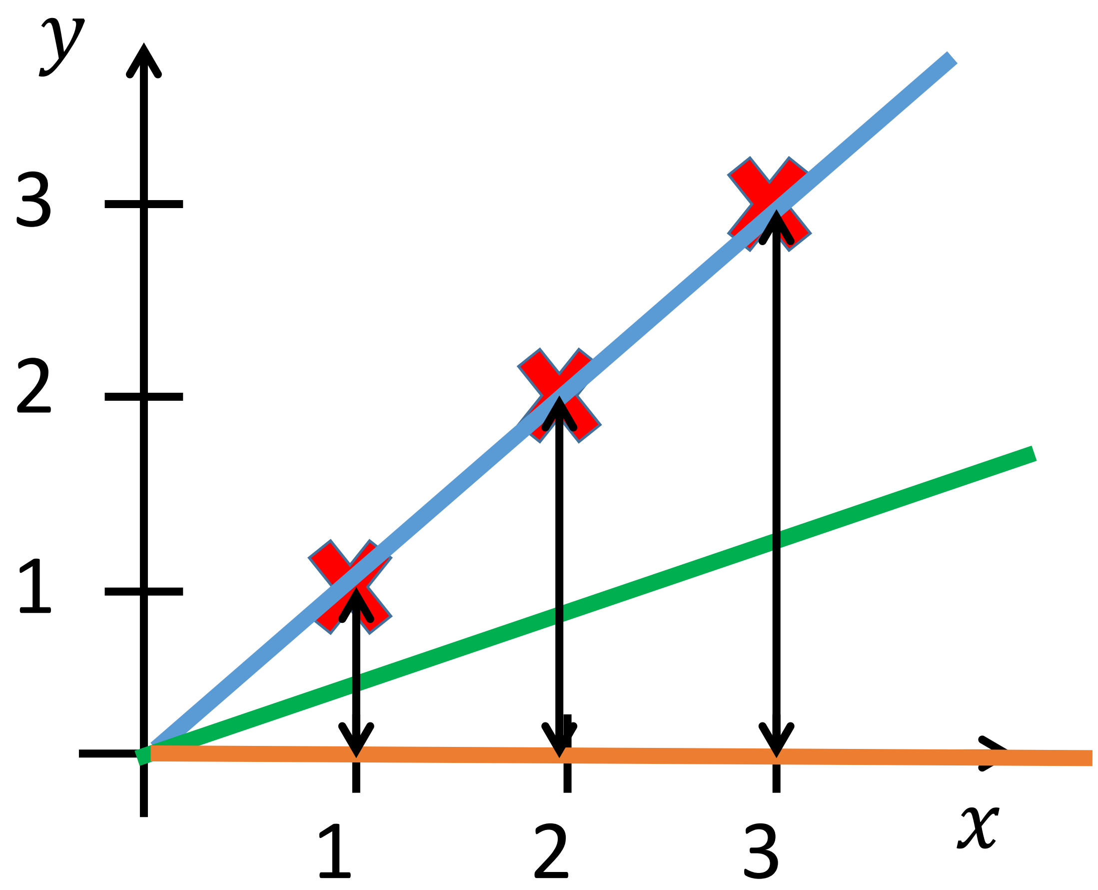
Slide credit: Andrew Ng

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

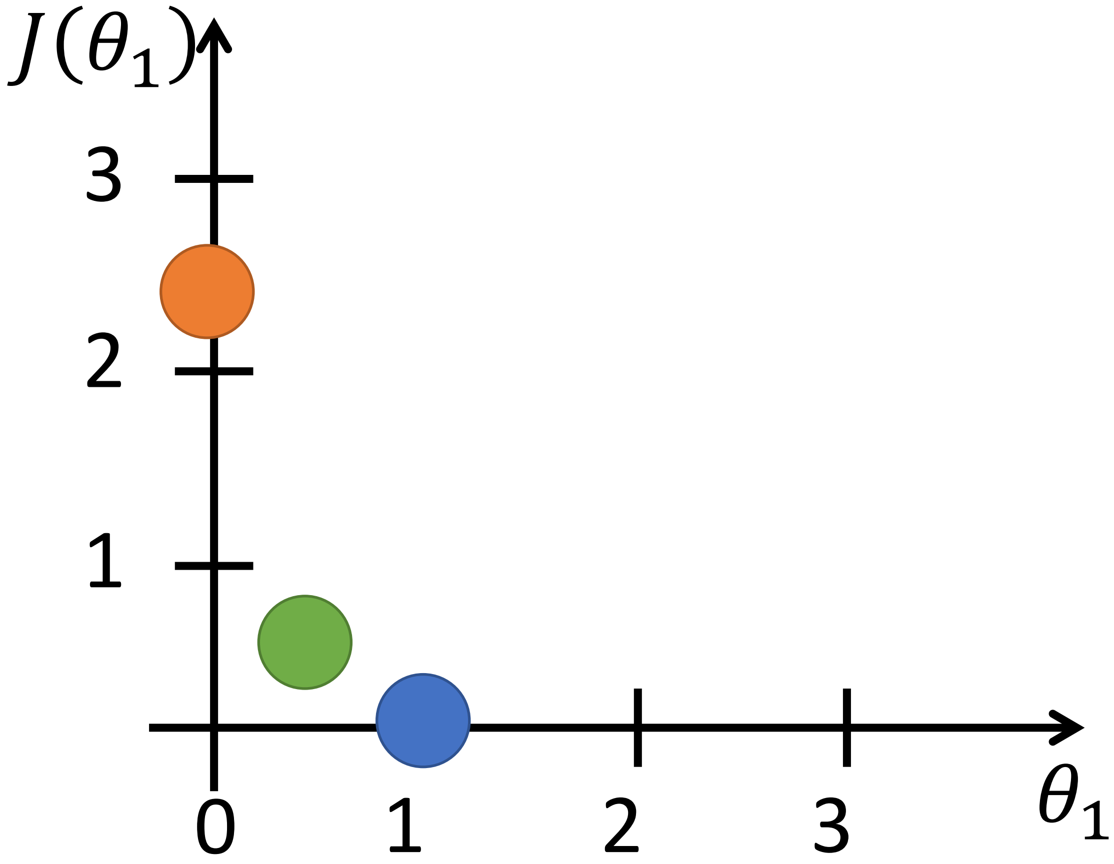


Slide credit: Andrew Ng

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

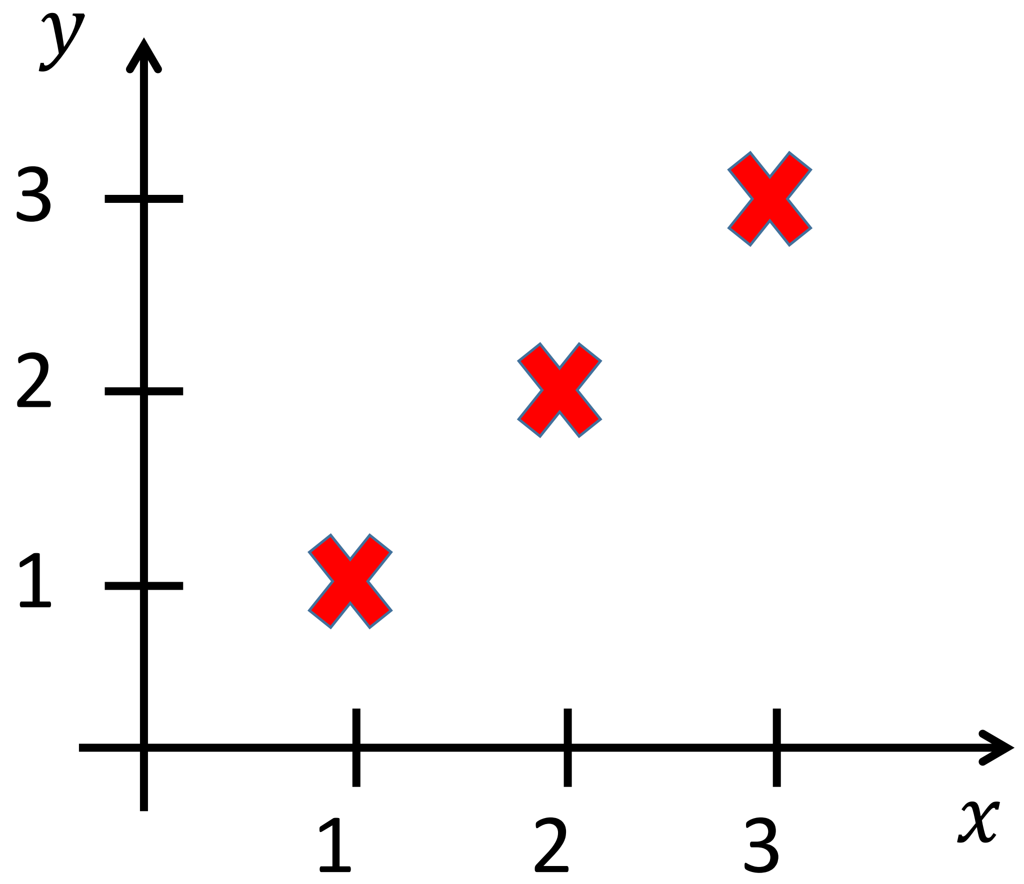


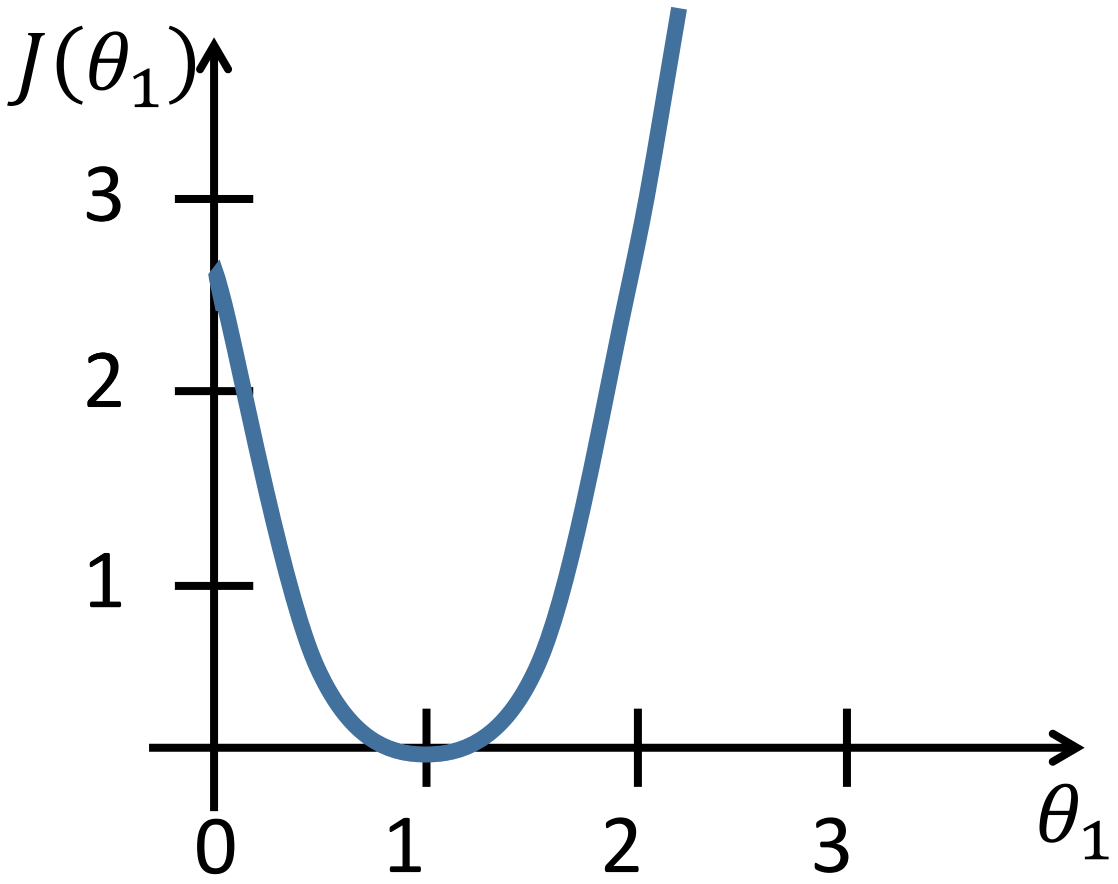
$J(\theta_1)$, function of θ_1



Slide credit: Andrew Ng

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





Slide credit: Andrew Ng

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

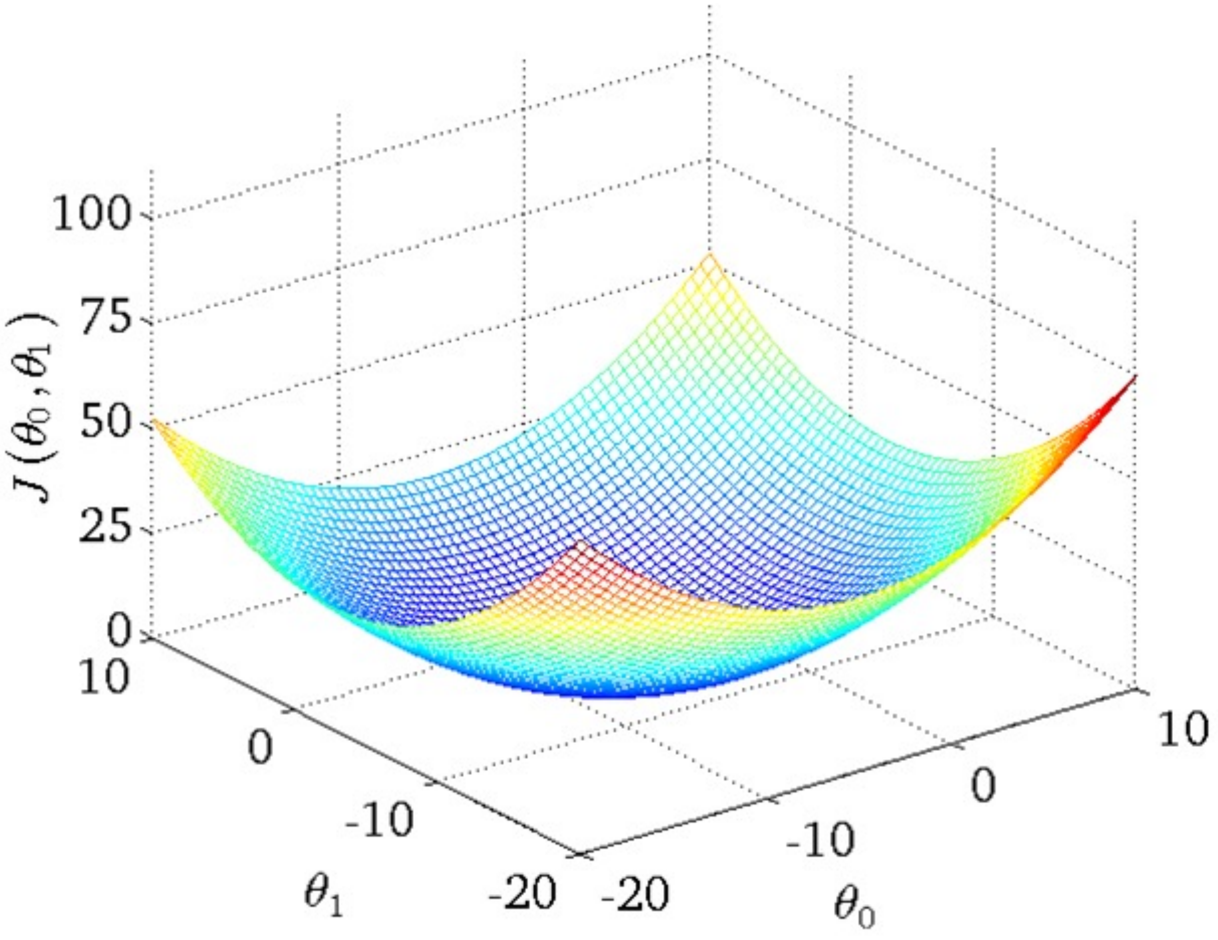
- **Parameters:** θ_0, θ_1

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

Slide credit: Andrew Ng

Cost function



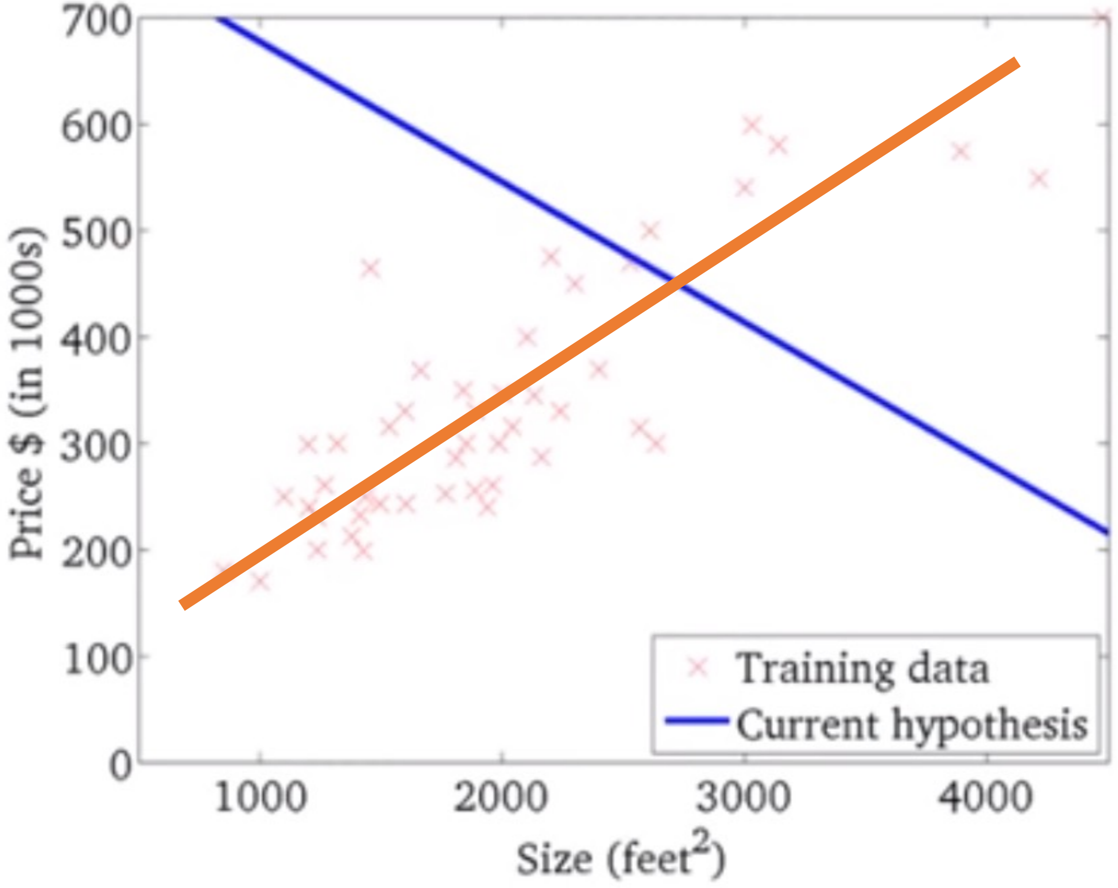
Slide credit: Andrew Ng

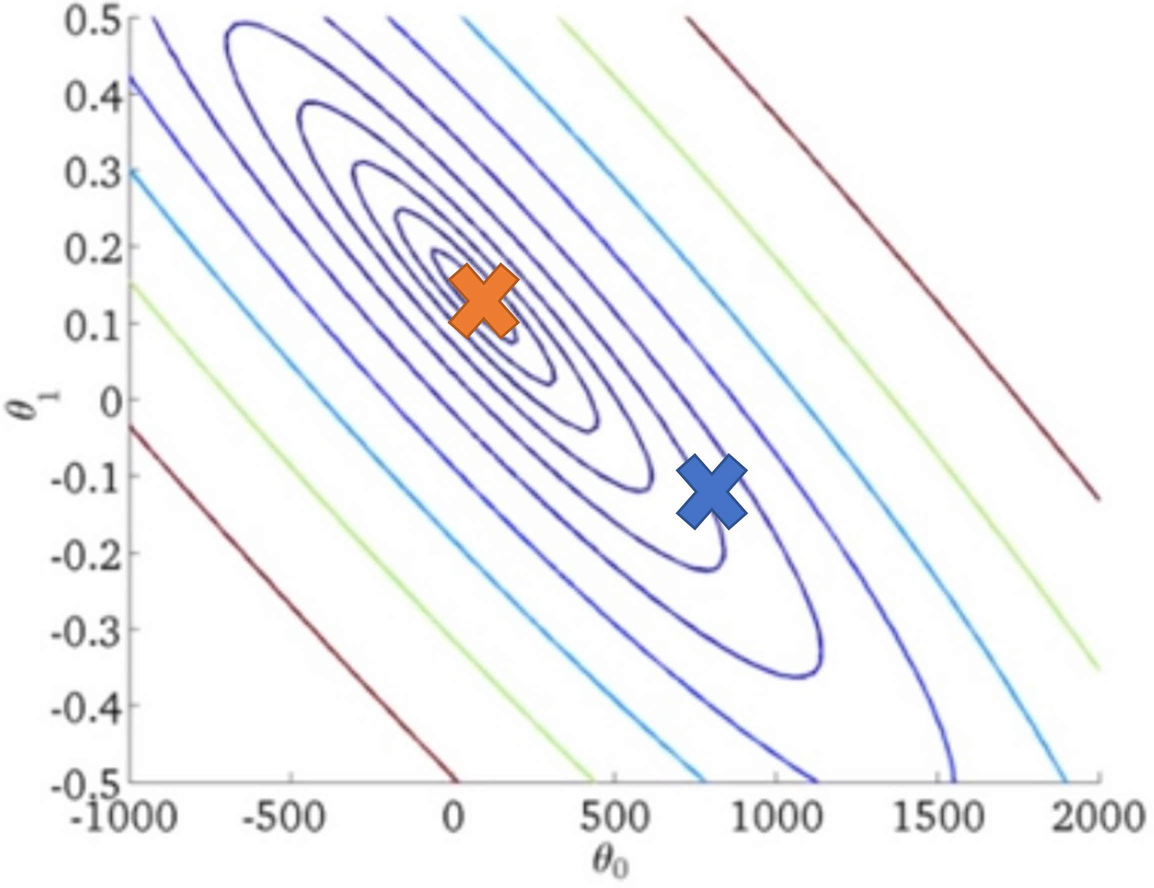
$h_\theta(x)$

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

Slide credit: Andrew Ng

Linear Regression

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

Gradient descent

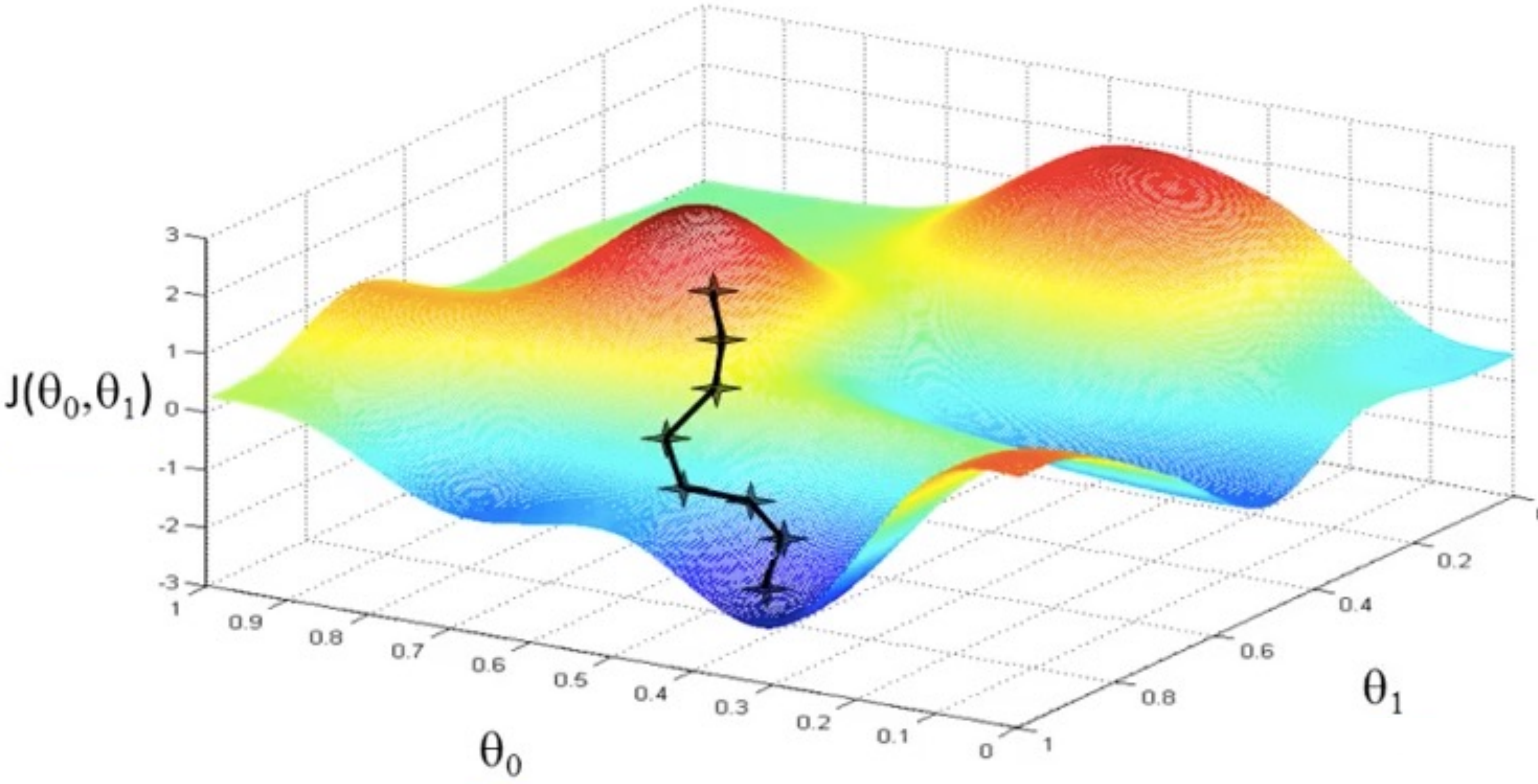
Have some function $J(\theta_0, \theta_1)$

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

Slide credit: Andrew Ng



Slide credit: Andrew Ng

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)

Slide credit: Andrew Ng

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:

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

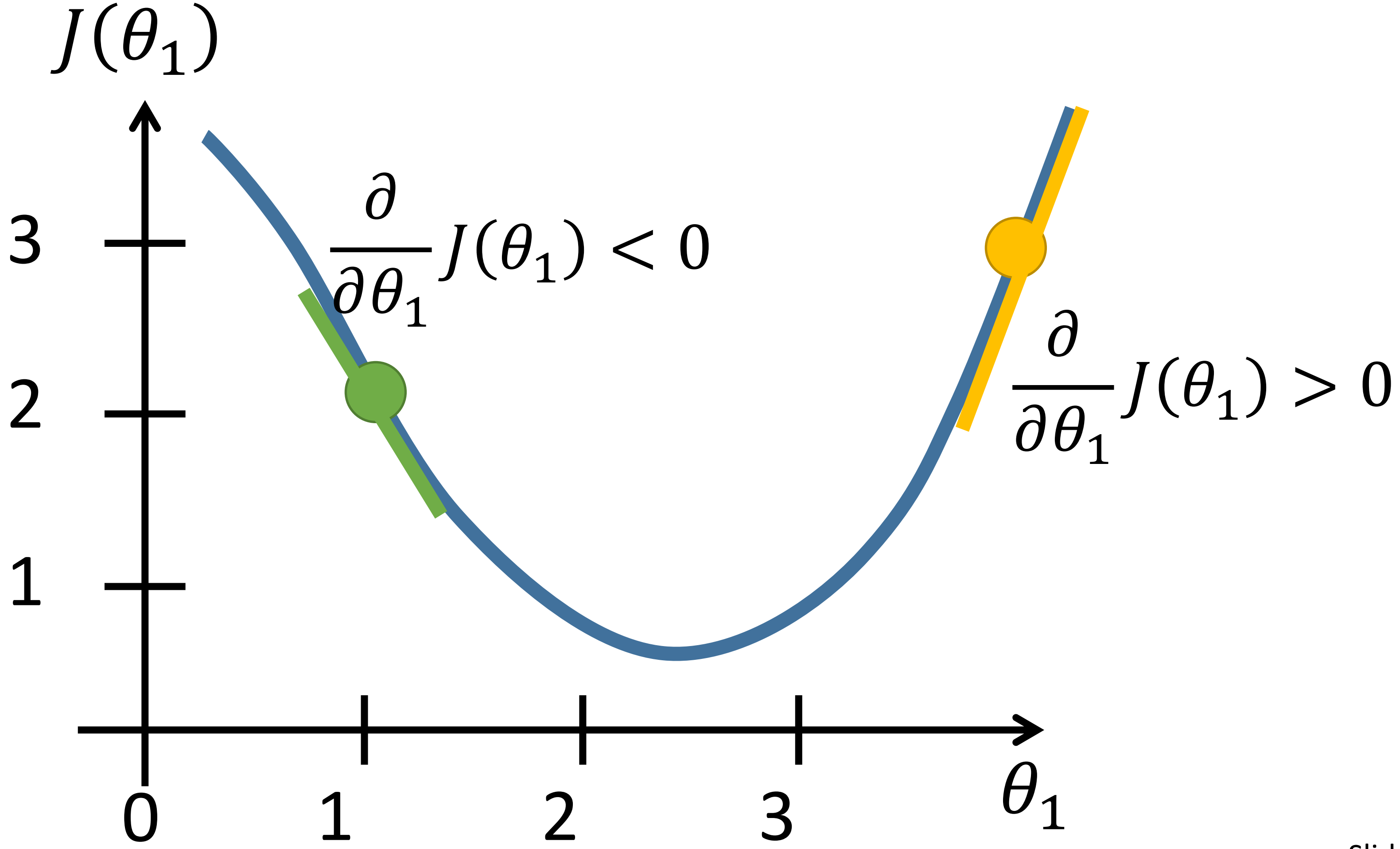
$$\theta_0 := \text{temp0}$$

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

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

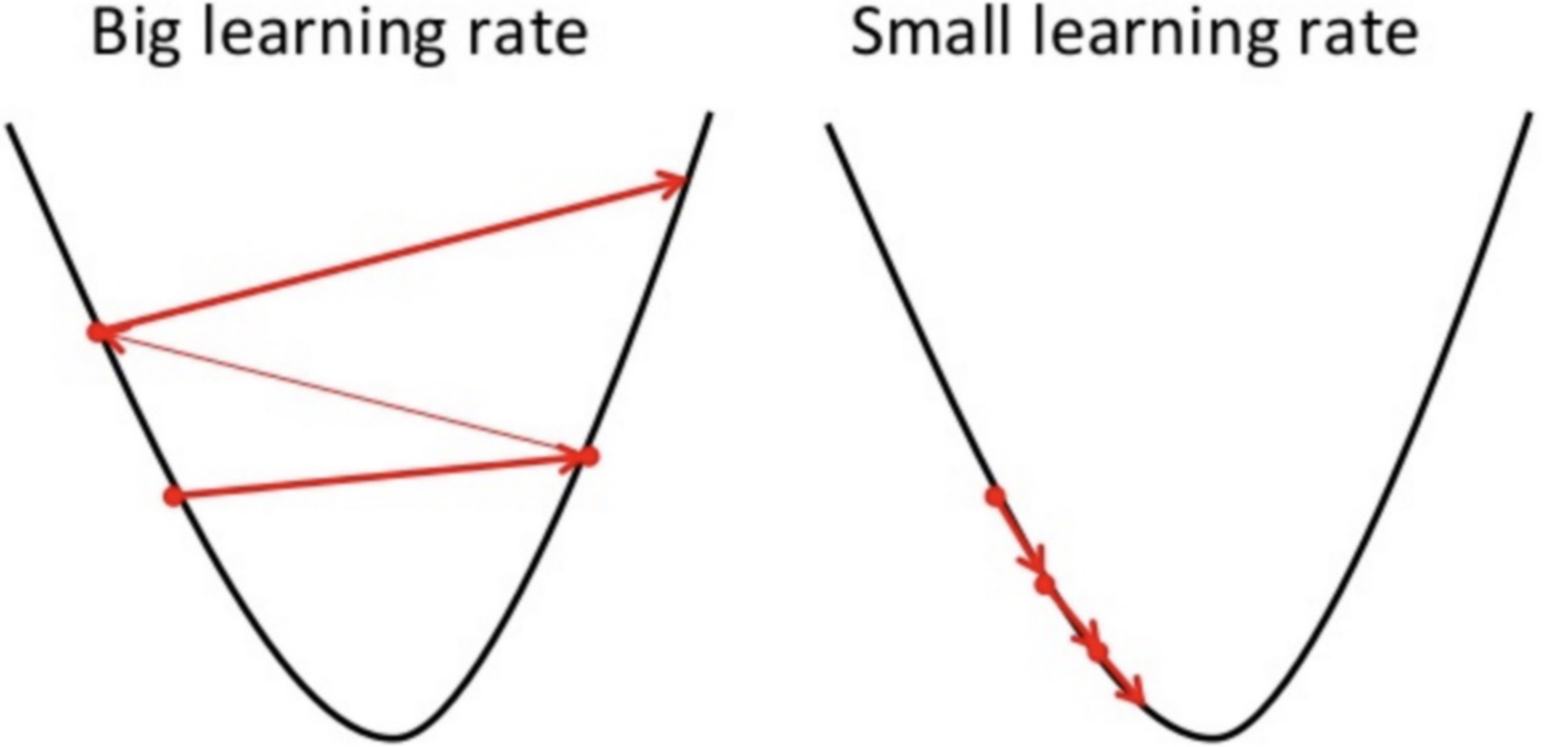
Slide credit: Andrew Ng

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



Slide credit: Andrew Ng

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

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

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

Slide credit: Andrew Ng

Computing partial derivative

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

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

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

Slide credit: Andrew Ng

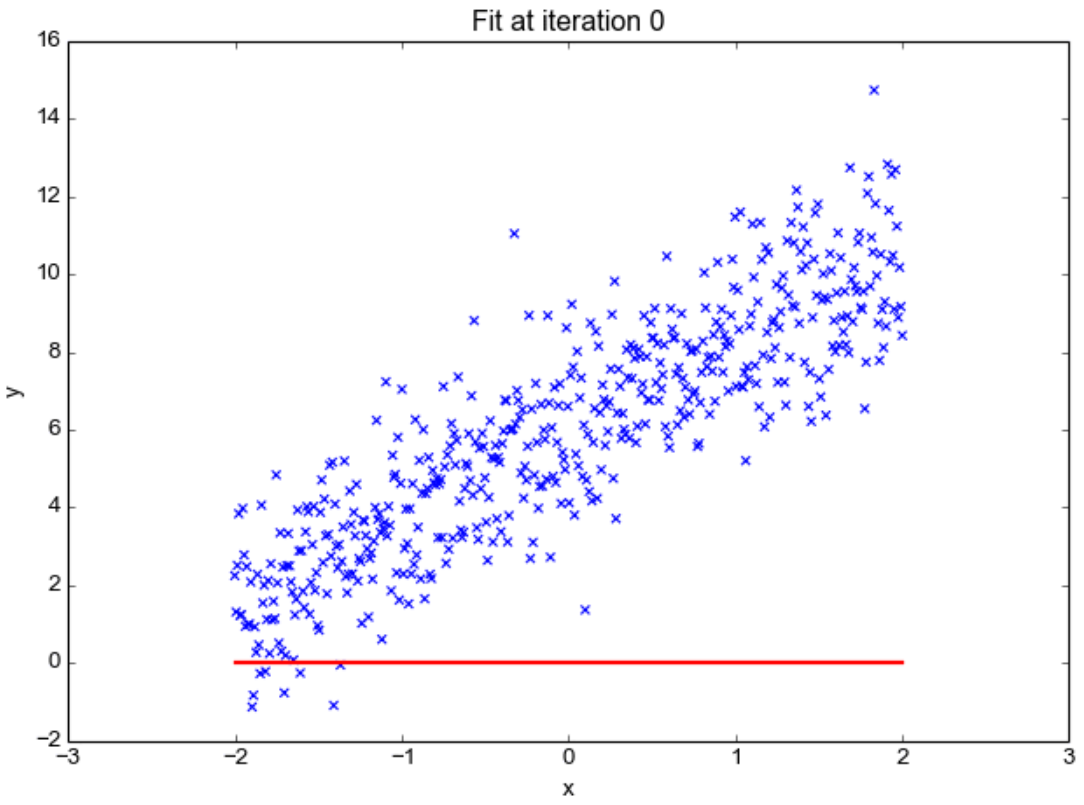
Gradient descent for linear regression

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

Slide credit: Andrew Ng

Batch gradient descent

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

Repeat until convergence{

m: Number of training examples

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

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

}

Slide credit: Andrew Ng

Linear Regression

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

Training dataset

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

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

Slide credit: Andrew Ng

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

$x_3^{(2)} =?$

$x_3^{(4)} =?$

Slide credit: Andrew Ng

Hypothesis

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

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

Slide credit: Andrew Ng

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

- For convenience of notation, define $x_0 = 1$ ($x_0^{(i)} = 1$ for all examples)
- $\boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in R^{n+1}$ $\qquad$ $\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in R^{n+1}$
- $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$
  $= \boldsymbol{\theta}^\top \boldsymbol{x}$

Slide credit: Andrew Ng

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{

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

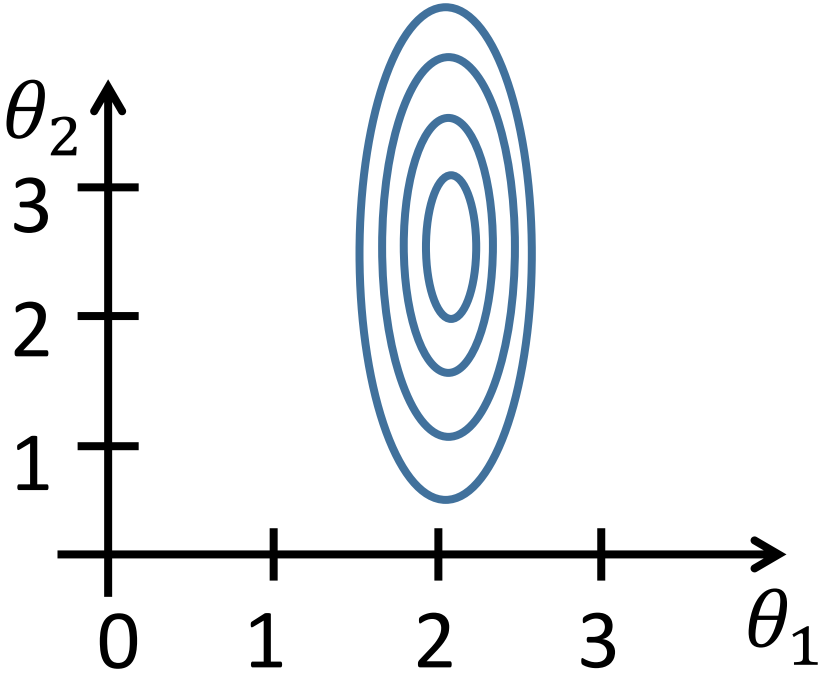
}

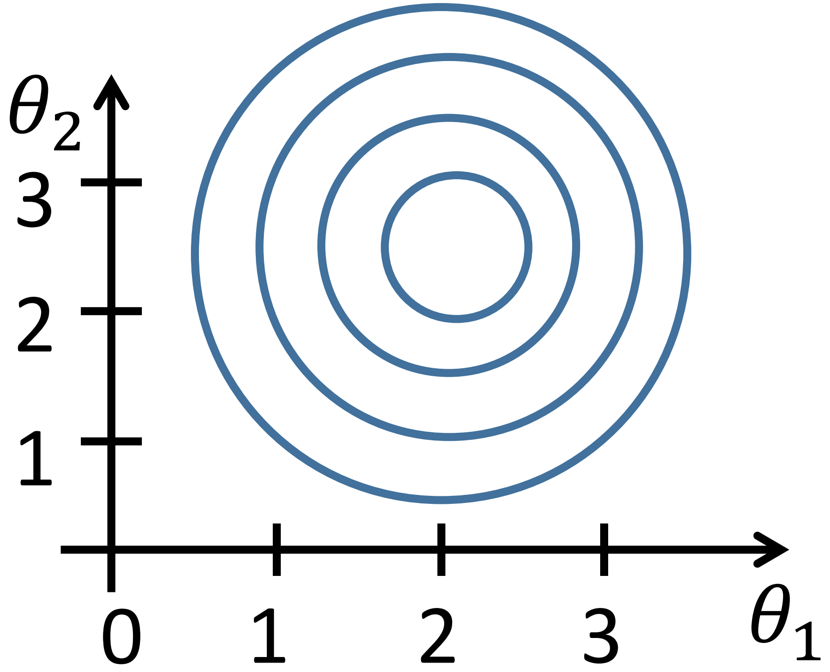
Simultaneously update
θ_j, for $j = 0, 1, \cdots, n$

Slide credit: Andrew Ng

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)





Slide credit: Andrew Ng

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

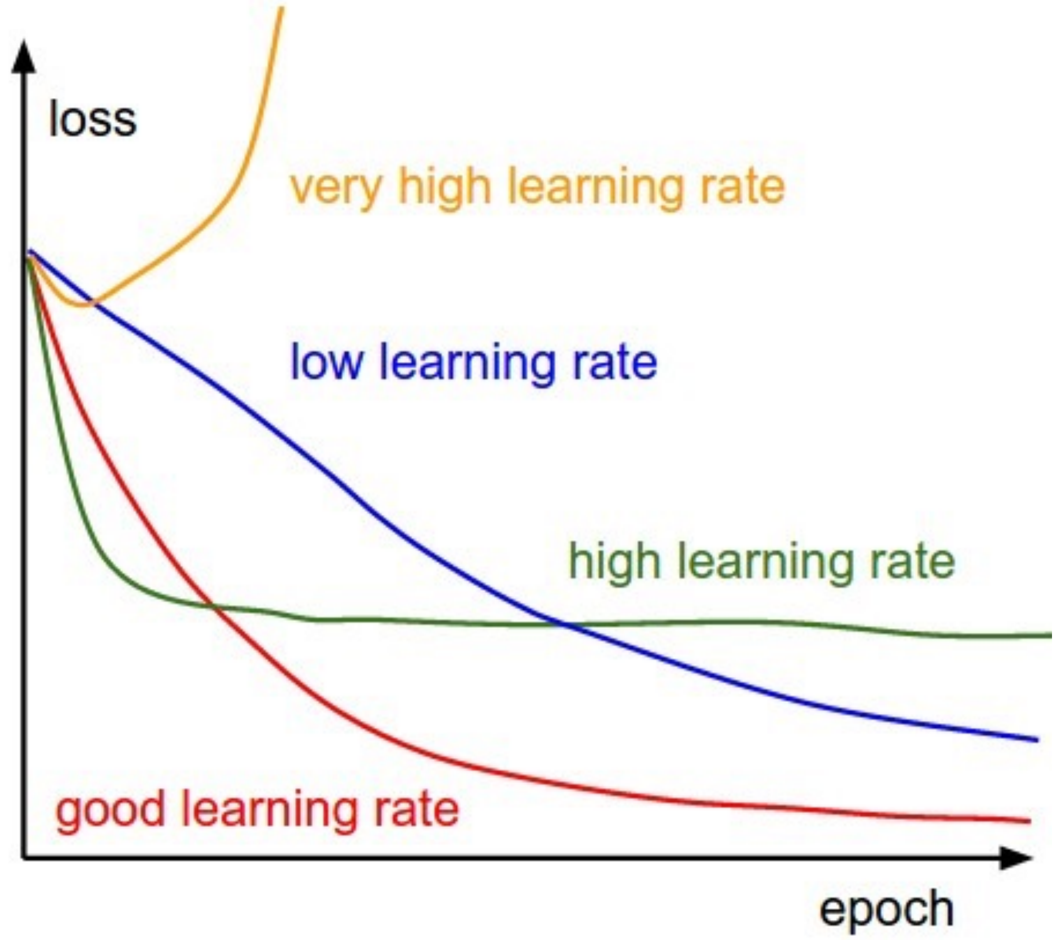


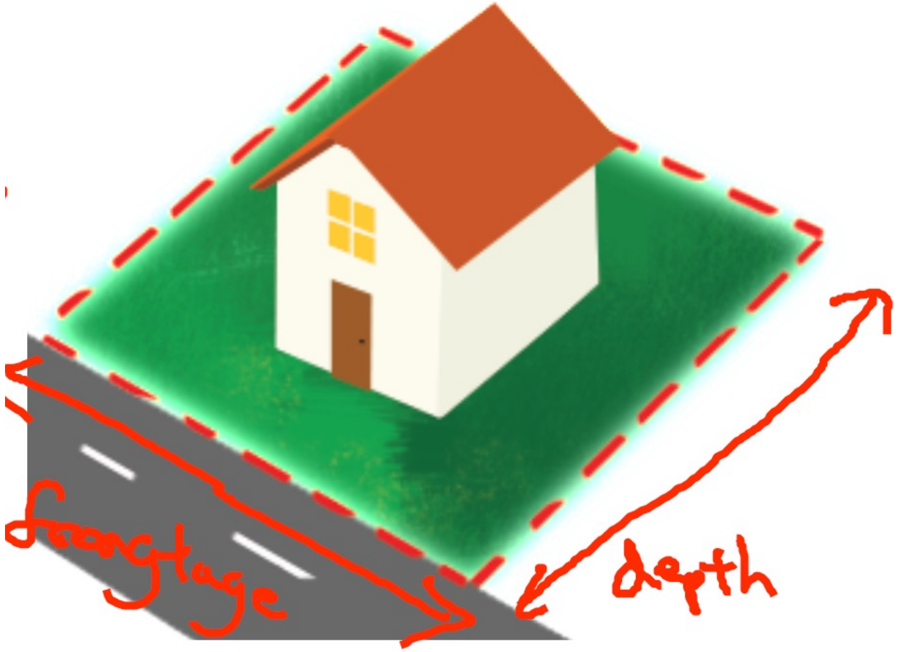
Image credit: CS231n@Stanford

House prices prediction

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

- Area

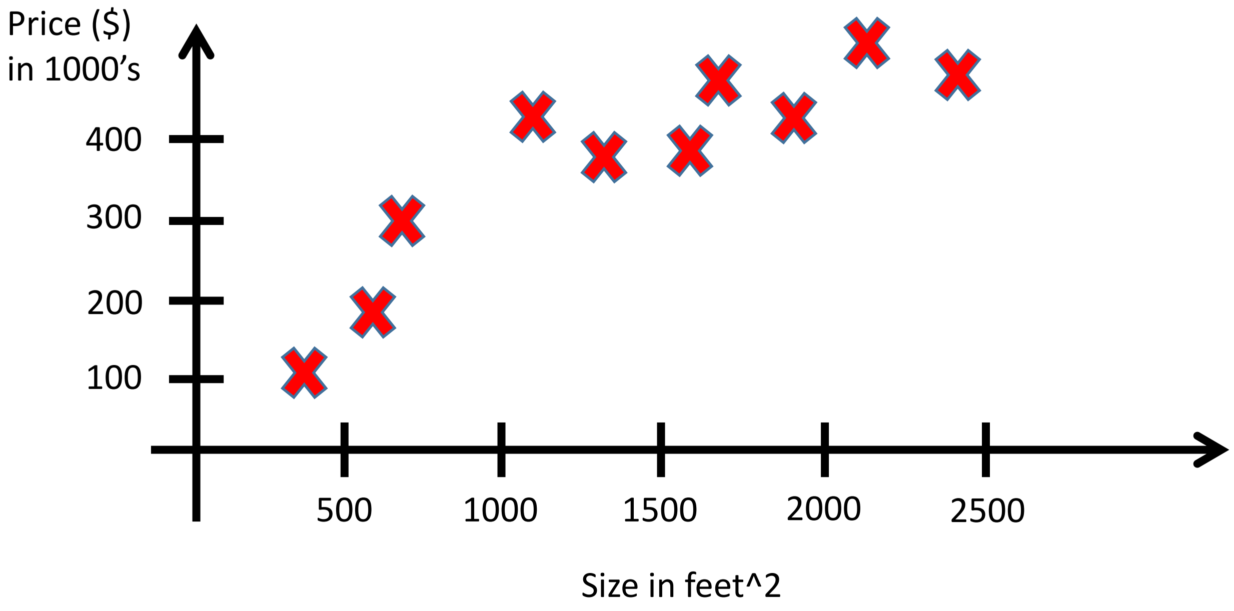
$$x = \text{frontage} \times \text{depth}$$



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

Slide credit: Andrew Ng

Polynomial regression



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

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

Slide credit: Andrew Ng

Linear Regression

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

(x_0)	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
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178
	…				…

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

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

Slide credit: Andrew Ng

Least square solution

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

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

  $= \frac{1}{2m}\|X\theta - y\|_2^2$
- $\frac{\partial}{\partial\theta}J(\theta) = 0$
- $\theta = (X^\top X)^{-1}X^\top y$

Justification/interpretation 1

- **Loss minimization**

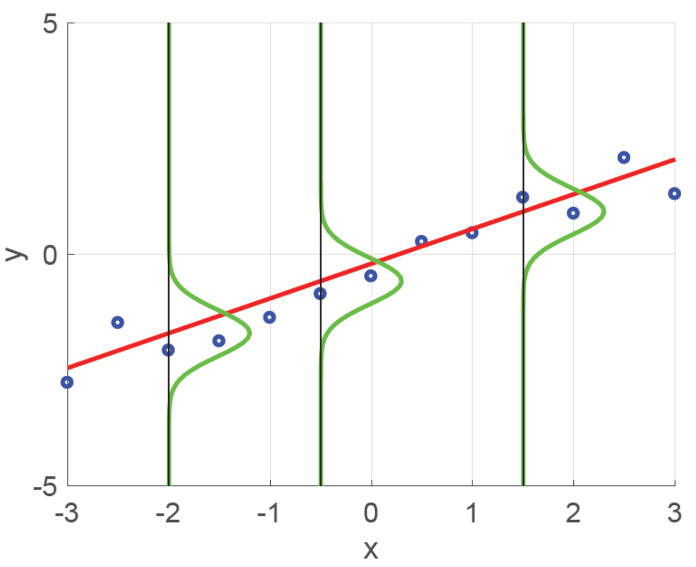
$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 = \frac{1}{m}\sum_{i=1}^{m} L_{ls}\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right)$$

- $L_{ls}(y, \hat{y}) = \frac{1}{2}\|y - \hat{y}\|_2^2$: Least squares loss

- Empirical Risk Minimization (ERM)

$$\frac{1}{m}\sum_{i=1}^{m} L_{ls}\left(y^{(i)}, \hat{y}\right)$$

Justification/interpretation 2



- **Probabilistic model**
- Assume linear model with Gaussian errors

$$p_\theta\left(y^{(i)}\middle|x^{(i)}\right) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{1}{2\sigma^2}\left(y^{(i)} - \theta^\top x^{(i)}\right)\right)$$

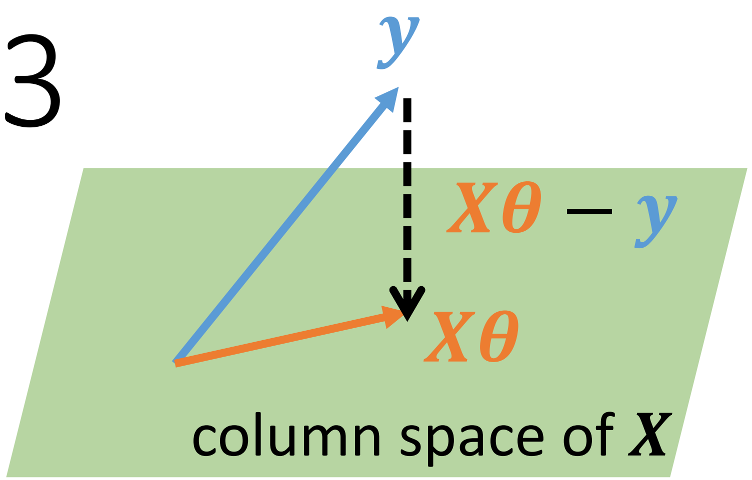
- Solving maximum likelihood

$$\underset{\theta}{\operatorname{argmin}} \prod_{i=1}^{m} p_\theta\left(y^{(i)}\middle|x^{(i)}\right)$$

$$\underset{\theta}{\operatorname{argmin}} \log\left(\prod_{i=1}^{m} p\left(y^{(i)}\middle|x^{(i)}\right)\right) = \underset{\theta}{\operatorname{argmin}}\ \frac{1}{2\sigma^2}\sum_{i=1}^{m}\frac{1}{2}\left(\theta^\top x^{(i)} - y^{(i)}\right)^2$$

Image credit: CS 446@UIUC

Justification/interpretation 3



- **Geometric interpretation**

$$\boldsymbol{X} = \begin{bmatrix} 1 & \leftarrow \boldsymbol{x}^{(1)} \rightarrow \\ 1 & \leftarrow \boldsymbol{x}^{(2)} \rightarrow \\ \vdots & \vdots \\ 1 & \leftarrow \boldsymbol{x}^{(m)} \rightarrow \end{bmatrix} = \begin{bmatrix} \uparrow & \uparrow & \uparrow & \cdots & \uparrow \\ \boldsymbol{z}_0 & \boldsymbol{z}_1 & \boldsymbol{z}_2 & & \boldsymbol{z}_n \\ \downarrow & \downarrow & \downarrow & & \downarrow \end{bmatrix}$$

- $\boldsymbol{X\theta}$: column space of $\boldsymbol{X}$ or span($\{\boldsymbol{z}_0, \boldsymbol{z}_1, \cdots, \boldsymbol{z}_n\}$)

- Residual $\boldsymbol{X\theta} - \mathbf{y}$ is <u>orthogonal</u> to the column space of $\boldsymbol{X}$
- $\boldsymbol{X}^\top(\boldsymbol{X\theta} - \mathbf{y}) = 0 \rightarrow (\boldsymbol{X}^\top\boldsymbol{X})\boldsymbol{\theta} = \boldsymbol{X}^\top\boldsymbol{y}$

m training examples, n features

Gradient Descent

- Need to choose α
- Need many iterations
- Works well even when n is large

Normal Equation

- No need to choose α
- Don't need to iterate
- Need to compute $(X^\top X)^{-1}$
- Slow if n is very large

Slide credit: Andrew Ng

Things to remember

- **Model representation**

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

- **Cost function** $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$
- **Gradient descent for linear regression**

  Repeat until convergence $\{\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}\}$
- **Features and polynomial regression**

  Can combine features; can use different functions to generate features (e.g., polynomial)
- **Normal equation** $\theta = (X^\top X)^{-1}X^\top y$