UCLA

CS M146: Introduction to Machine Learning
Linear Regression

Aditya Grover

UCLA

The instructor gratefully acknowledges Sriram Sankararaman (UCLA) and Andrew Ng (Stanford) for some of the materials used in these slides, and many others who made their course materials freely available online.

https://aditya-grover.github.io/ @adityagrover_

Recap: Overview of ML

Definition: Machine Learning (ML) is the study of algorithms that improve their performance **P,** at some task **T**, with experience **E**.
A well-defined learning task is given by <**T**, **P**, **E**>.
[Tom Mitchell (1998)]

3 Types of Learning:

Supervised, Unsupervised, Reinforcement Learning

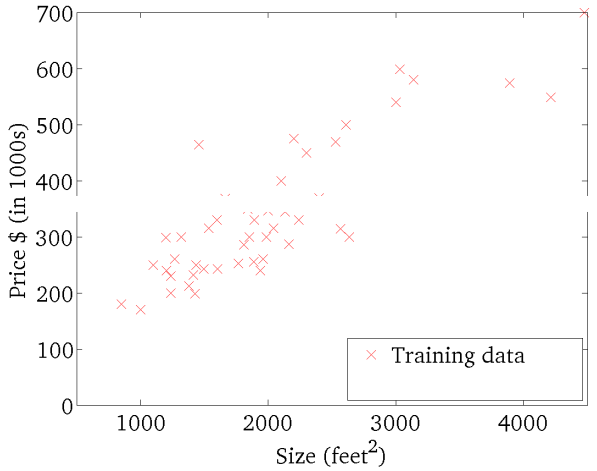
Plan for Today

- Supervised Learning
  - **Linear Regression**

Regression: Example

Housing prices in Portland

Living area (feet^2)	Price (1000$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮



Given a dataset of housing prices, can we learn to predict the prices of other houses in Portland as a function of their living area sizes?

Regression – Problem Setup

Given

A training dataset consisting of n labelled **training instances**

- Input **features/attributes** $\mathbf{X} = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\}$ where $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$
- Corresponding **labels** $\boldsymbol{y} = \{y^{(1)}, \ldots, y^{(n)}\}$ where $y^{(i)} \in \mathbb{R}$

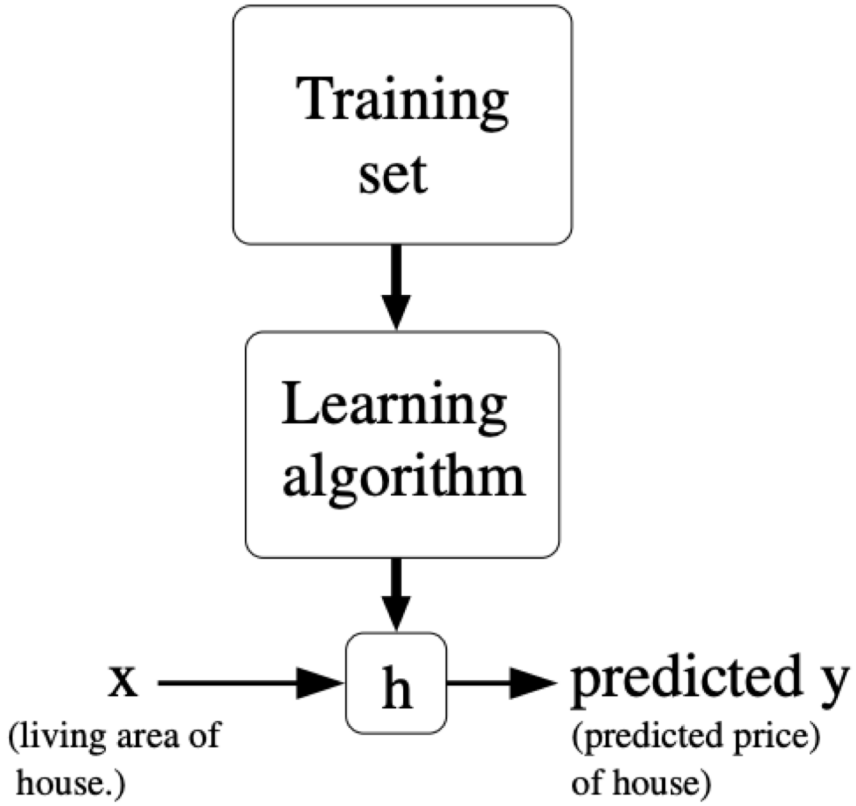
Output

A **hypothesis function** $h: \mathbb{R}^d \rightarrow \mathbb{R}$ from a **hypothesis class** $h \in \mathcal{H}$ such that $h(\boldsymbol{x}) \approx y$

Aside: Notation scheme for the course

- Every time we see a new term, it will be colored **blue**
- Vectors are always in **bold**, e.g., $\boldsymbol{x}^{(1)}, \boldsymbol{\theta}$
- Vector elements and scalars are not in bold, e.g., $x_1^{(1)}, \theta_2, y^{(3)}$
- Why $\boldsymbol{x}^{(1)}$ as opposed to:
  - $\boldsymbol{x}^1$ to avoid confusion with exponentiation
  - $\boldsymbol{x_1}$ to avoid confusion with indexing of vector elements
- **If you spot a typo, let us know and we'll fix it!**

Regression – 3 Step Recipe



Steps for learning h

1. Represent hypothesis class $\mathcal{H}$
2. Define a **loss**
3. Optimize loss to find best $h \in \mathcal{H}$

Linear Regression – Step 1

- In **linear** regression, we represent hypothesis class $\mathcal{H}$ as the set of all **parameterized linear functions**

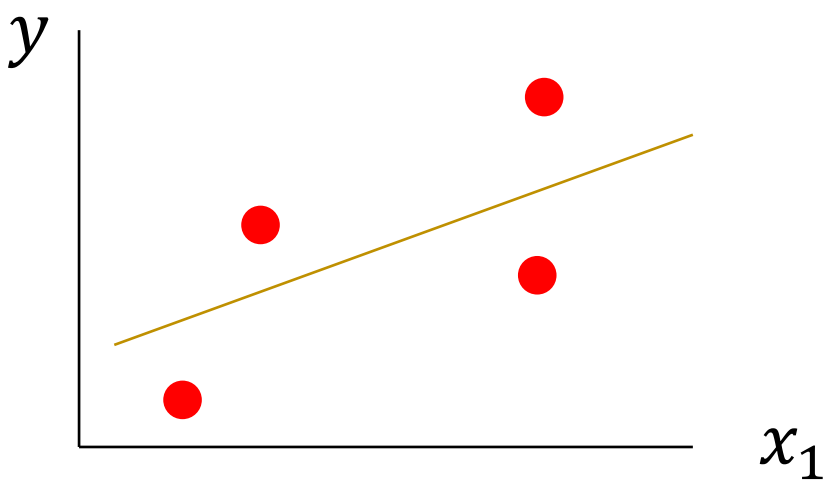
$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d$$

- **Note:** linearity is defined with respect to θ_i's (and not x_i's)
- $\boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_d] \in \mathbb{R}^{d+1}$ are the **parameters** of hypothesis class $\mathcal{H}$.
- Alternate terminologies:
  - θ_0 : **bias**
  - $\theta_{1:d}$: **weights**
- **Vectorization:** Define $x_0 = 1$ such that $\boldsymbol{x} \in \mathbb{R}^{d+1}$

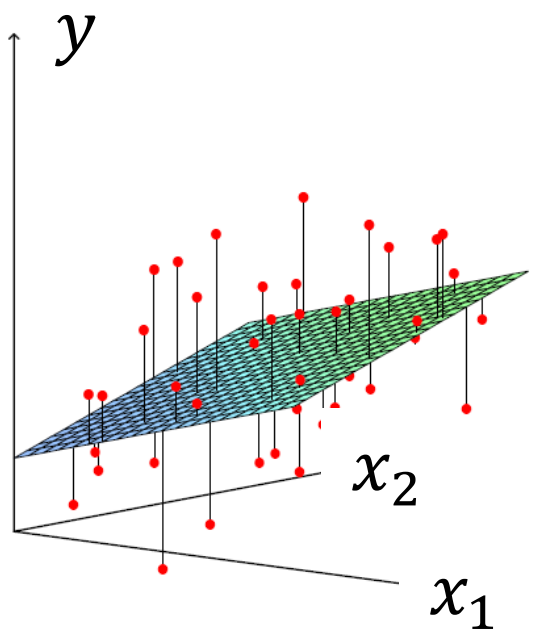
$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=0}^{d} \theta_j x_j = \boldsymbol{\theta}^T \boldsymbol{x}$$

Geometric Intuition

$d = 1, h_{\boldsymbol{\theta}}(\boldsymbol{x})$: straight line



$d = 2, h_{\boldsymbol{\theta}}(\boldsymbol{x})$: plane



$d > 2, h_{\boldsymbol{\theta}}(\boldsymbol{x})$: hyperplane

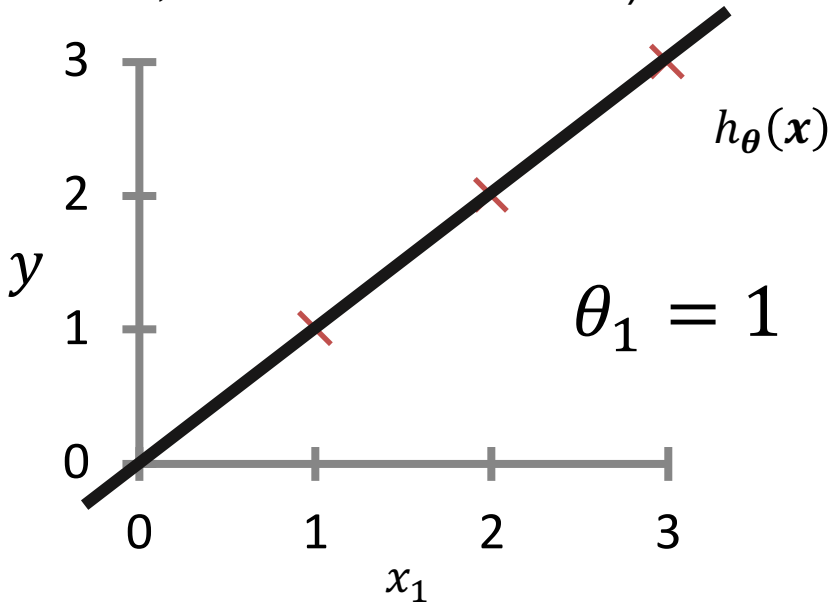
Intuition Behind Loss Function

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

For intuition, let us assume $d = 1$ and $\theta_0 = 0$. Hence $\boldsymbol{\theta} = [0, \theta_1]$.

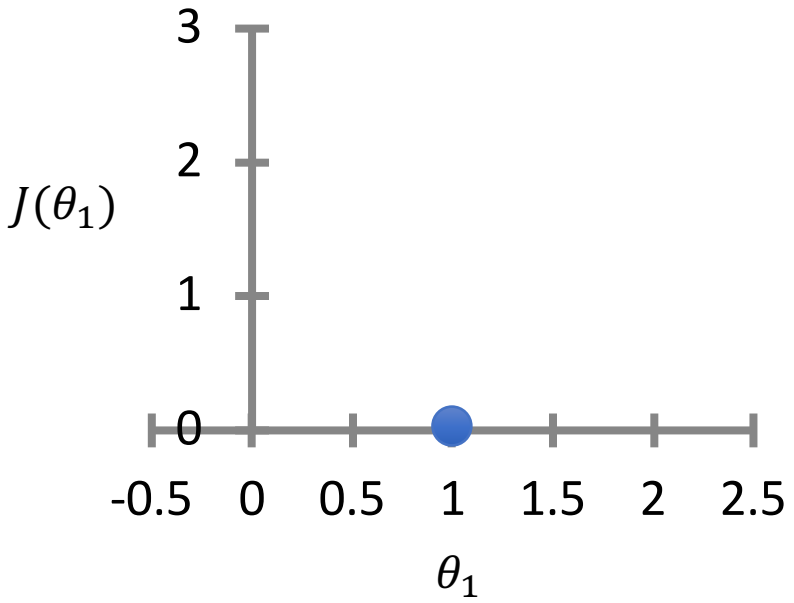
$h_{\boldsymbol{\theta}}(\boldsymbol{x})$

(for fixed $\boldsymbol{\theta}$, this is a function of $\boldsymbol{x}$)



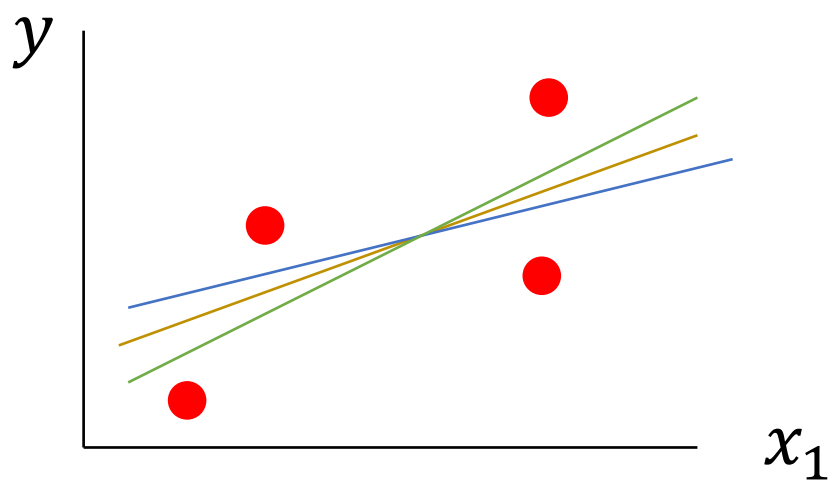
$J(\theta_1)$

(function of the parameter θ_1)



Based on example by Andrew Ng

Linear Regression – What Next?



Steps for learning h

1. Represent hypothesis class $\mathcal{H}$
2. **Define a loss objective**
3. Minimize loss to find best $h \in \mathcal{H}$

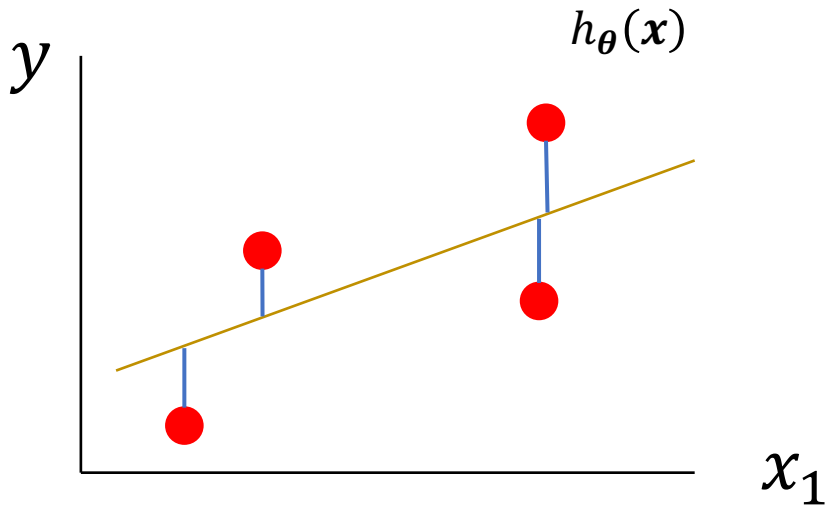
Many hypothesis in $\mathcal{H}$.

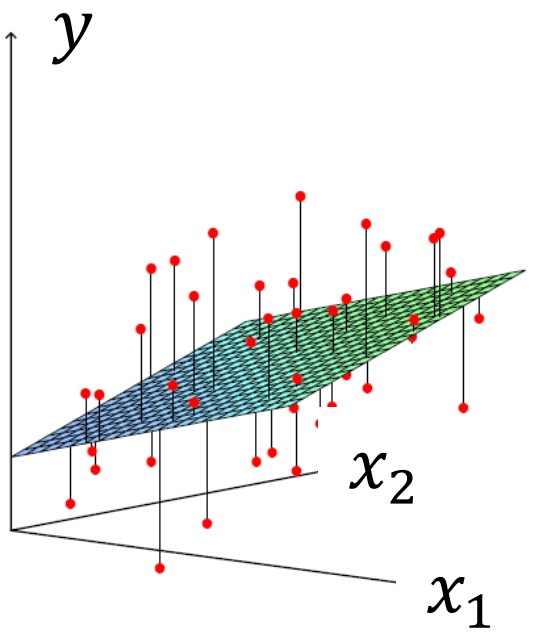
Which one is best hypothesis? – Step 2

Least Squares Linear Regression – Step 2

- **Loss function:** A way to score hypothesis. Lower loss is better.
- Least squares loss function

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





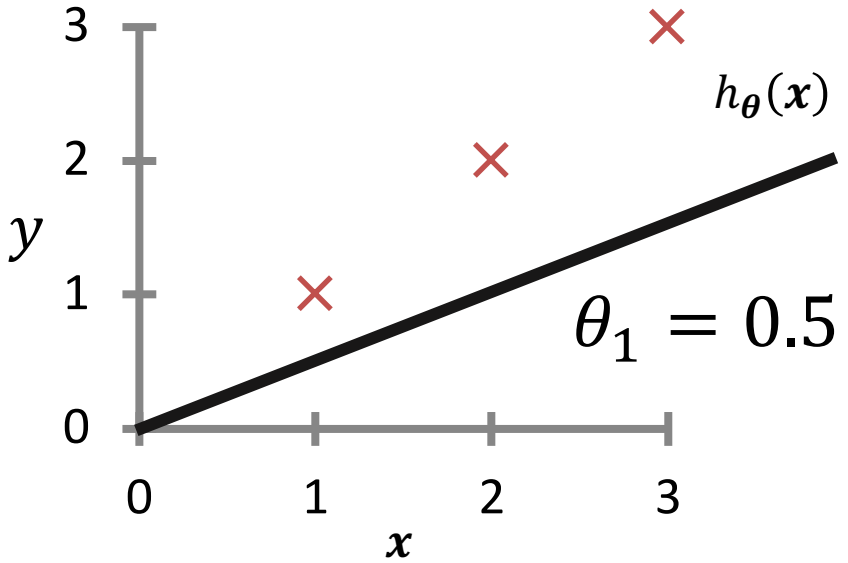
Intuition Behind Loss Function

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

For intuition, let us assume $d = 1$ and $\theta_0 = 0$ Hence $x \in \mathbb{R}$ so $\boldsymbol{\theta} = [0, \theta_1]$.

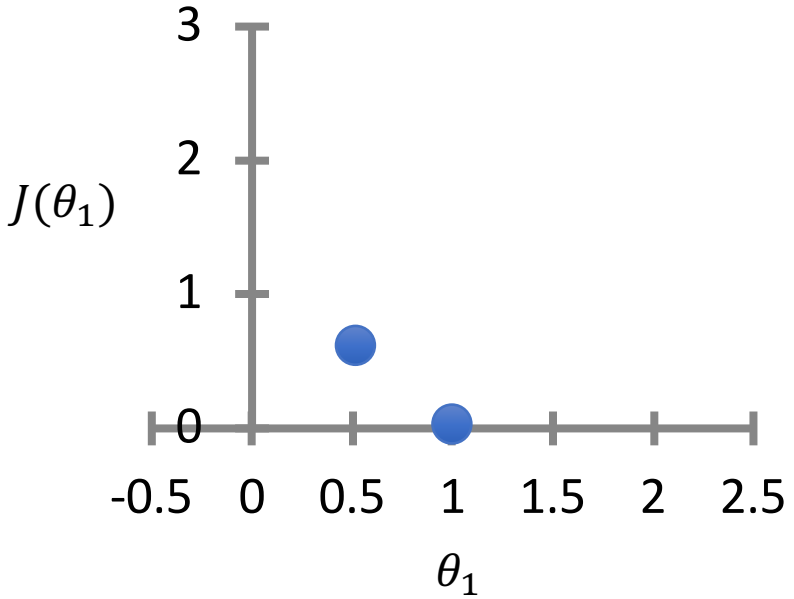
$h_{\boldsymbol{\theta}}(\boldsymbol{x})$

(for fixed $\boldsymbol{\theta}$, this is a function of $\boldsymbol{x}$)



$J(\theta_1)$

(function of the parameter θ_1)



$$J([0, 0.5]) = \frac{1}{2 \times 3}\left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right] \approx 0.58$$

Based on example by Andrew Ng

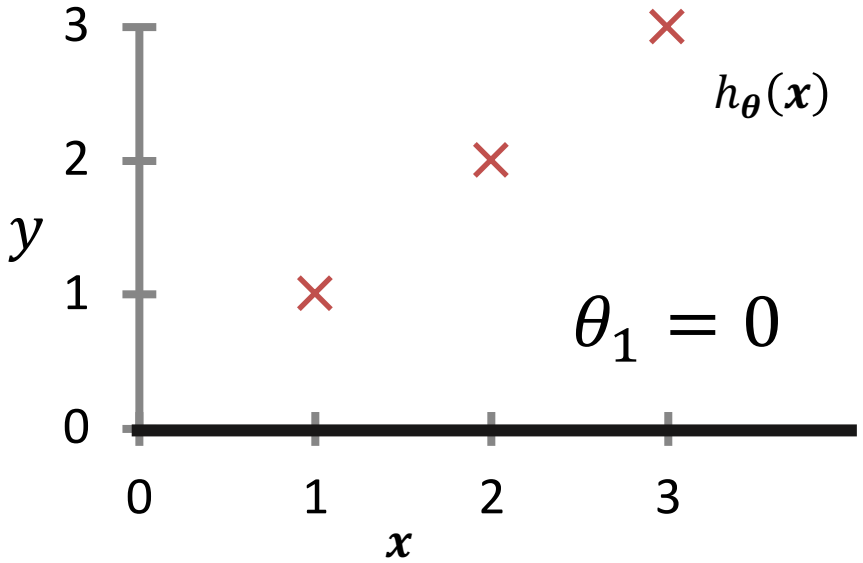
Intuition Behind Loss Function

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

For intuition, let us assume $d = 1$ and $\theta_0 = 0$ Hence $x \in \mathbb{R}$ so $\boldsymbol{\theta} = [0, \theta_1]$.

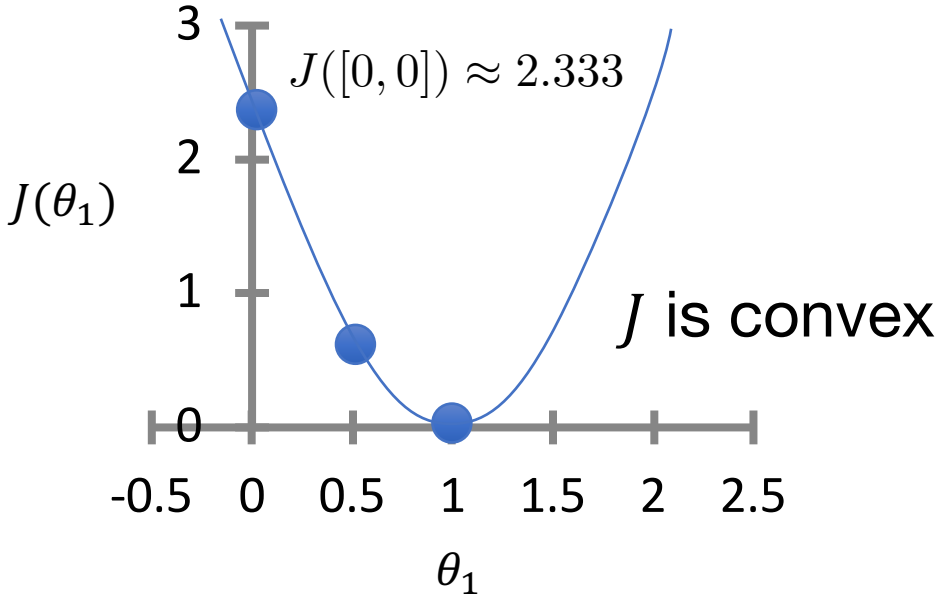
$h_{\boldsymbol{\theta}}(\boldsymbol{x})$

(for fixed $\boldsymbol{\theta}$, this is a function of $\boldsymbol{x}$)



$J(\theta_1)$

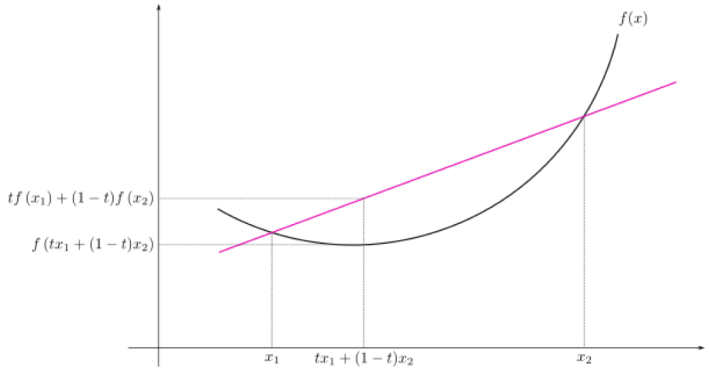
(function of the parameter θ_1)



Based on example by Andrew Ng

Background: Convex Functions

- **Definition:** A real-valued function f is **convex** if the line segment joining any two points lies above the function



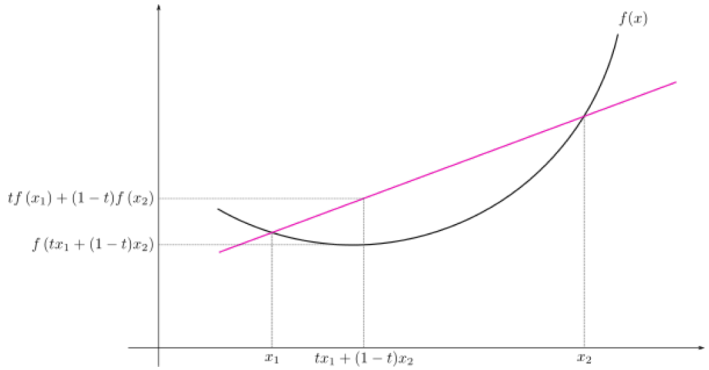
- A function f is convex iff for all $0 \leq t \leq 1$ and all $x_1, x_2 \in X$:

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$$

- For **strictly convex** functions, replace $\leq$ by $<$ above

Background: Convex Functions

- **(Stated without proof)** Convex functions have a unique optima

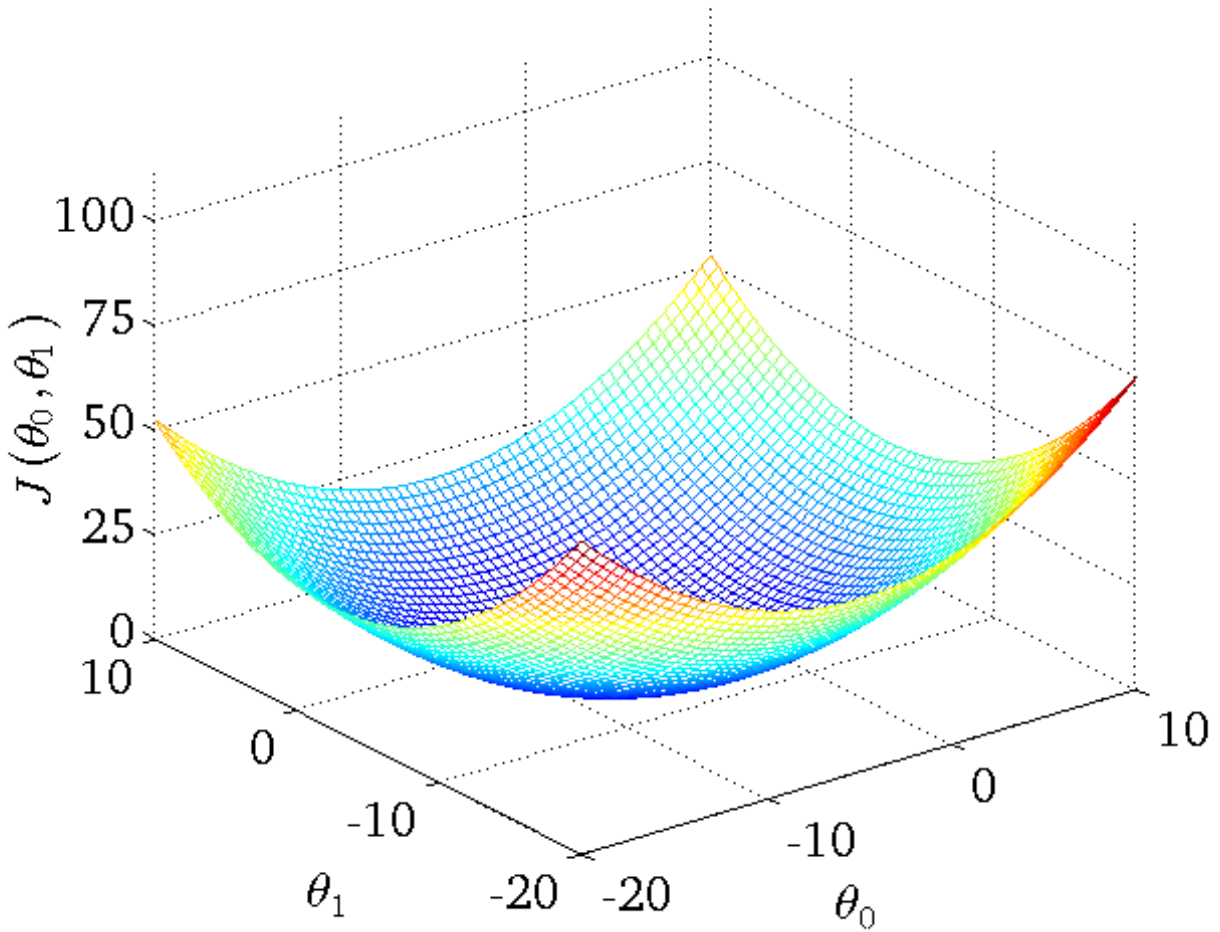


- **(Stated without proof)**

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

$J(\boldsymbol{\theta})$ is **convex** and **differentiable** w.r.t. $\boldsymbol{\theta}$

Intuition Behind Loss Function

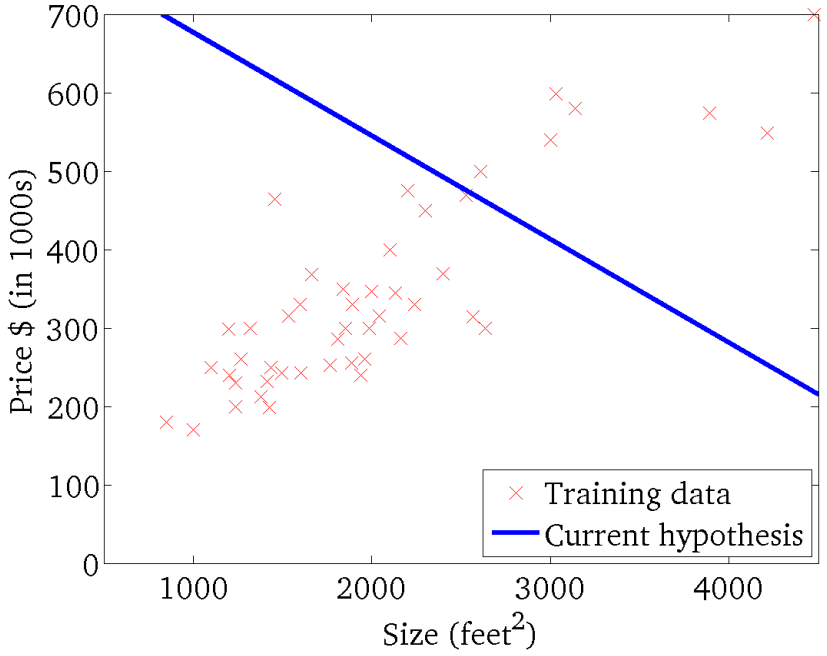


Slide by Andrew Ng

Intuition Behind Loss Function

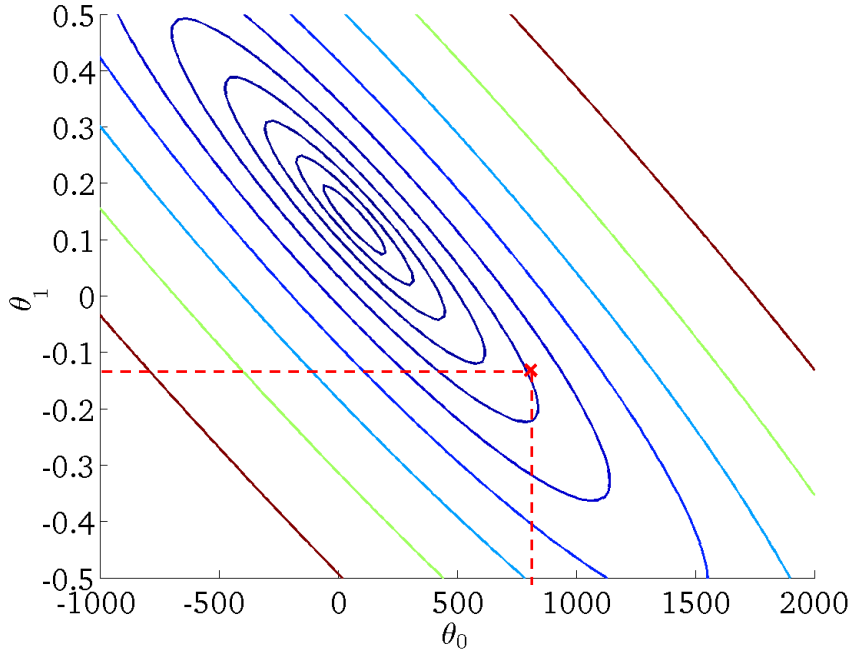
$h_{\boldsymbol{\theta}}(\boldsymbol{x})$

(for fixed θ_0, θ_1, this is a function of $\boldsymbol{x}$)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

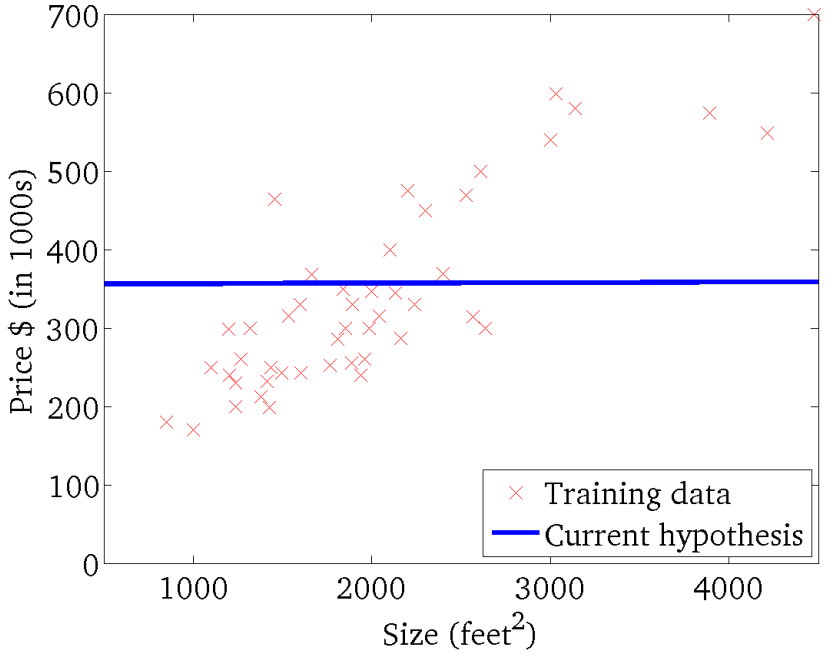


Slide by Andrew Ng

Intuition Behind Loss Function

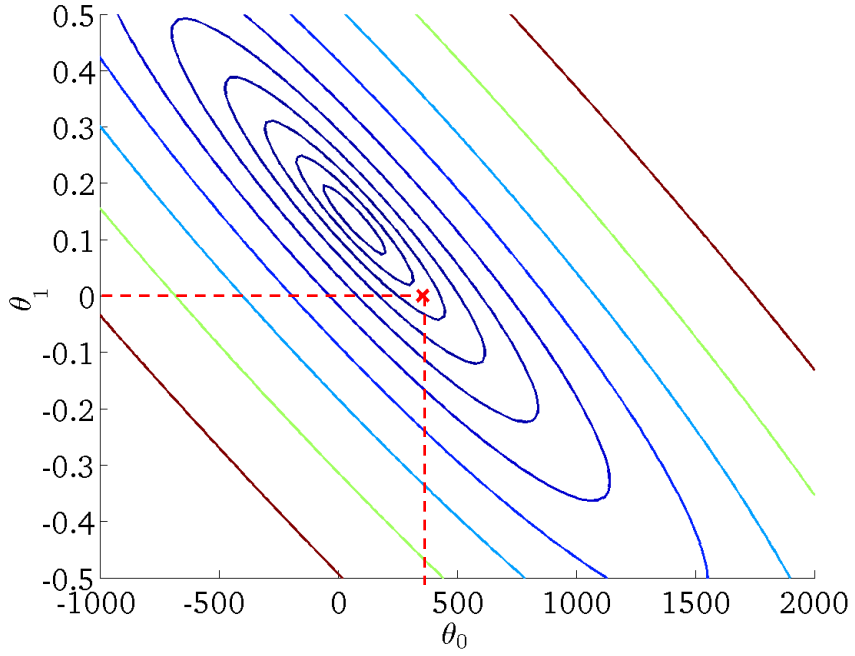
$h_{\boldsymbol{\theta}}(\boldsymbol{x})$

(for fixed θ_0, θ_1, this is a function of $\boldsymbol{x}$)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

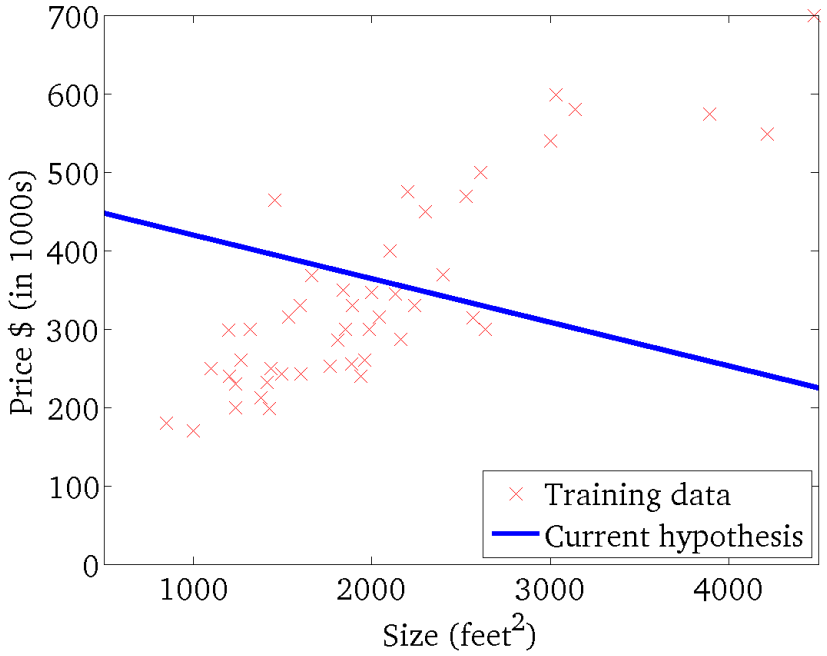


Slide by Andrew Ng

Intuition Behind Loss Function

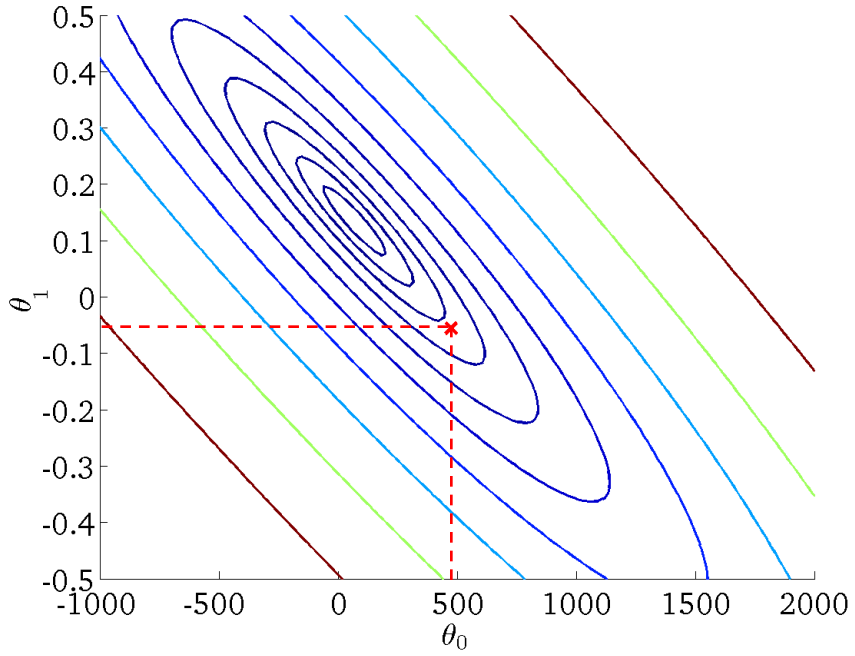
$h_{\boldsymbol{\theta}}(\boldsymbol{x})$

(for fixed θ_0, θ_1, this is a function of $\boldsymbol{x}$)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

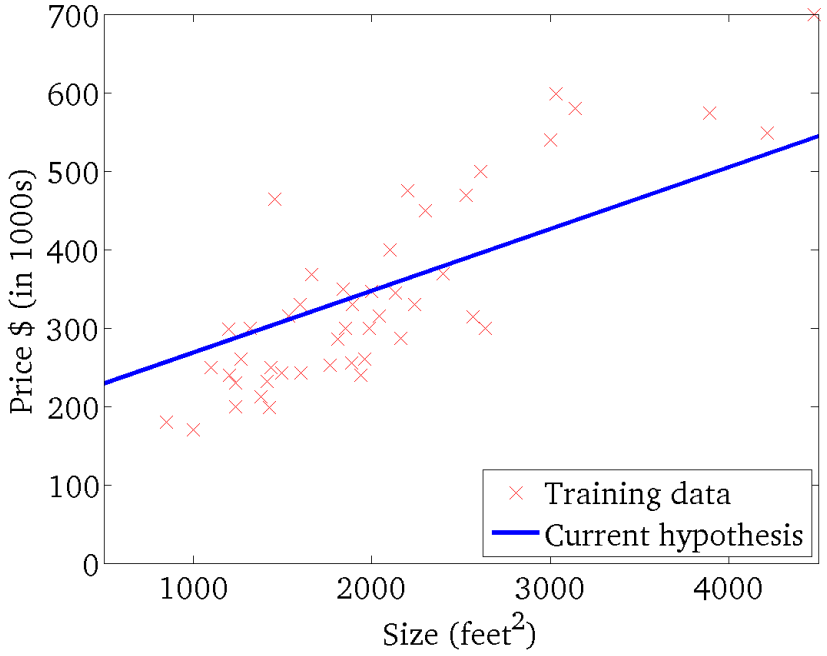


Slide by Andrew Ng

Intuition Behind Loss Function

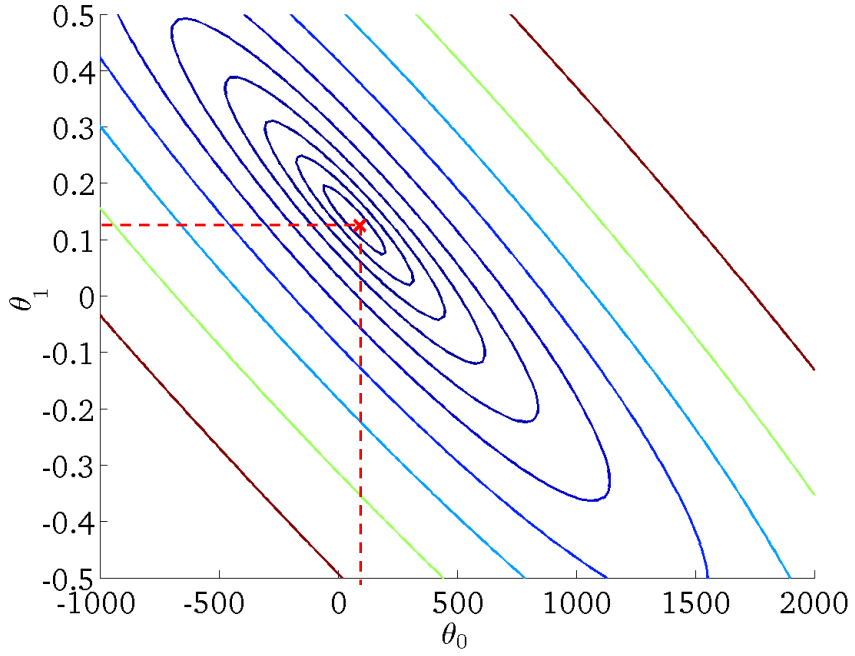
$h_{\boldsymbol{\theta}}(\boldsymbol{x})$

(for fixed θ_0, θ_1, this is a function of $\boldsymbol{x}$)



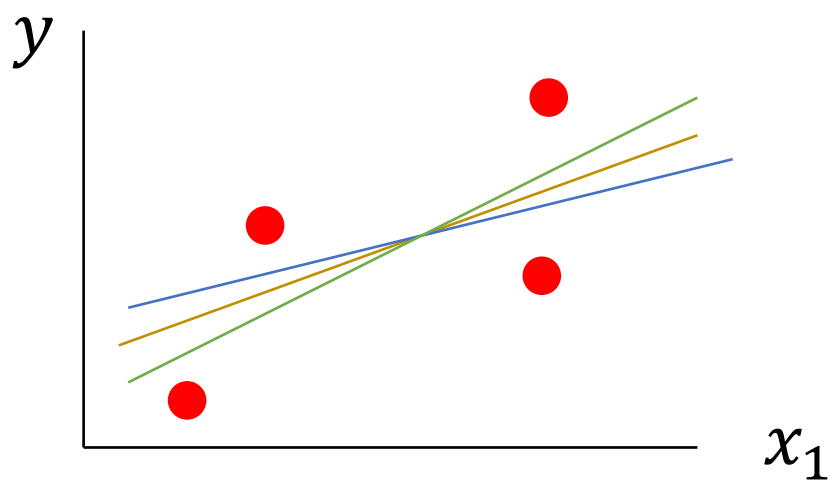
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



Slide by Andrew Ng

Linear Regression – What Next?



Steps for learning h

1. Represent hypothesis class $\mathcal{H}$
2. Define a loss objective
3. **Minimize loss to find best $h \in \mathcal{H}$**

Many hypothesis in $\mathcal{H}$.

How to find the best hypothesis? – Step 3

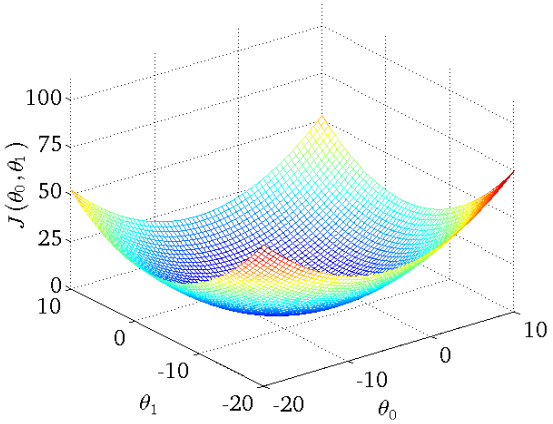
Optimizing Loss Functions – Step 3

- Least squares loss function

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

- Fit by solving

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

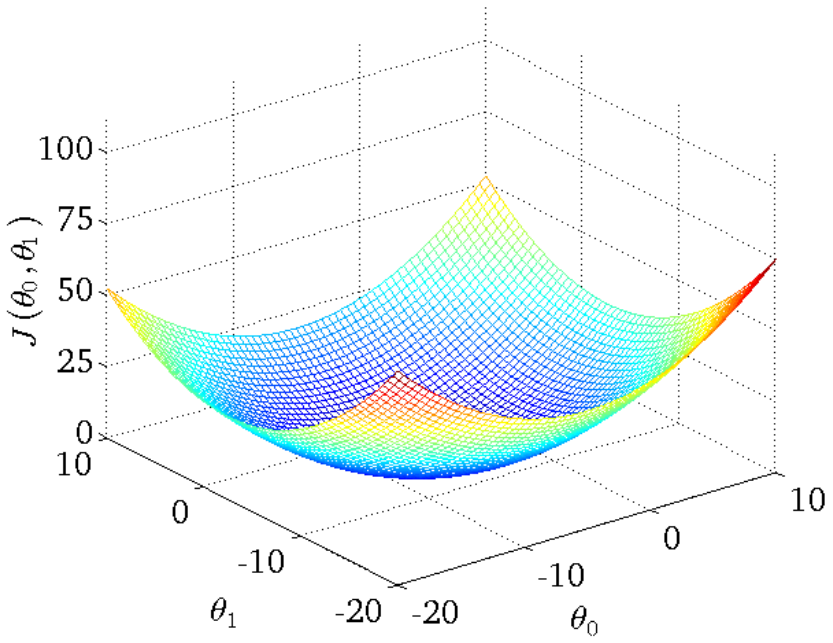


Two optimization algorithms:

- **Gradient Descent**
- **Normal Equation**

Gradient Descent - Intuition

Think of loss at any point as its height above ground level



How will you minimize it?

Gradient Descent - Basic Search Procedure

- Choose initial value for $\boldsymbol{\theta}$
- Until we reach a minimum:
  - Choose a new value for $\boldsymbol{\theta}$ to reduce $J(\boldsymbol{\theta})$

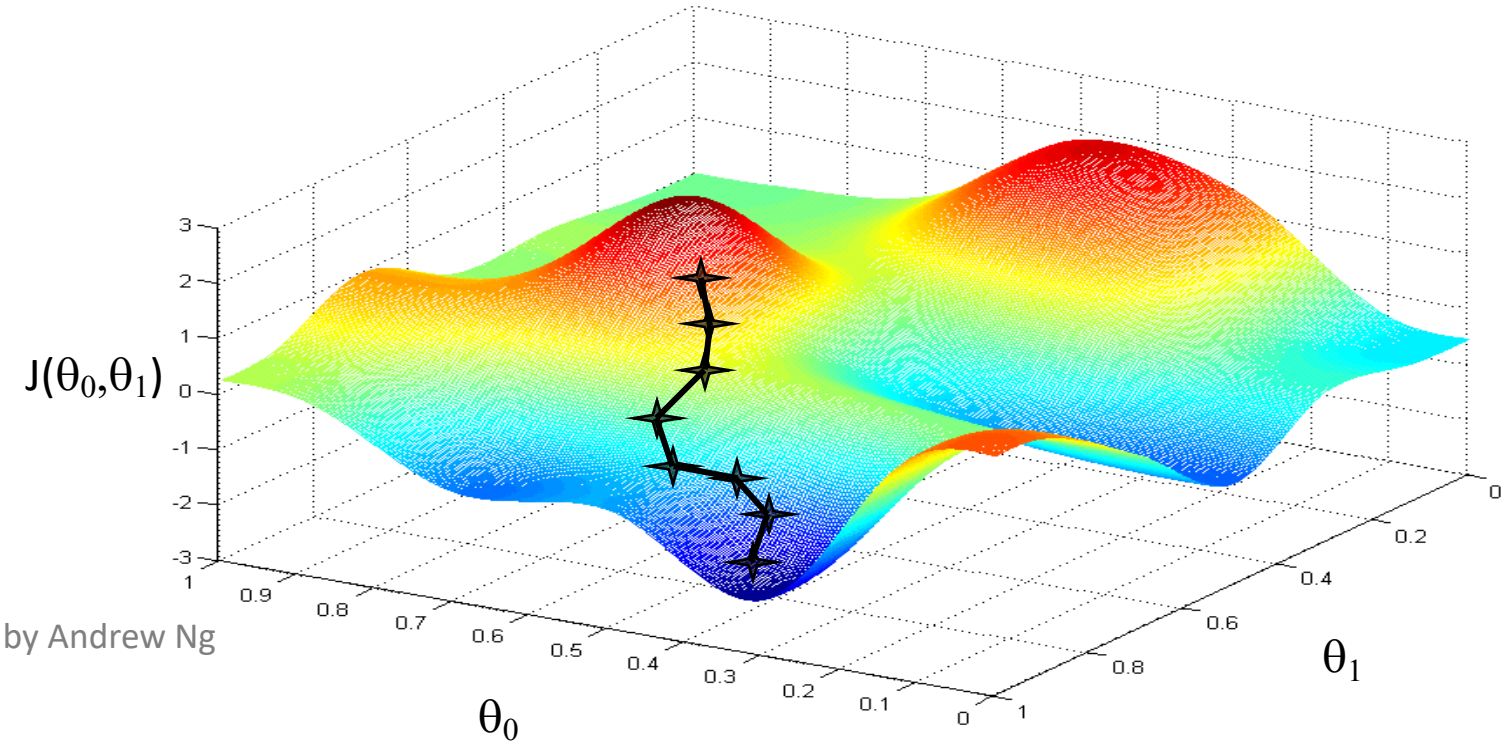
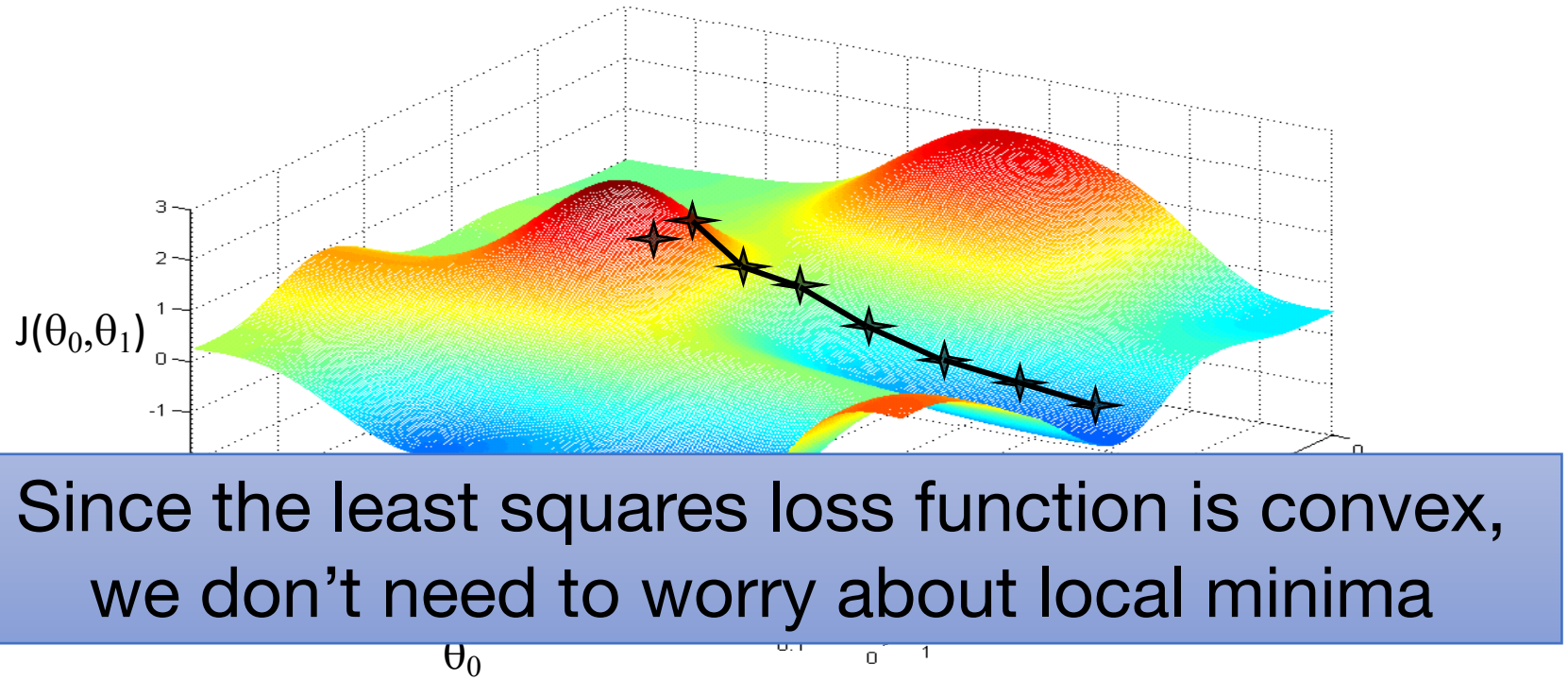


Figure by Andrew Ng

Basic Search Procedure

- Choose initial value for $\boldsymbol{\theta}$
- Until we reach a minimum:
  - Choose a new value for $\boldsymbol{\theta}$ to reduce $J(\boldsymbol{\theta})$



Since the least squares loss function is convex, we don't need to worry about local minima

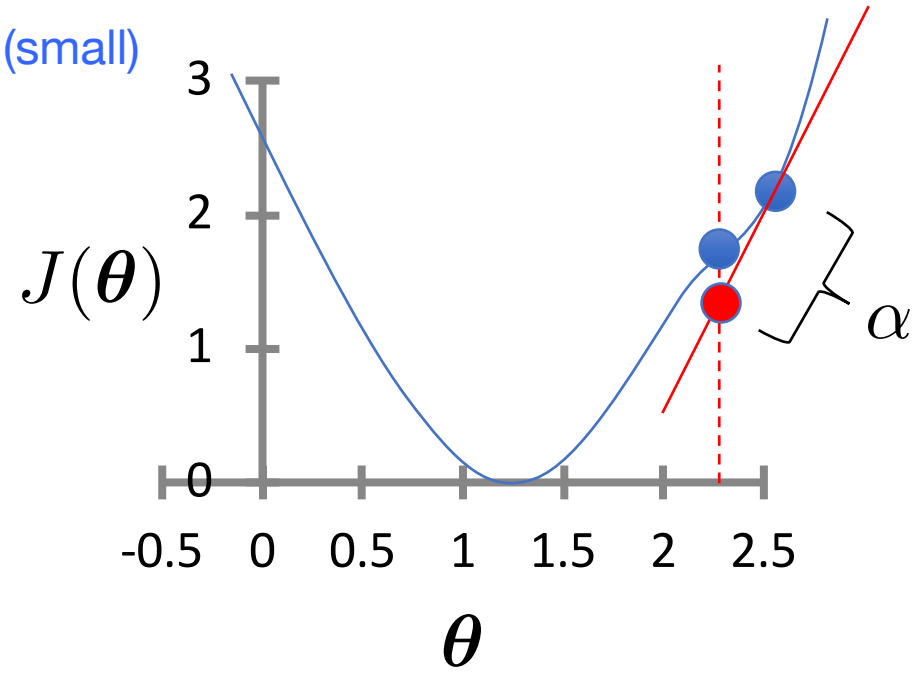
Gradient Descent

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

simultaneous update
for j = 0 ... d

step size or learning rate (small)
e.g., α = 0.05



Gradient Descent

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

simultaneous update
for j = 0 ... d

For Linear Regression:

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

Gradient Descent

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

simultaneous update
for j = 0 ... d

For Linear Regression:

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

$$= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right)^2$$

Gradient Descent

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

simultaneous update
for j = 0 ... d

For Linear Regression:

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)^2 \\
&= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right)^2 \\
&= \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right) \times \frac{\partial}{\partial \theta_j} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right)
\end{aligned}$$

Gradient Descent

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

simultaneous update
for j = 0 ... d

For Linear Regression:

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)^2 \\
&= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right)^2 \\
&= \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right) \times \frac{\partial}{\partial \theta_j} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right) \\
&= \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right) x_j^{(i)}
\end{aligned}$$

Gradient Descent

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

simultaneous update
for j = 0 ... d

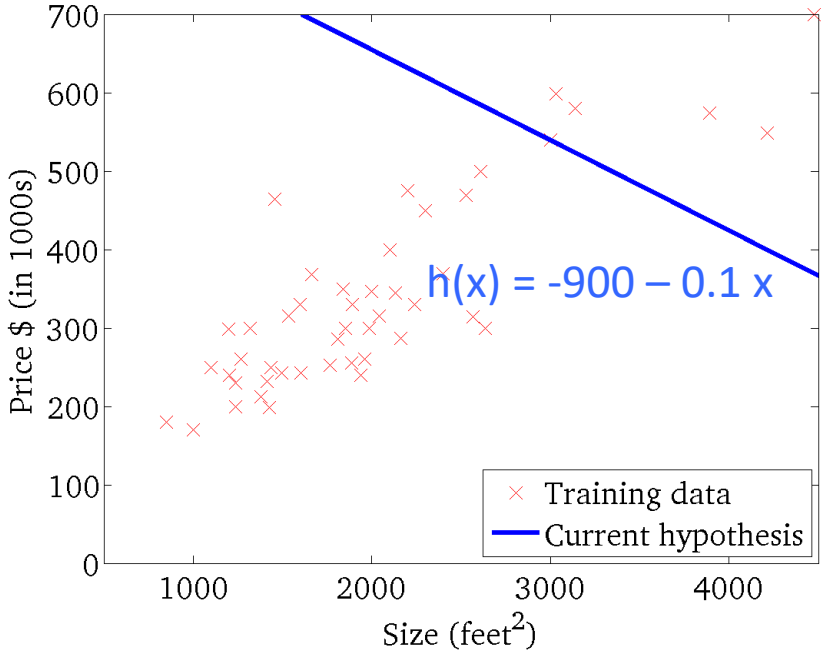
- To achieve simultaneous update
  - At the start of each GD iteration, compute $h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)$
  - Use this stored value in the update step loop
- Assume convergence when $\|\boldsymbol{\theta}_{new} - \boldsymbol{\theta}_{old}\|_2 < \epsilon$

L_2 norm: $\|\boldsymbol{v}\|_2 = \sqrt{\sum_i v_i^2} = \sqrt{v_1^2 + v_2^2 + \ldots + v_{|v|}^2}$

Gradient Descent

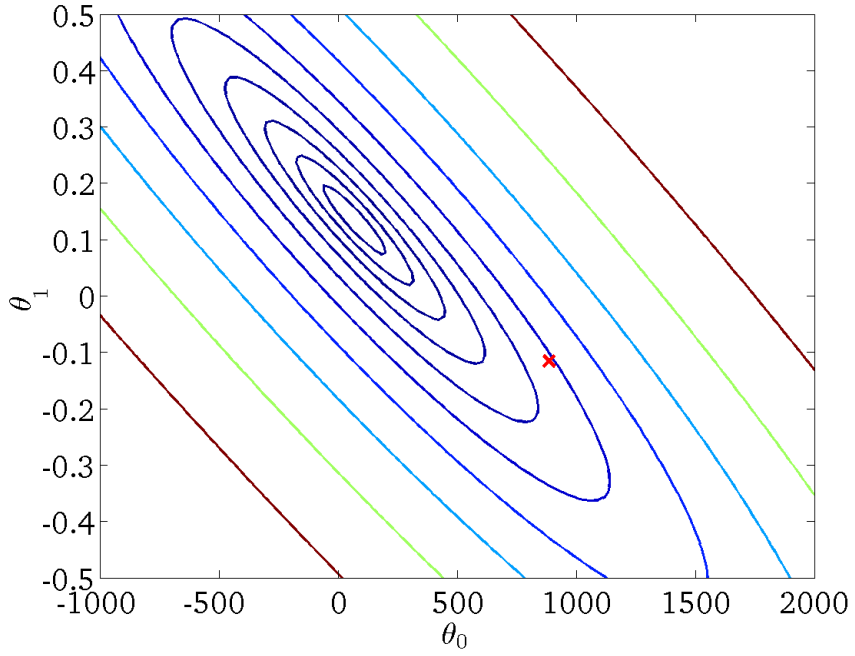
$h_\theta(x)$

(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)

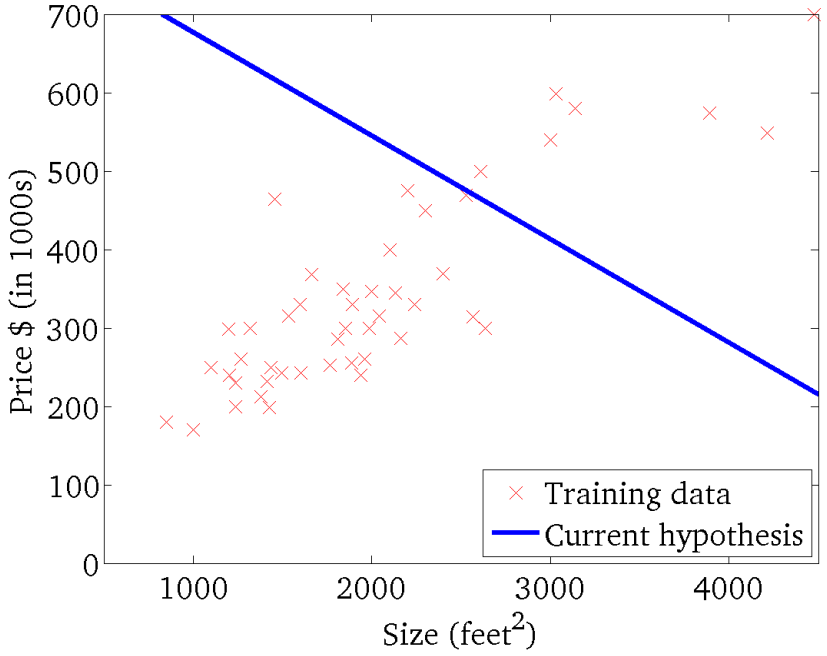


Slide by Andrew Ng

Gradient Descent

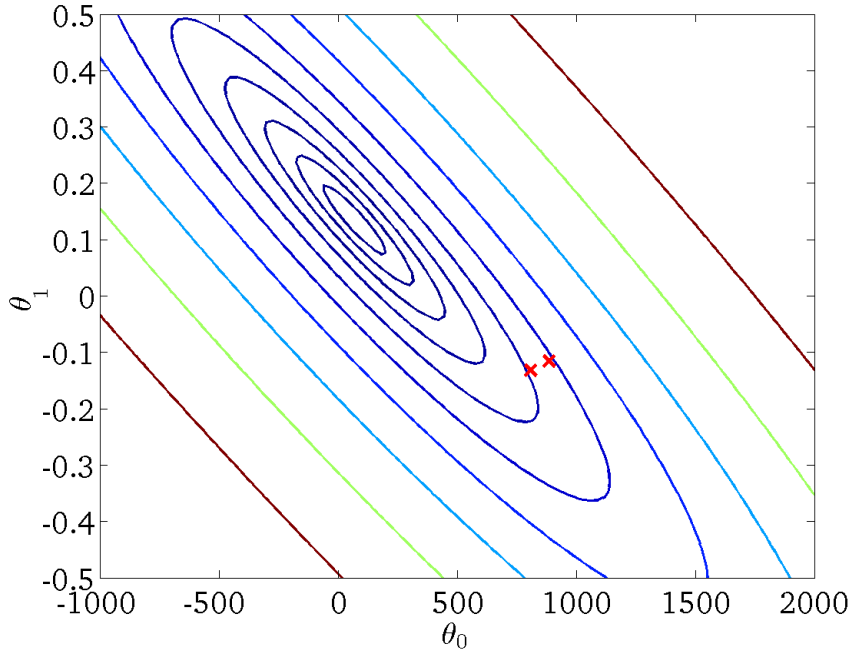
$h_{\boldsymbol{\theta}}(\boldsymbol{x})$

(for fixed θ_0, θ_1, this is a function of $\boldsymbol{x}$)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

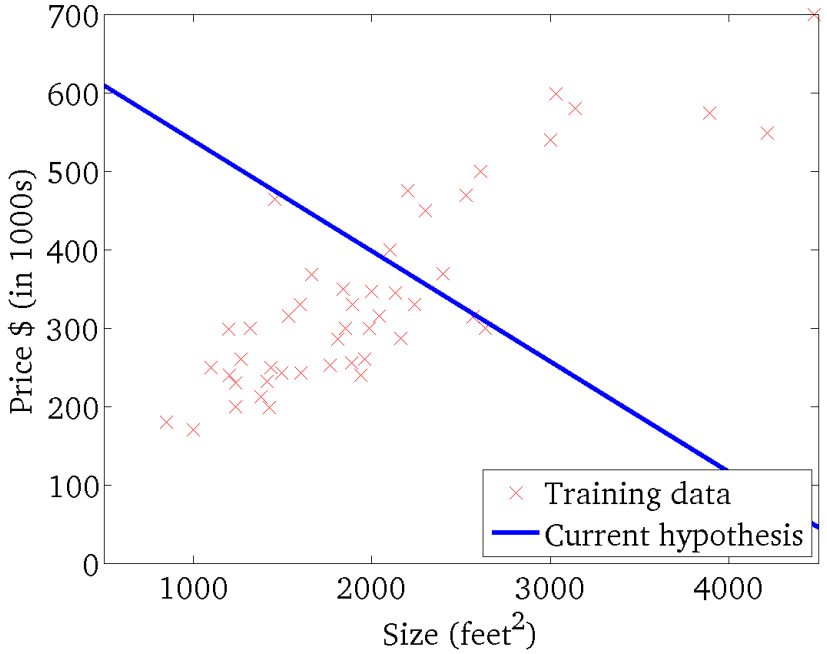


Slide by Andrew Ng

Gradient Descent

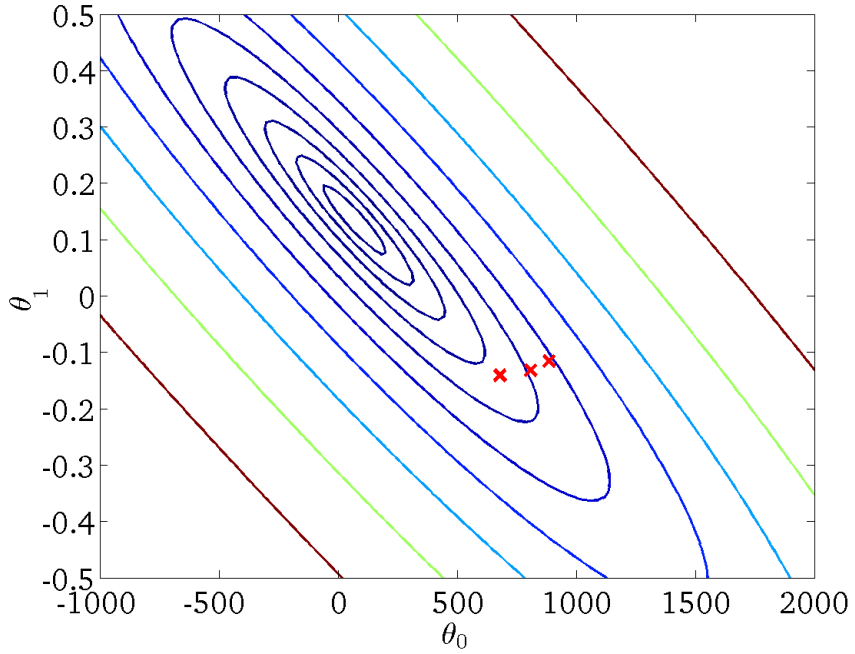
$h_{\boldsymbol{\theta}}(\boldsymbol{x})$

(for fixed θ_0, θ_1, this is a function of $\boldsymbol{x}$)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

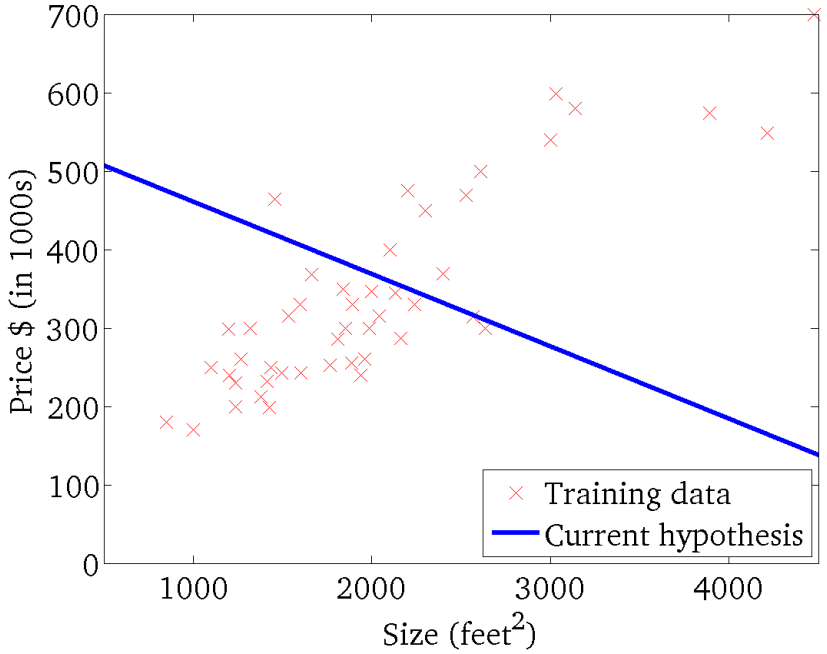


Slide by Andrew Ng

Gradient Descent

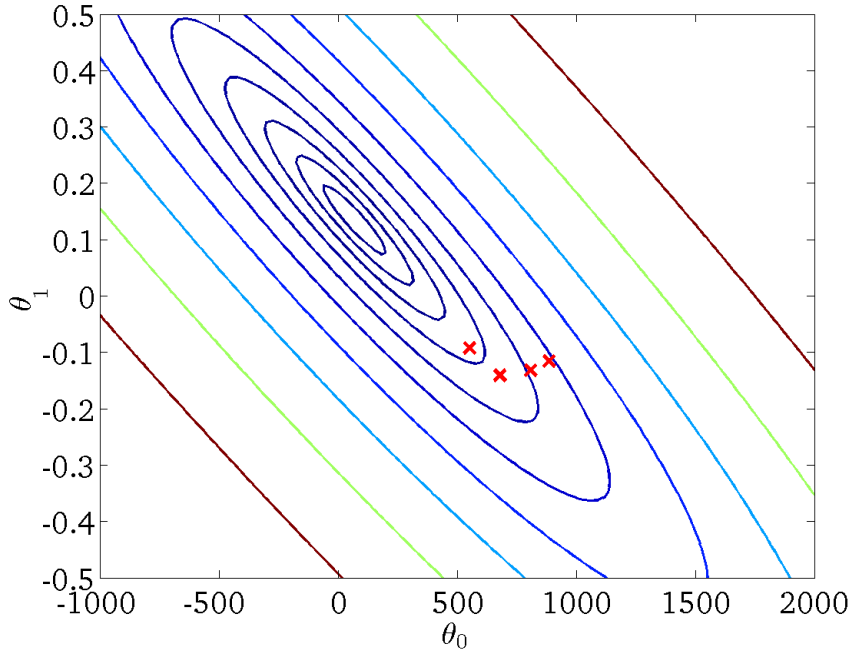
$h_{\boldsymbol{\theta}}(\boldsymbol{x})$

(for fixed θ_0, θ_1, this is a function of $\boldsymbol{x}$)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

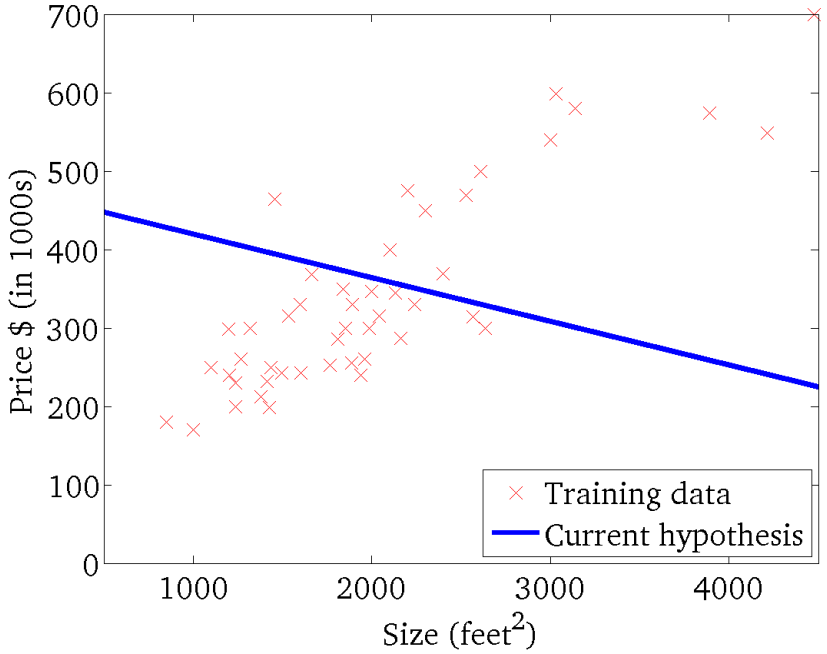


Slide by Andrew Ng

Gradient Descent

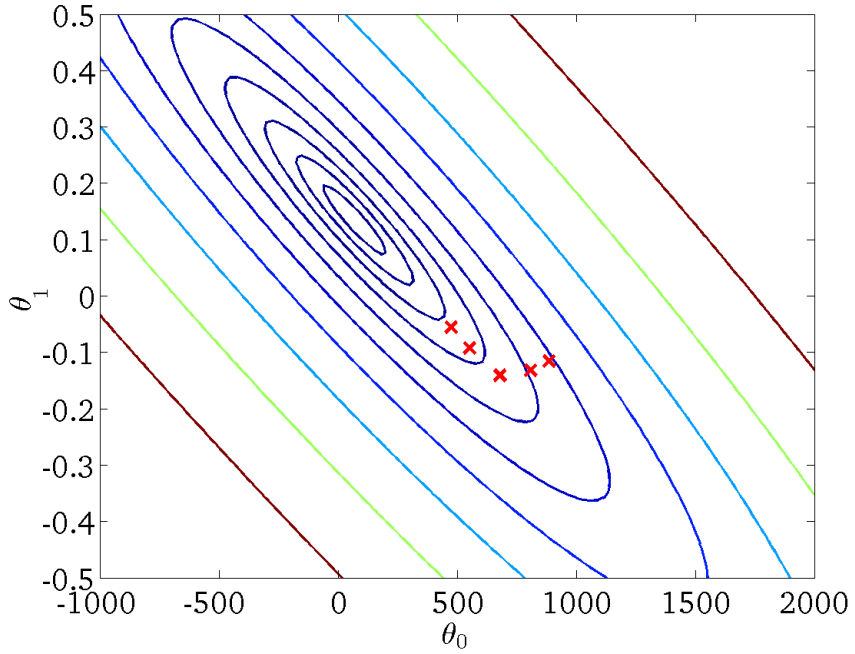
$h_{\boldsymbol{\theta}}(\boldsymbol{x})$

(for fixed θ_0, θ_1, this is a function of $\boldsymbol{x}$)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

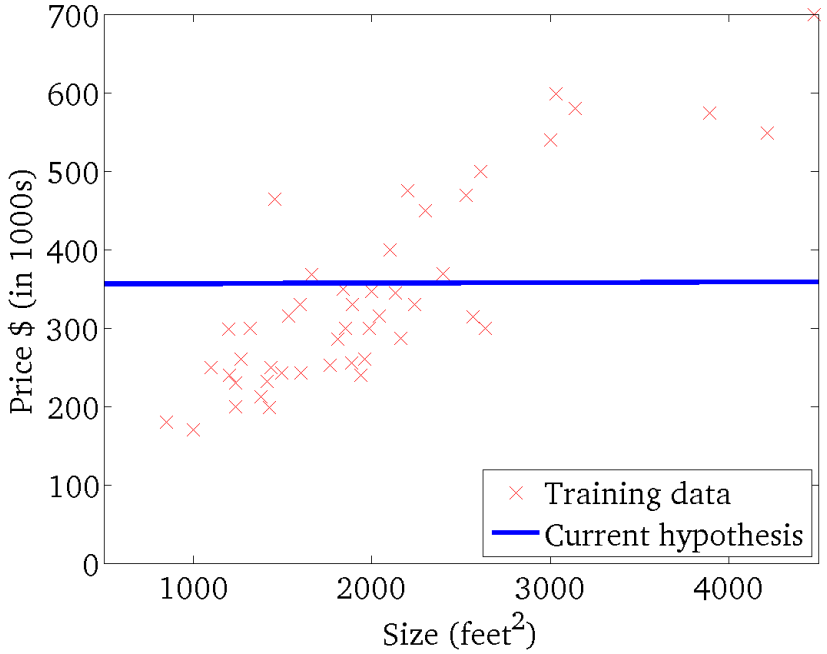


Slide by Andrew Ng

Gradient Descent

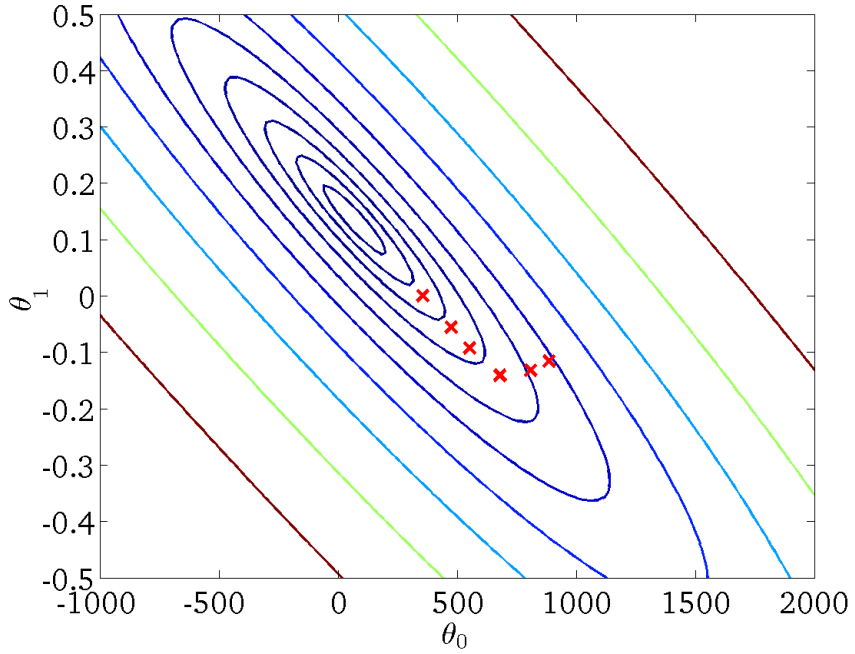
$h_{\boldsymbol{\theta}}(\boldsymbol{x})$

(for fixed θ_0, θ_1, this is a function of $\boldsymbol{x}$)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

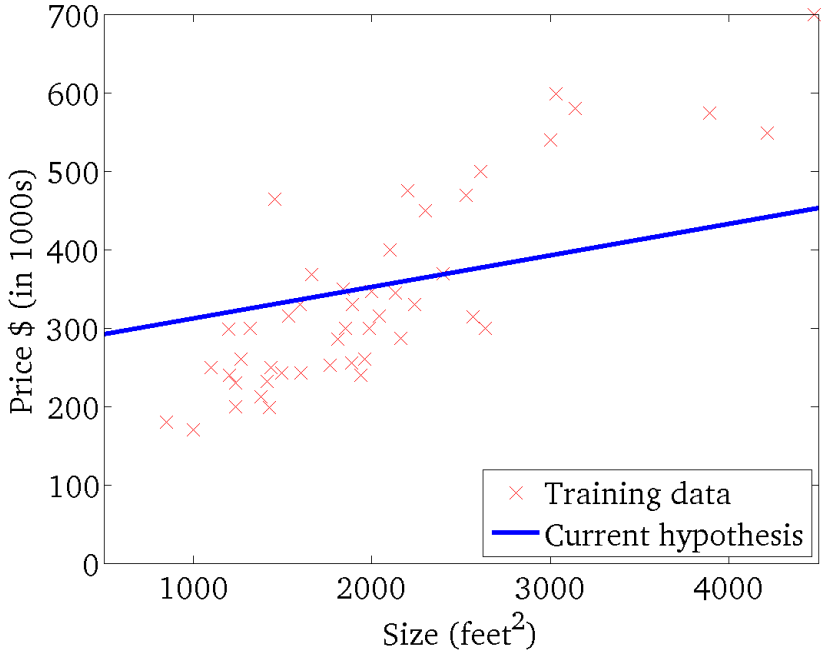


Slide by Andrew Ng

Gradient Descent

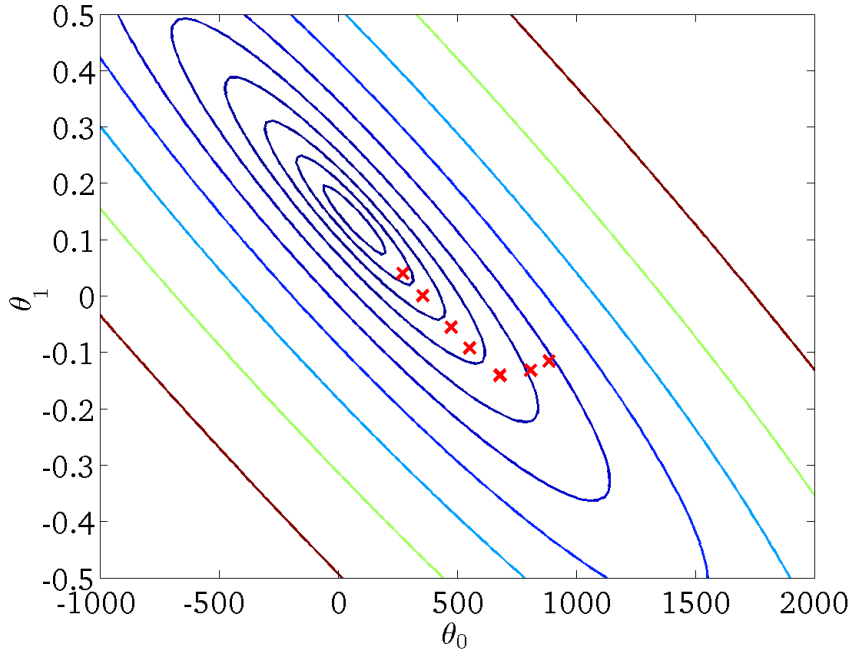
$h_{\boldsymbol{\theta}}(\boldsymbol{x})$

(for fixed θ_0, θ_1, this is a function of $\boldsymbol{x}$)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

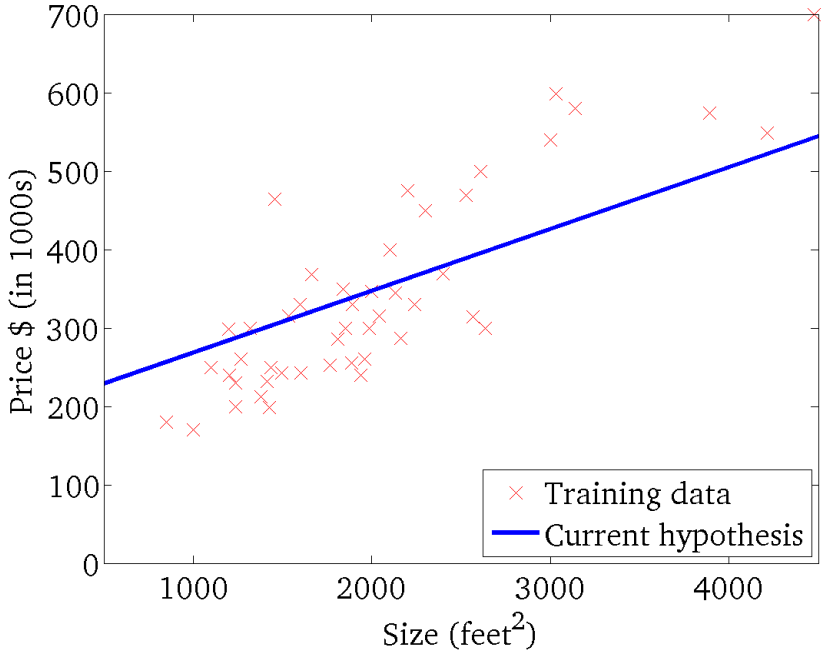


Slide by Andrew Ng

Gradient Descent

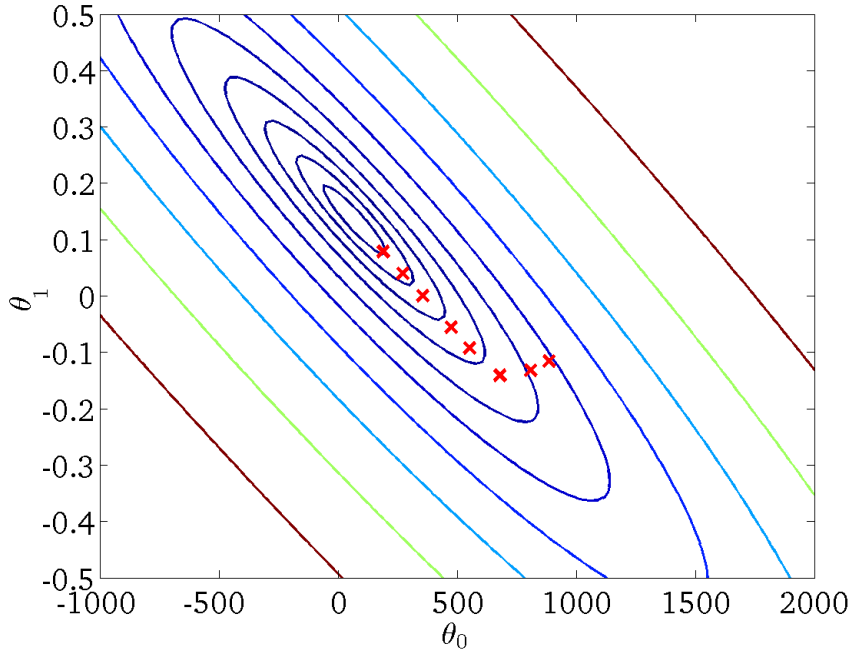
$h_{\boldsymbol{\theta}}(\boldsymbol{x})$

(for fixed θ_0, θ_1, this is a function of $\boldsymbol{x}$)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

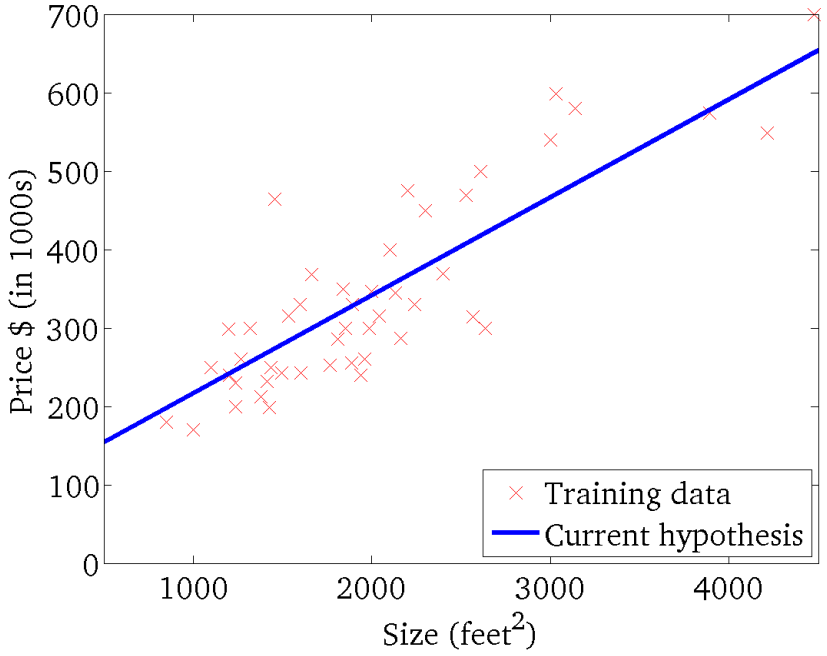


Slide by Andrew Ng

Gradient Descent

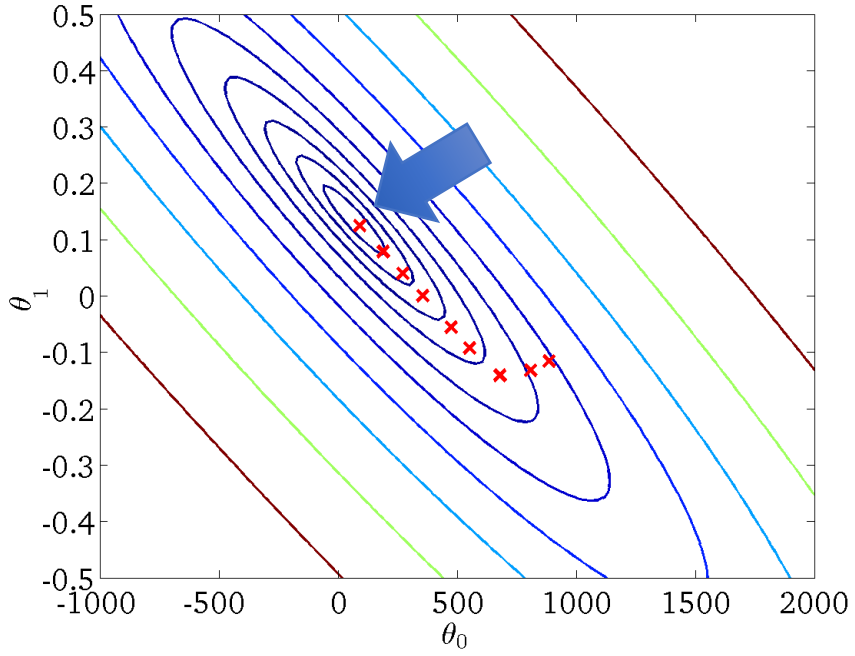
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

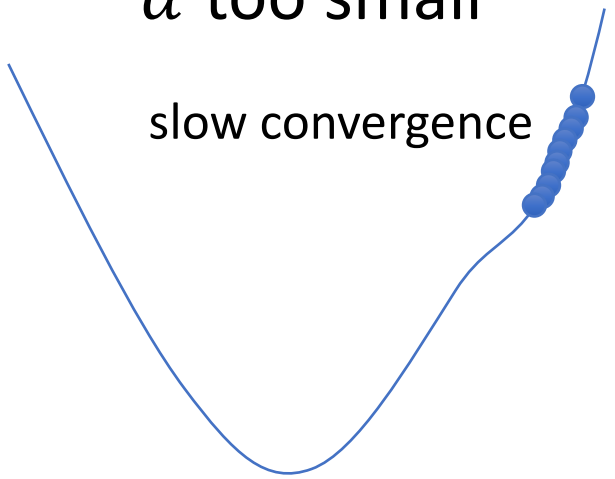
(function of the parameters θ_0, θ_1)



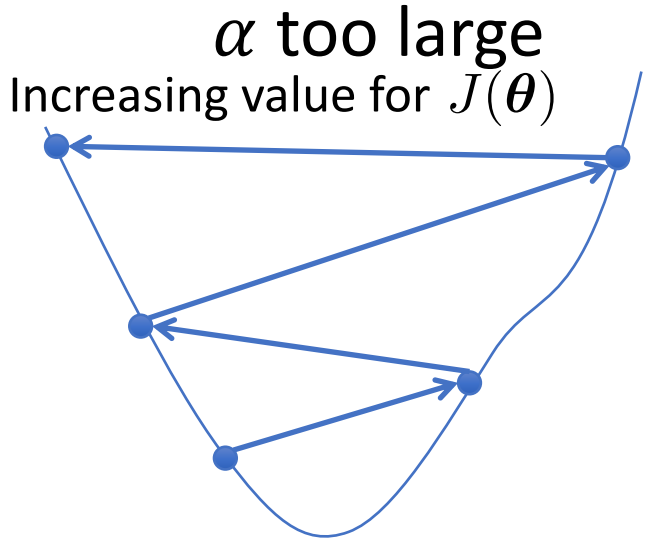
Slide by Andrew Ng

Learning rate α

α too small



α too large

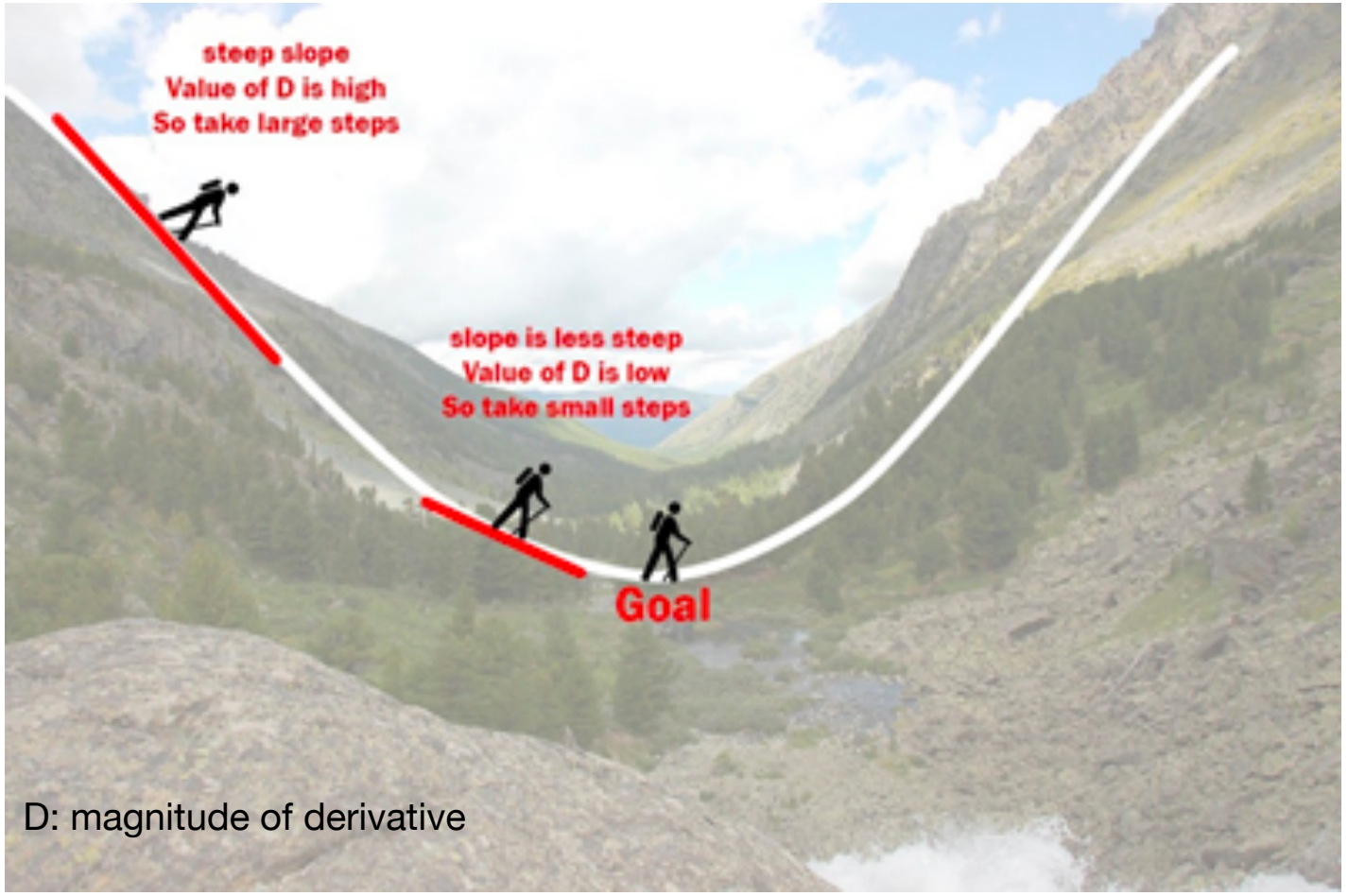


- May overshoot the minimum
- May fail to converge
- May even diverge

To see if gradient descent is working, print out $J(\theta)$ each iteration

- The value should decrease at each iteration
- If it doesn't, adjust α

Heuristic for setting learning rate α



D: magnitude of derivative

Fig src: Aarthi Kasirajan

Scaling to large datasets

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

where $\ell(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$ is the mean-squared error of $h_{\boldsymbol{\theta}}(\boldsymbol{x})$ for $(\boldsymbol{x}^{(i)}, y^{(i)})$

- Modern datasets for ML are very large
  - E.g., vision and language datasets have billions of training instances
- Computing average gradient over n training instances requires $O(n)$ time
  - Very slow for an iterative algorithm when n is large!
  - Also known as **Full Batch Gradient Descent (GD)**
- **Stochastic gradient descent (SGD):** compute gradient for every example

  Iterate over $i = 1, 2, \ldots, n$

  $$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

GD vs SGD

- Batch gradient descent

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)} \right) \boldsymbol{x}^{(i)}$$

- Stochastic gradient descent

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)} \right) \boldsymbol{x}^{(i)}$$

GD vs SGD

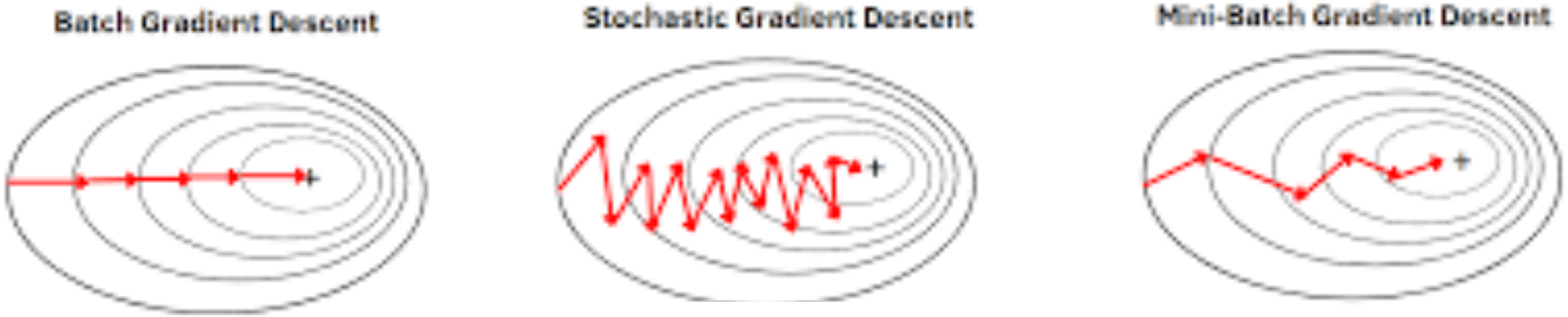
- Advantages of SGD: Scalability to large datasets
  - Memory efficient: single example is processed at a time
  - Computationally cheap: single example needed to compute gradient
  - Implicit regularization (advanced ML course)
- Disadvantages of SGD
  - High noise in the gradient. Can take many updates to converge
  - Cannot exploit modern hardware optimized for matrix operations

Mini-batch Gradient Descent

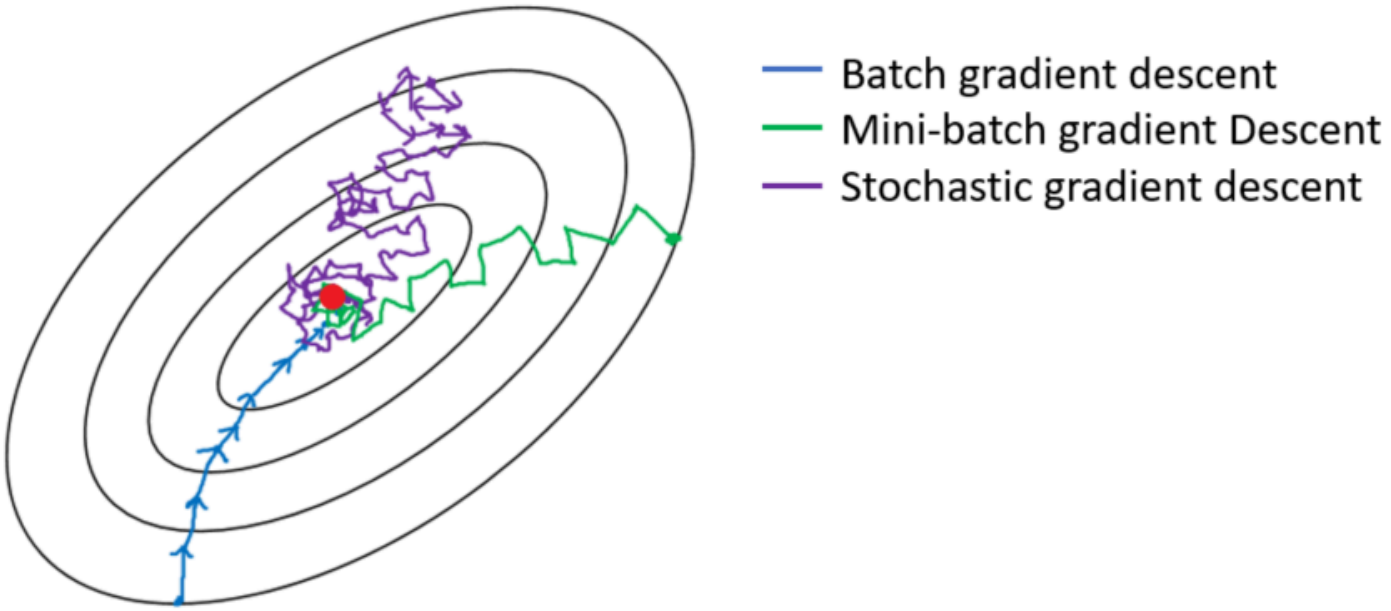
- **Mini-batch GD (MBGD):** Sample a batch of B points D_B at random from the full dataset D without replacement

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \frac{1}{B} \sum_{(\boldsymbol{x}^{(i)}, y^{(i)}) \in D_B} \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

- Compromise between (full batch) GD and SGD
  - When $B = 1$, MBGD is same as SGD
  - When $B = n$, MBGD is same as GD



GD vs SGD vs MBGD



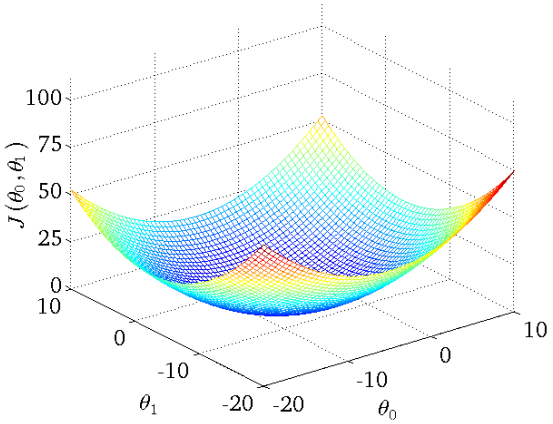
Optimizing Loss Functions – Step 3

- Least squares loss function

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

- Fit by solving

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$



Two optimization algorithms:

- **Gradient Descent**
- **Normal Equation** – next!

Vectorization

- Benefits of vectorization
  - More compact equations
  - Faster code (using optimized matrix libraries)
- Consider our model:

$$h(\boldsymbol{x}) = \sum_{j=0}^{d} \theta_j x_j$$

- Let

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \qquad \boldsymbol{x}^\intercal = \begin{bmatrix} 1 & x_1 & \dots & x_d \end{bmatrix}$$

- Can write the model in vectorized form as

$$h(\boldsymbol{x}) = \boldsymbol{\theta}^\intercal \boldsymbol{x}$$

Vectorization

- Consider our model for n instances:

$$h\left(\boldsymbol{x}^{(i)}\right) = \sum_{j=0}^{d} \theta_j x_j^{(i)}$$

- Let

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \quad \mathbb{R}^{(d+1)\times 1}$$

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \dots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix} \quad \mathbb{R}^{n\times(d+1)}$$

- Can write the model in vectorized form as

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{X}\boldsymbol{\theta}$$

Vectorization

- For the linear regression cost function:

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

$$= \frac{1}{2n}\sum_{i=1}^{n}\left(\boldsymbol{\theta}^{\intercal}\boldsymbol{x}^{(i)} - y^{(i)}\right)^2$$

$$= \frac{1}{2n}\underbrace{(\boldsymbol{X\theta} - \boldsymbol{y})^{\intercal}}_{\mathbb{R}^{1\times n}}\underbrace{(\boldsymbol{X\theta} - \boldsymbol{y})}_{\mathbb{R}^{n\times 1}}$$

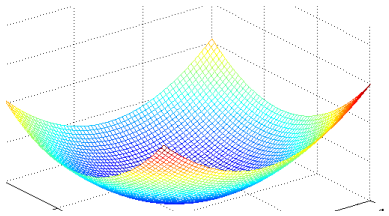
$\mathbb{R}^{n\times(d+1)}$

$\mathbb{R}^{(d+1)\times 1}$

Let:

$$\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

Normal Equations - Closed Form Solution



- Instead of using GD, solve for optimal $\boldsymbol{\theta}$ analytically

$$\frac{\partial}{\partial \boldsymbol{\theta}} J(\boldsymbol{\theta}) = 0$$

- Derivation:

$$\mathcal{J}(\boldsymbol{\theta}) = \frac{1}{2n} (\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y})^\intercal (\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y})$$

1 x 1

$$\propto \boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}^\intercal \boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{y} + \boldsymbol{y}^\intercal \boldsymbol{y}$$

$$\propto \boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{X}\boldsymbol{\theta} - 2\boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{y} + \boldsymbol{y}^\intercal \boldsymbol{y}$$

Take derivative and set equal to 0, then solve for $\boldsymbol{\theta}$:

$$\frac{\partial}{\partial \boldsymbol{\theta}} (\boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{X}\boldsymbol{\theta} - 2\boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{y} + \cancel{\boldsymbol{y}^\intercal \boldsymbol{y}}) = 0$$

$$(\boldsymbol{X}^\intercal \boldsymbol{X})\boldsymbol{\theta} - \boldsymbol{X}^\intercal \boldsymbol{y} = 0$$

$$(\boldsymbol{X}^\intercal \boldsymbol{X})\boldsymbol{\theta} = \boldsymbol{X}^\intercal \boldsymbol{y}$$

Closed Form Solution: $\boldsymbol{\theta} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}$

Closed Form Solution

- Can obtain $\boldsymbol{\theta}$ by simply plugging $\boldsymbol{X}$ and $\boldsymbol{y}$ into

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

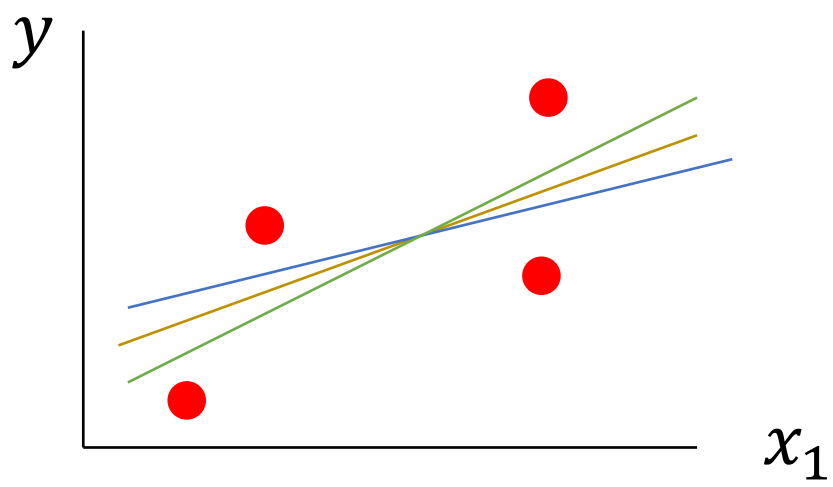
$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \dots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

- If $\boldsymbol{X}^\intercal \boldsymbol{X}$ is not invertible (i.e., singular), may need to:
  - Use pseudo-inverse instead of the inverse
    - In python, `numpy.linalg.pinv(a)`
  - Remove redundant (not linearly independent) features
  - Remove extra features to ensure that $d \leq n$

Gradient Descent v.s. Normal Equations

Gradient Descent	Normal Equations
• Requires multiple iterations • Need to choose learning rate α • Works well when n is large • Can support incremental learning	• Non-iterative • No need for α • Slow if d is large – Computing $(\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X})^{-1}$ is roughly O(d^3)

Linear Regression



Steps for learning h

1. Represent hypothesis class $\mathcal{H}$
2. Define a loss objective
3. Minimize loss to find best $h \in \mathcal{H}$

Summary so far

3 Steps:

- Represent hypothesis class

  Linear functions: $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x}$ (linear in both $\boldsymbol{\theta}$ and $\boldsymbol{x}$)

- Define a loss function

  Squared error loss: $J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)} \right)^2$

- Optimize loss to find best hypothesis

  Gradient descent: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)} \right) \boldsymbol{x}^{(i)}$

  Normal equations: $\widehat{\boldsymbol{\theta}} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \dots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$