Foundations of Data Science

DS 3001

Data Science Program

Department of Computer Science

Worcester Polytechnic Institute

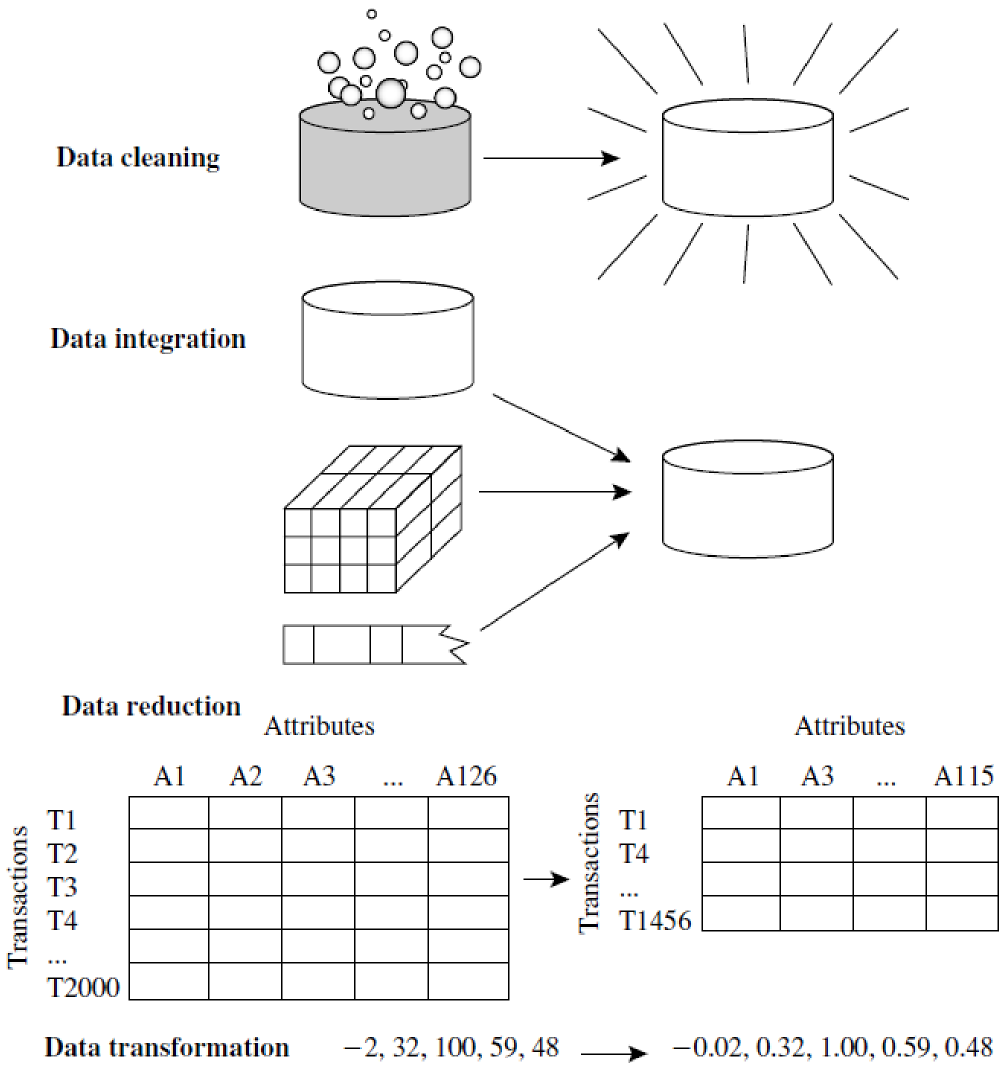
Instructor: Prof. Kyumin Lee

Project Teams

1. Clay Oshiro-Leavitt, Hunter Caouette, Nick Alescio
2. Danielle Angelini, Elijah Ellis, Ryan Candy, Rob Wondolowski
3. Eva (Yingbing) Lu, Manasi Danke, Erica Lee, Jonathan Dang
4. Arianna Kan, Yihan Lin, Margaret Goodwin, Ken Snoddy
5. Yang Gao, Jose Li, Sarah Burns, Daniel McDonough
6. Noah Puchovsky, Katherine Handy, Alex Tavares, Angelica Puchovsky
7. Armando Zubillaga, Gabriel Rodrigues, Humberto Leon, Joao Omena de Lucena
8. Edward Carlson, Samuel Goldman, Nick Krichevsky, Christopher Myers
9. Jessie White, Lindsay MacInnis, Bao Huynh, Ziqian Zeng
10. Suverino Frith, Nicholas Odell, Fay Whittall, Johvanni Perez
11. Alp Piskin, Robert Scalfani, Jake Barefoot, Mark Bernardo
12. Amanda Chan, Nugzar Chkhaidze, Luke Gebler
13. Daniel Pelaez, Nathan Savard, Kate Sincaglia

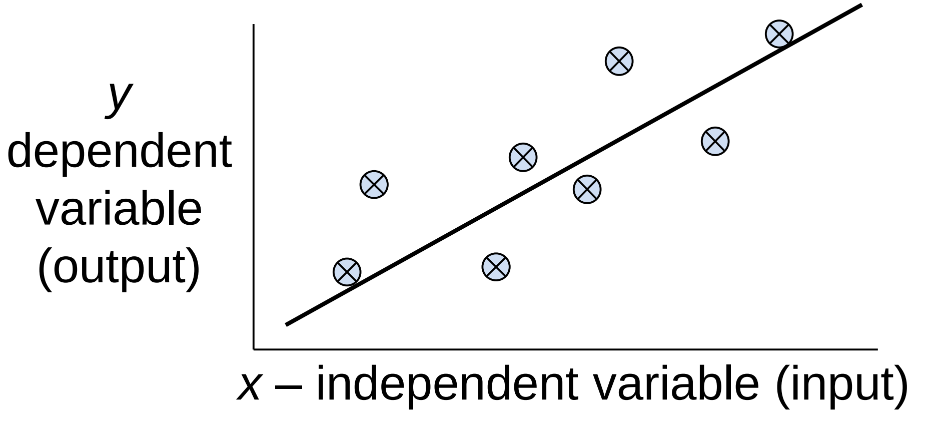
- So far, 49 students expressed their preferences

Forms of Data Preprocessing



Regression

- In regression the output is continuous
  - Function Approximation
  - Also a supervised learning
    - Given the "right answer" for each example in the data.
- Many models could be used – Simplest is linear regression
  - Fit data with the best hyper-plane which "goes through" the points



Training set of housing prices

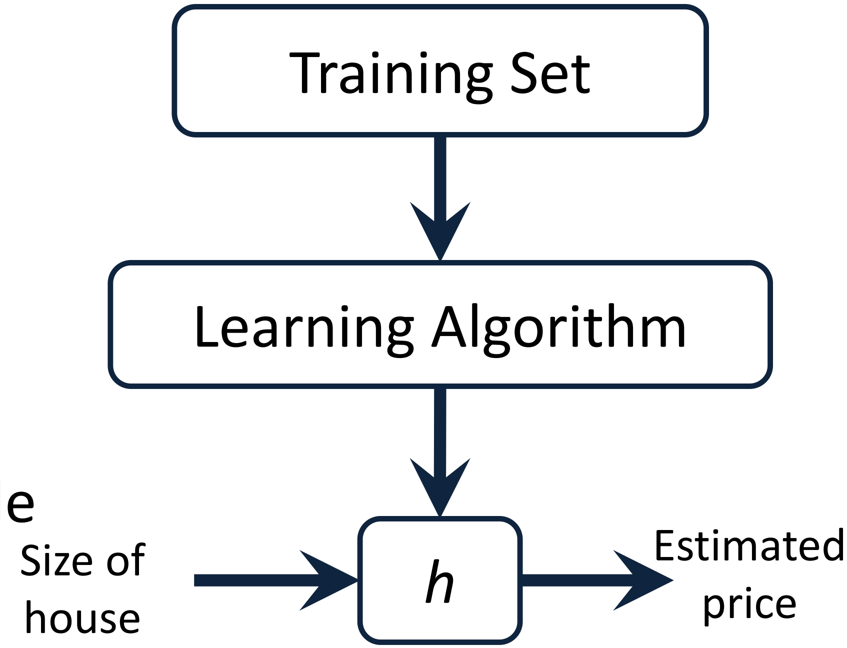
Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

Notation:

n = Number of training examples

x's = "input" variables / features

y's = "output" variable / "target" variable



Question : How to describe ***h***?

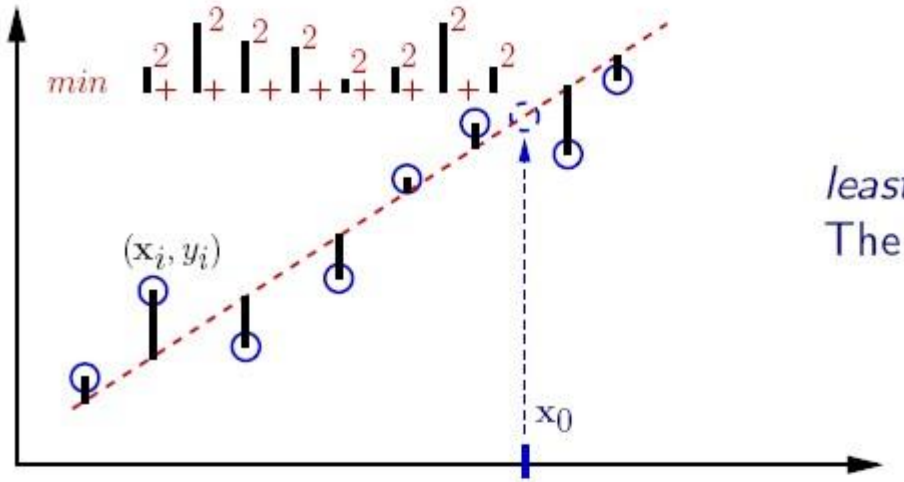
Linear Regression

- Hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ \ldots + \theta_d x_d = \sum_{j=0}^{d} \theta_j x_j$$

Assume $x_0 = 1$

- Fit model by minimizing sum of squared errors



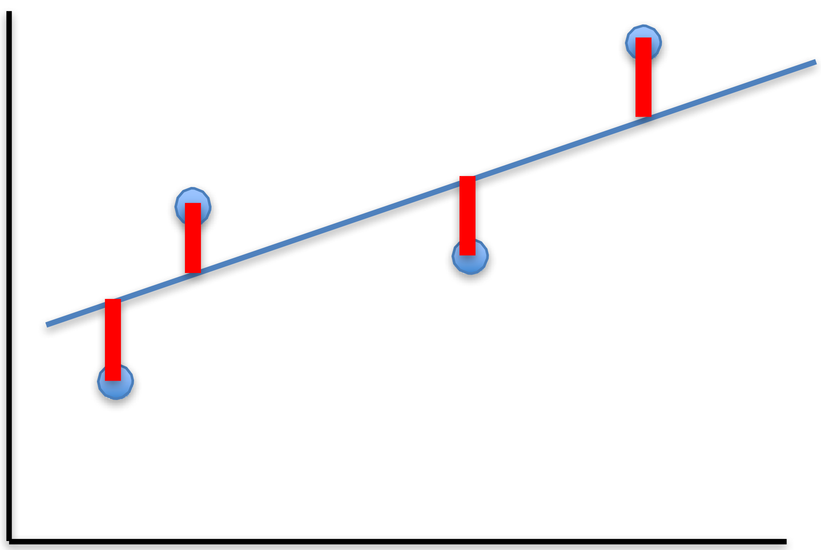
least squares (LSQ)
The fitted line is used as a predictor

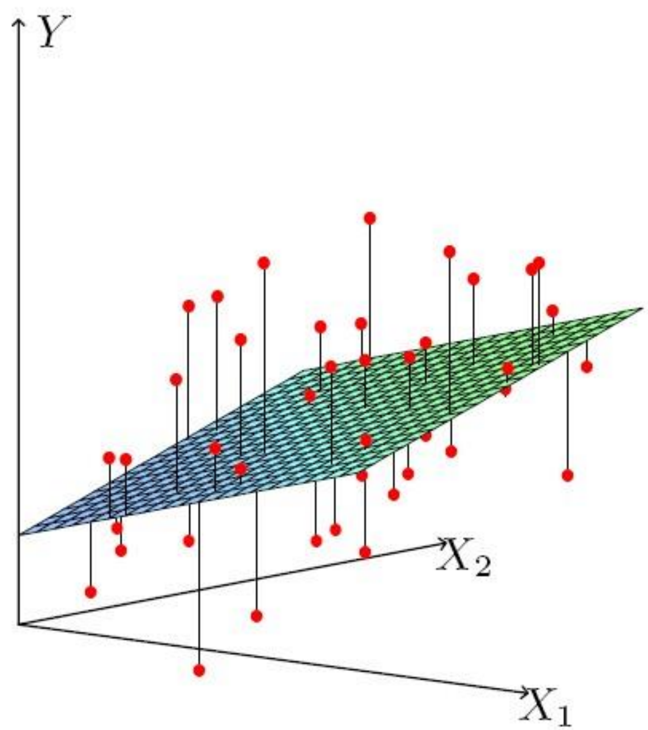
Least Squares Linear Regression

- Cost Function

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

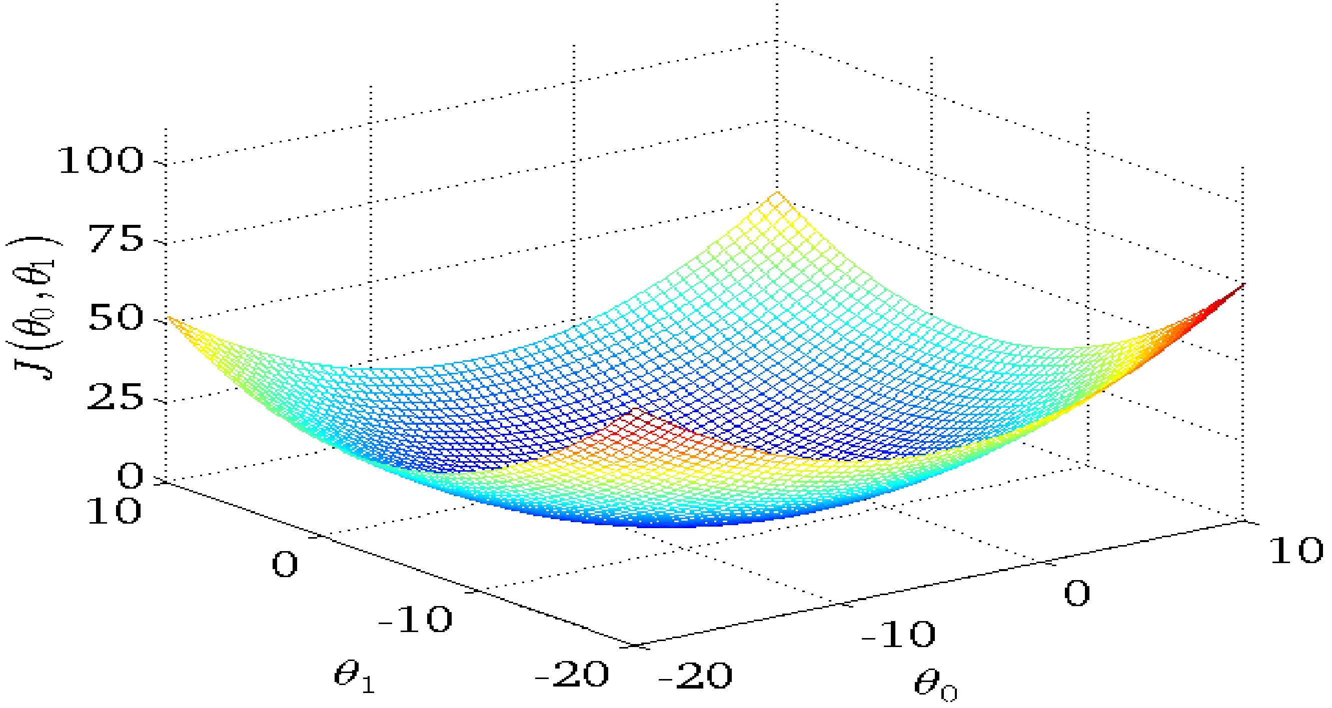
- Fit by solving $\min_\theta J(\theta)$





Basic Search Procedure

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



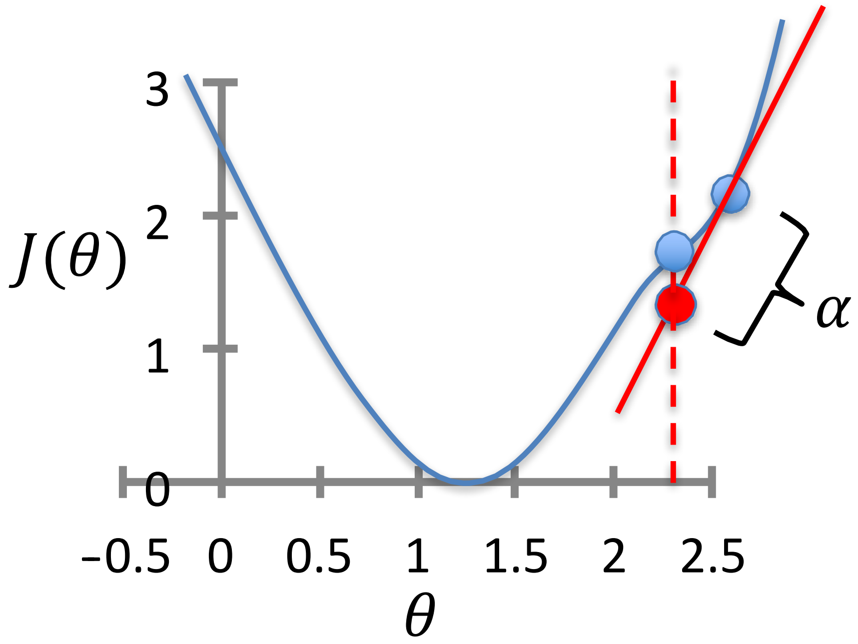
Gradient Descent

- Initialize θ
- Repeat until convergence

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

simultaneous update
for j = 0 … d

learning rate (small)
e.g., α = 0.05



Gradient Descent

- Initialize θ
- Repeat until convergence

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

simultaneous update
for j = 0 … d

For Linear Regression: $\frac{\partial}{\partial \theta_j} J(\theta) = \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^{n} (h_\theta(x^{(i)}) - y^{(i)})^2$

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

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

$$= \frac{1}{n} \sum_{i=1}^{n} (\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)}) \times x_j^{(i)}$$

http://mccormickml.com/2014/03/04/gradient-descent-derivation/

Gradient Descent for Linear Regression

- Initialize θ
- Repeat until convergence

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

simultaneous update
for j = 0 … d

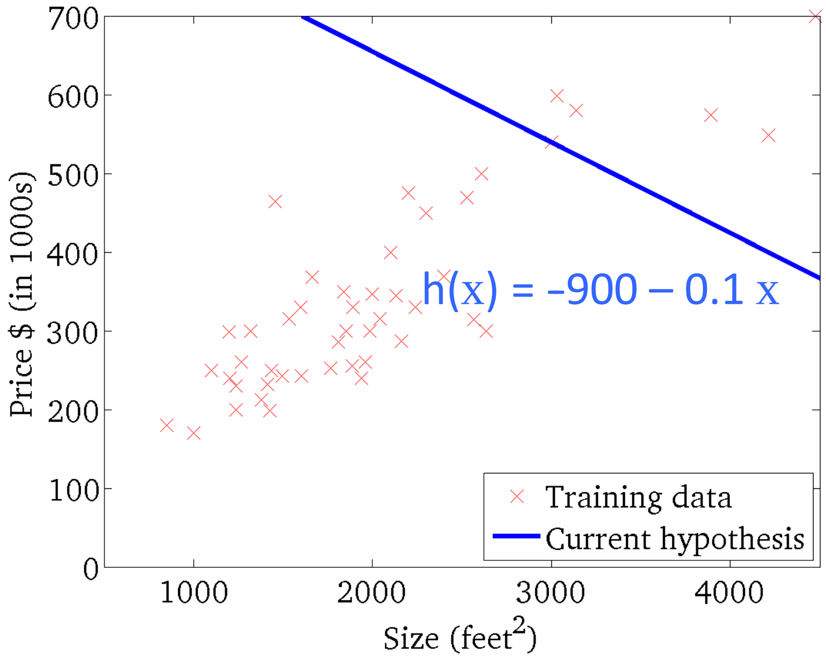
- To achieve simultaneous update
  - At the start of each GD iteration, compute $h_\theta\left(x^{(i)}\right)$
  - Use this stored value in the update step loop
- Assume convergence when $\left\|\theta_{new} - \theta_{old}\right\|_2 < \epsilon$

L2 norm: $\|v\|_2 = \sqrt{\sum_i v_i^2} = \sqrt{v_1^2 + v_2^2 + \ \ldots + v_{|v|}^2}$

Gradient Descent

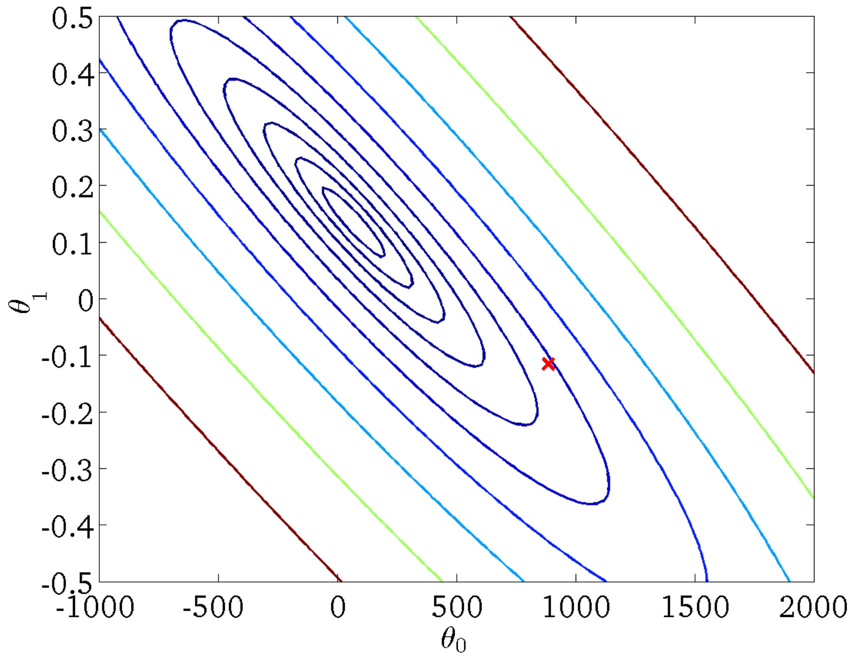
$$h_\theta(x)$$

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



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

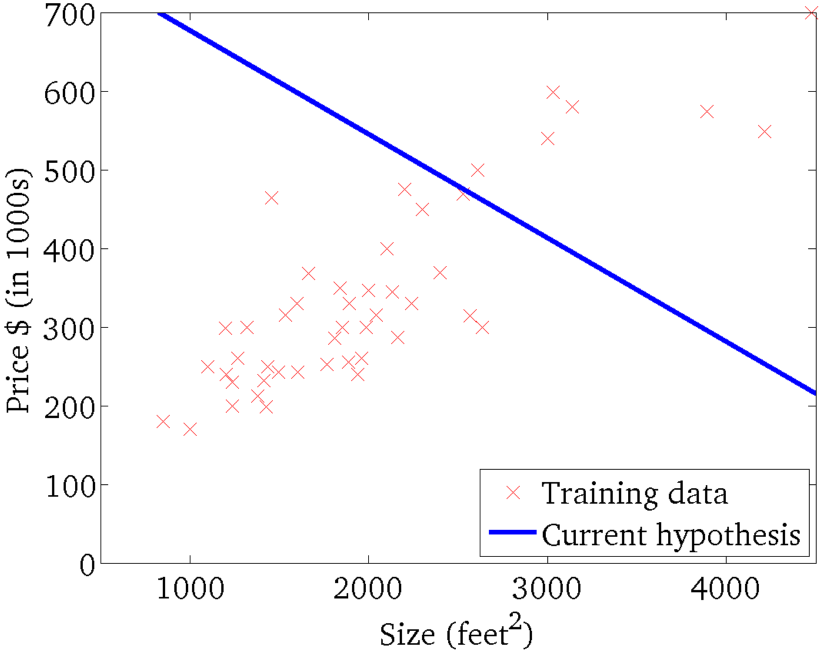
(function of the parameters θ_0, θ_1)



Gradient Descent

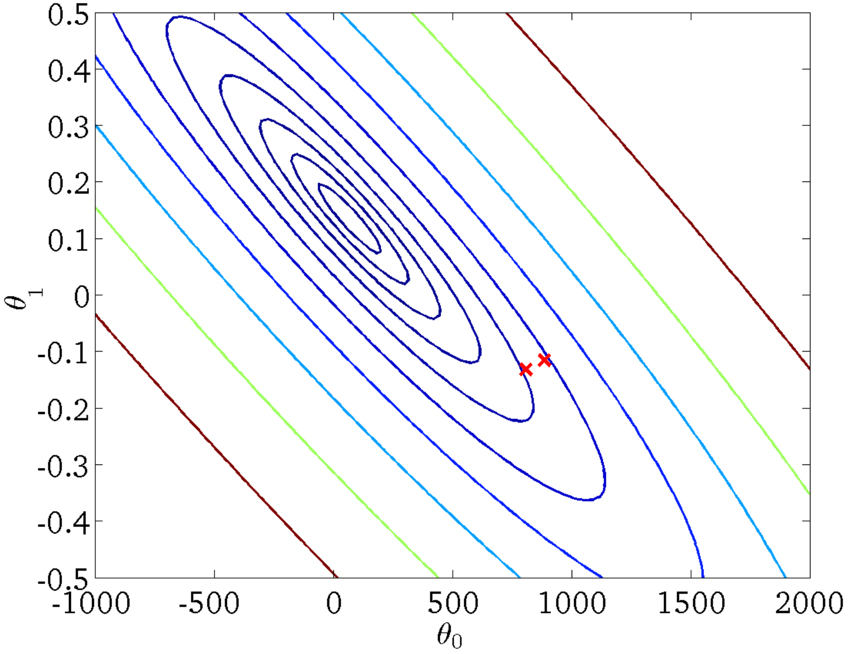
$$h_\theta(x)$$

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



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

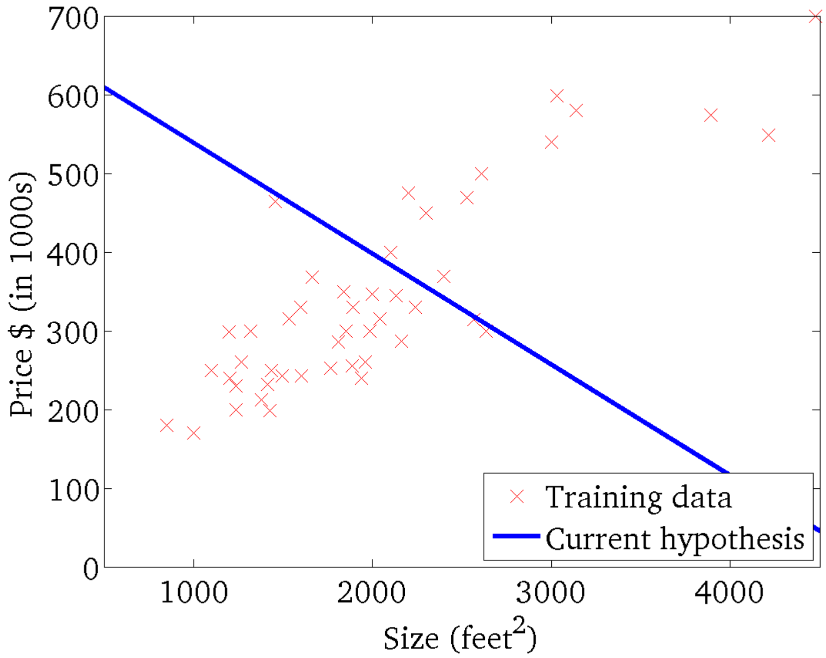
(function of the parameters θ_0, θ_1)



Gradient Descent

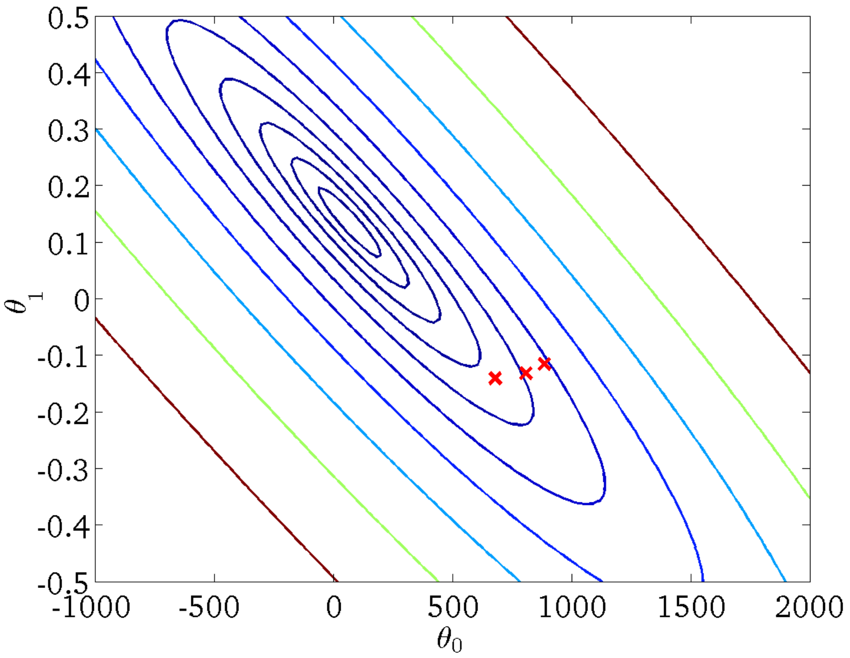
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

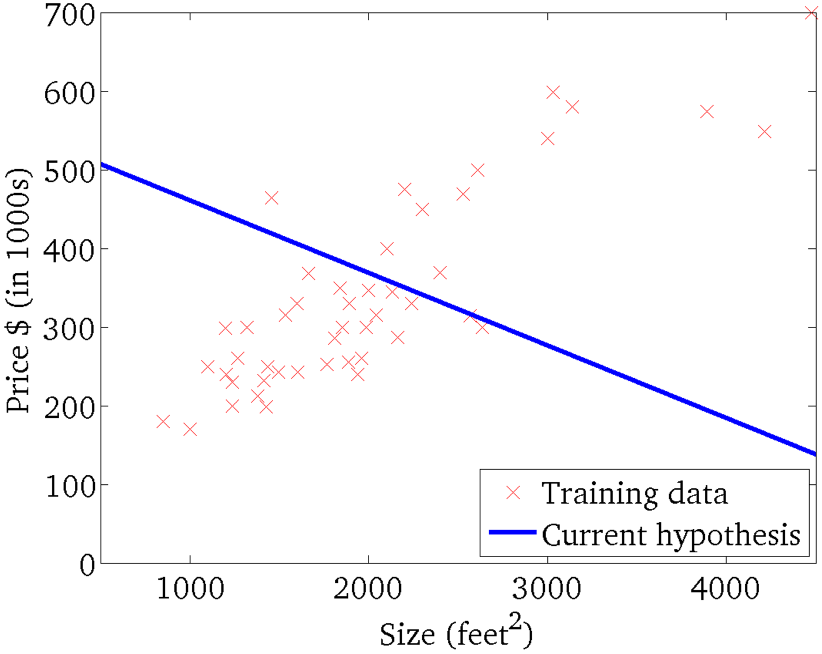
(function of the parameters θ_0, θ_1)



Gradient Descent

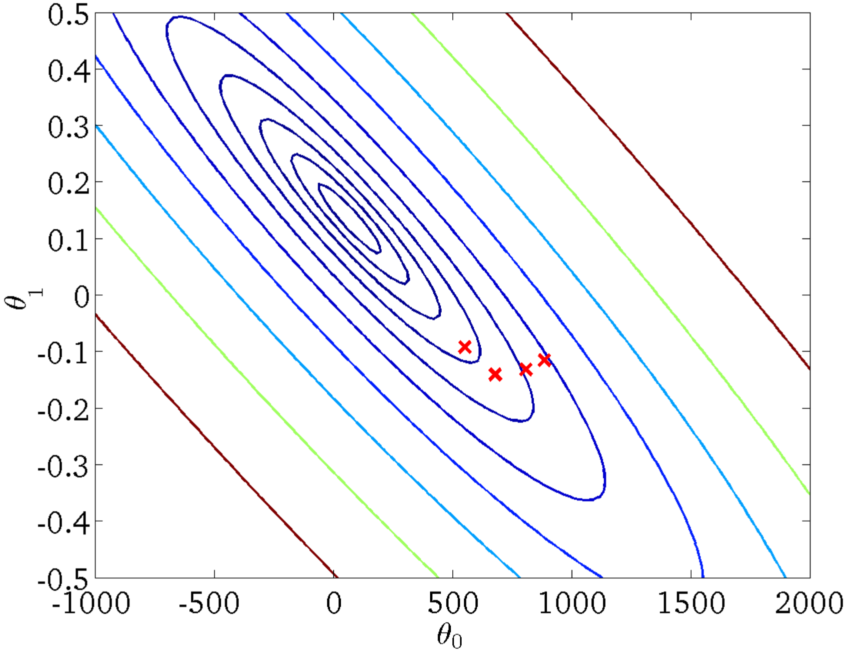
$$h_\theta(x)$$

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



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

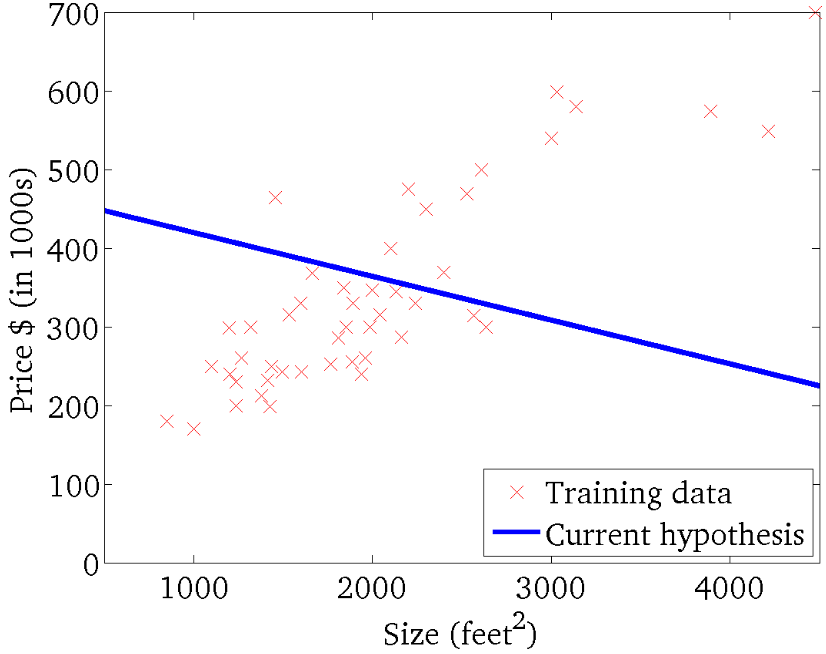
(function of the parameters θ_0, θ_1)



Gradient Descent

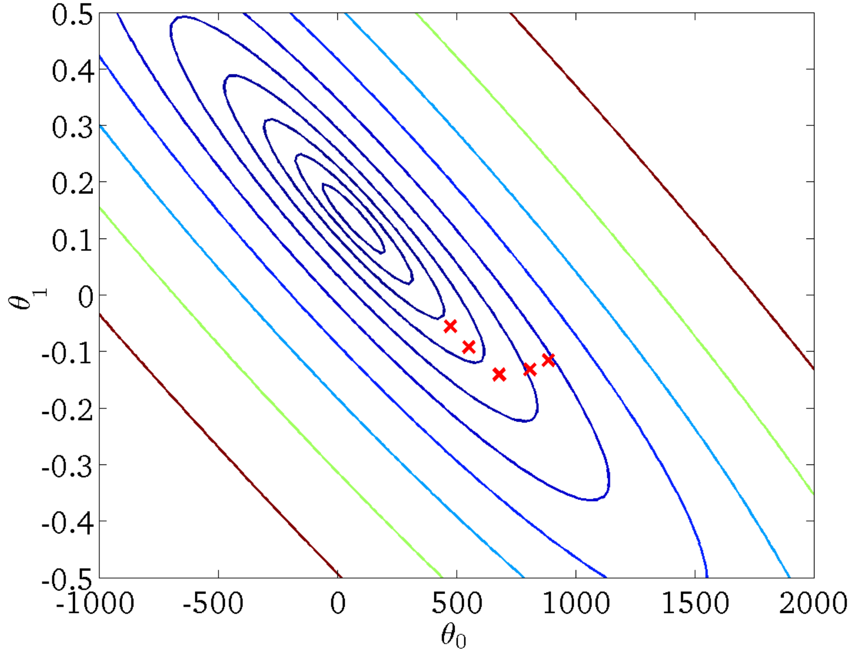
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

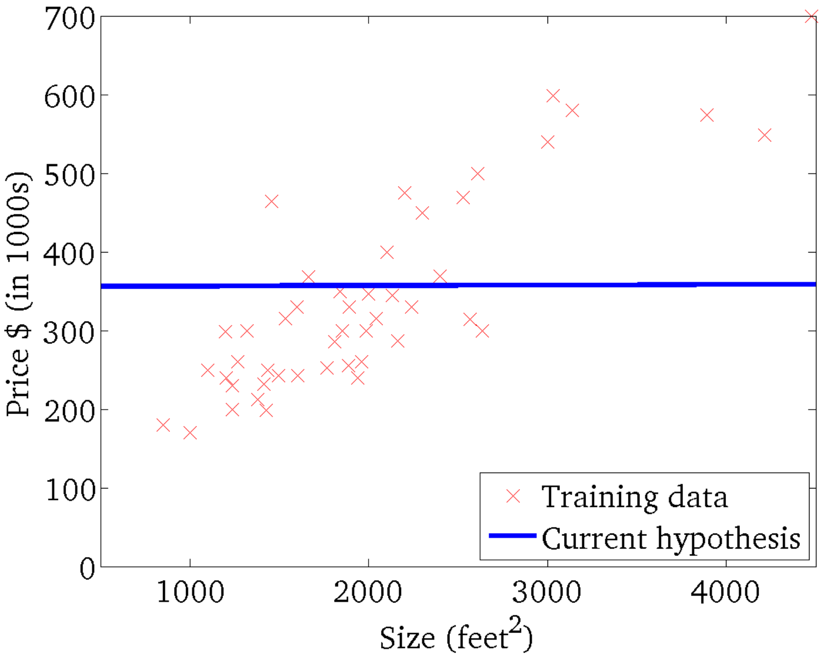
(function of the parameters θ_0, θ_1)



Gradient Descent

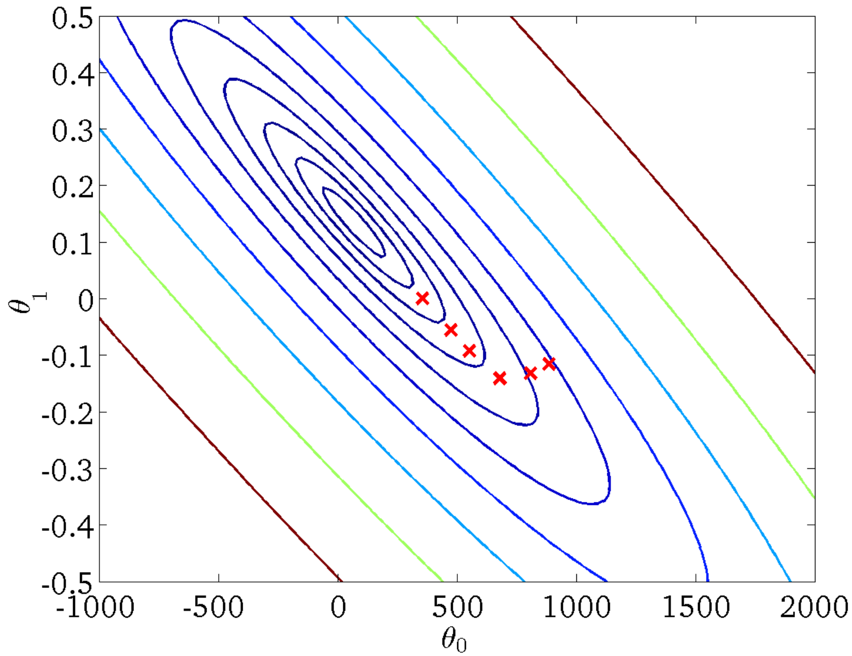
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

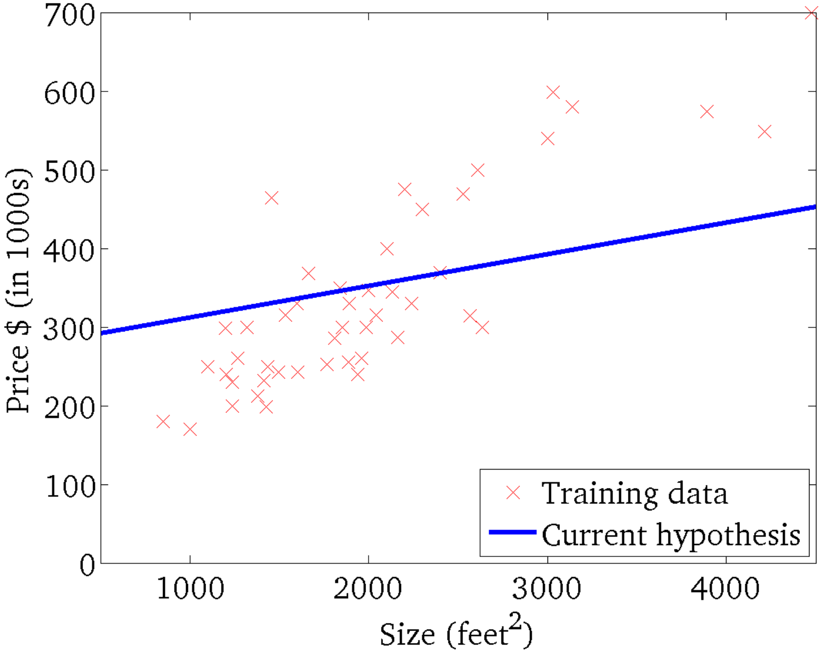
(function of the parameters θ_0, θ_1)



Gradient Descent

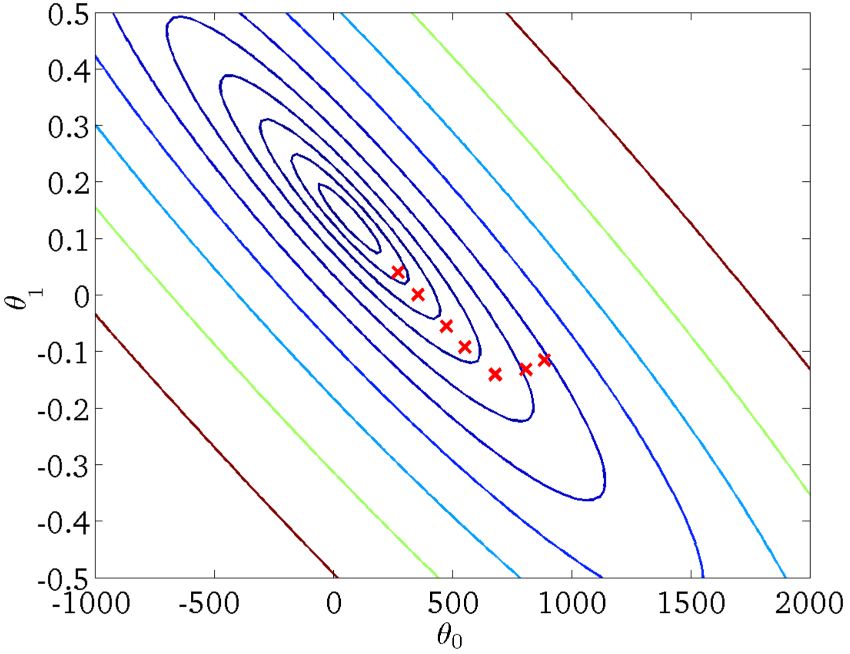
$$h_\theta(x)$$

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



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

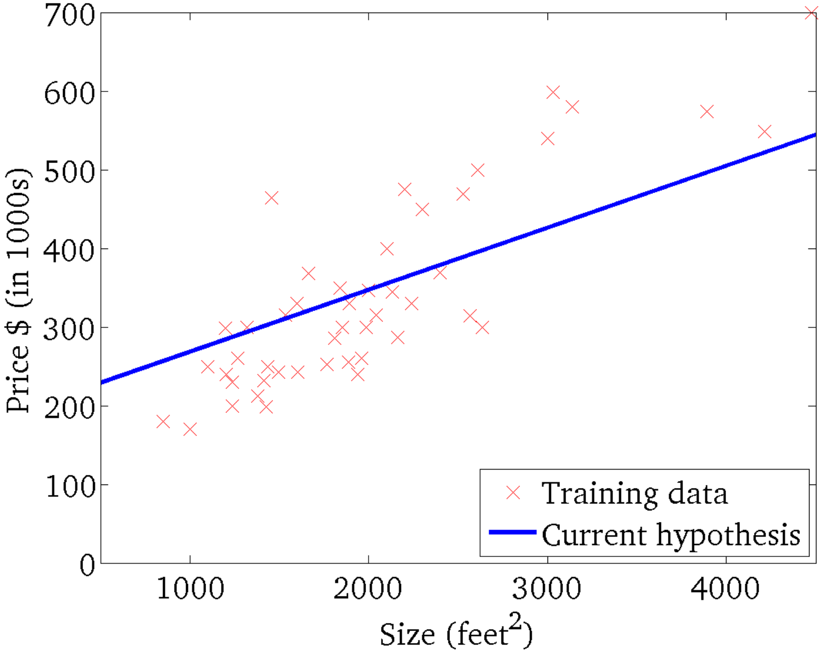
(function of the parameters θ_0, θ_1)



Gradient Descent

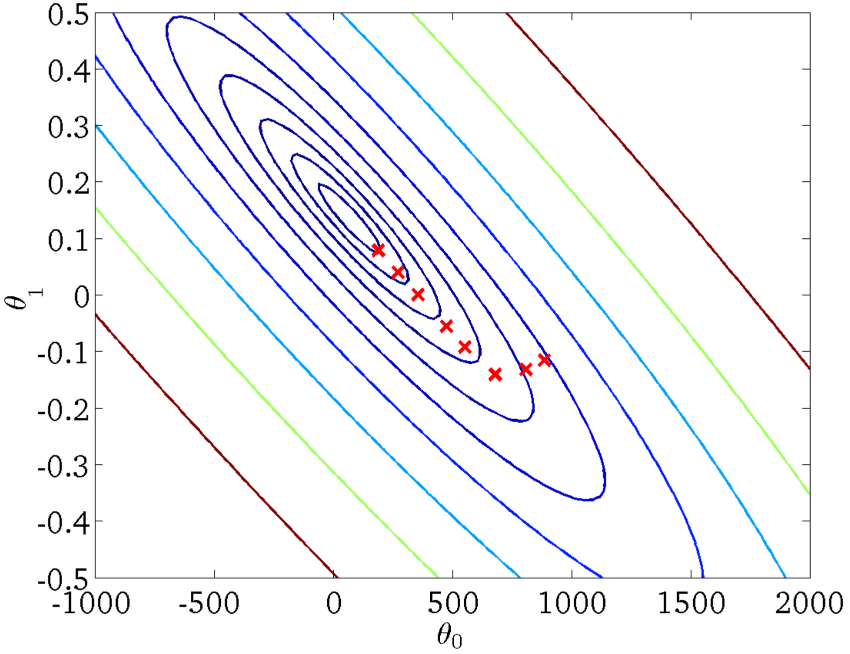
$$h_\theta(x)$$

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



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

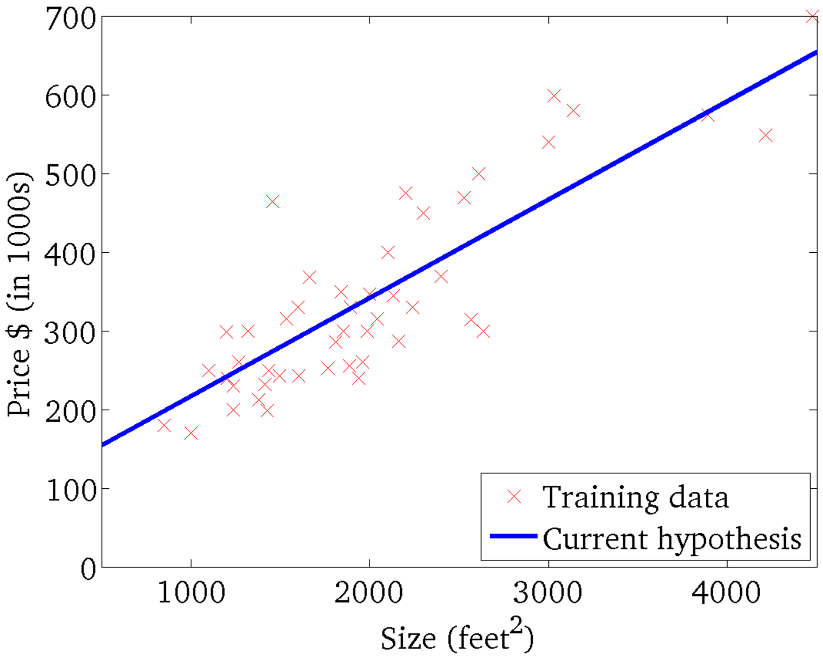
(function of the parameters θ_0, θ_1)



Gradient Descent

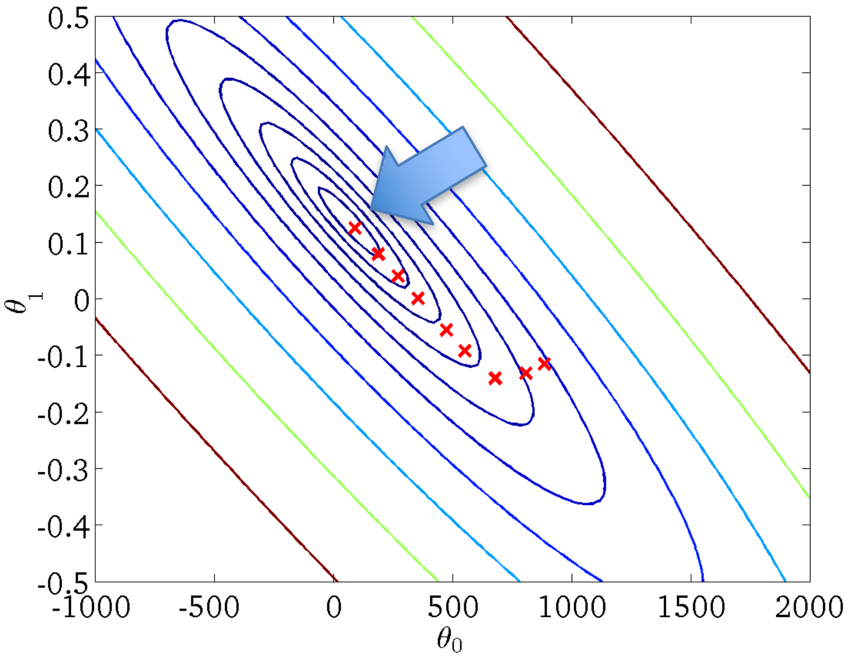
$h_\theta(x)$

(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)



Linear Regression (Big Picture)

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$

Cost function:

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

Gradient descent:

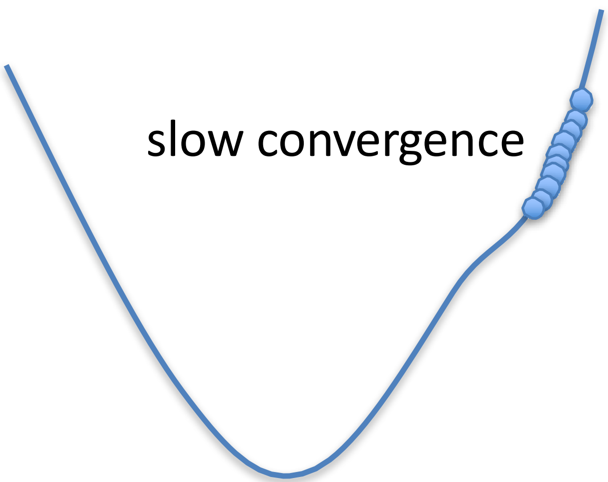
Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$$

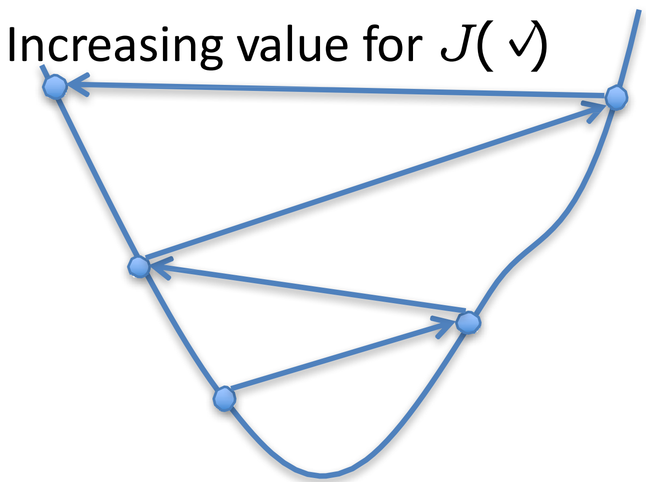
$\}$ (simultaneously update for every $j = 0, \ldots, n$)

Choosing 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

Linear Algebra Concepts

- Vector in $\mathbb{R}^d$ is an ordered set of *d* real numbers
  - e.g., v = [1,6,3,4] is in $\mathbb{R}^4$
  - "[1,6,3,4]" is a column vector: $\begin{pmatrix} 1 \\ 6 \\ 3 \\ 4 \end{pmatrix}$
  - as opposed to a row vector: $\begin{pmatrix} 1 & 6 & 3 & 4 \end{pmatrix}$
- An *m*-by-*n* matrix is an object with m rows and n columns, where each entry is a real number:

$$\begin{pmatrix} 1 & 2 & 8 \\ 4 & 78 & 6 \\ 9 & 3 & 2 \end{pmatrix}$$

Linear Algebra Concepts

- Transpose: flips a matrix over its diagonal ╲

$$\begin{pmatrix} a \\ b \end{pmatrix}^T = \begin{pmatrix} a & b \end{pmatrix} \qquad \begin{pmatrix} a & b \\ c & d \end{pmatrix}^T = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$$

  - Note: $(Ax)^T = x^T A^T$ (We'll define multiplication soon…)

- Vector norms:
  - L_p norm of $\mathbf{v} = (\mathbf{v}_1, \ldots, v_k)$ is $\left(\sum_i |v_i|^p\right)^{\frac{1}{p}}$
  - Common norms: L_1, L_2
  - $L_{infinity} = \max_i |v_i|$
- Length of a vector v is $L_2(v)$

Linear Algebra Concepts

- Vector dot product: $u \bullet v = \begin{pmatrix} u_1 & u_2 \end{pmatrix} \bullet \begin{pmatrix} v_1 & v_2 \end{pmatrix} = u_1 v_1 + u_2 v_2$
  - Note: dot product of u with itself = length$(u)^2$ = $\|u\|_2^2$

- Matrix product:

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

$$AB = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

Vectorization

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

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

- Let

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

- 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

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \qquad \boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \dots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix}$$

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

- Can write the model in vectorized form as $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{X}\boldsymbol{\theta}$

Vectorization

- For the linear regression cost function:

$$
\begin{aligned}
J(\boldsymbol{\theta}) &= \frac{1}{2n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)^2 \\
&= \frac{1}{2n} \sum_{i=1}^{n} \left(\boldsymbol{\theta}^{\intercal} \boldsymbol{x}^{(i)} - y^{(i)} \right)^2 \\
&= \frac{1}{2n} \underbrace{(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y})^{\intercal}}_{\mathbb{R}^{1\times n}} (\underbrace{\overset{\mathbb{R}^{n\times(d+1)}}{\boldsymbol{X}} \overset{\mathbb{R}^{(d+1)\times 1}}{\boldsymbol{\theta}}}_{\mathbb{R}^{n\times 1}} - \underbrace{\boldsymbol{y}}_{\mathbb{R}^{n\times 1}})
\end{aligned}
$$

Let:

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

Mining and Analytics: Classification + Decision Trees

Classification: Definition

- Given a collection of records (training set)
  - Each record contains a set of attributes, one of the attributes is the class.
- Find a model for class attribute as a function of the values of other attributes.

- Goal: previously unseen records should be assigned a class as accurately as possible.
  - A test set is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

Classification: Direct Marketing

- Goal: Reduce cost of mailing by *targeting* a set of consumers likely to buy a new cell-phone product.
- Approach:
  - Use the data for a similar product introduced before.
  - We know which customers decided to buy and which decided otherwise. This *{buy, don't buy}* decision forms the *class attribute*.
  - Collect various demographic, lifestyle, and company-interaction related information about all such customers.
    - Type of business, where they stay, how much they earn, etc.
  - Use this information as input attributes to learn a classifier model.

Classification: Fraud Detection

- Goal: Predict fraudulent cases in credit card transactions.
- Approach:
  - Use credit card transactions and the information on its account-holder as attributes.
    - When does a customer buy, what does he buy, how often he pays on time, etc
  - Label past transactions as fraud or fair transactions. This forms the class attribute.
  - Learn a model for the class of the transactions.
  - Use this model to detect fraud by observing credit card transactions on an account.

Illustrating Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set