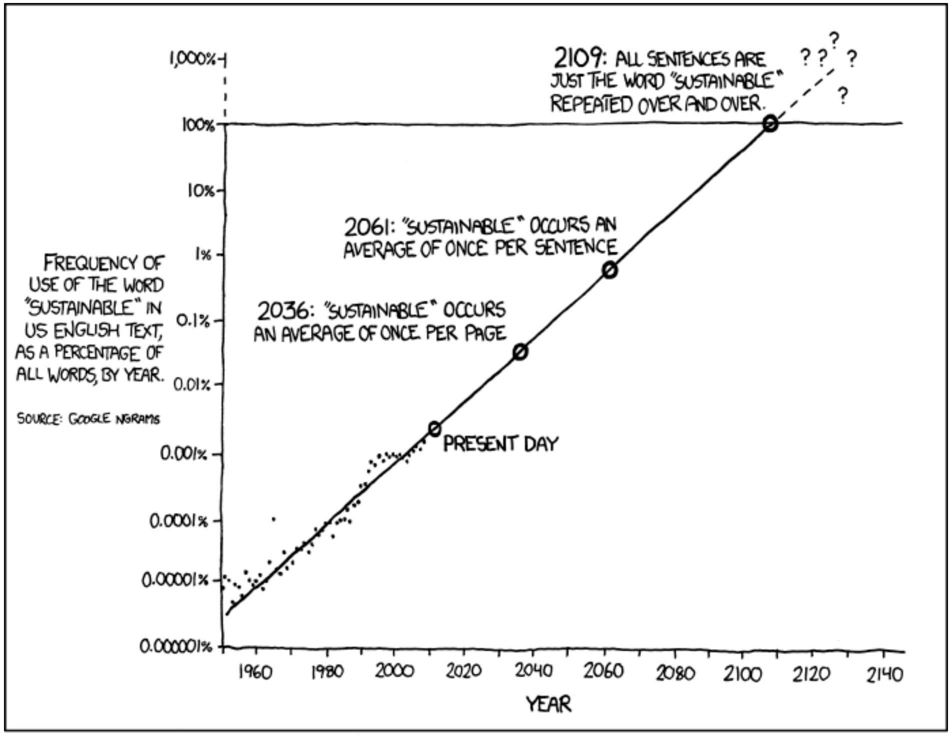
Linear Regression



THE WORD "SUSTAINABLE" IS UNSUSTAINABLE.

https://xkcd.com/1007/

Slides from:

Andrew Ng

CSC411: Machine Learning and Data Mining, Winter 2018

Michael Guerzhoy and Lisa Zhang

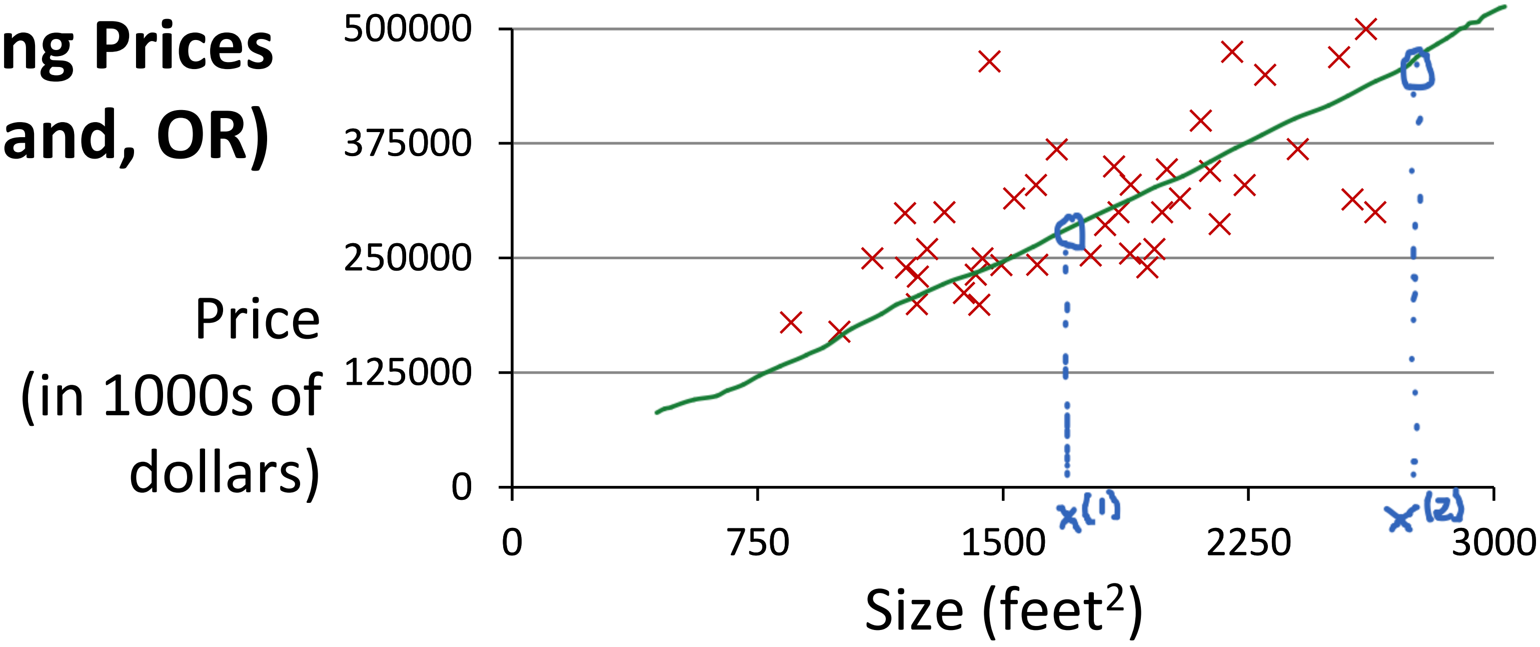
Training set of housing prices (Portland, OR)

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

Notation:

m = Number of training examples
x’s = “input” variable / features
y’s = “output” variable / “target” variable

Housing Prices (Portland, OR)

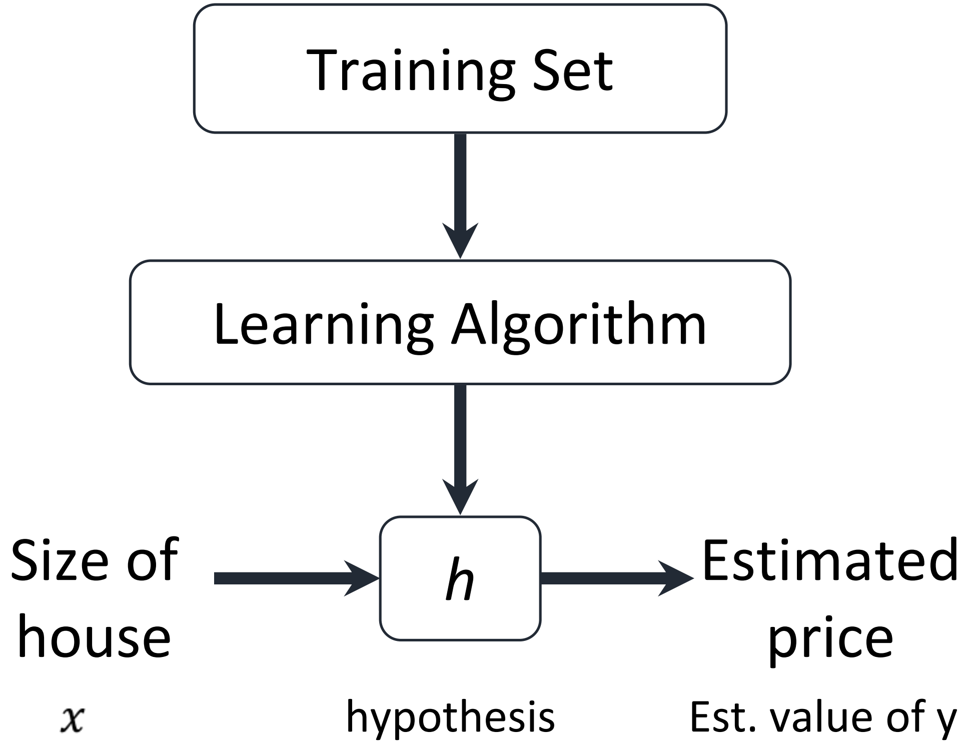


Supervised Learning

Given the “right answer” for each example in the data.

Regression Problem

Predict real-valued output



Training Set

Learning Algorithm

Size of house

h

Estimated price

x

hypothesis

Est. value of y

H maps x's to y's.

How do we represent ***h*** ?

- We represent hypotheses about the data using the parameters $\theta = (\theta_0, \theta_1)$
- If the data is correctly predicted according to hypothesis h_θ, then $y \approx h_\theta(x) = \theta_0 + \theta_1 x$
- The learning algorithm finds the best hypothesis h_θ for the training set
- We can then estimate the values of y for the test set using that h_θ
- If $h_\theta(x)$ is a linear function of a real number x, this procedure is called linear regression.

Training Set

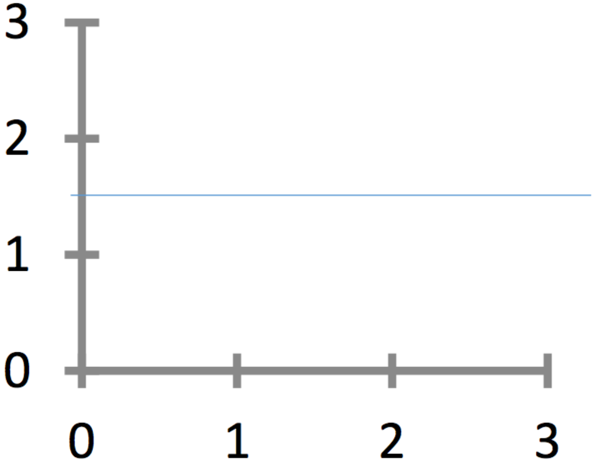
Size in feet^2 (x)	**Price ($) in 1000's (y)**
2104	460
1416	232
1534	315
852	178
…	…

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

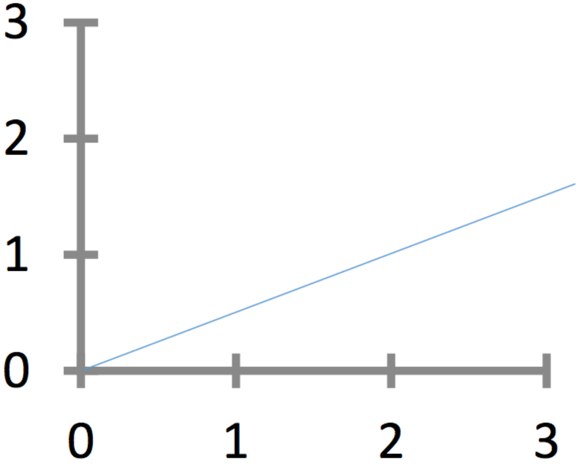
θ_i's: Parameters

How to choose θ_i's ?

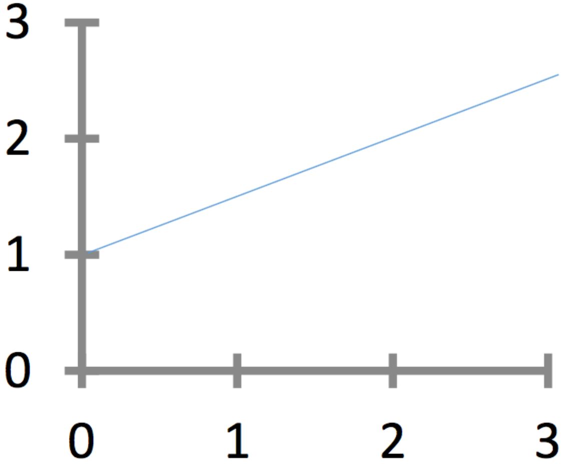
$$h_\theta(x) = \theta_0 + \theta_1 x$$



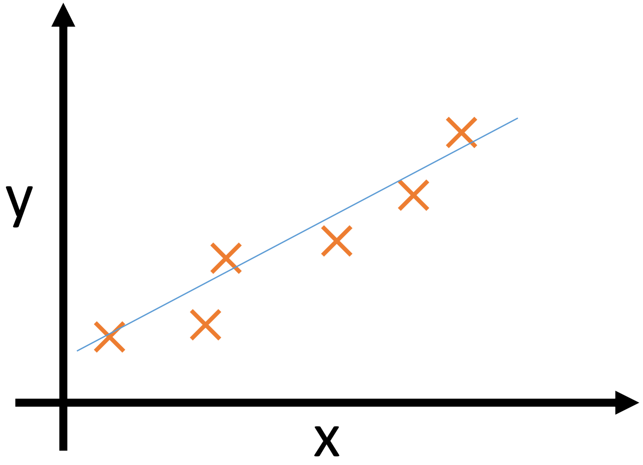
$\theta_0 = 1.5$
$\theta_1 = 0$



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



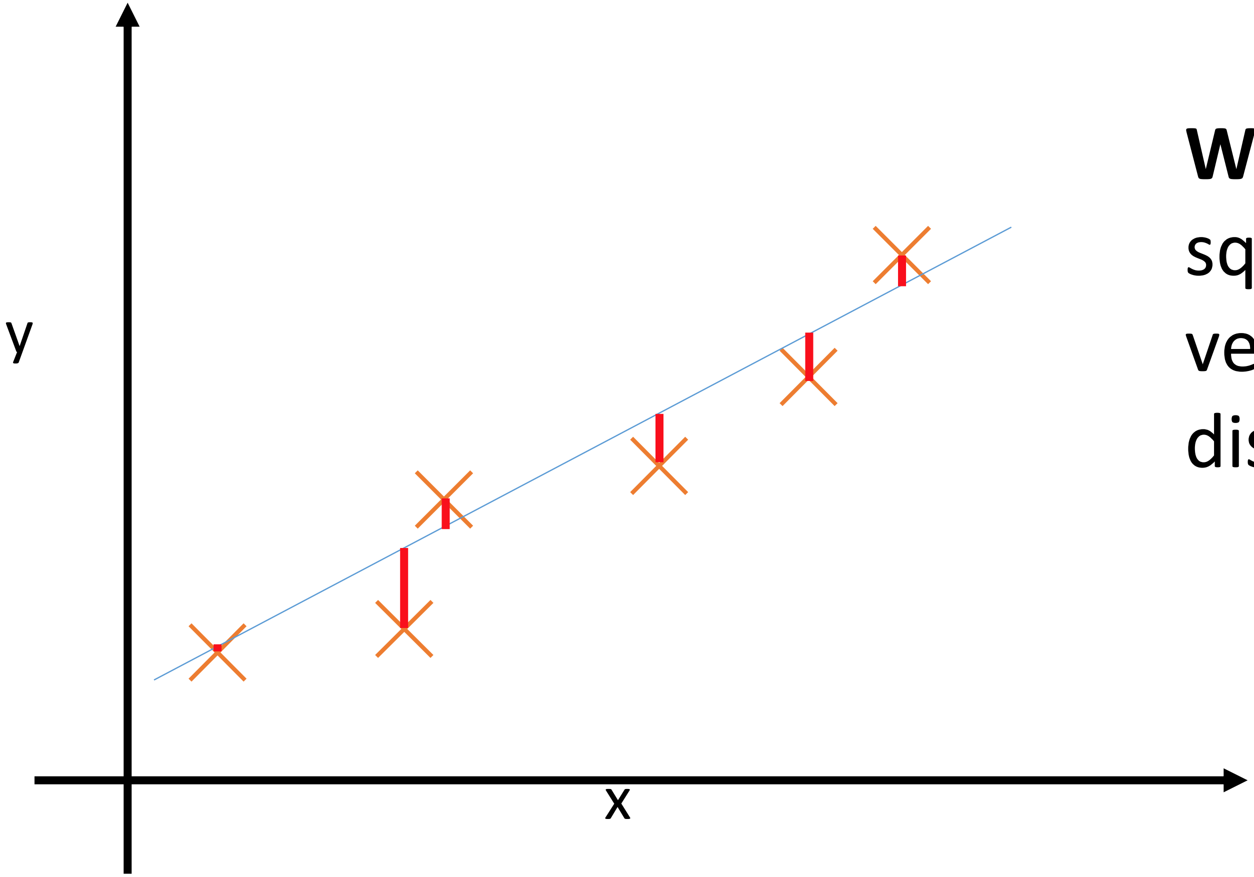
$\theta_0 = 1$
$\theta_1 = 0.5$



But what does “close” mean?

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

Quadratic cost function – on the board



We choose:
squared
vertical
distance

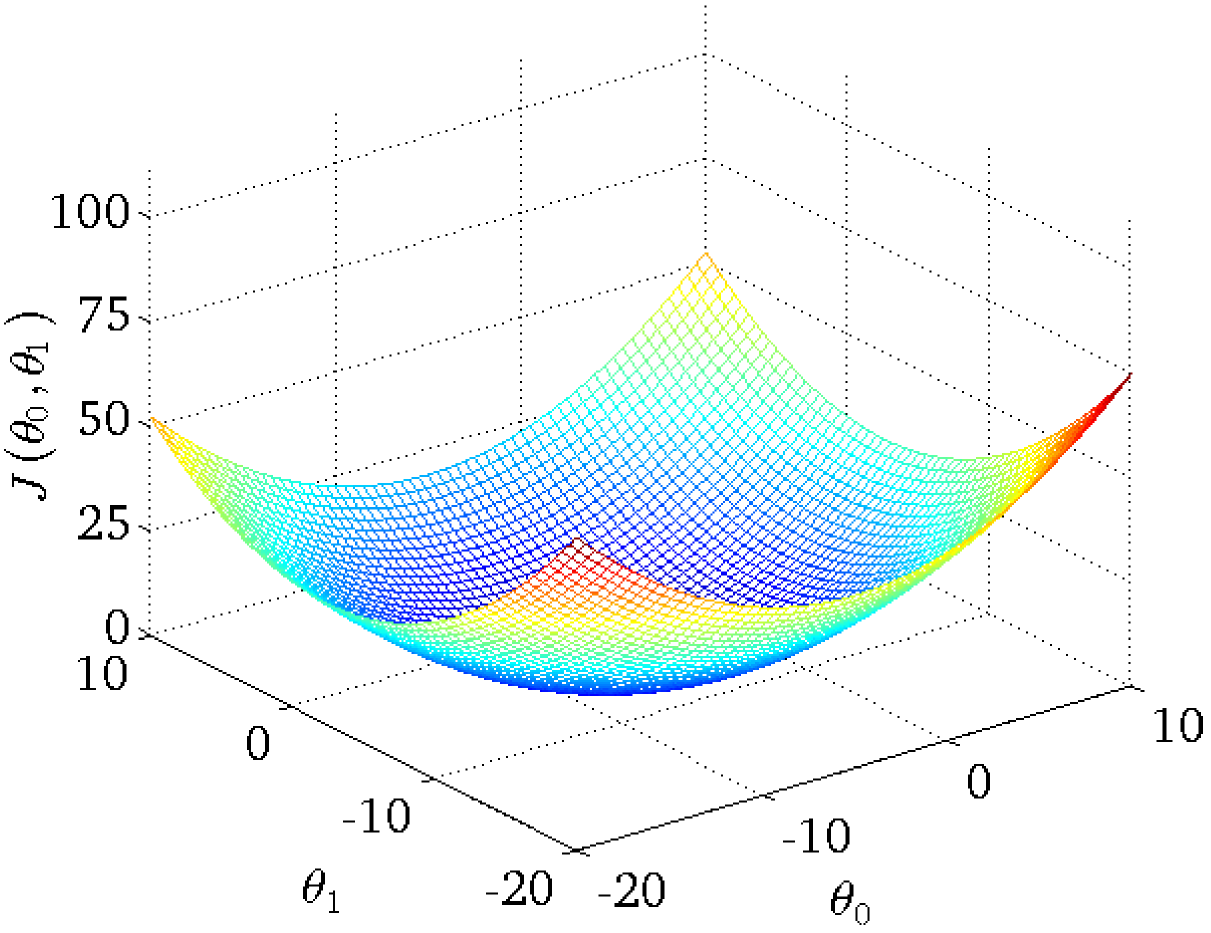
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

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

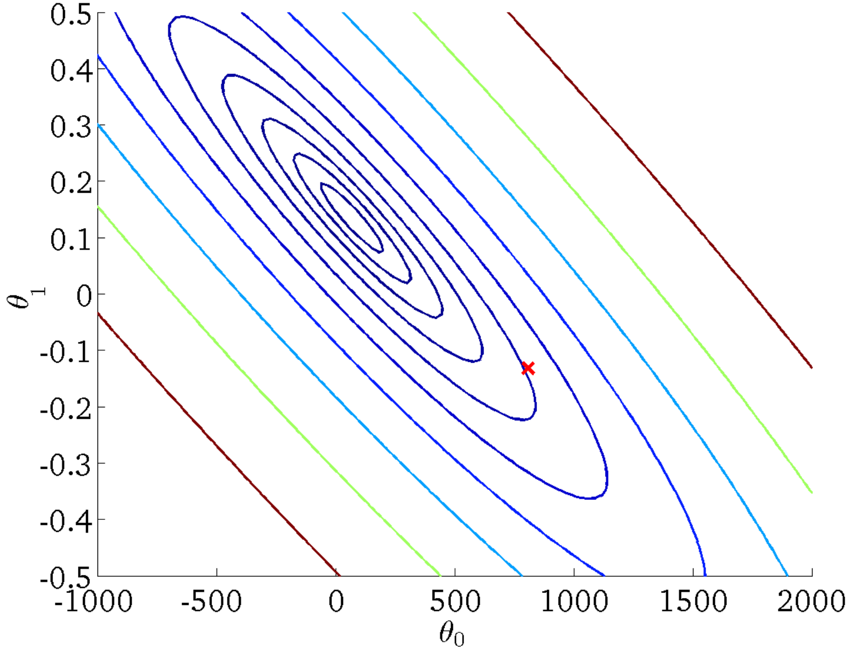
Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Cost Function Surface Plot



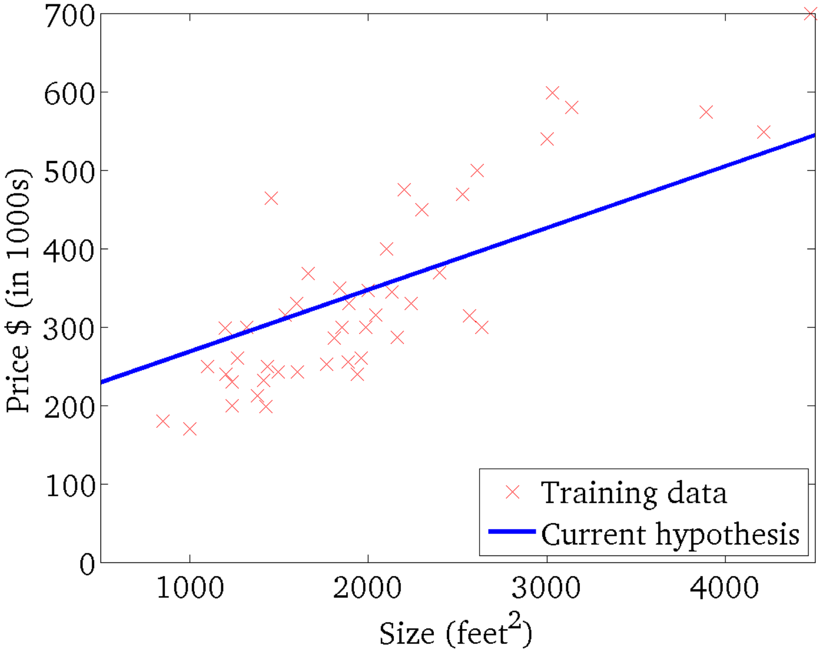
Contour Plots

- For a function F(x, y) of two variables, assigned different colours to different values of F
- Pick some values to plot
- The result will be *contours* – curves in the graph along which the values of F(x, y) are constant



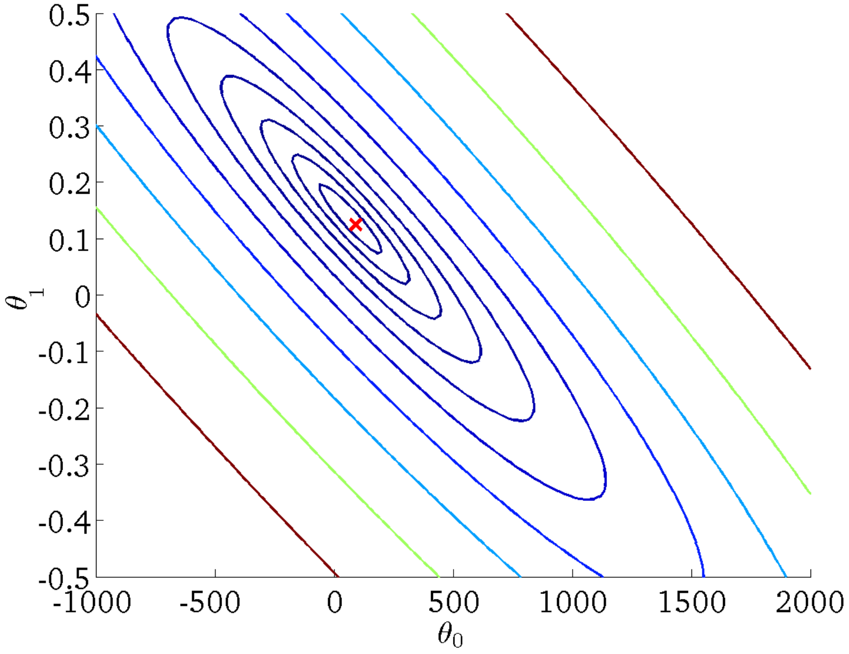
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

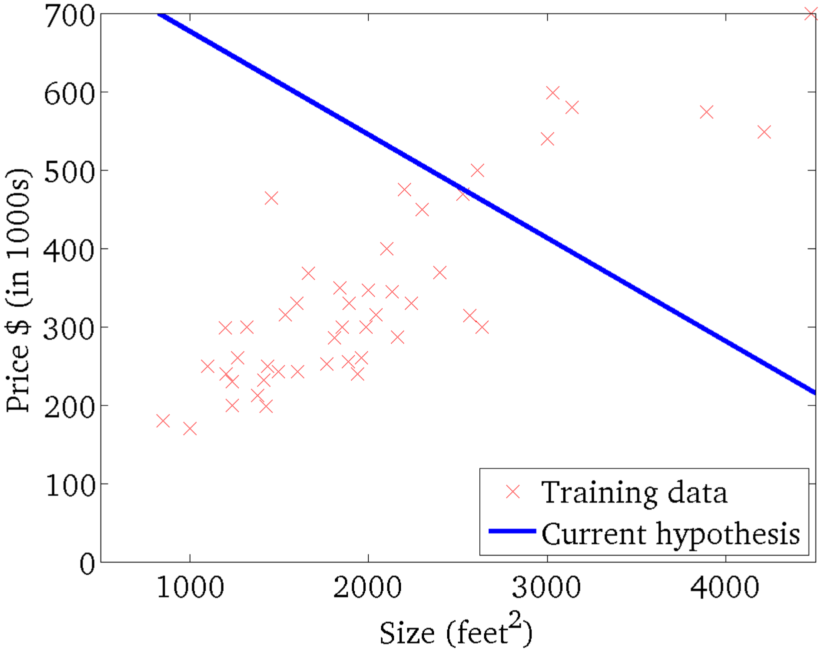
(function of the parameters θ_0, θ_1)



Cost Function Contour Plot

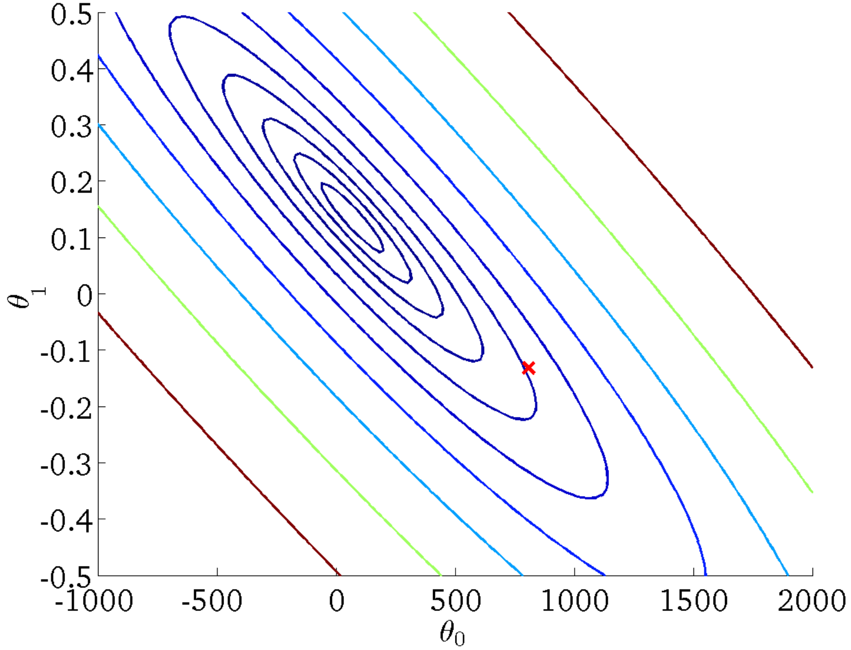
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

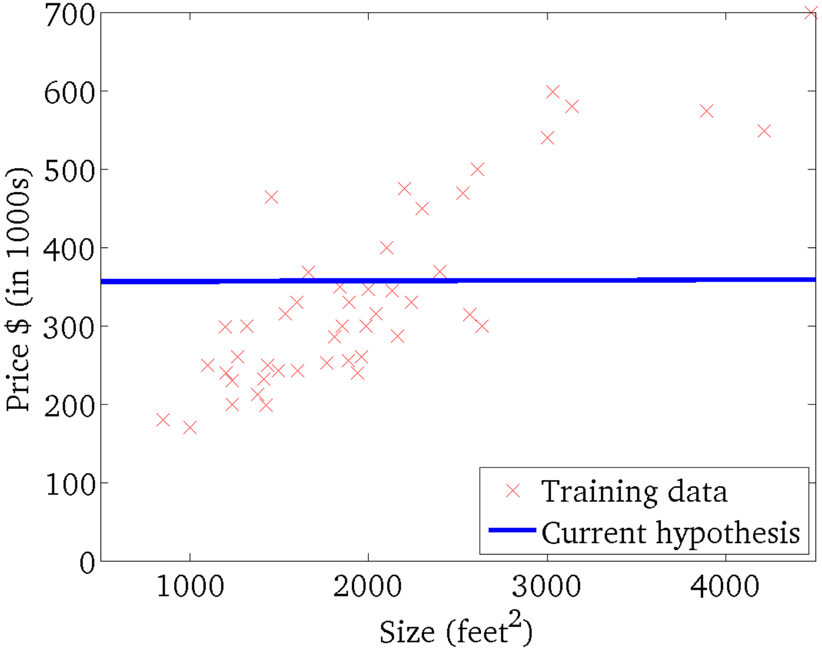


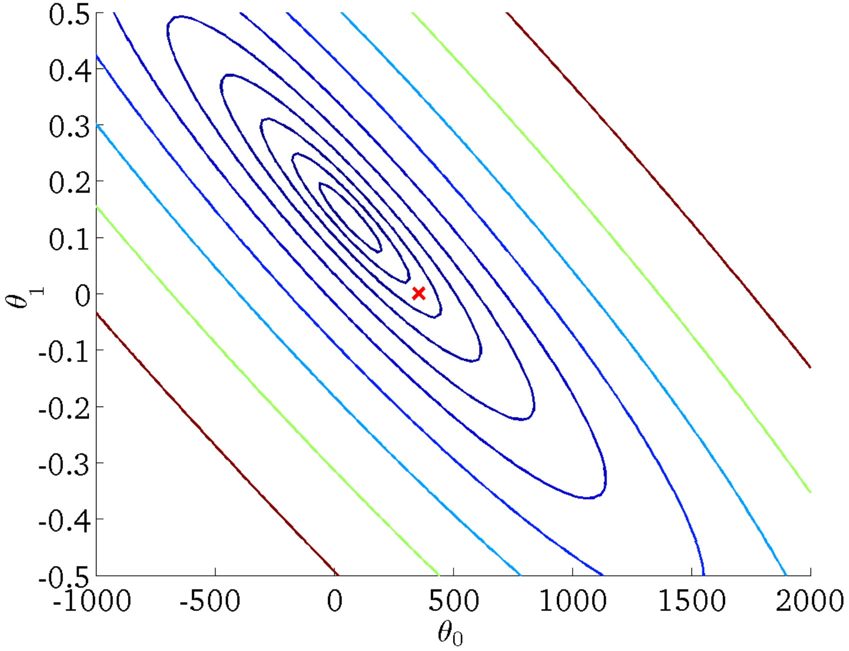
$h_\theta(x)$

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

$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)





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

Want $\min_{\theta_0,\theta_1} J(\theta_0, \theta_1)$

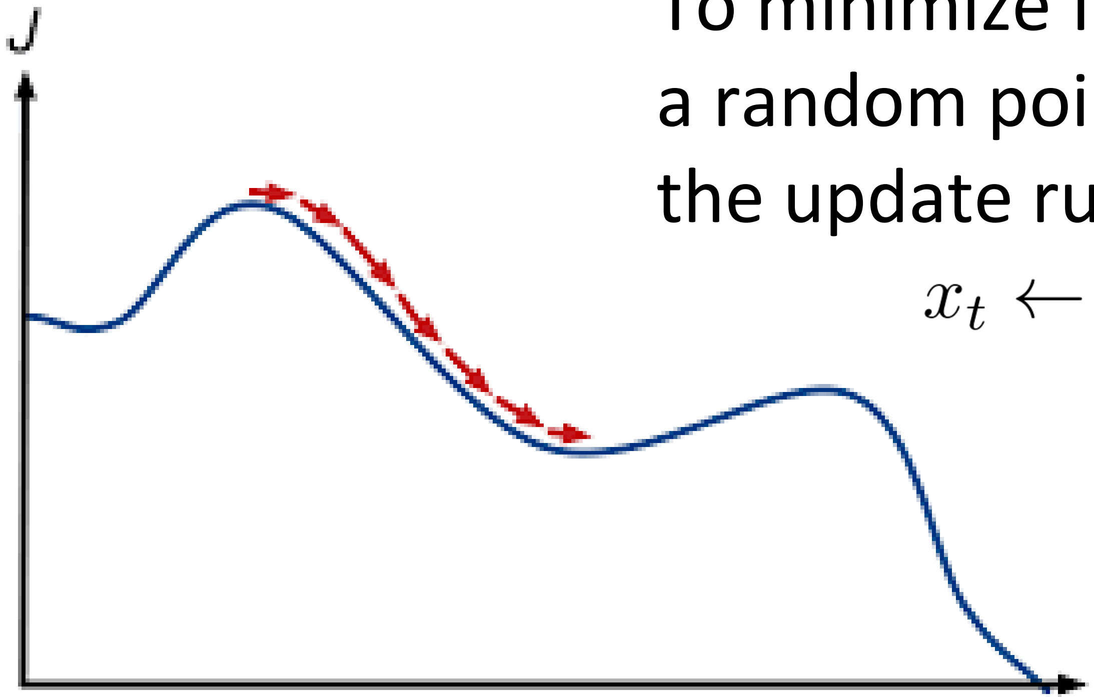
Outline:

- 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 a minimum

Gradient Descent on the board

Gradient Descent in 1D

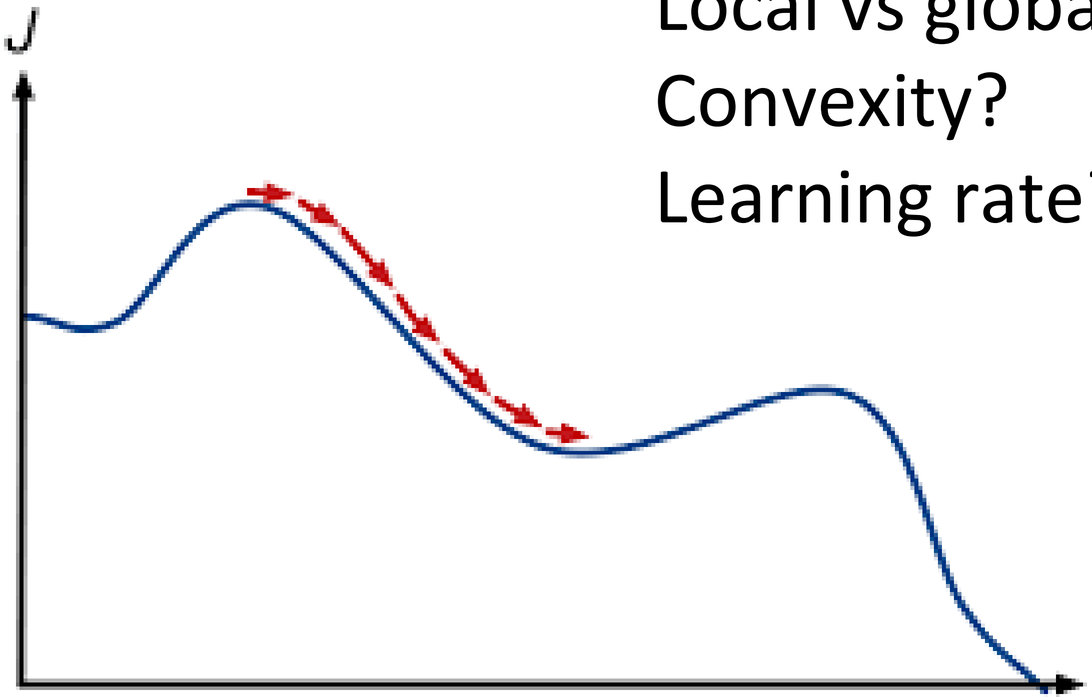


To minimize f(x), we start with a random point and iterate with the update rule:

$$x_t \leftarrow x_{t-1} - \alpha \frac{df}{dx}(x_{t-1})$$

Image Credit: http://www.deepideas.net/deep-learning-from-scratch-iv-gradient-descent-and-backpropagation/

Things to consider:



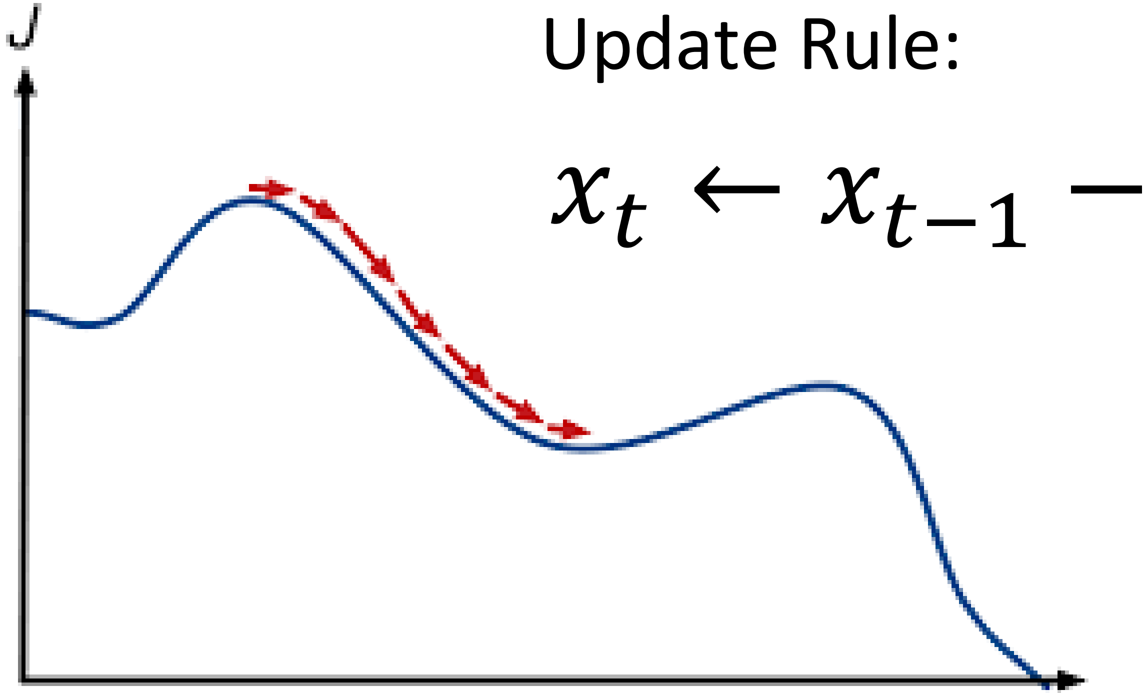
Local vs global minima?

Convexity?

Learning rate? (α)

Image Credit: http://www.deepideas.net/deep-learning-from-scratch-iv-gradient-descent-and-backpropagation/

Gradient Descent in Higher Dimensions

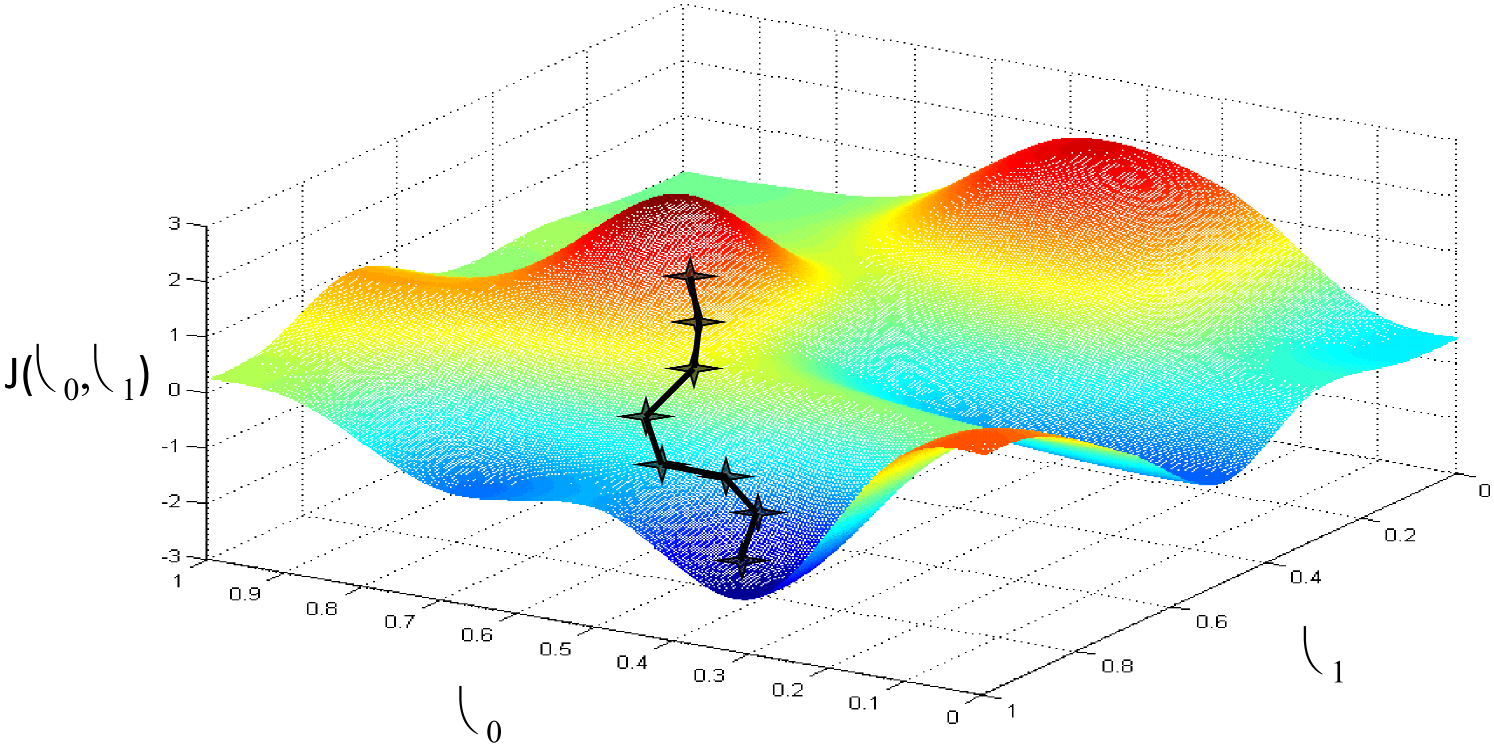


Update Rule:

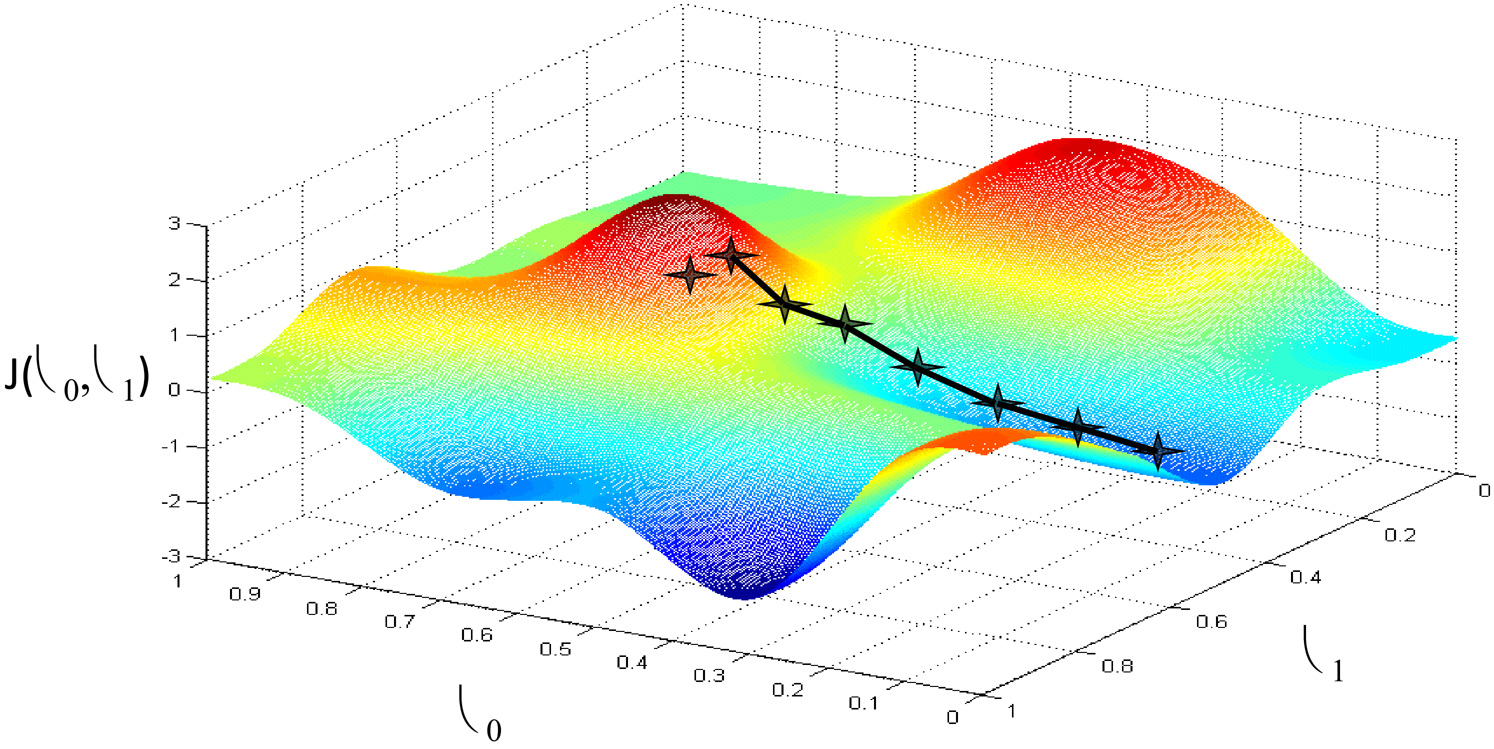
$$x_t \leftarrow x_{t-1} - \alpha \nabla f(x_{t-1})$$

Image Credit: http://www.deepideas.net/deep-learning-from-scratch-iv-gradient-descent-and-backpropagation/

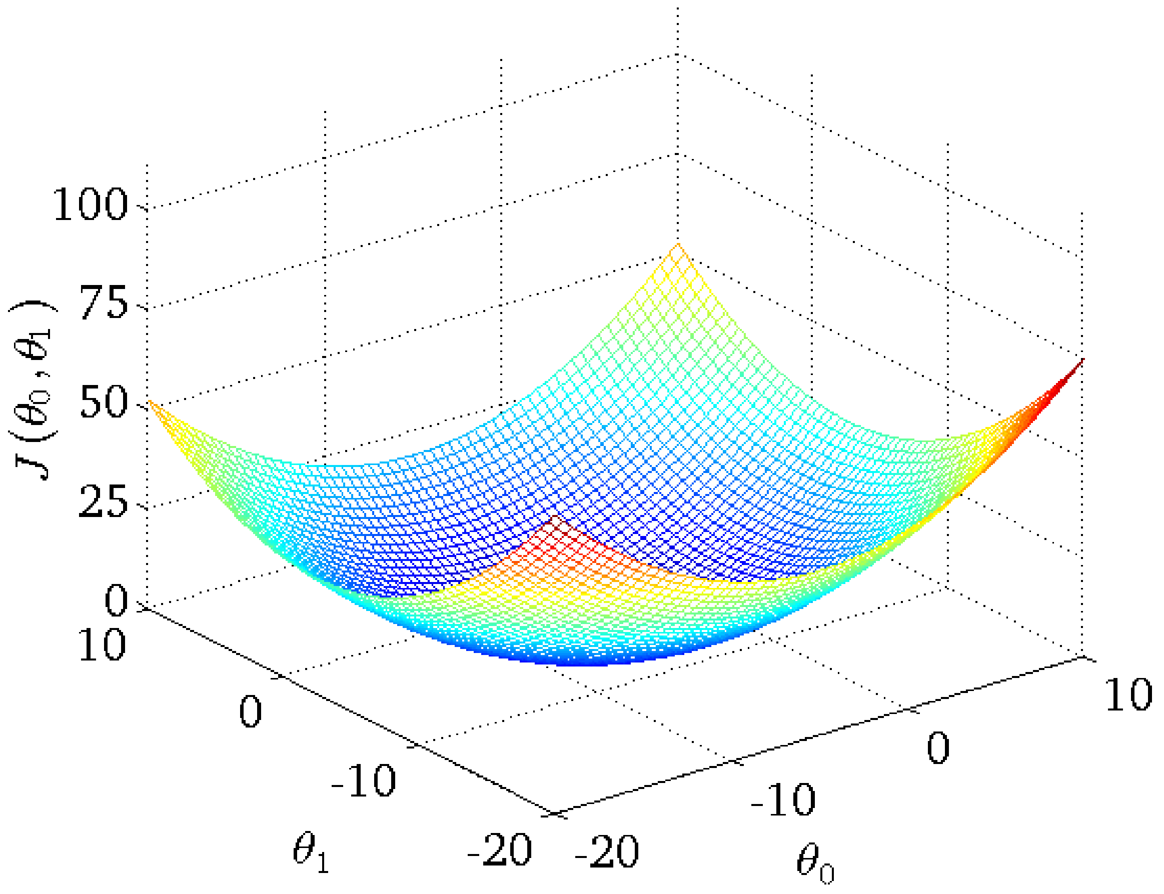
Gradient, on the board



J(θ0,θ1)
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ0
0
0.2
0.4
0.6
0.8
1
θ1


$J(\theta_0,\theta_1)$
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
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1

For Linear Regression, J is bowl-shaped (“convex”)



Gradient Descent Example

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

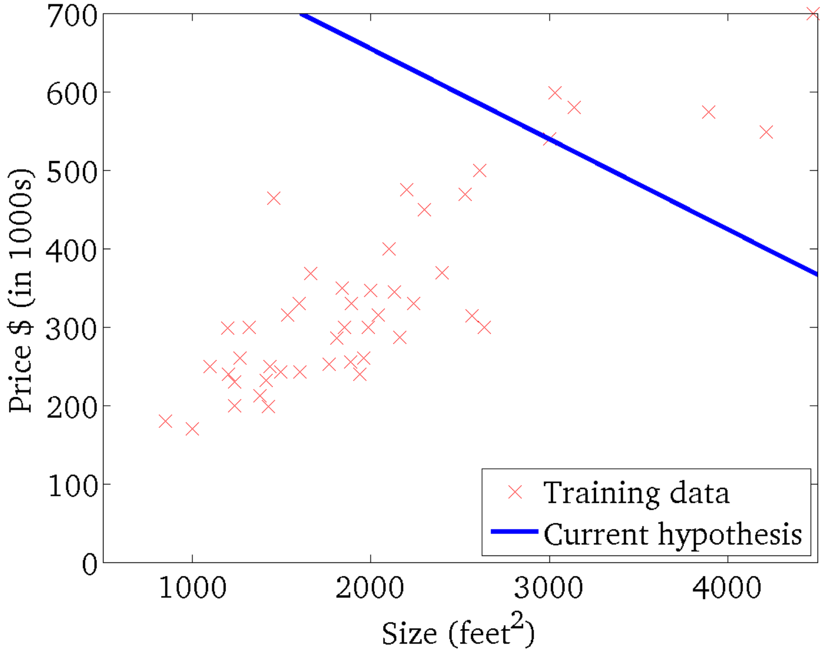
Parameters: θ_0, θ_1

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

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

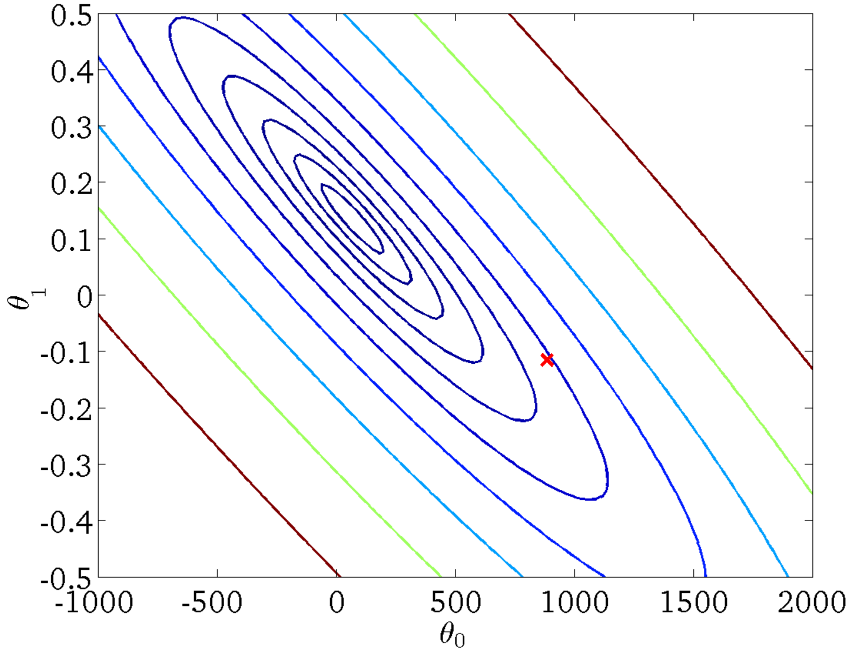
$h_\theta(x)$

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



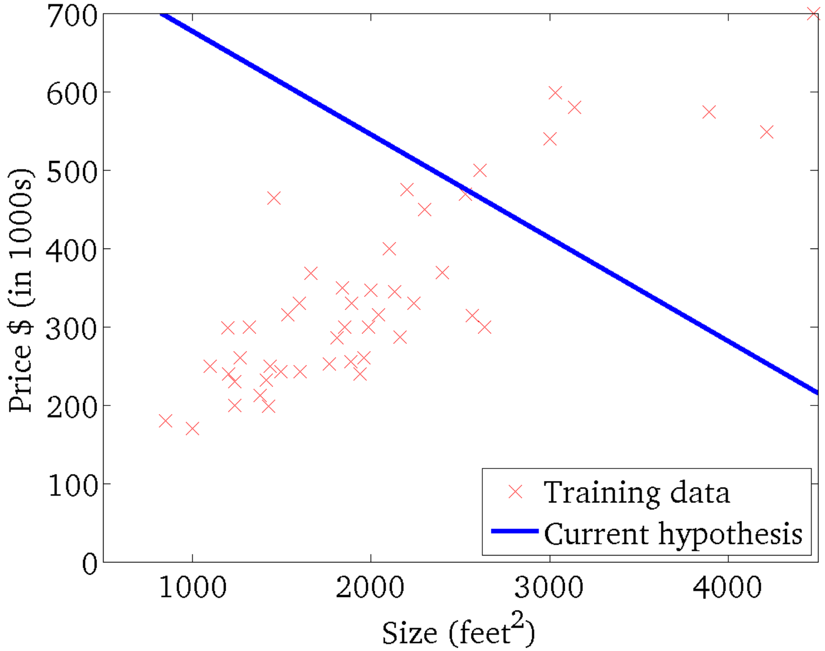
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



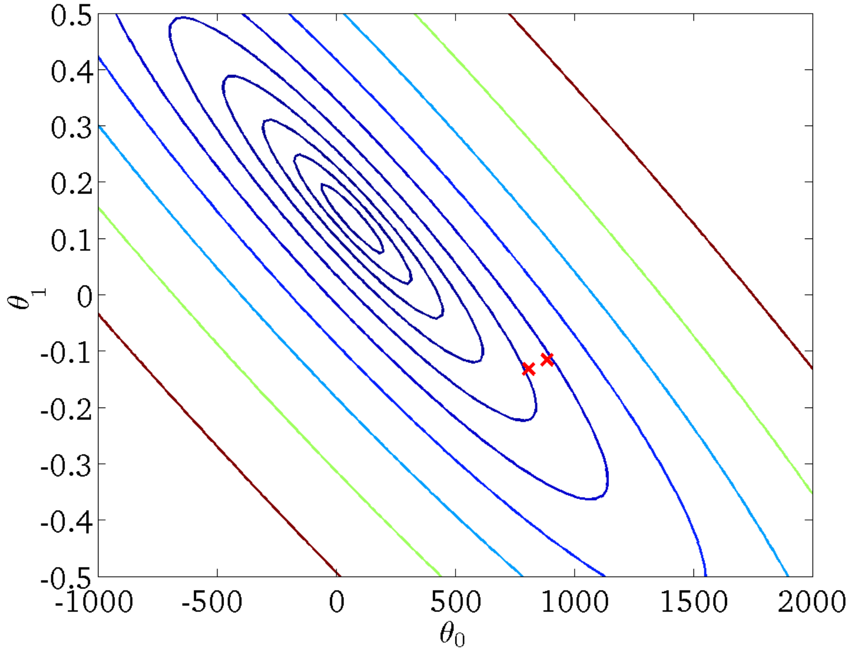
$h_\theta(x)$

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



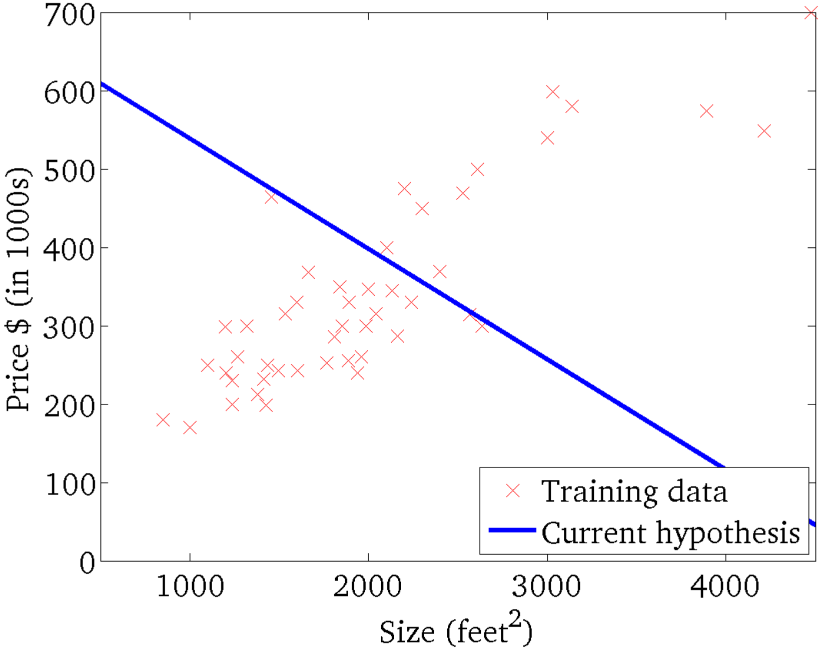
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



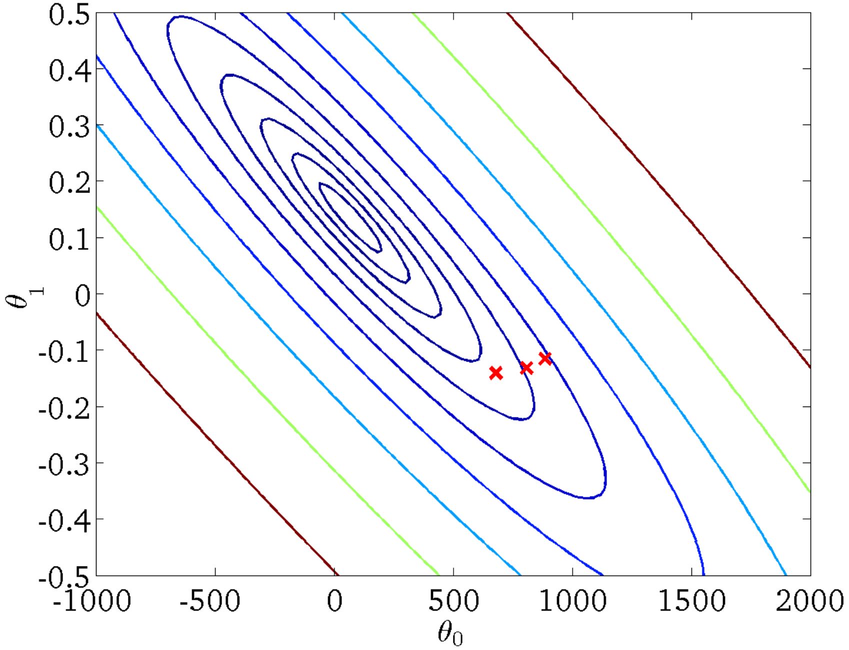
$h_\theta(x)$

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



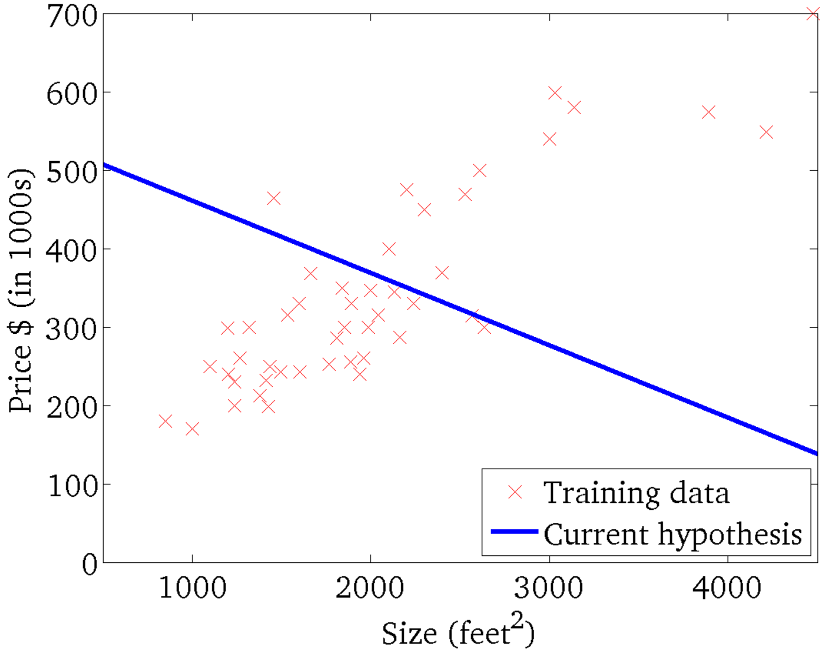
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



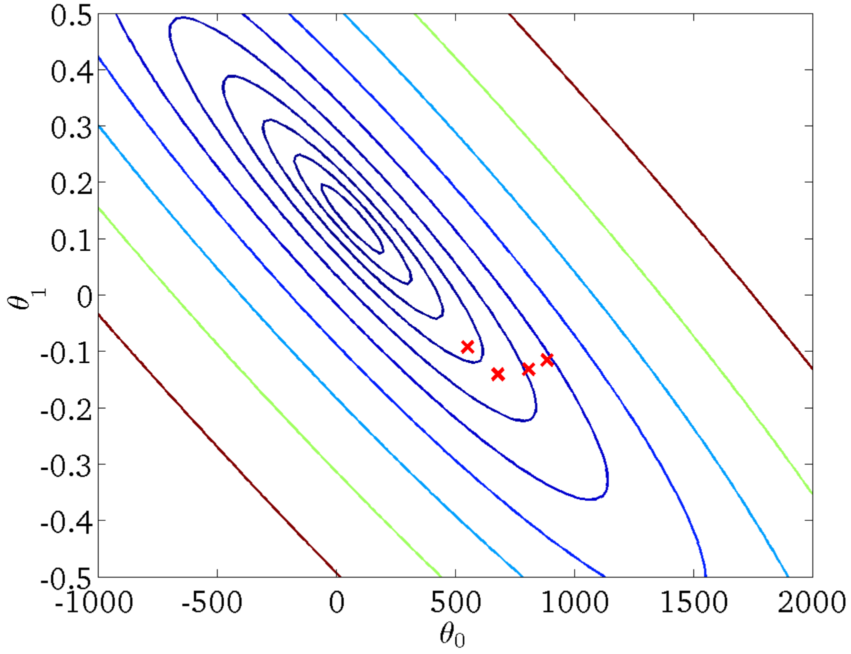
$h_\theta(x)$

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



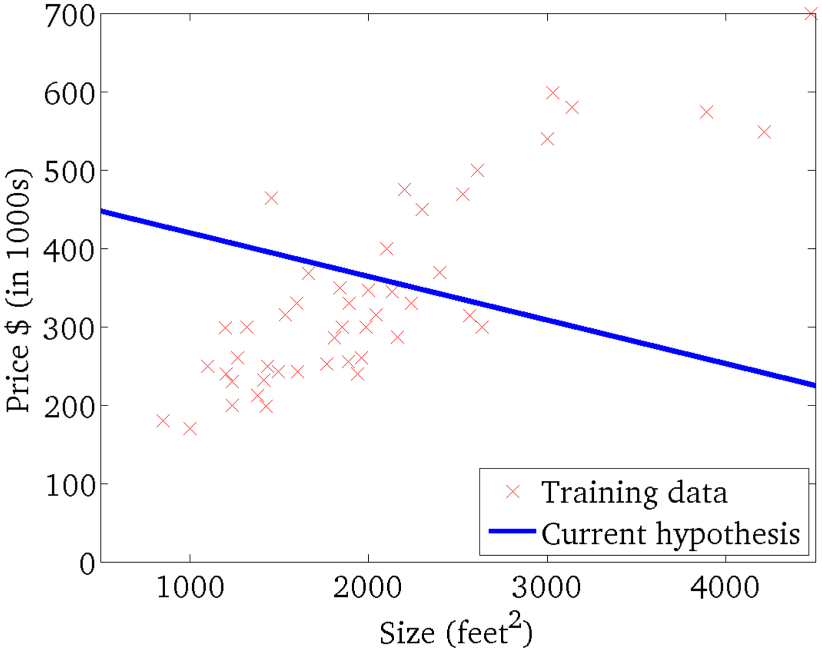
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



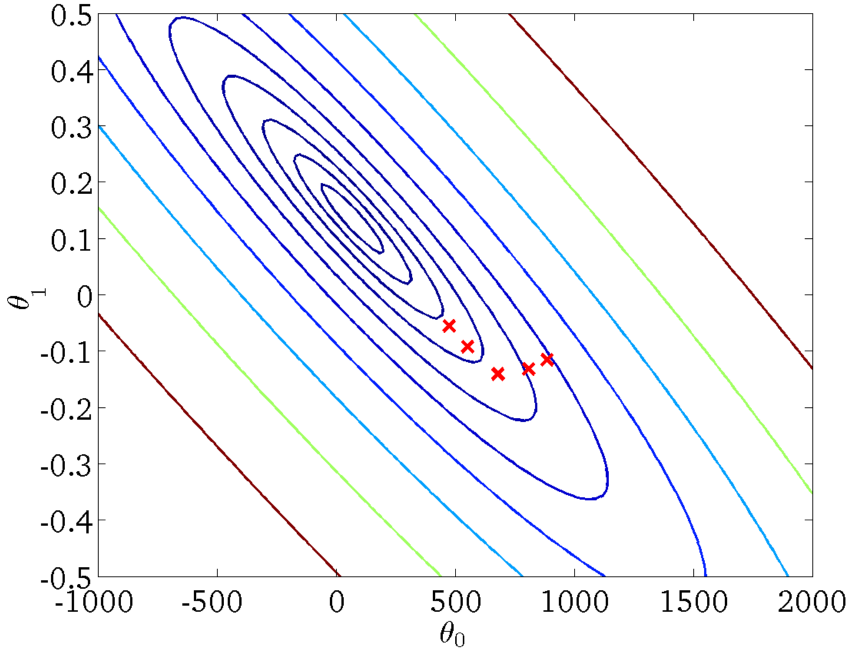
$$h_\theta(x)$$

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



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

(function of the parameters θ_0, θ_1)

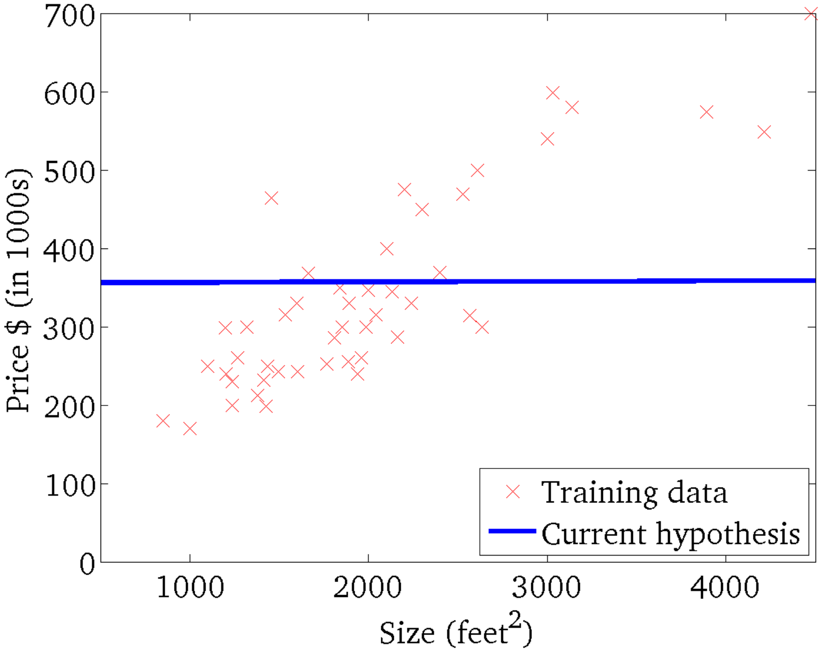


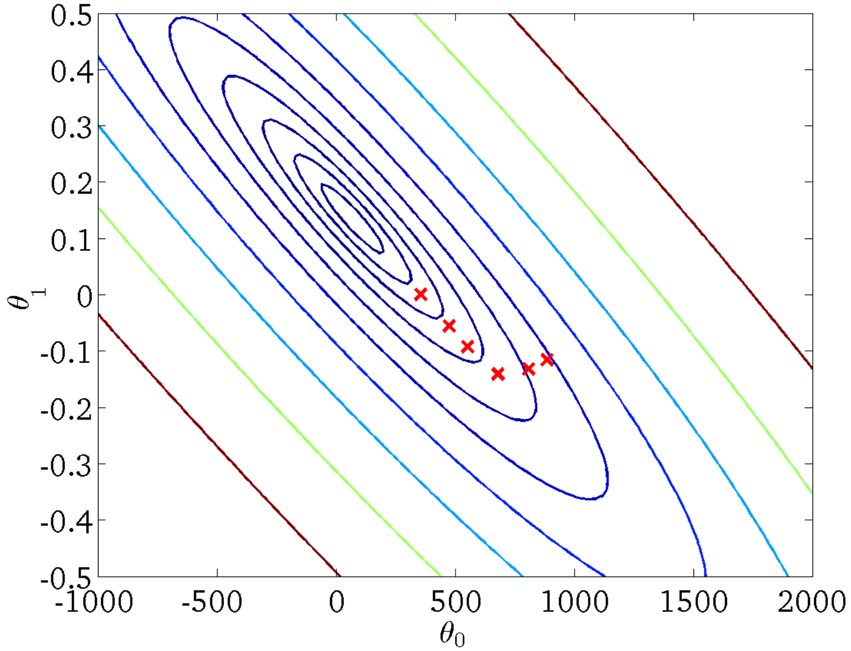
$h_\theta(x)$

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

$J(\theta_0, \theta_1)$

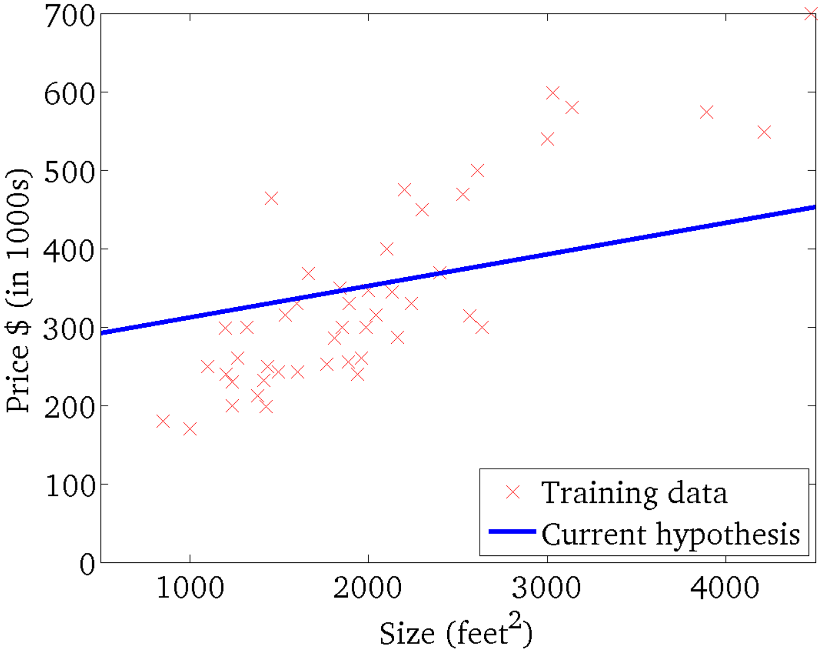
(function of the parameters θ_0, θ_1)





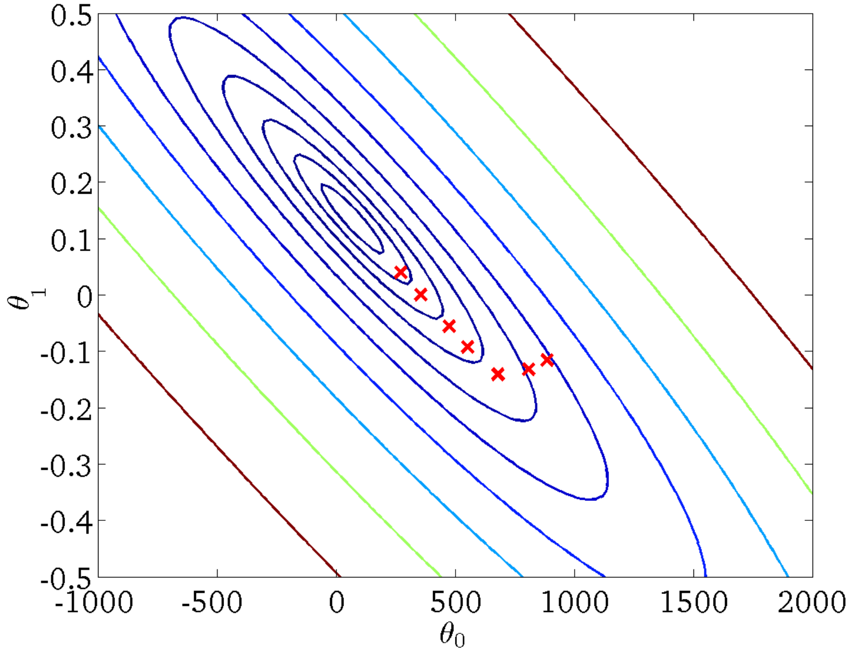
$h_\theta(x)$

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



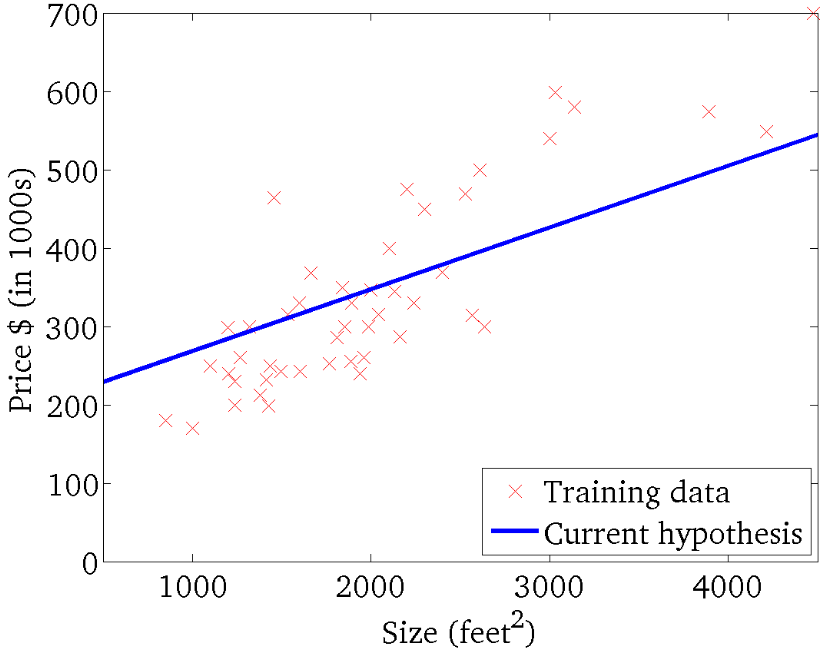
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



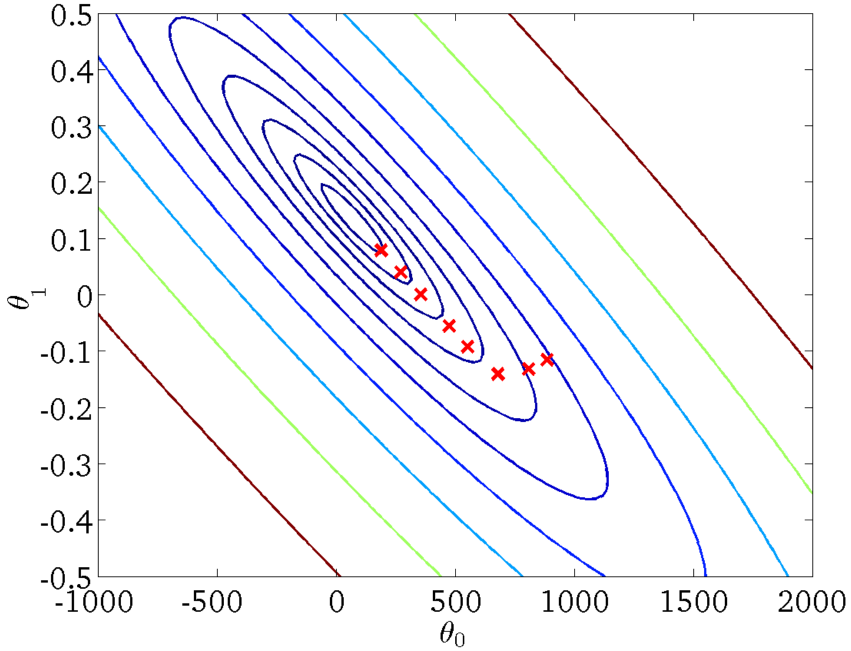
$h_\theta(x)$

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



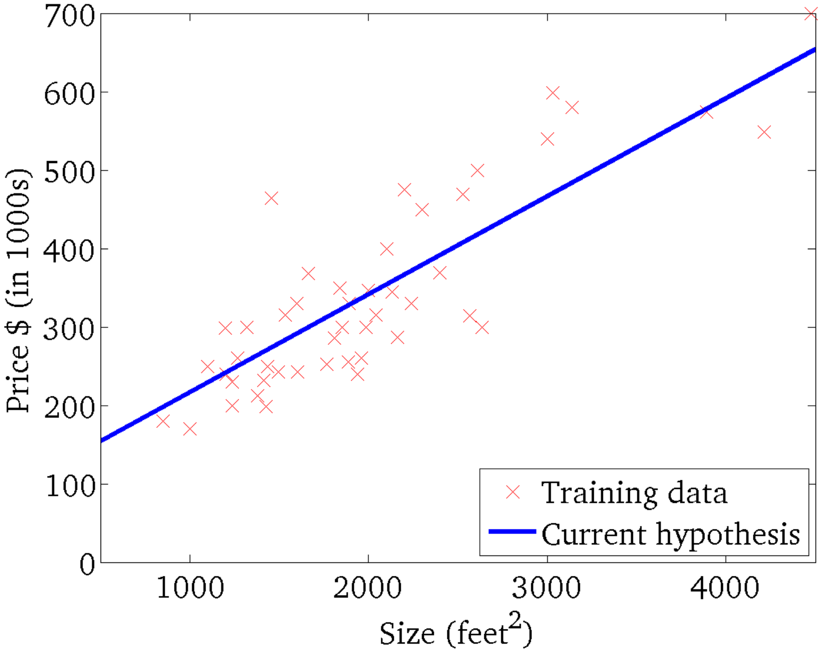
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



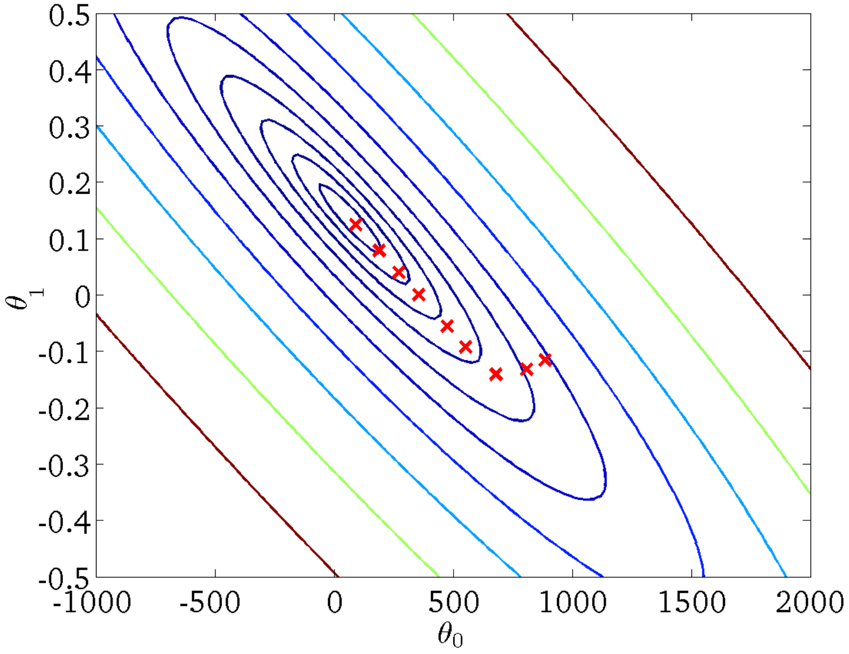
$$h_\theta(x)$$

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

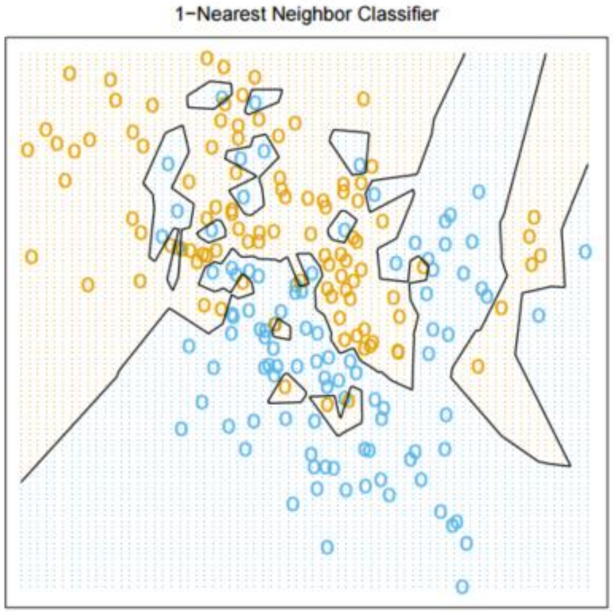


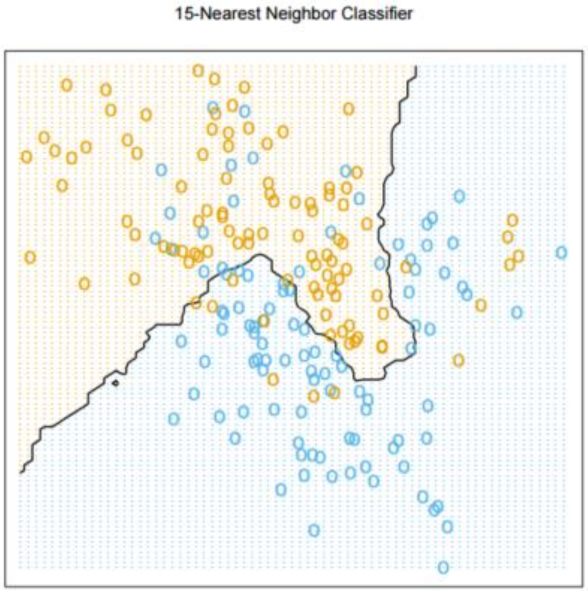
$$J(\theta_0, \theta_1)$$

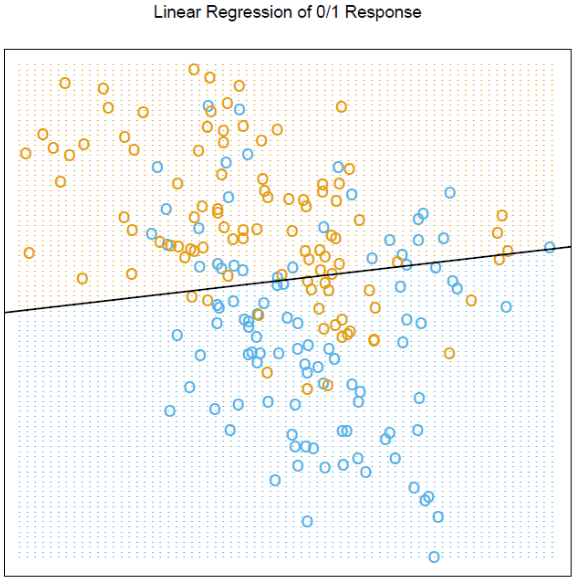
(function of the parameters θ_0, θ_1)



Linear Regression vs. k-Nearest Neighbours







Orange: y = 1
Blue: y = 0

Linear Regression vs. k-Nearest Neighbours

- Linear Regression: the boundary can only be linear
- Nearest Neighbours: the boundary can more complex
- Which is better?
    - Depends on what the *actual boundary* looks like
    - Depends on whether we have enough data to figure out the *correct* complex boundary