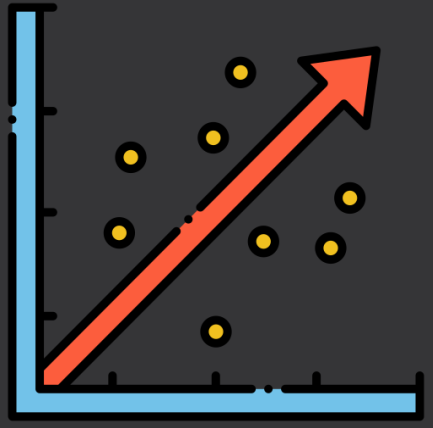


Simple Linear Regression

Lecture 7 – HCCDA-AI

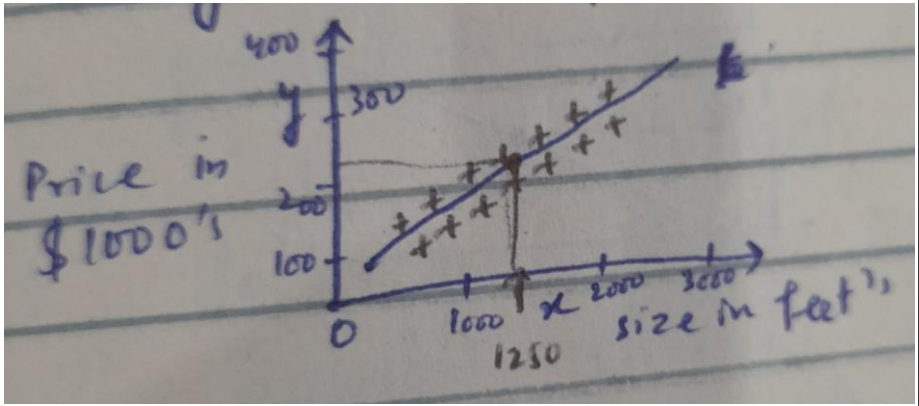
Imran Nawar

Simple Linear Regression

- Linear Regression with one variable.
- A type of regression algorithms that models the relationship between a dependent variable and a single variable.

$y = mx + c$ *line equation / slope-intercept form*

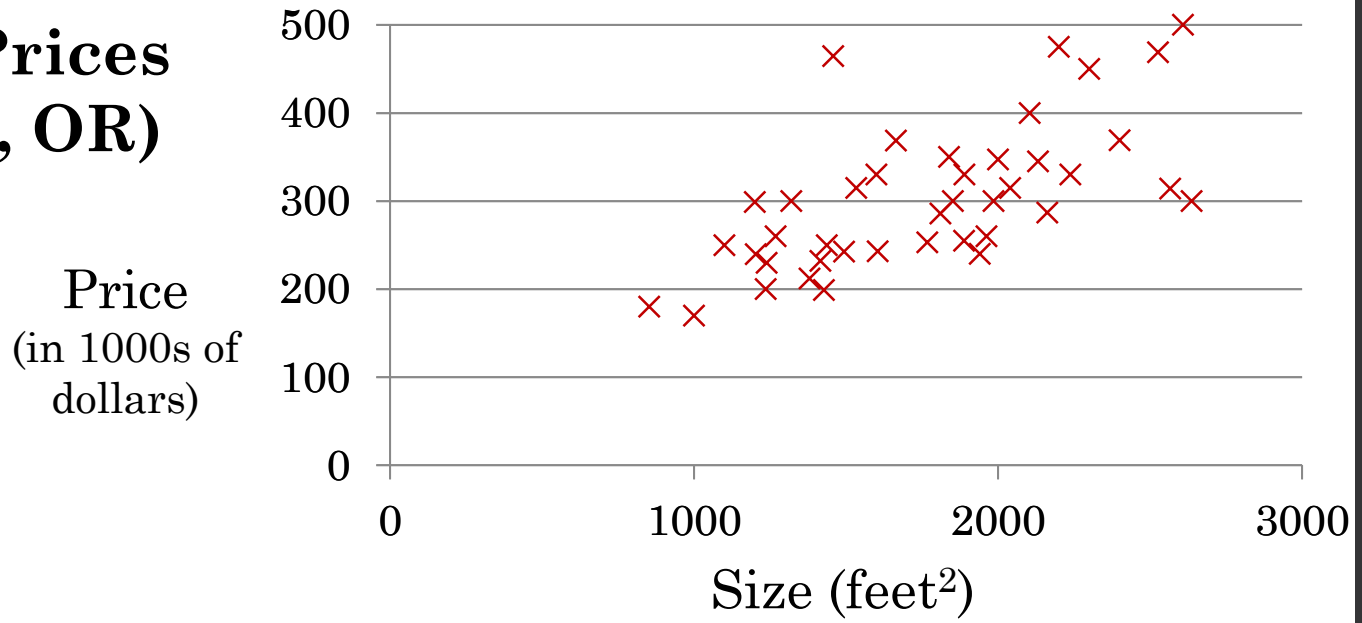
- **Independent variables** → input features
- **Dependent variables** → What we want to predict
- Finding a line that fits the data
- The training set is a subset of your data on which your model will learn how to predict the dependent variable with the independent variables.



Linear regression with one variable

Model Representation

Housing Prices (Portland, OR)



Supervised Learning

Given the "right answer" for each example in the data.

Regression Problem

Predict real-valued output

Training set of housing prices (Portland, OR)

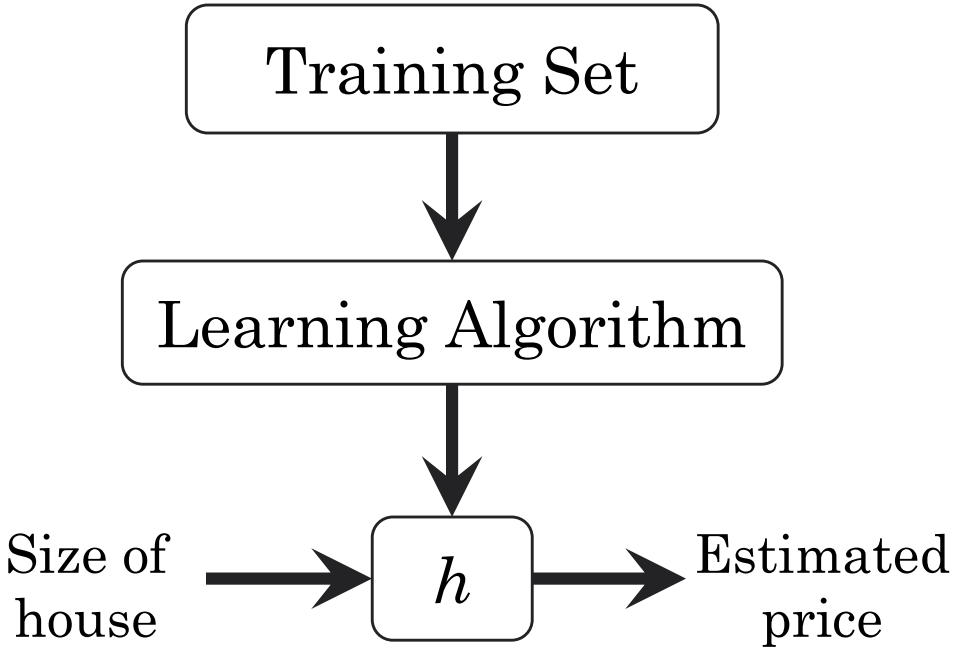
Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

Notation:

m = Number of training examples

x's = "input" variable / features

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



How do we represent h ?

Linear regression with one variable.
Univariate linear regression.

Linear regression with one variable

Cost Function

Training Set

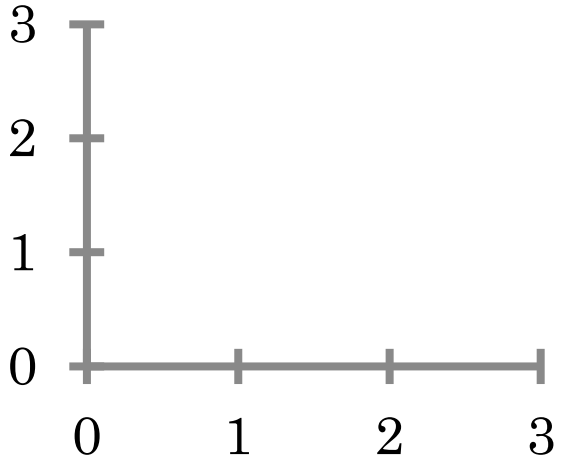
Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

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

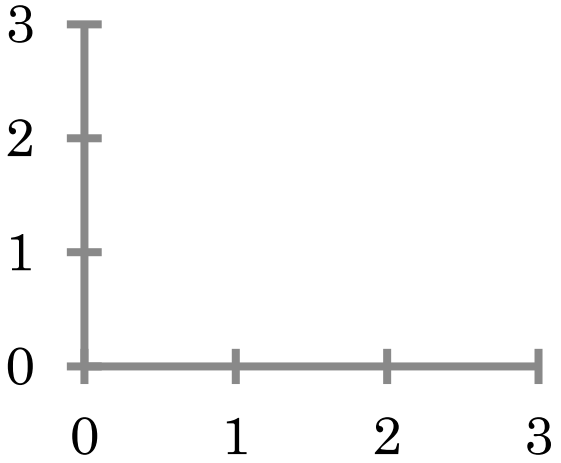
θ_i's: Parameters

How to choose θ_i's ?

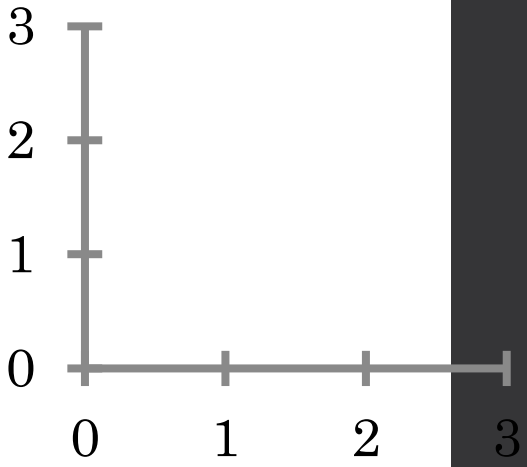
$h_\theta(x) = \theta_0 + \theta_1 x$



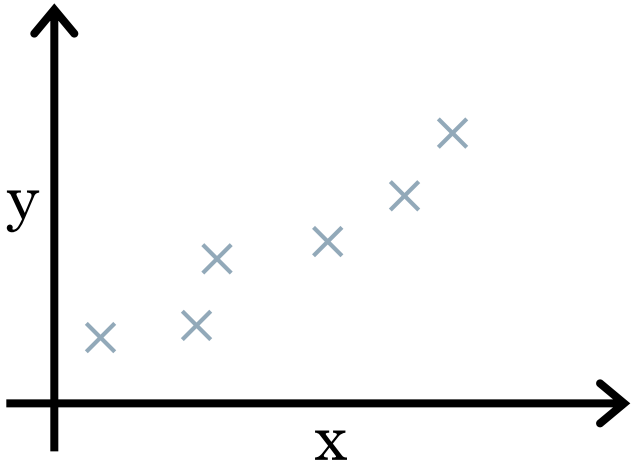
$\theta_0 = 1.5$
$\theta_1 = 0$



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



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



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

Cost Function

- Cost function represent the error between the actual value and the predicted value.
- A cost function, also known as a loss function or objective function, is a mathematical function that measures the difference between the predicted values of a model and the actual values in a dataset.

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

- Cost function can be different based on different models. As previously discussed, this is the cost function for linear regression.
- Lesser the cost function value, the more accurate will be the prediction. Higher the cost function value the lesser accurate will be the prediction.

Linear regression with one variable

Cost function intuition I

Hypothesis:

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

Parameters:

$$\theta_0, \theta_1$$

Cost Function:

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

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

Simplified

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

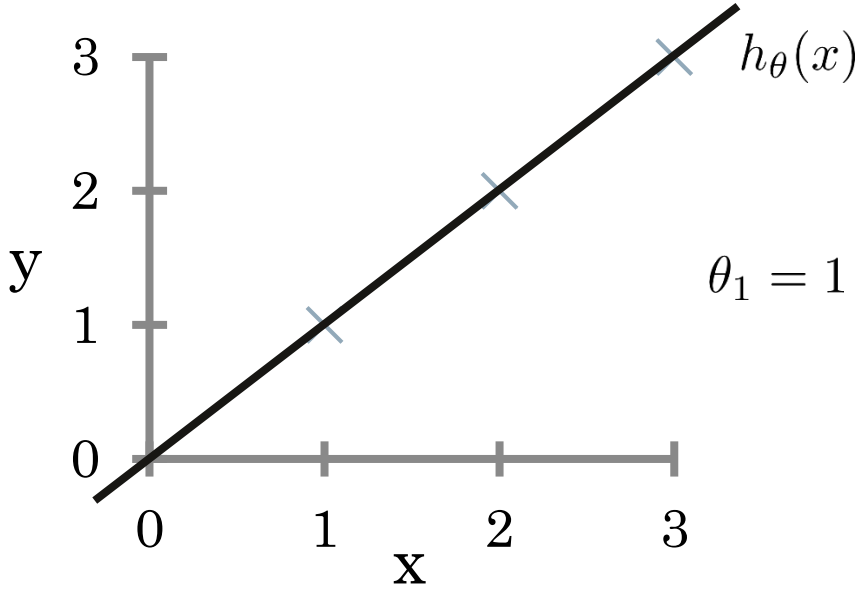
$$\theta_1$$

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

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

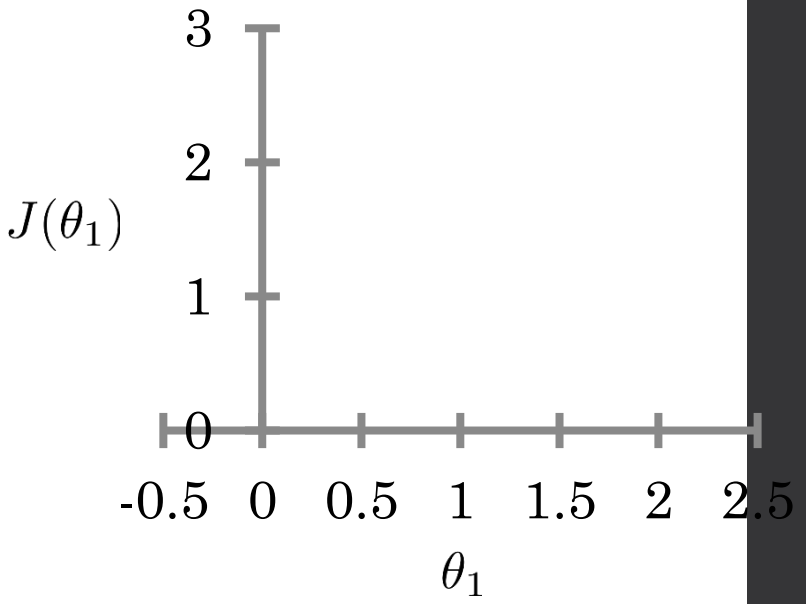
$h_\theta(x)$

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



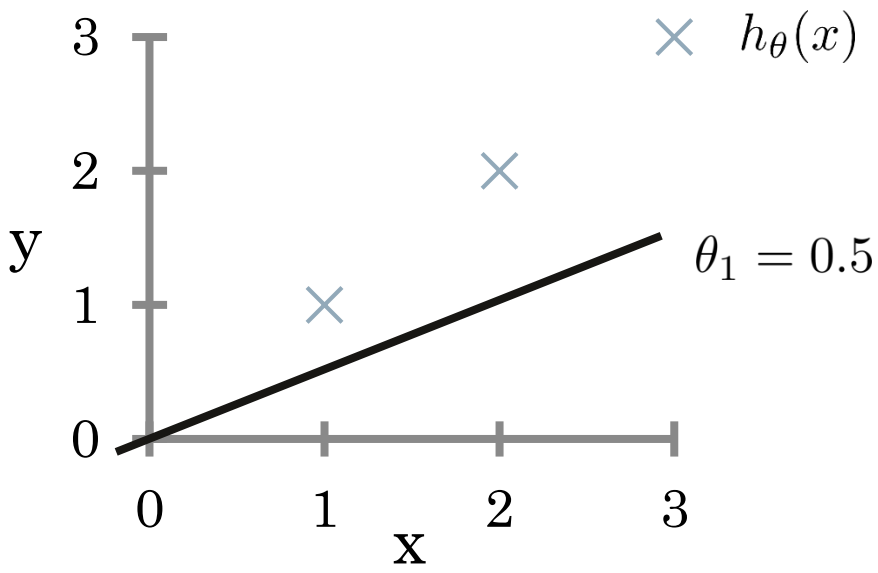
$J(\theta_1)$

(function of the parameter θ_1)



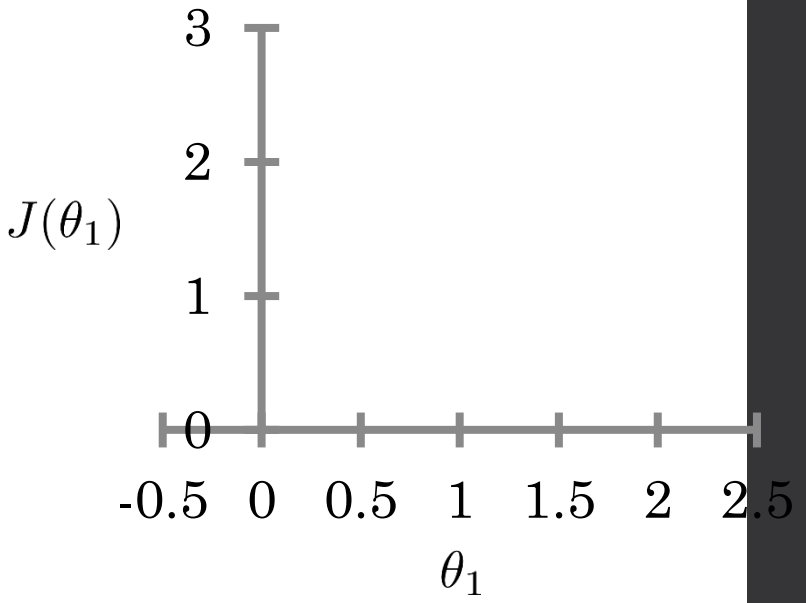
$h_\theta(x)$

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



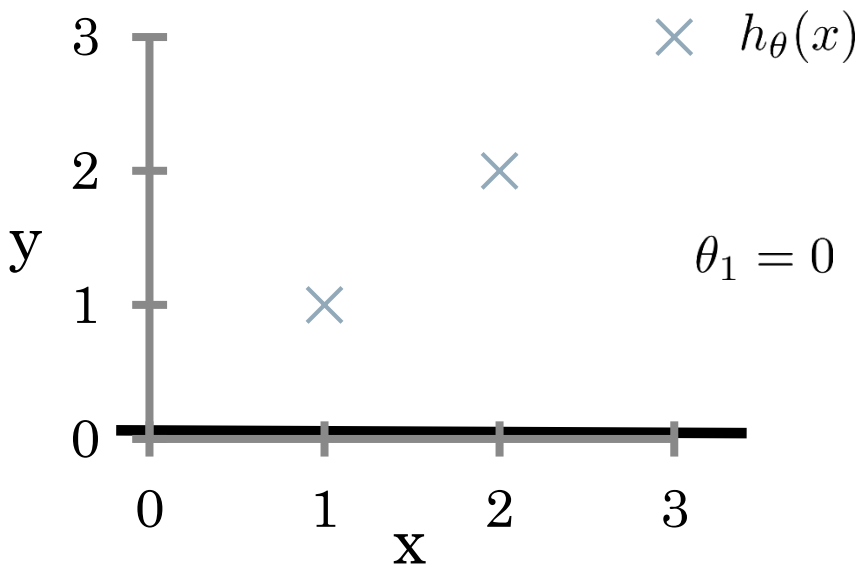
$J(\theta_1)$

(function of the parameter θ_1)



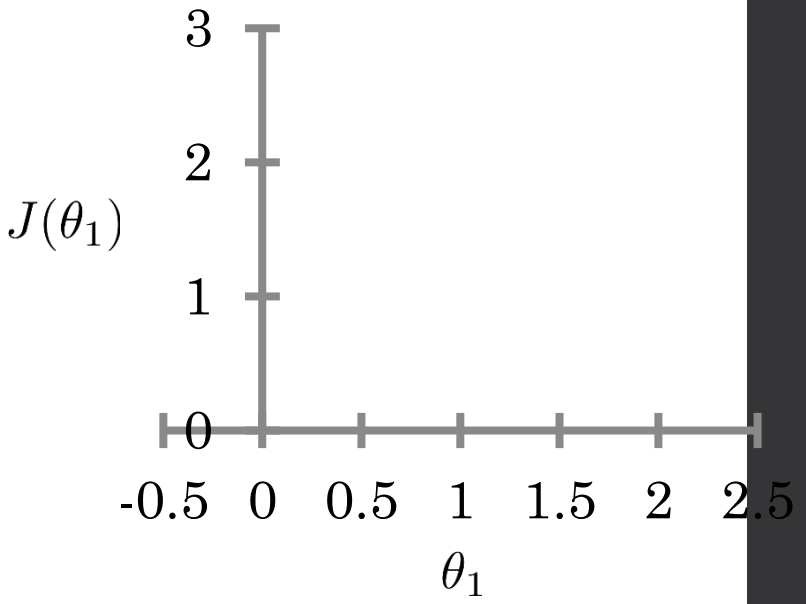
$h_\theta(x)$

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



$J(\theta_1)$

(function of the parameter θ_1)



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

- This is our objective function for linear regression.
- The optimization objective for our learning algorithm is we want to chose the value for θ, that minimize $J(\theta)$.

Types of Cost Functions:

The choice of cost function depends on the specific task and the type of model being trained. Here are some common types of cost functions:

1. Mean Squared Error (MSE):
   - Used in regression problems.
   - $MSE = \frac{1}{2m}\sum_{i=1}^{m}(y_i - \hat{y_i})^2$
2. Mean Absolute Error (MAE):
   - Used in regression tasks.
   - $MAE = \frac{1}{m}\sum_{i=1}^{m}|y_i - \hat{y_i}|$
3. Binary Cross Entropy Loss:
   - Used in binary classification problems.
4. Categorical Cross Entropy Loss:
   - Used in Multiclass classification problems.

Linear regression with one variable

Cost function intuition II

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

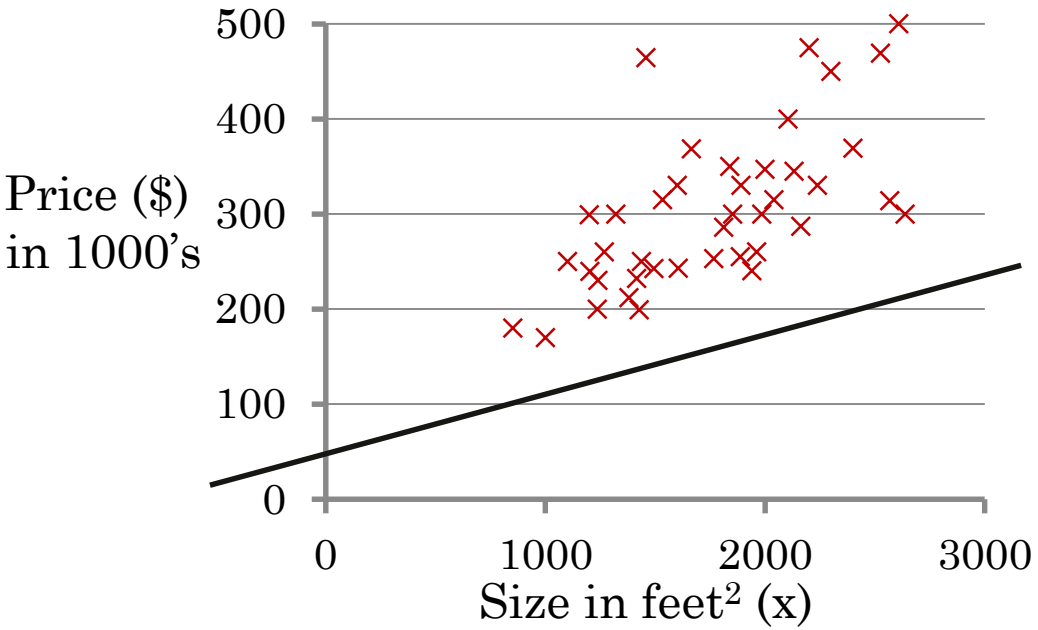
Parameters: θ_0, θ_1

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

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

$h_\theta(x)$

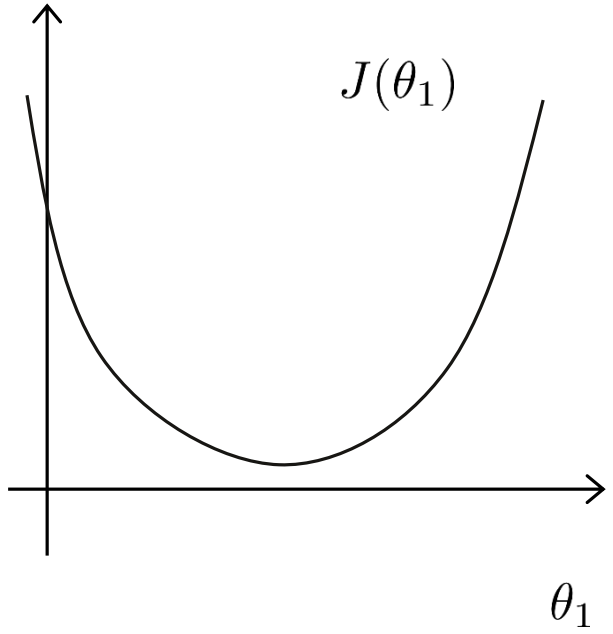
(for fixed θ_0, θ_1 , this is a function of x)



$h_\theta(x) = 50 + 0.06x$

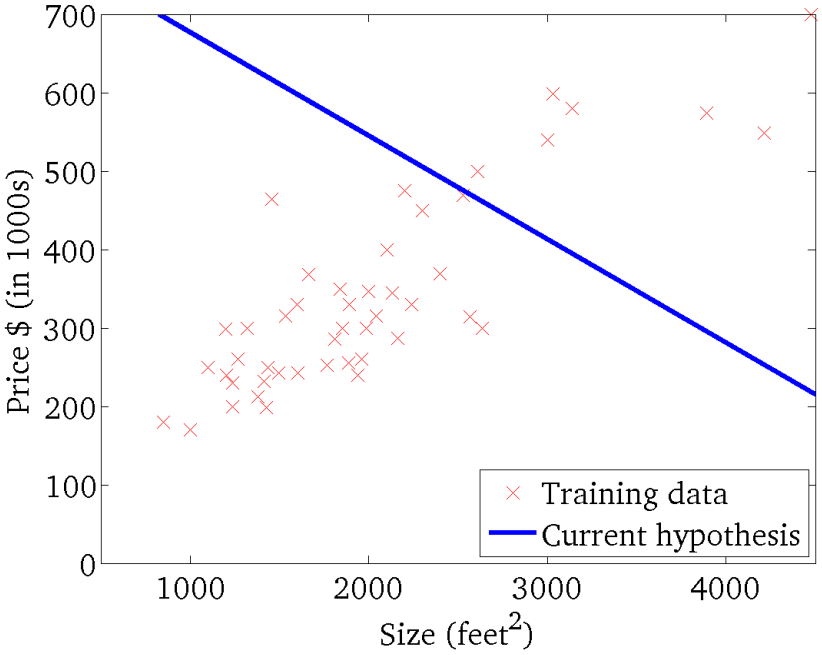
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



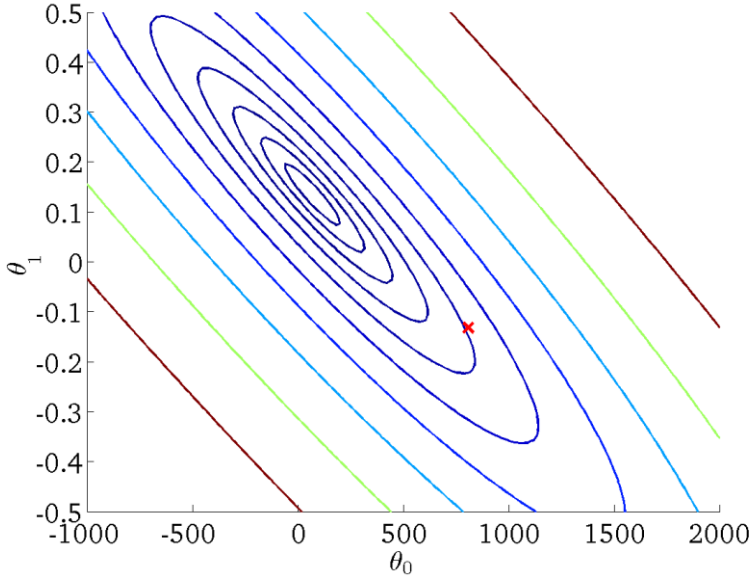
$h_\theta(x)$

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



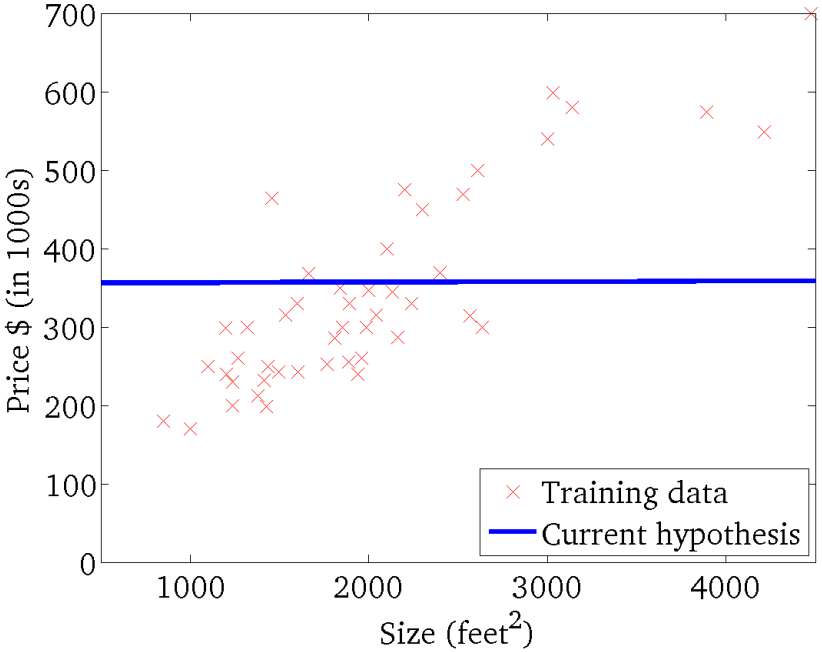
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



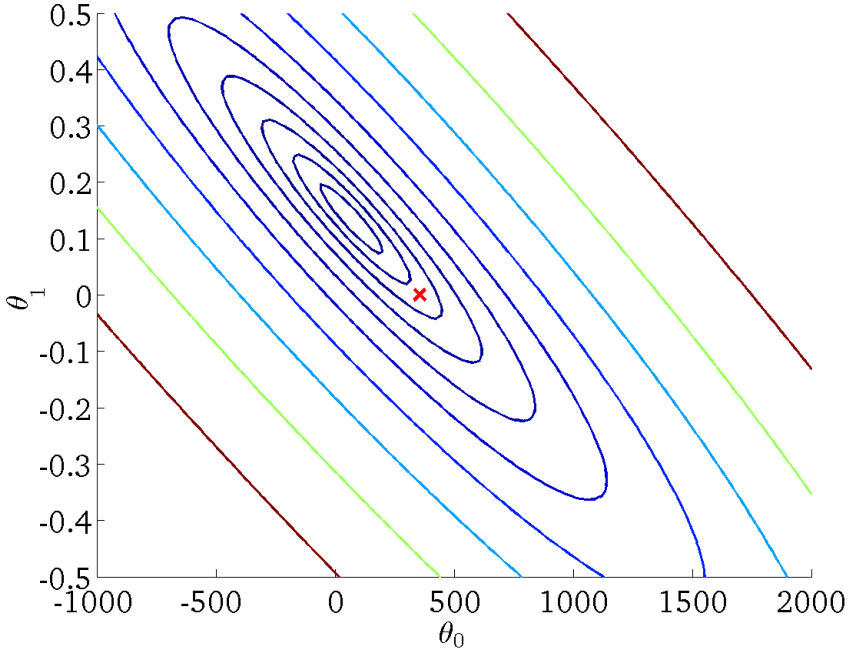
$h_\theta(x)$

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



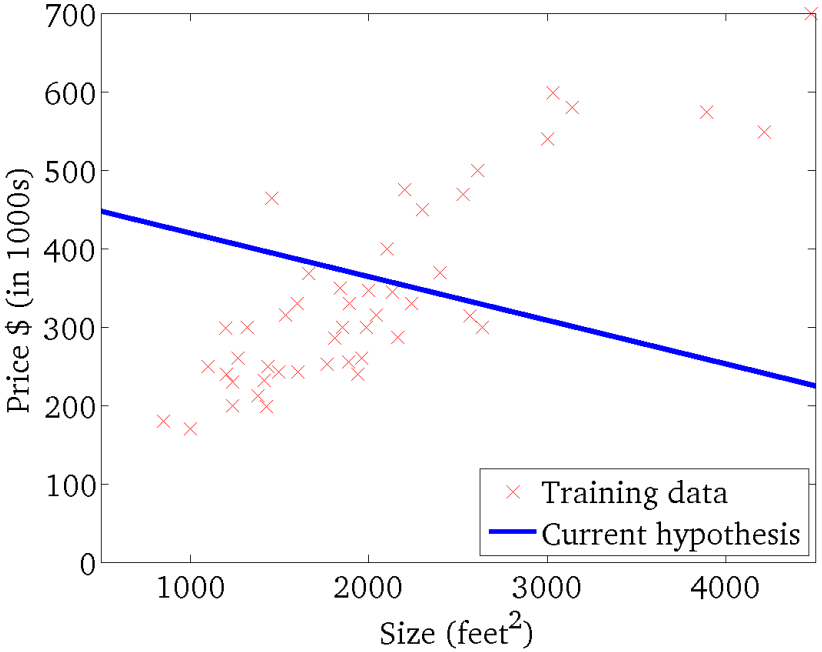
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



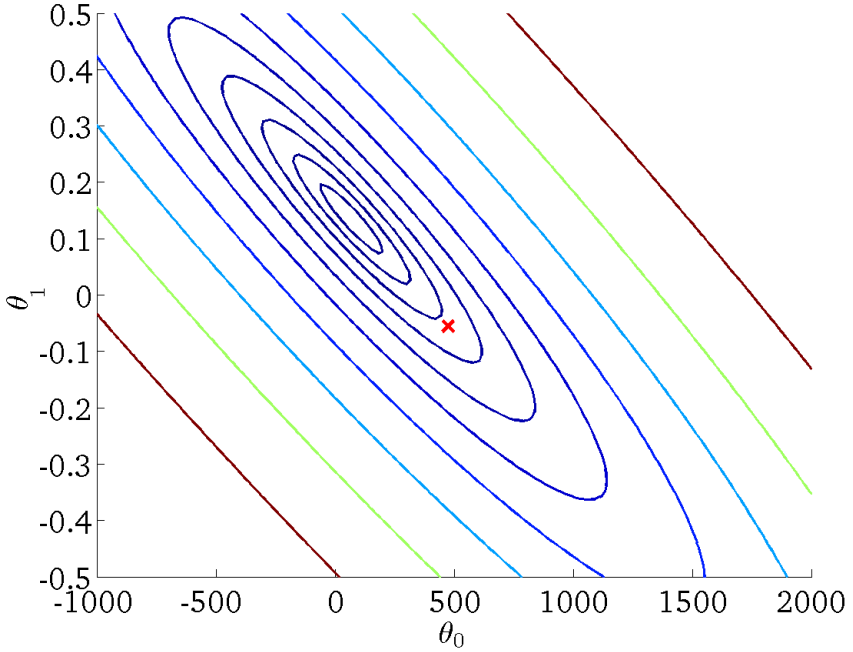
$h_\theta(x)$

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



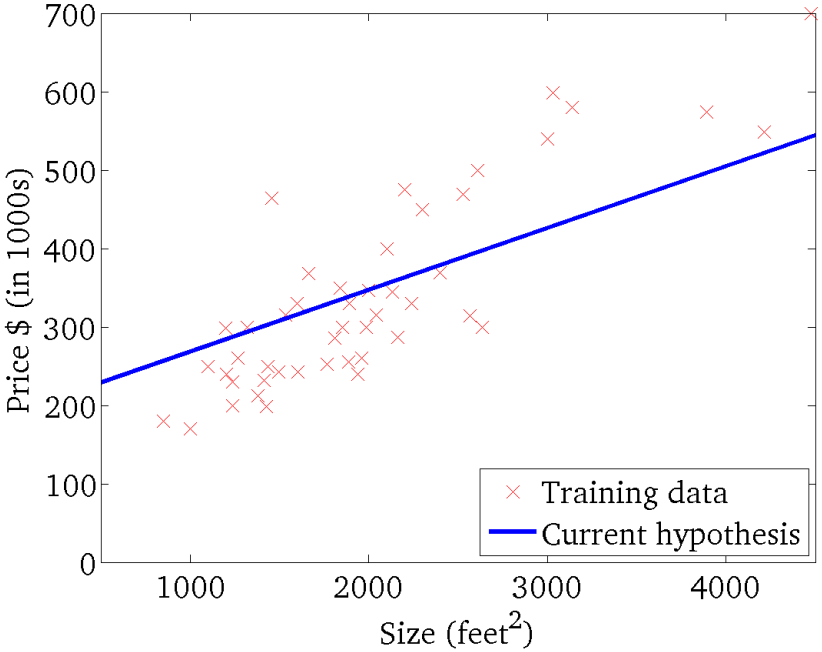
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



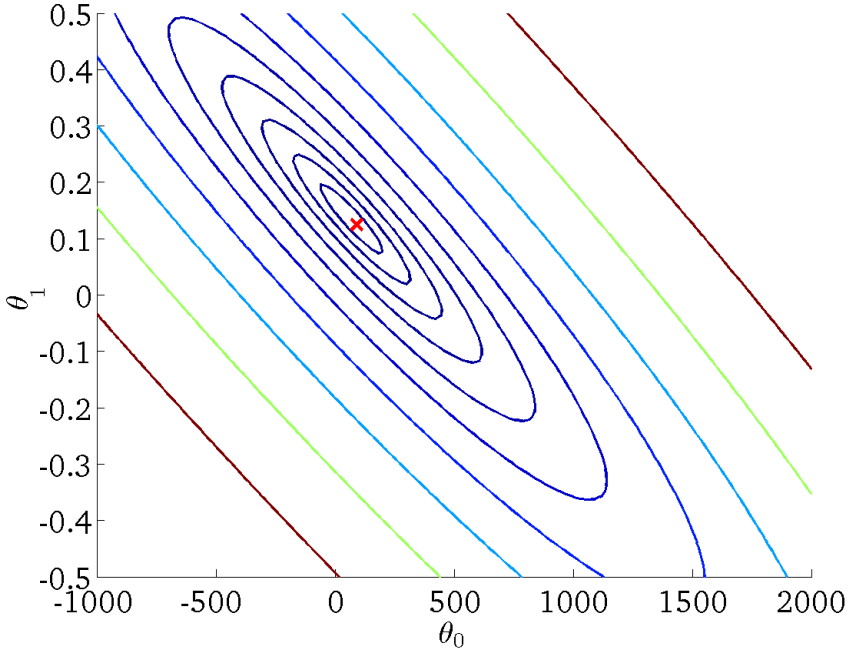
$h_\theta(x)$

(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)



Minimizing the Cost Function

- To find the best-fit line, we need to minimize the cost function. This is done by adjusting model parameters to reduce the error.

- **Our goal:** Find values of θ_0 and θ_1 that minimize the cost function.
- We need a systematic approach to find these optimal parameters.

- **Key Question**: How do we adjust θ_0 and θ_1 to reduce our prediction errors?
- **Gradient Descent:** An iterative algorithm that steps towards the minimum.

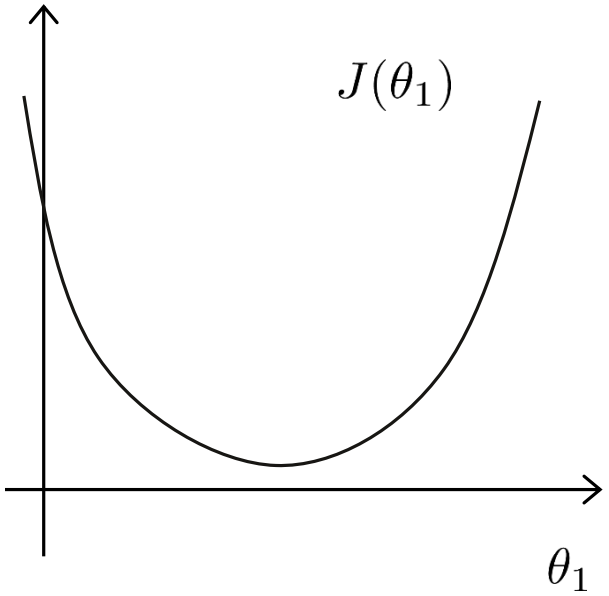
Next, we explore **Gradient Descent**, an optimization algorithm for this purpose.

Linear regression with one variable

Gradient Descent

Gradient Descent

- Gradient descent is an optimization algorithm used to minimize the cost function of a model by iteratively adjusting the parameters of the model.
- It is used to minimize the cost function value so that our straight line fits bests to the dataset.
- With the gradient descent we are taking steps to reach to it's minimum.

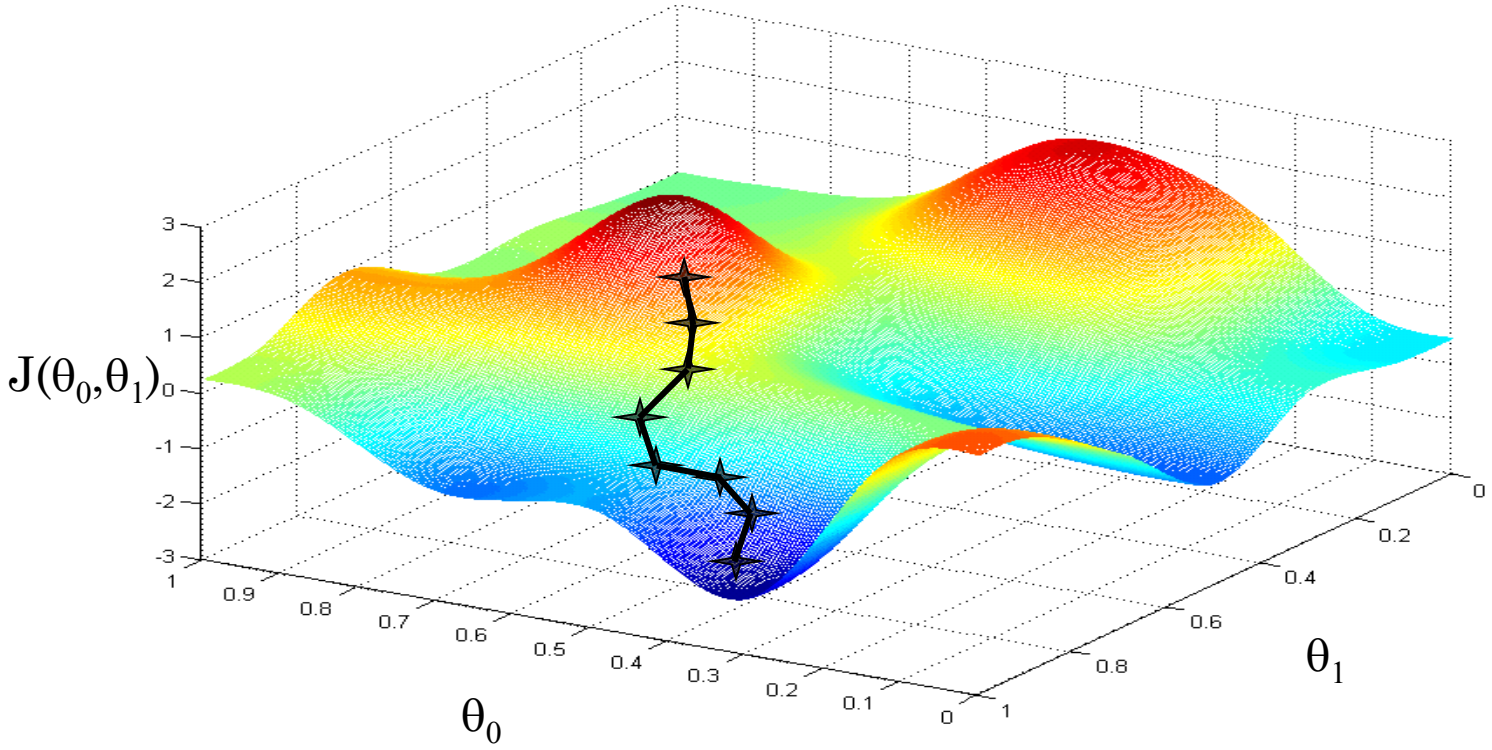


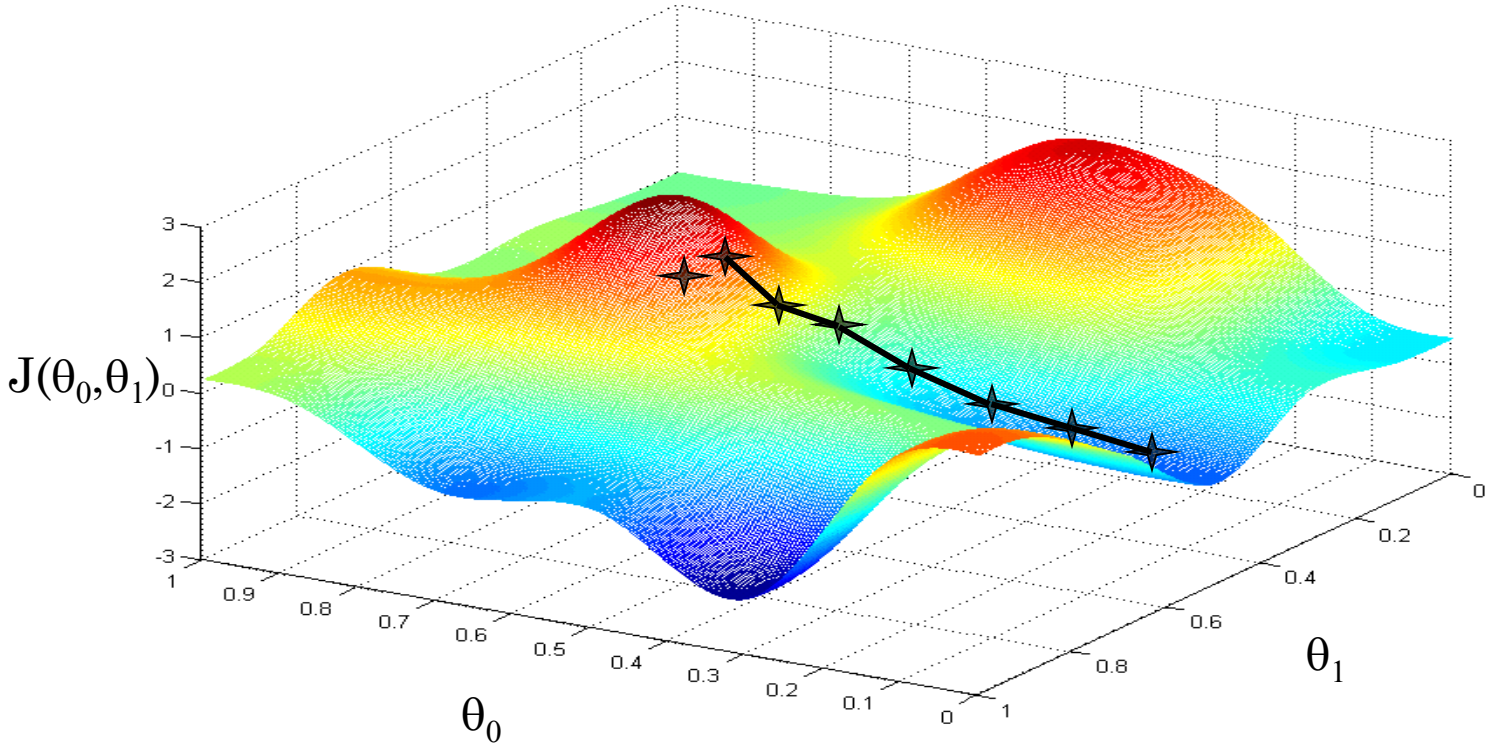
Have some function $J(\theta_0, \theta_1)$

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


$J(\theta_0,\theta_1)$
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


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

Gradient descent algorithm

repeat until convergence {

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

}

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:

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_1 := \text{temp1}$$

Key Components of Gradient Descent

- J(θ_0, θ_1) is the cost function that measures the difference between the predicted values and the actual values.
- α is the learning rate, which determines the size of step of each iteration.
- It is multiplied to scale the gradient. (α is a positive constant)
- The learning rate is a hyper parameter that needs to be chosen carefully. If it is too small, the algorithm may take a long time to converge, while if it is too large the algorithm may overshoot the minimum and fail to converge.
- $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ is the partial derivative of the cost function J with respect to parameter θ_j
- The derivative of a function represents the rate at which the function's output (dependent variable) changes with respect to it's input (independent variable).

Linear regression with one variable

Gradient Descent Intuition

Gradient descent algorithm

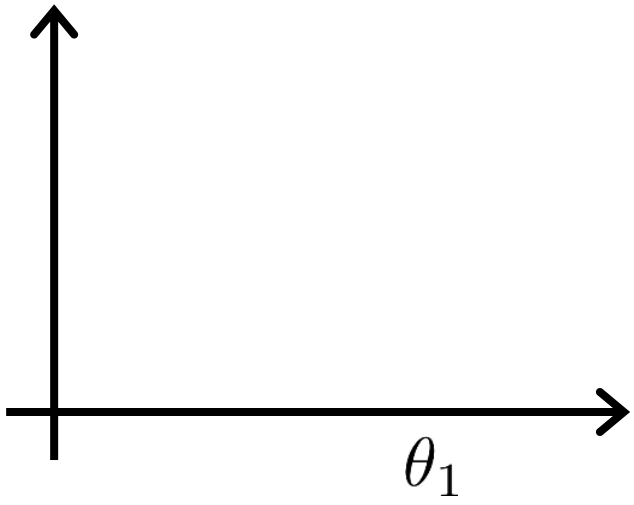
repeat until convergence {

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

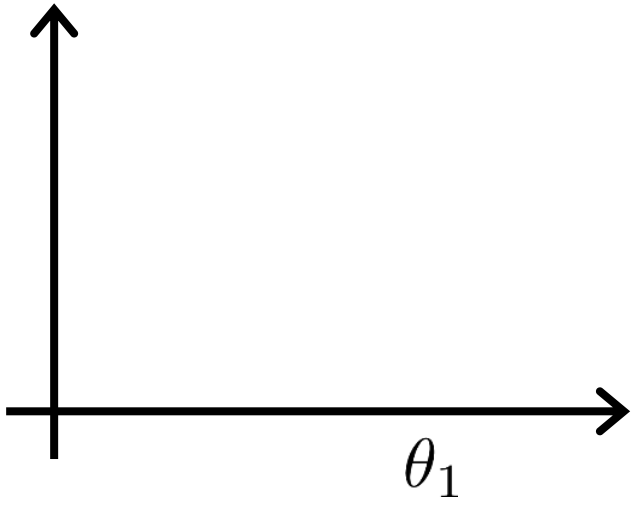
}

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

If α is too small, gradient descent can be slow.



If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.



Linear regression with one variable

Gradient descent for linear regression

Gradient descent algorithm

repeat until convergence {

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

(for $j = 1$ and $j = 0$)

}

Linear Regression Model

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

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

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) =$$

$$j = 0 : \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) =$$

$$j = 1 : \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) =$$

Derivative of cost function in Linear Regression

- Figure out partial derivative with respect to θ_0, θ_1

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

- In order for us to differentiate at a certain point we want to ignore the (∑) summation sign and the value of m. Because we are finding the derivative at a single point.

- For finding the derivative at a composite function we typically apply the chain rule.

Composite Function:

$$\mathrm{F} = f(g(x))$$

- **Chain rule:** The derivative of the outer function multiplied to the derivative of the inner function.

Derivative of cost function with respect to θ_0:

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

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

- $\theta_1 x^{(i)}$ *and* $y^{(i)}$) are constant items.

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

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

Derivative of cost function with respect to θ_1:

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

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

- θ_0 *and* $y^{(i)}$ are constant items.

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

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

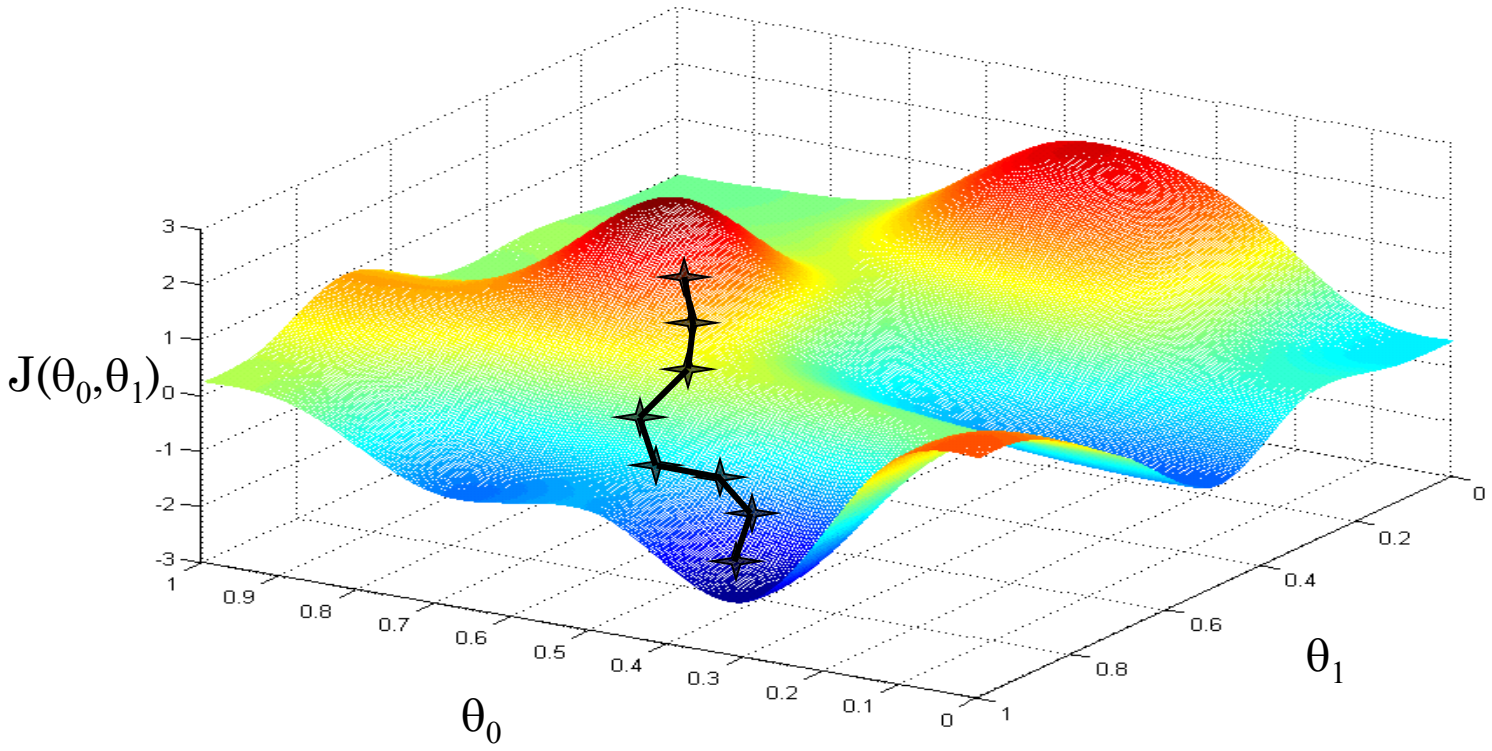
Gradient descent algorithm

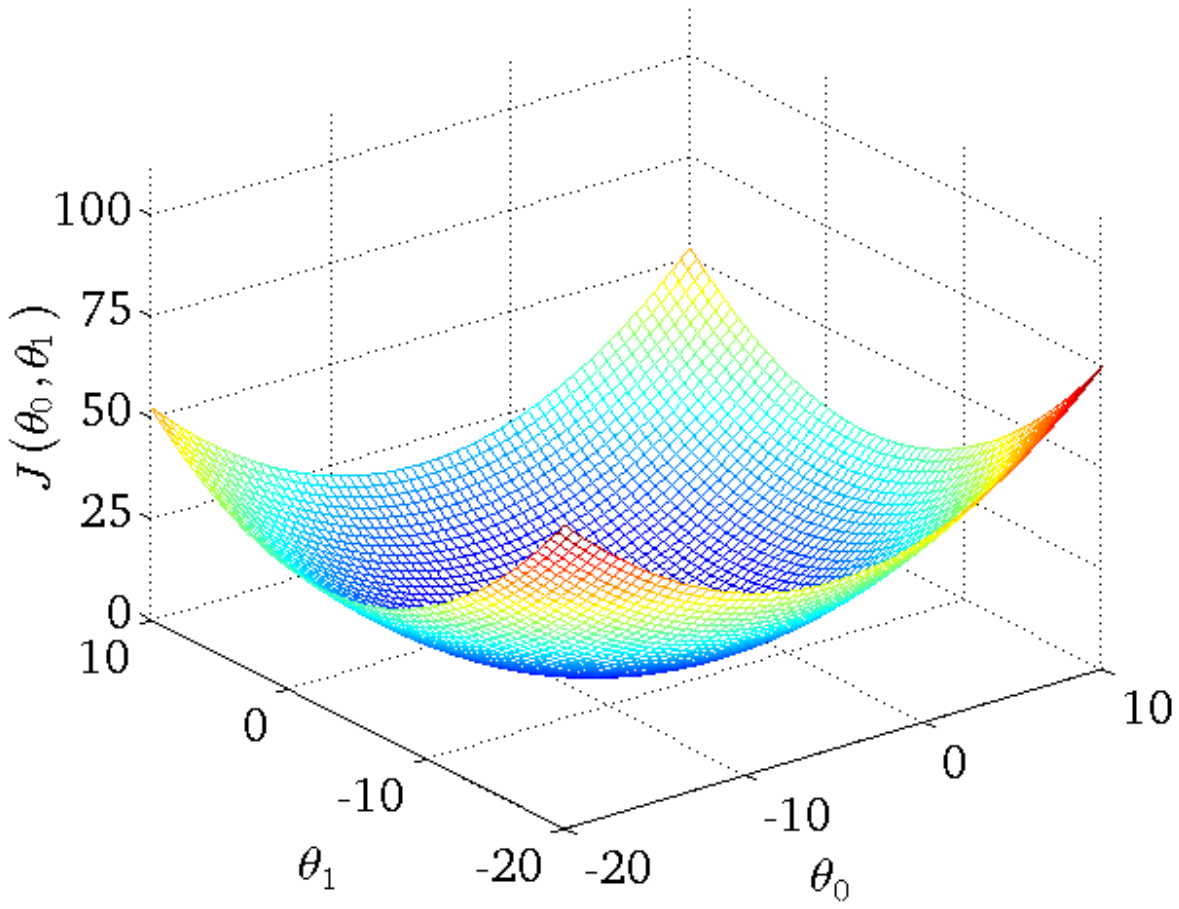
repeat until convergence {

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

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

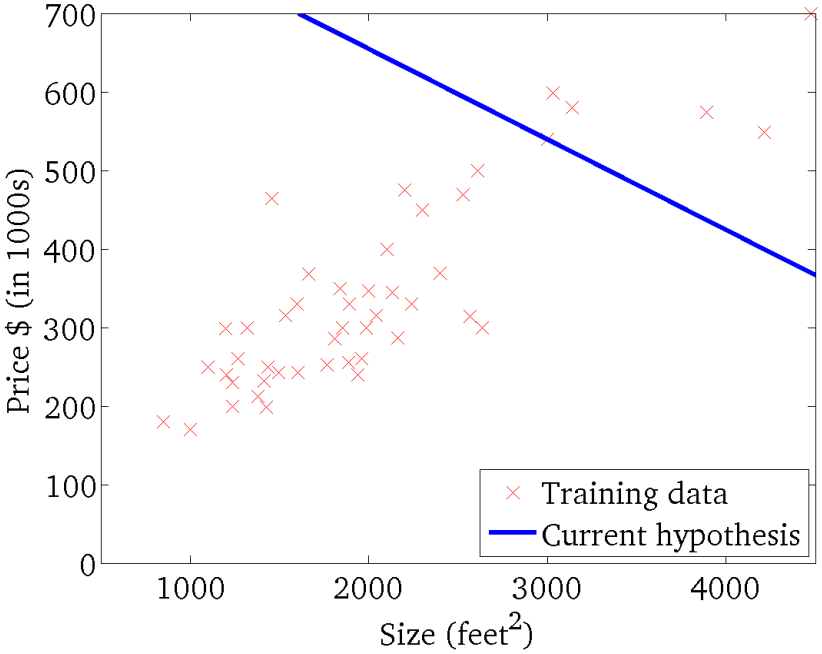
} update θ_0 and θ_1 simultaneously


$J(\theta_0,\theta_1)$
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


100
75
50
25
0
$J(\theta_0, \theta_1)$
10
0
-10
-20
θ_1
-20
-10
0
10
θ_0

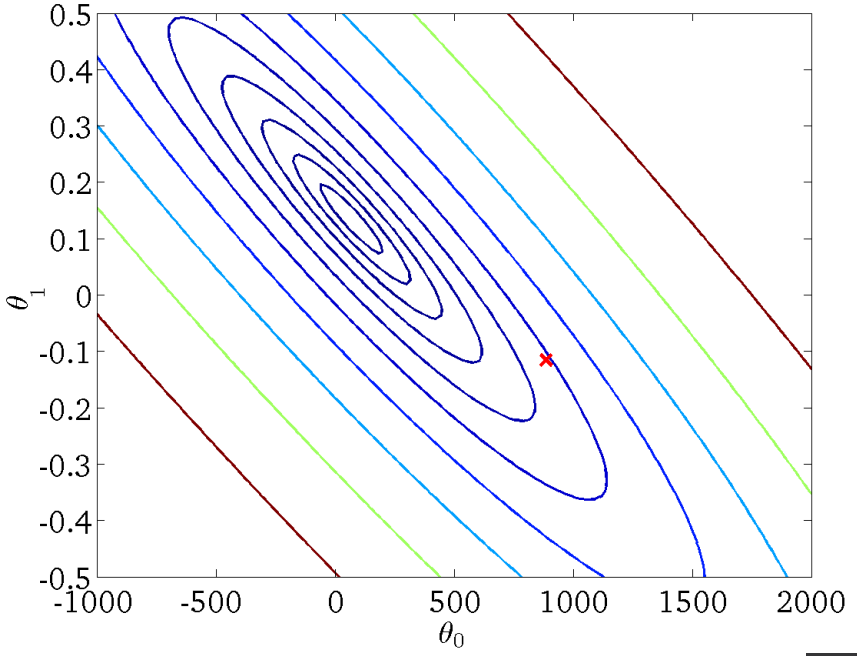
$h_\theta(x)$

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



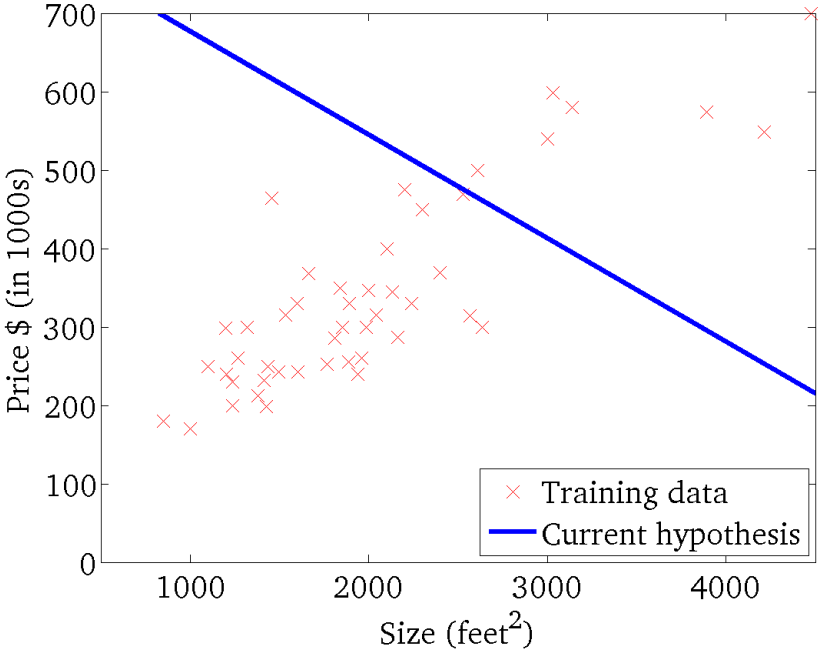
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



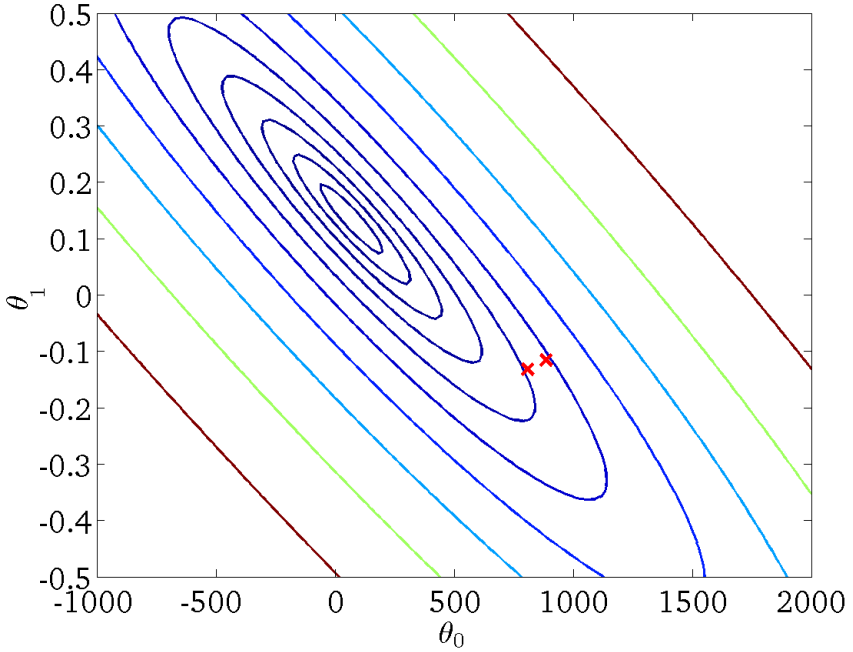
$h_\theta(x)$

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



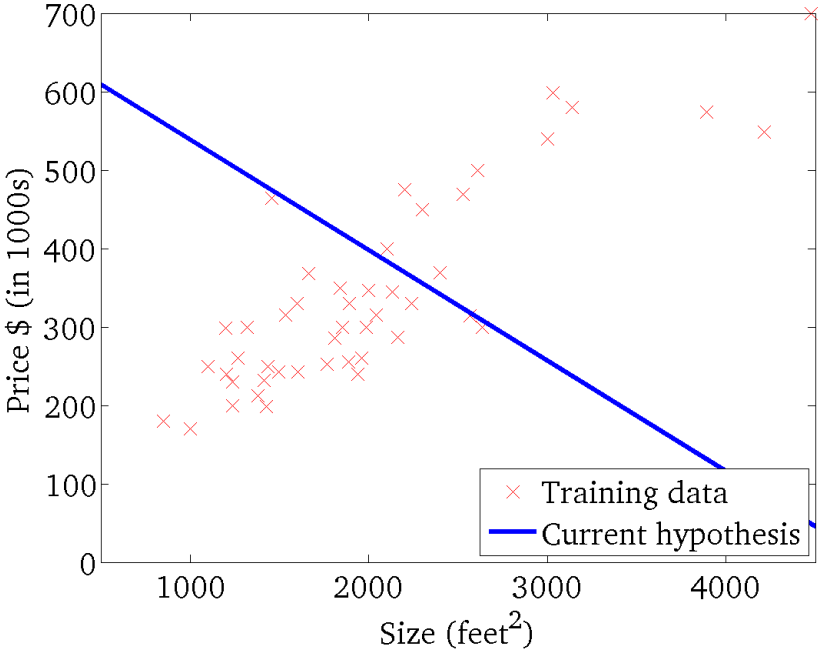
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



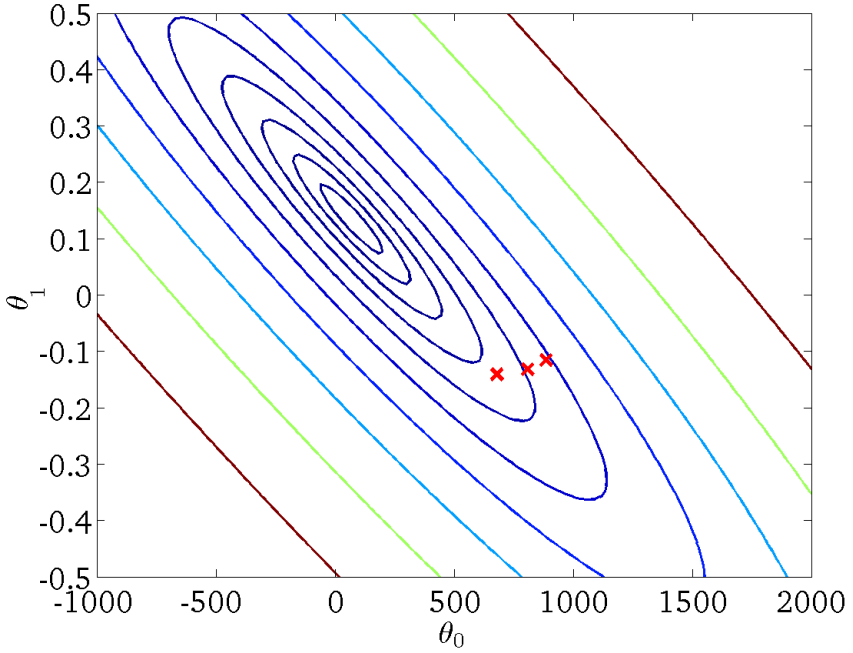
$h_\theta(x)$

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



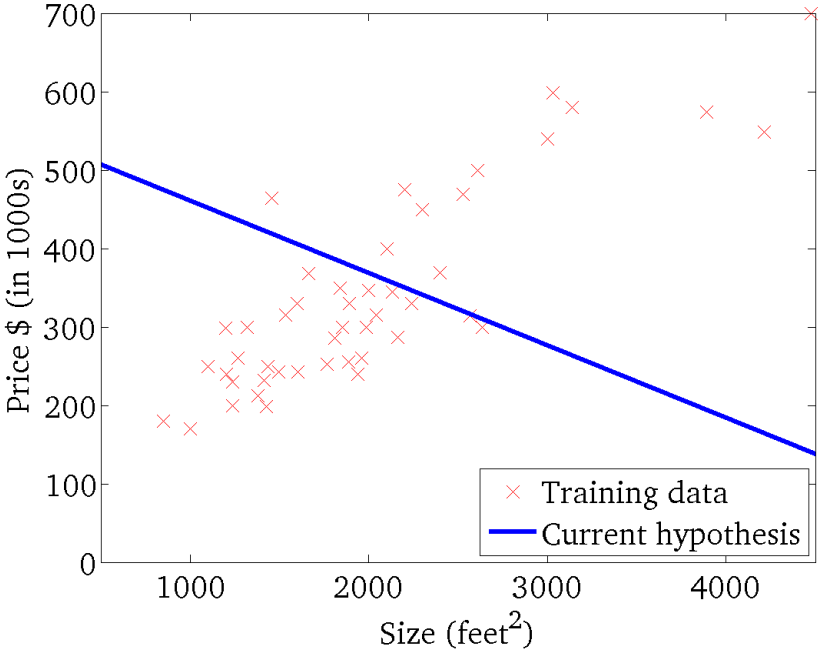
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



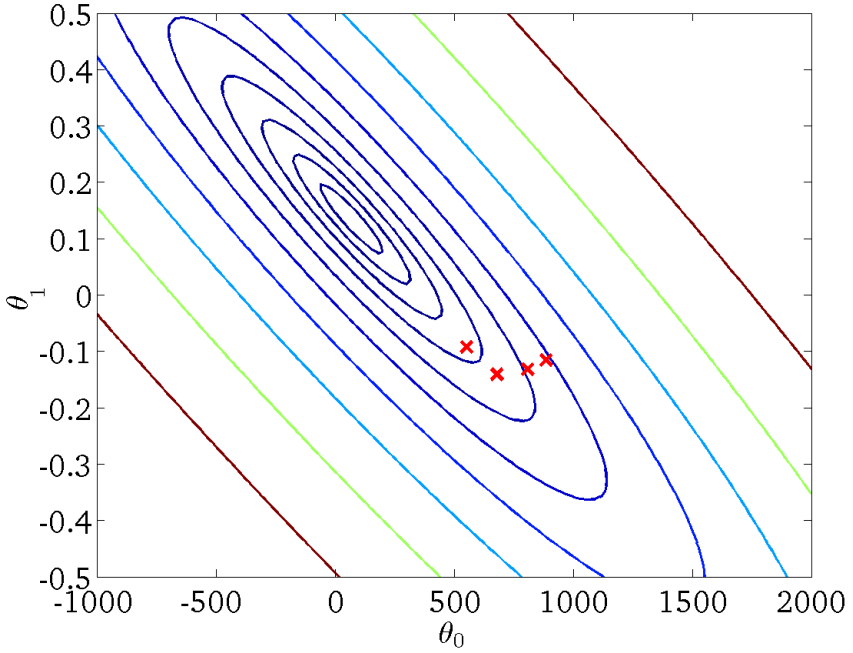
$h_\theta(x)$

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



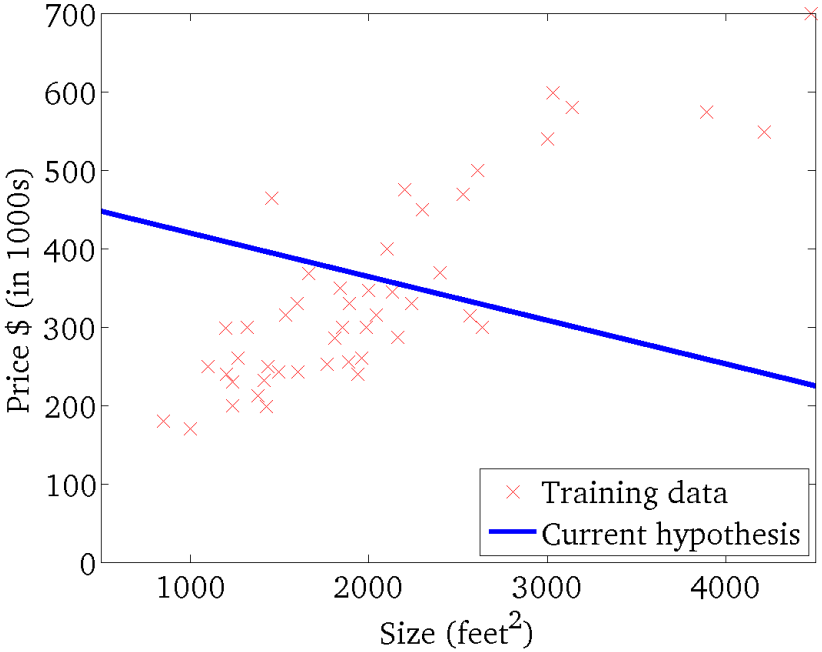
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



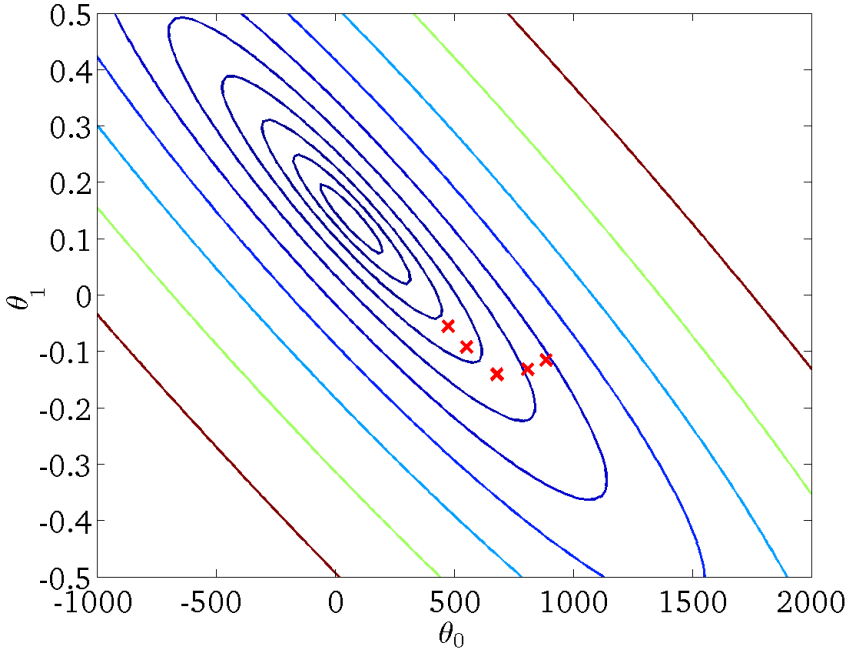
$h_\theta(x)$

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



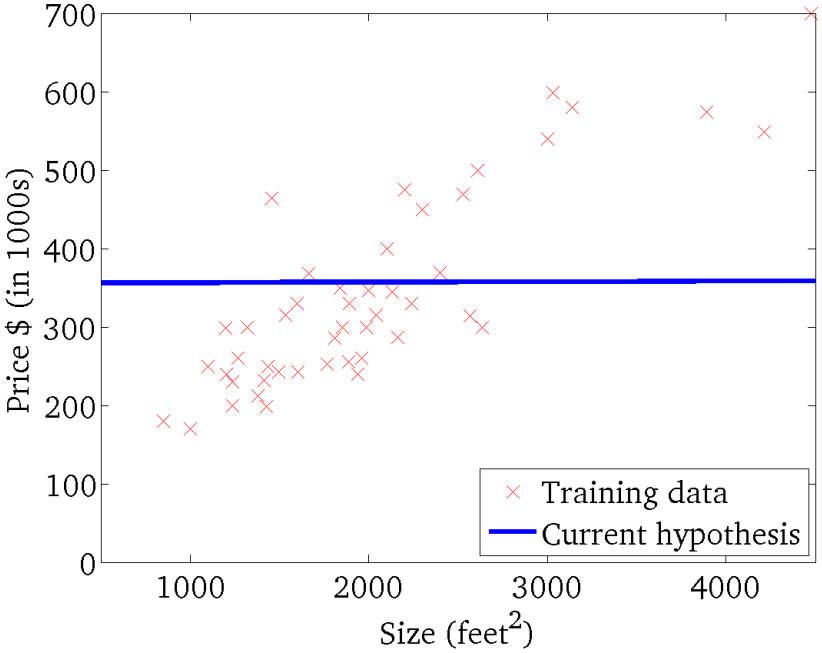
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



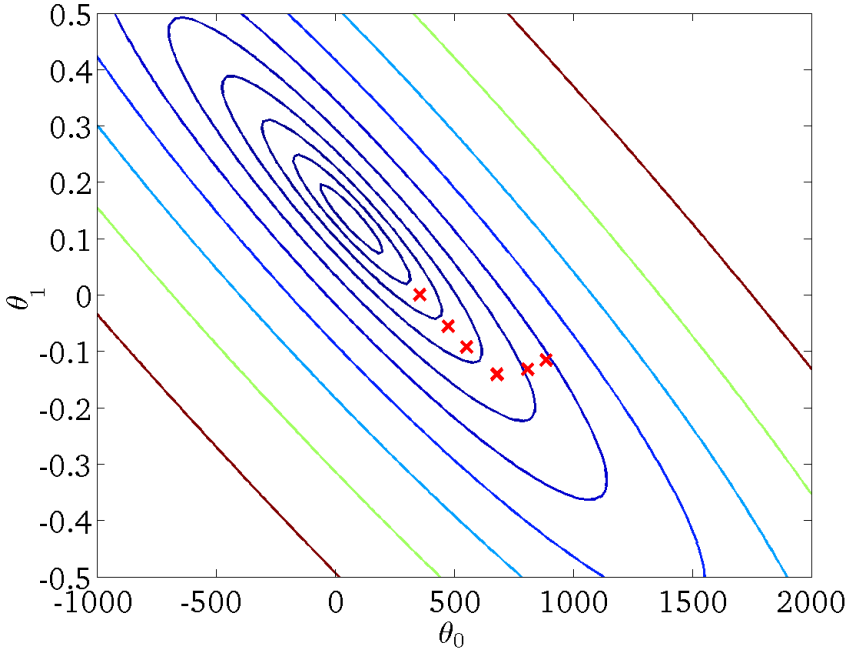
$h_\theta(x)$

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



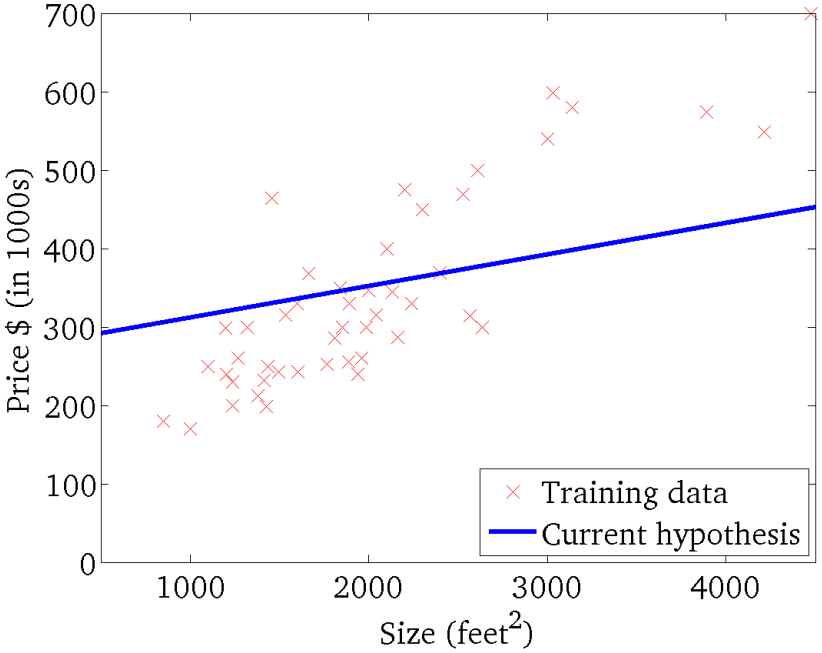
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



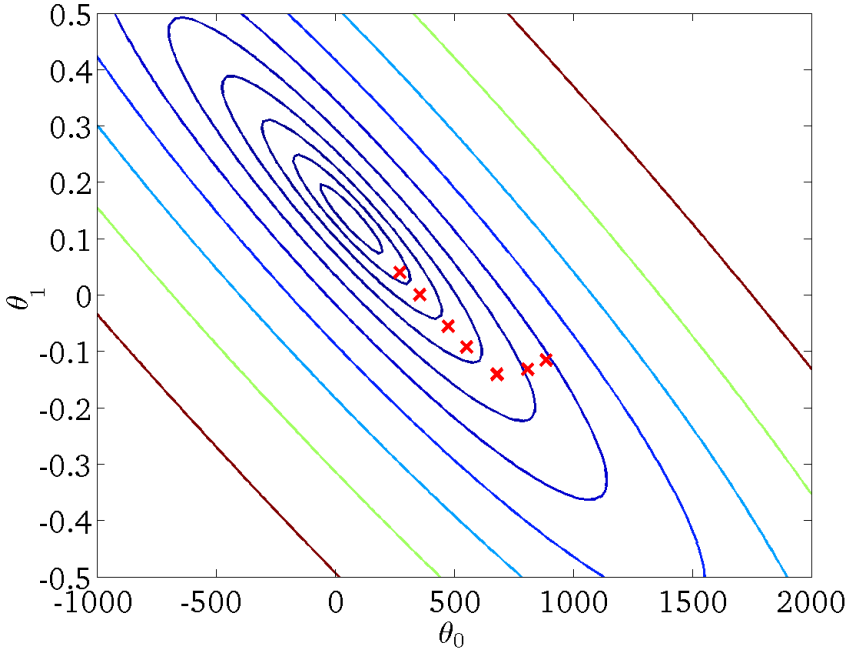
$h_\theta(x)$

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



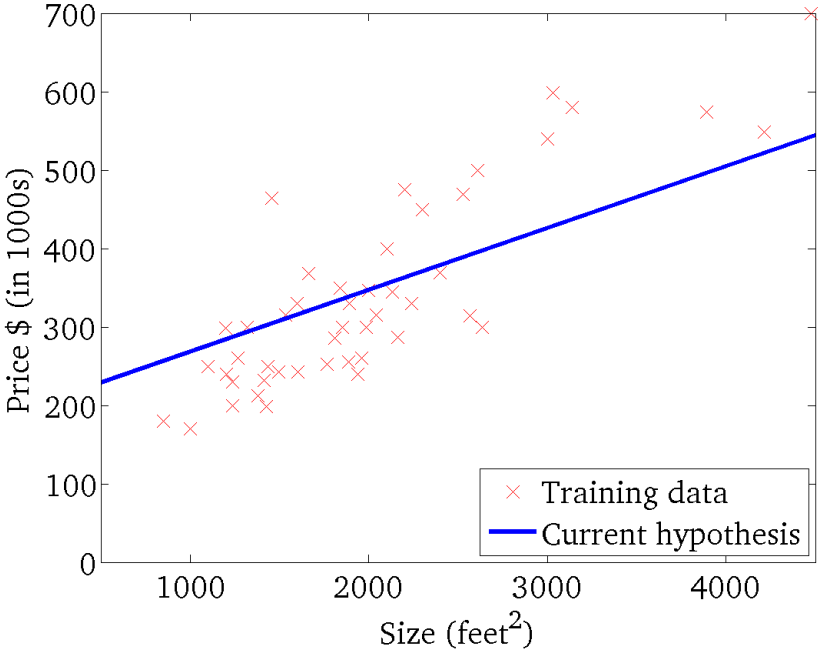
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



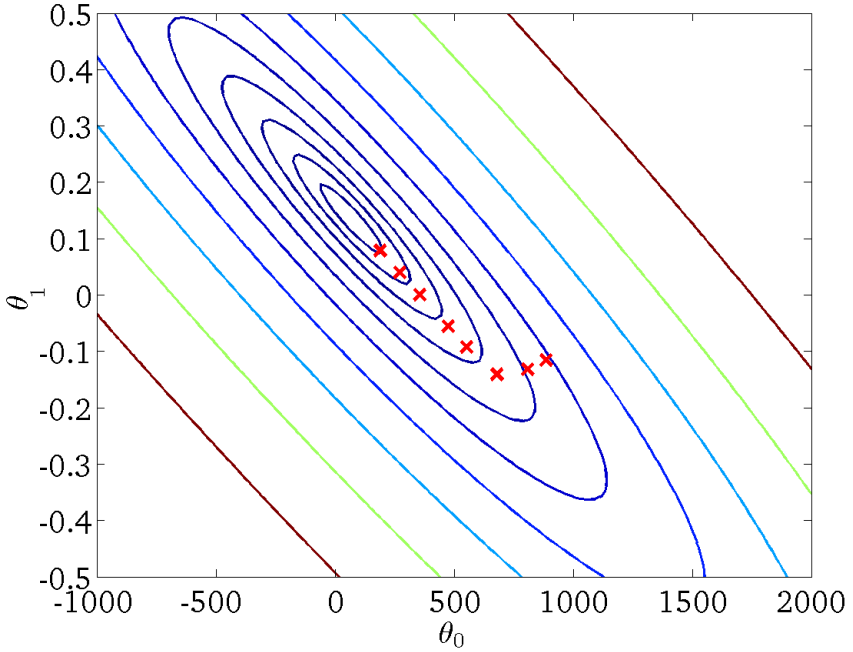
$h_\theta(x)$

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



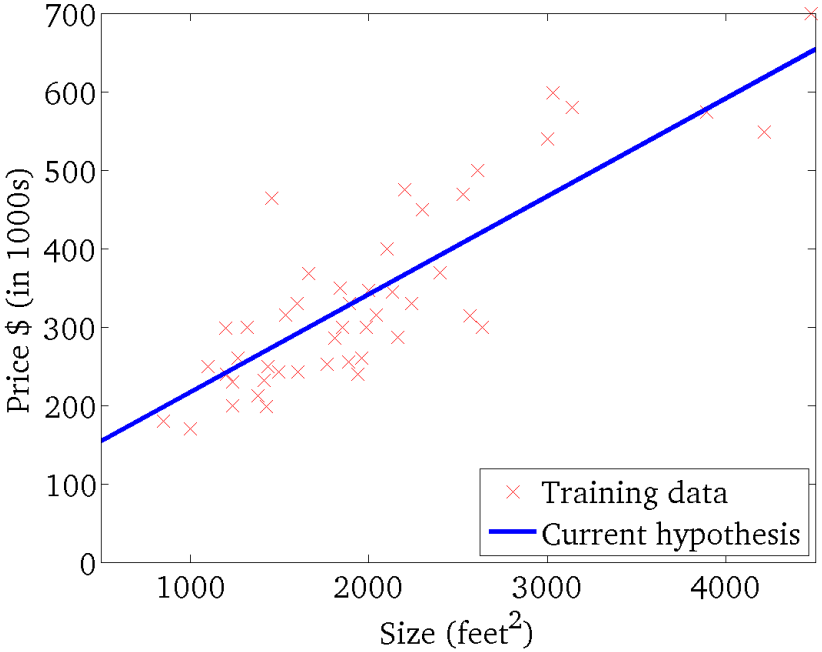
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



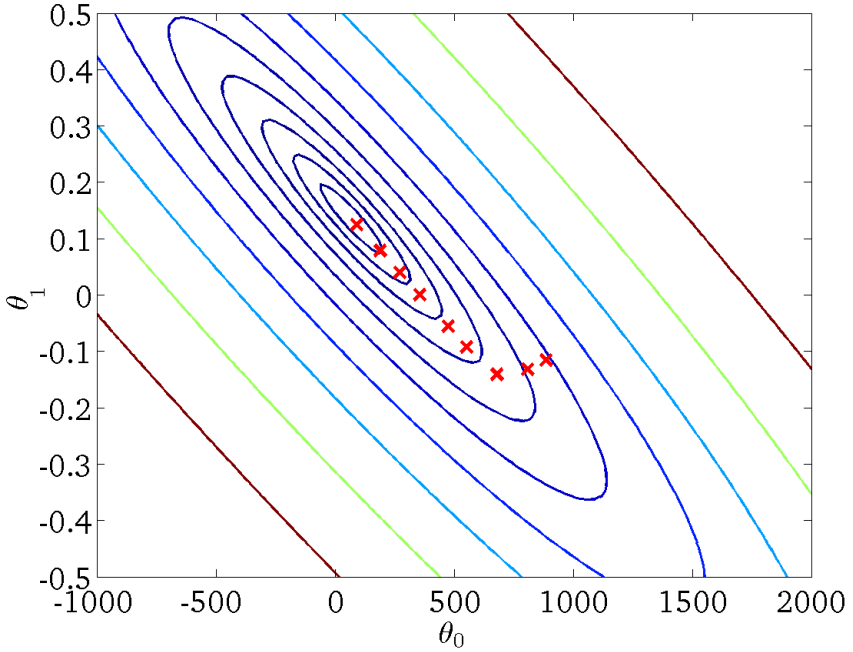
$h_\theta(x)$

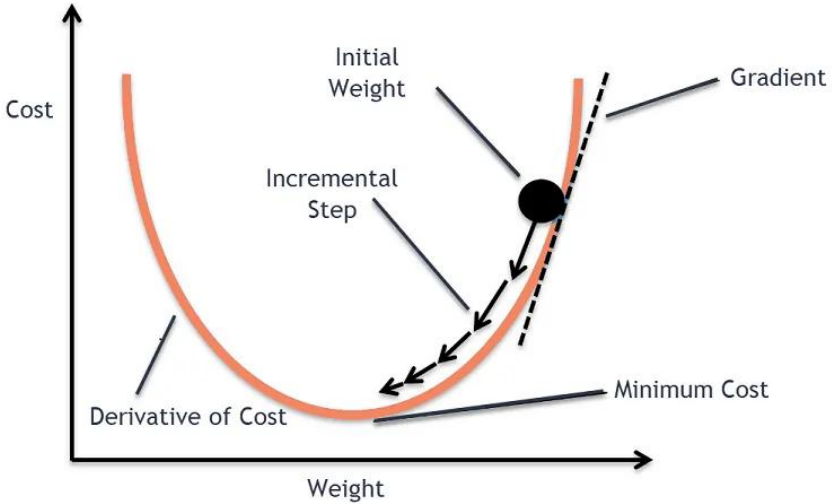
(for fixed θ_0, θ_1, this is a function of x)

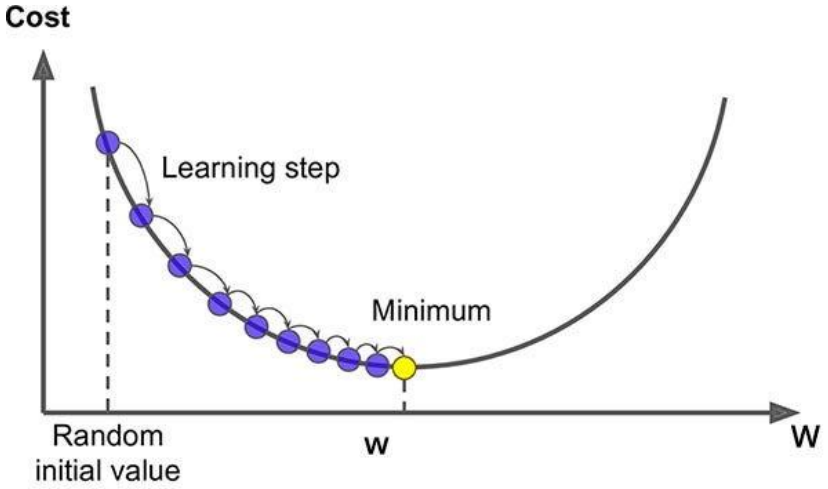


$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

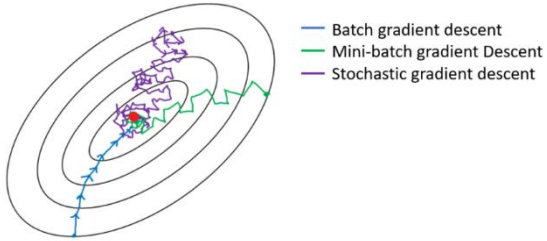



Cost
Initial
Weight
Gradient
Incremental
Step
Minimum Cost
Derivative of Cost
Weight

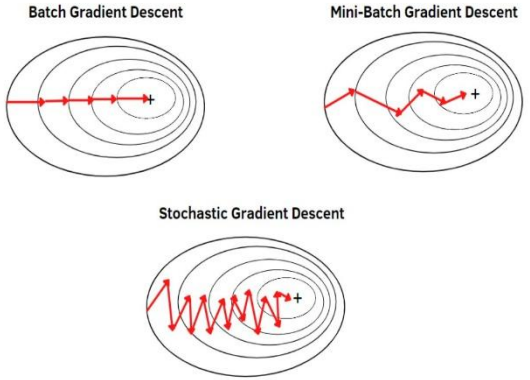

Cost
Learning step
Minimum
Random
initial value
w
W

Notable Gradient Descent Algorithms

- Different variations of Gradient Descent help optimize learning efficiency:



- **Batch Gradient Descent (BGD):**
  - Uses the entire dataset to compute gradients, ensuring stable updates but slower processing for large datasets.
  - "Batch": Each step of gradient descent uses all the training examples.
  - BGD is computationally expensive since it requires processing the entire dataset in each iteration.



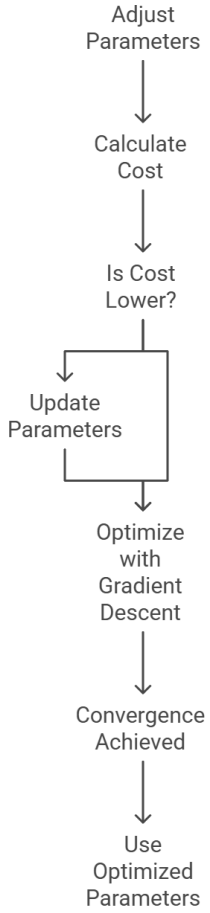
- **Stochastic Gradient Descent (SGD):**
  - Updates parameters using one data point at a time, making it faster but more noisy in convergence.
- **Mini-Batch Gradient Descent:**
  - A balance between BGD and SGD, updating parameters using small batches of data for efficient and stable learning.
- These variations impact **speed, stability, and accuracy** in training models.

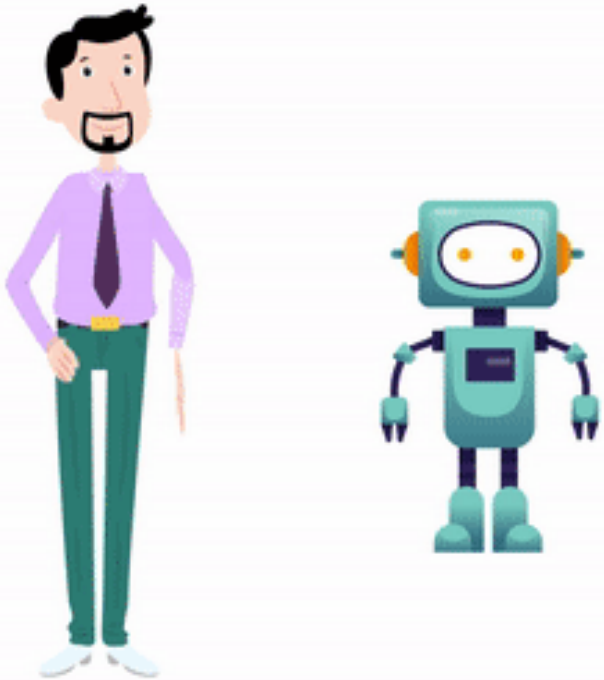
How do Models Learn?

- Machine Learning models learn by adjusting their parameters to minimize the cost function, which measures the difference between predicted and actual values.
- **Cost Function (Mean Squared Error):**

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

- A lower cost indicated better predictions.
- To minimize cost function, machine learning algorithms use optimization techniques like gradient descent.
- **Optimization with Gradient Descent:**
  - Iteratively updates parameters in the direction that reduces cost.
  - Continues until convergence to (or near) the optimal values.
- Once trained, the model uses these optimized parameters to make predictions based on new data.





Thank You