Machine Learning

Contents…

2. **Classic Machine Learning**
   - **Linear regressions**
   - Logistic regressions

CHAPTER 2:
Classic Machine Learning: regression

What is ML?

- Arthur Samuel (1959). Machine learning:
  - a field of study that gives computers the ability to learn without being explicitly programmed.
- Tom Mitchell (1998): Well-proposed Learning Problem:
  - "A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E."
  - Suppose your email program watches which emails you do or do not mark as spam, and based on that learns how to better filter spam.
  - What is the task T in this setting?
    - ✓ Classifying emails as spam or not spam
  - What is the experience in this setting?
    - ✓ Watching your label email as spam or not spam
  - What is the performance of this setting?
    - ✓ The number of emails correctly classified as spam or not spam

Types of ML

Supervised

- ✓ have a target column
- ✓ Data points have a known outcome

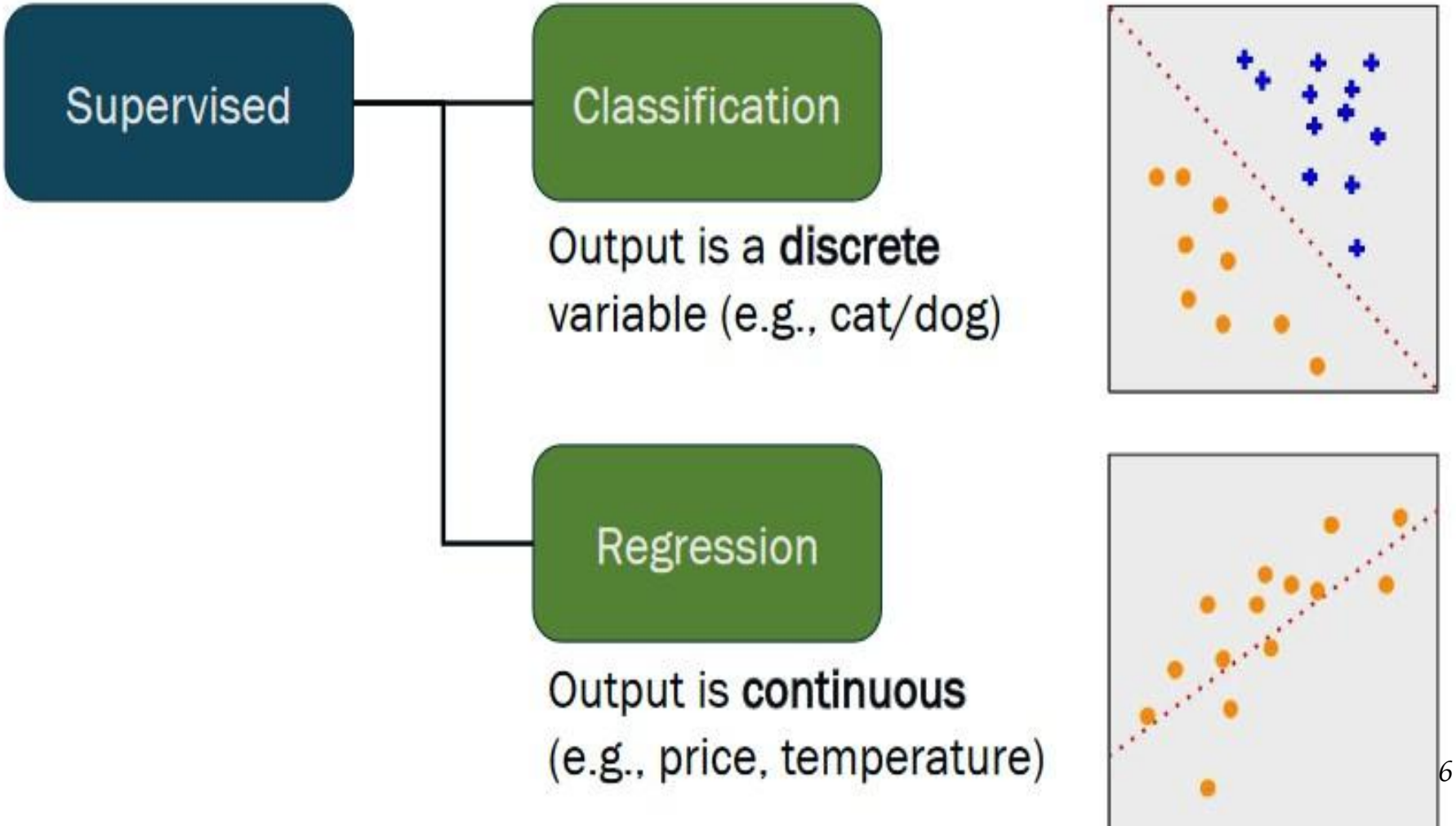
Unsupervised

- ✓ Doesn't have a target column
- ✓ Data points have unknown outcome

Semi-supervised

- ✓ An intermediate learning
- ✓ Model learns both from labeled and unlabeled data

Supervised Learning Types



Supervised Learning:
classification vs regression

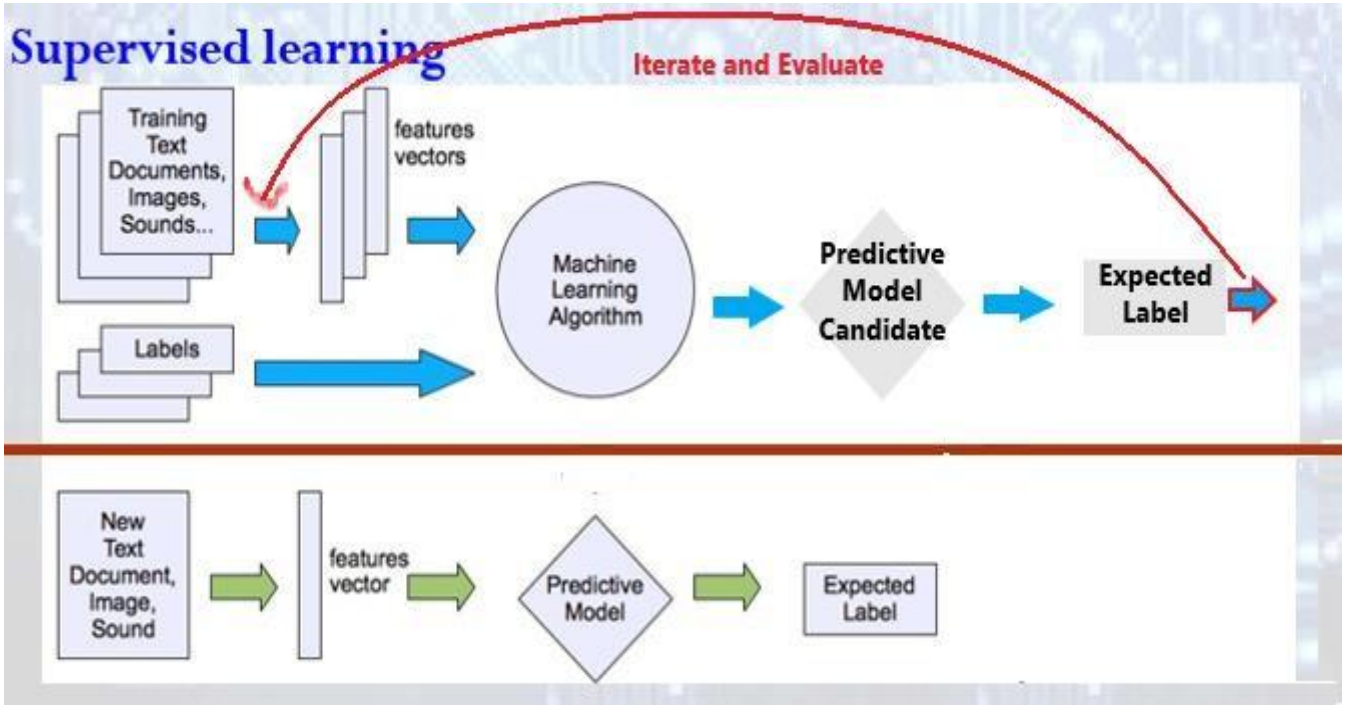
- Suppose you are working on weather prediction, and your weather station makes one of three predictions for each day's weather: Sunny, Cloudy or Rainy. You'd like to use a learning algorithm to predict tomorrow's weather.
  - Would you treat this as a classification or a regression problem?
    - multiclass classification
- Given data about the size of houses and number of bed rooms on the real estate market, try to predict their price.
  - Would you treat price as a function of size and #bedrooms as a classification or a regression problem?
    - regression

Learning – a two step process

- Model : A learning algorithm
  - A model is a small thing that captures a larger thing
  - A good model is going to omit unimportant details while retaining what's important
  - But we need to do so in a way that preserves the features or relationships that were interested in.
- Model Construction
  - A training set is used to create the model.
  - The model is represented as classification rules, decision trees, or mathematical formula
- Model Usage
  - the test set is used to see how well it works for classifying future or unknown objects

How ML works:

classic ML process



- The general learning approach: First, the dataset needs to be transformed into a representation, most often a list of vectors, which can be used by the learning algorithm. The learning algorithm choses a model and efficiently searches for the model's parameters.
- After we have selected a model that has been fitted on the training dataset, we can use the test dataset to estimate how well it performs on this unseen data to estimate the so-called generalization error

How ML works:

classic ML process

- From a technical perspective, every classic machine learning problem is composed of several key choices to be made in a standard pipeline of five steps.

- **Step 1: Feature Extraction**
  - Derive compact and uncorrelated features to represent raw data.
- **Step 2: Choose a proper model (hypothesis function)**
  - Based on the nature of the given problem, choose a good machine learning model from the candidates listed below
    - ✓ Linear models
    - ✓ Logistic sigmoid, softmax
    - ✓ Nonlinear kernels
    - ✓ Decision trees
  - constrain the function family to be learned from
- **Step 3: Choose a learning criterion (cost/objective function)**
  - Choose an appropriate learning criterion from the candidates Listed below, which forms an objective function of model parameters.
    - ✓ Mean squared error
    - ✓ Minimum classification error
    - ✓ Minimum cross-entropy
  - Choose certain criteria to measure how well the selected models fits over the training data as a function of unknown model parameters

How ML works:

classic ML process

- From a technical perspective, every classic machine learning problem is composed of several key choices to be made in a standard pipeline of five steps.

- **Step 4: Choose an optimization algorithm**
  - Considering the characteristics of the derived objective function, use an appropriate optimization algorithm from Lists below to learn the model parameters.
    - ✓ Grid search, Gradient descent, SGD , ADAM, RMSprop
  - Once the objective functions are determined, machine learning is turned into a standard optimization problem, where the objective function needs to be maximized or minimized with respect to the unknown model parameters.
  - Estimate the parameters in the hypothesis function

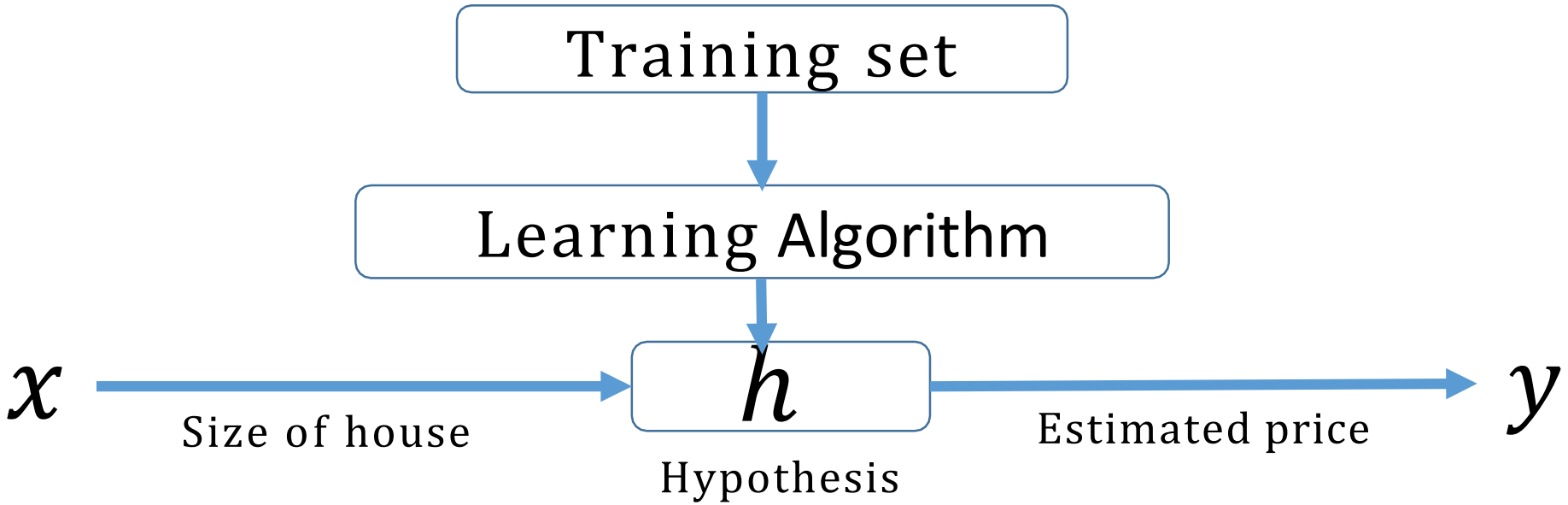
- **Step 5: Perform empirical evaluation**
  - Use held-out data to empirically evaluate the performance of learned models
  - In practice, the performance of the learned models can always be empirically evaluated based on a held-out data set that is not used anywhere in the earlier steps

Learning: model representations

- To describe the supervised learning problem slightly more formally, our goal is, given a training set, to learn a hypothesis function

$$h : X \rightarrow Y$$

so that h(x) is a "good" predictor for the corresponding value of y



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

Shorthand $h(x)$

How Does ML Work: ML framework

$$y_p = f(\Theta, \mathbf{x})$$

X: the input

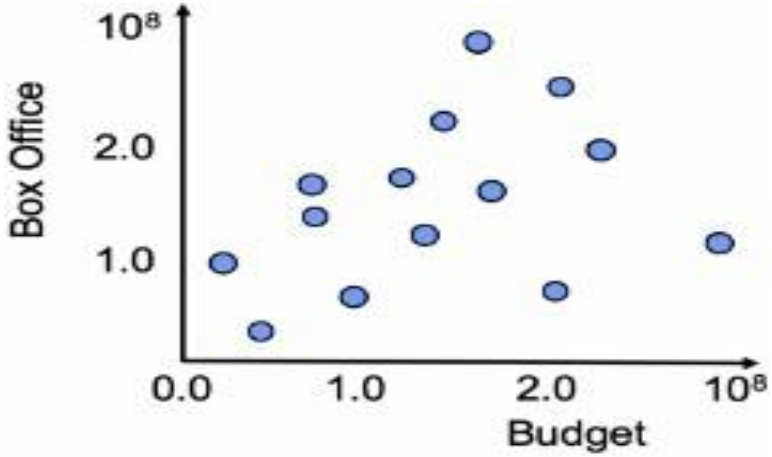
Y_p : output (values predicted by the model)

Θ: represents parameters of the model (1 or more variables)

- **Training:** given a training set of labeled examples $\{(x_1,y_1), \ldots, (x_n,y_n)\}$, estimate the prediction function f by minimizing the prediction error on the training set
- **Testing:** apply f to a never before seen test example x and output the predicted value y = f(x)

Linear regression

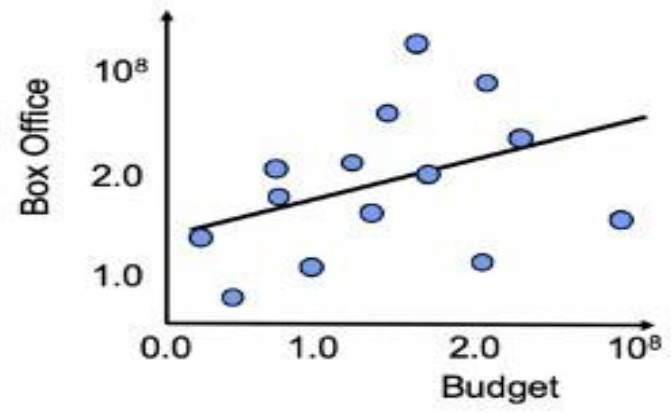
- **(Univariate) Linear regression**
  - involves finding the "best" line to fit two attributes (or variables) so that one attribute can be used to predict the other.



- **predict the Box Office revenue of a movie using the marketing budget only**

Linear regression

- **(Univariate) Linear regression**
  - involves finding the "best" line to fit two attributes (or variables) so that one attribute can be used to predict the other.



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

box office revenue

coefficient 0

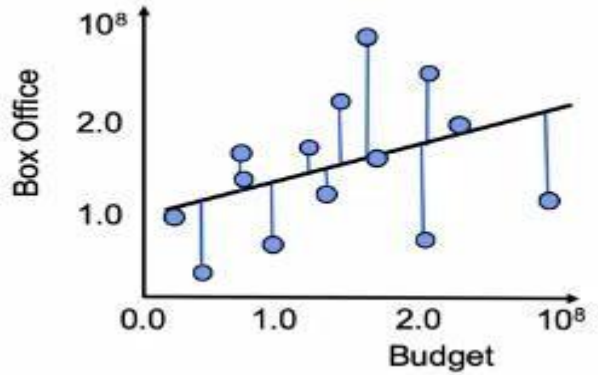
coefficient 1

movie budget

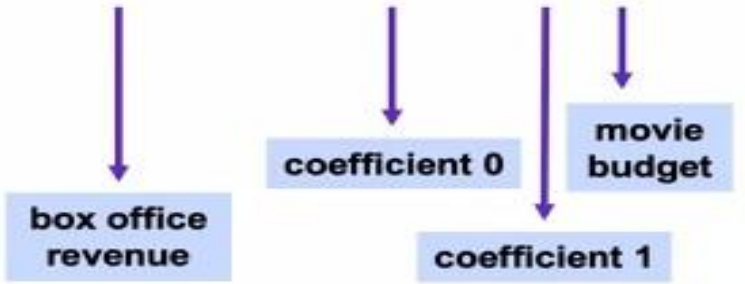
- **predict the Box Office revenue of a movie using the marketing budget only**

Linear regression

- **(Univariate) Linear regression**
  - involves finding the "best" line to fit two attributes (or variables) so that one attribute can be used to predict the other.
- **Multivariate linear regression**
  - an extension of linear regression, where more than two attributes are involved and the data are fit to a multidimensional surface.

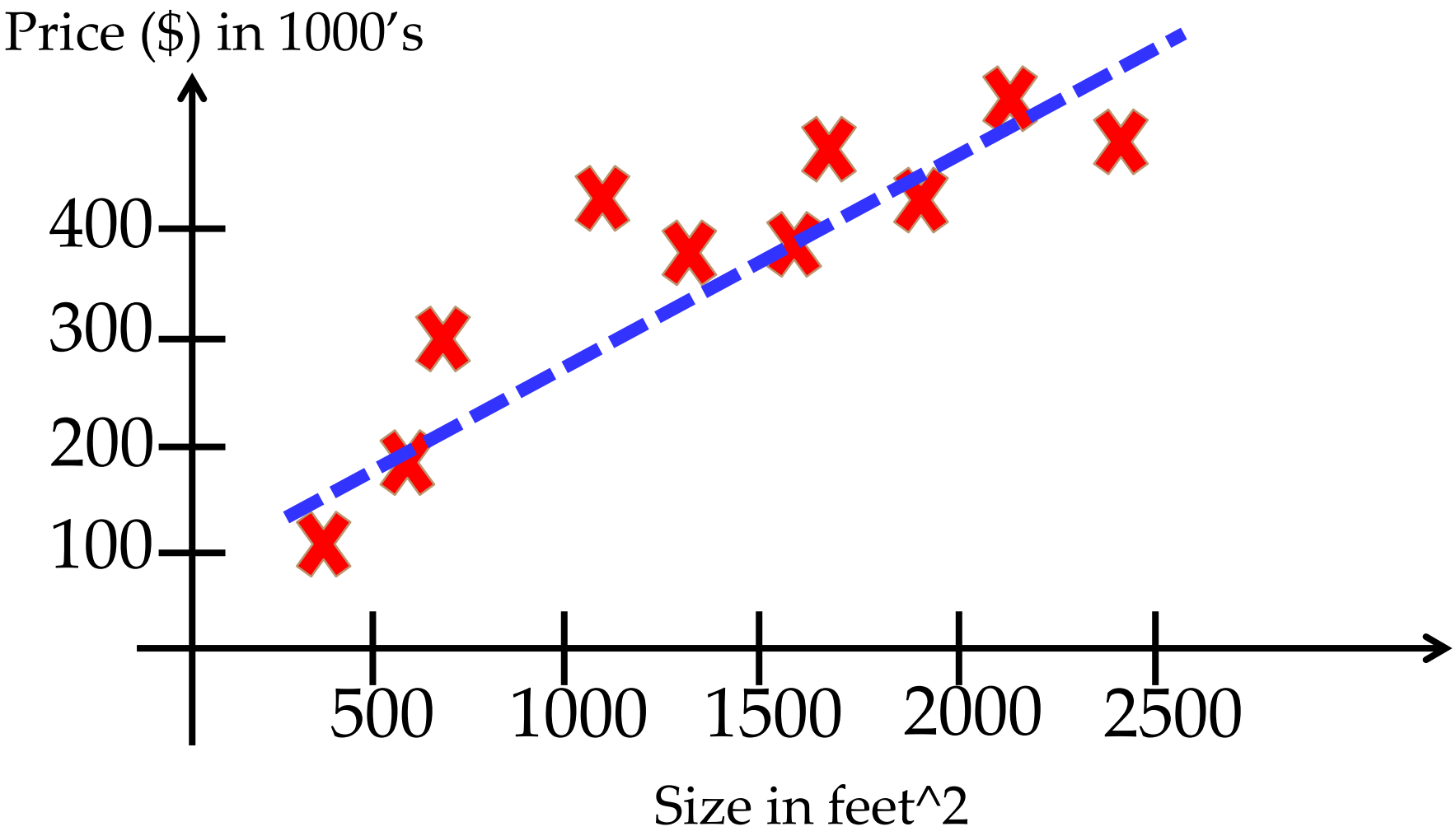


$$h_\theta(x) = \theta_0 + \theta_1 x$$



- **with this line we can predict what the box office revenue will be for a new movie**

Linear regression: house pricing predictions



Linear regression: notations

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

$m = 47$

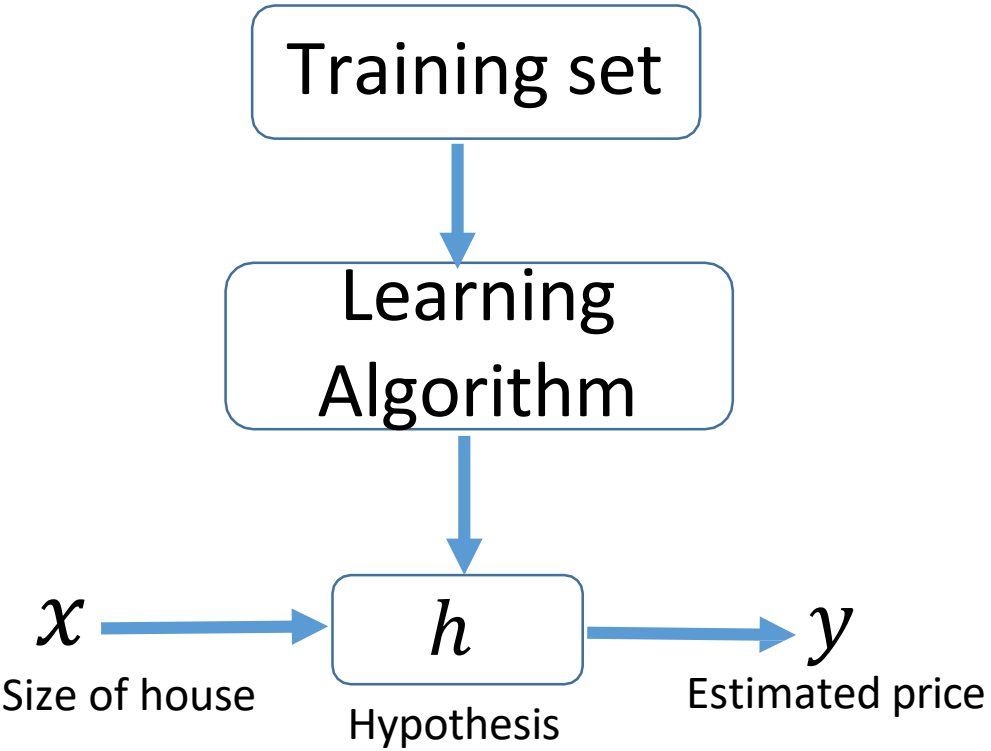
- Notation:
  - m = Number of training examples
  - x = Input variable / features
  - y = Output variable / target variable
  - (x, y) = One training example
  - $(x^{(i)}, y^{(i)})$ = i^{th} training example

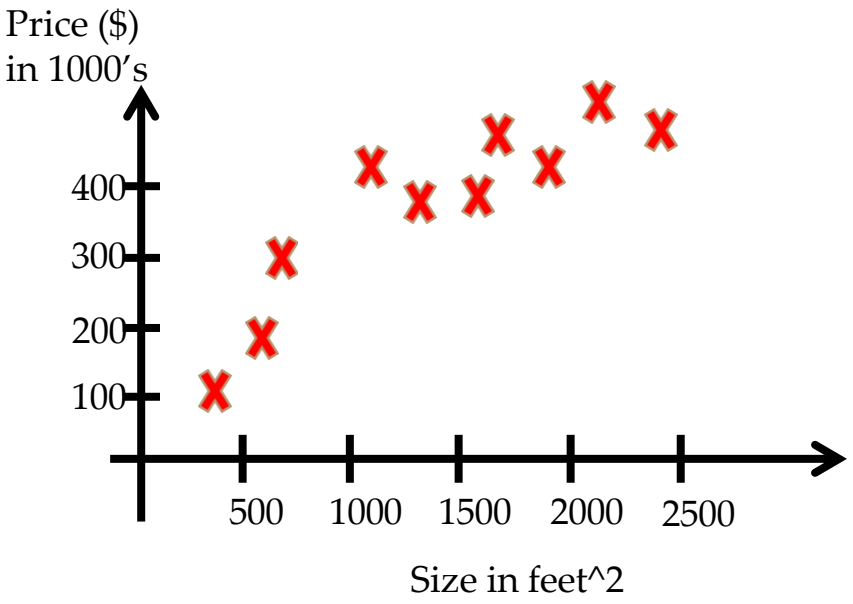
- Examples:
  - $x^{(1)} = 2104$
  - $x^{(2)} = 1416$
  - $y^{(1)} = 460$

Linear regression

- Model representation
- Cost function
- Gradient descent
- Gradient descent for linear regression

Linear regression: model representations





Univariate linear regression

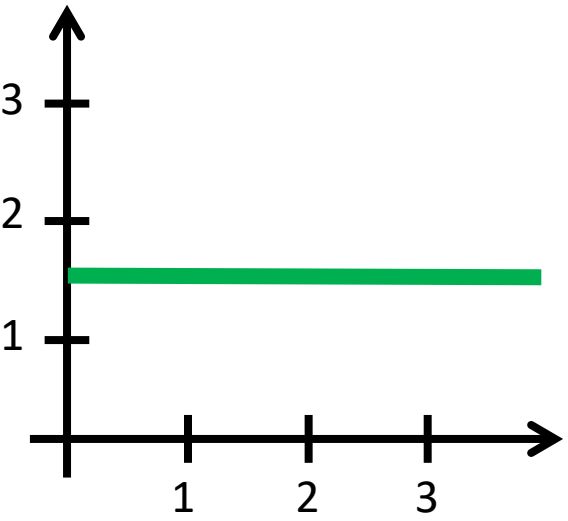
$$y = h_\theta(x) = \theta_0 + \theta_1 x$$

Shorthand $h(x)$

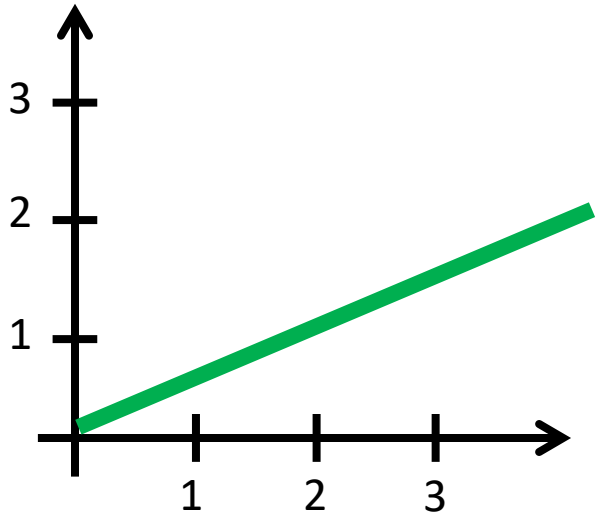
Parameters of the mode:
θ_0 *and* θ_1

Linear regression: model representations

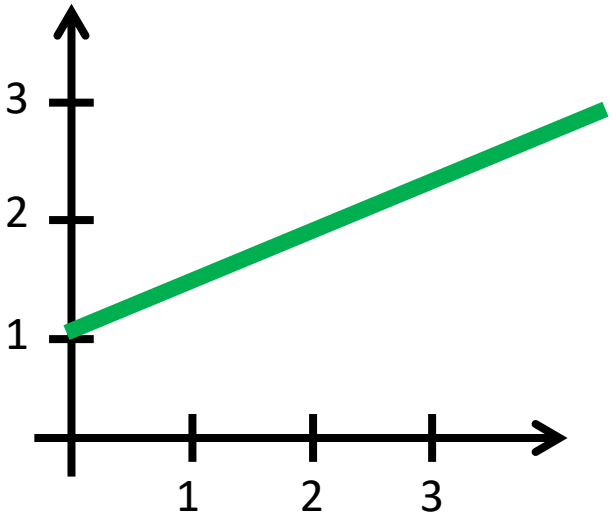
- $h_\theta(x) = \theta_0 + \theta_1 x$
- With different choices of the parameter's θ_0 and θ_1, we get different hypothesis, different hypothesis functions



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

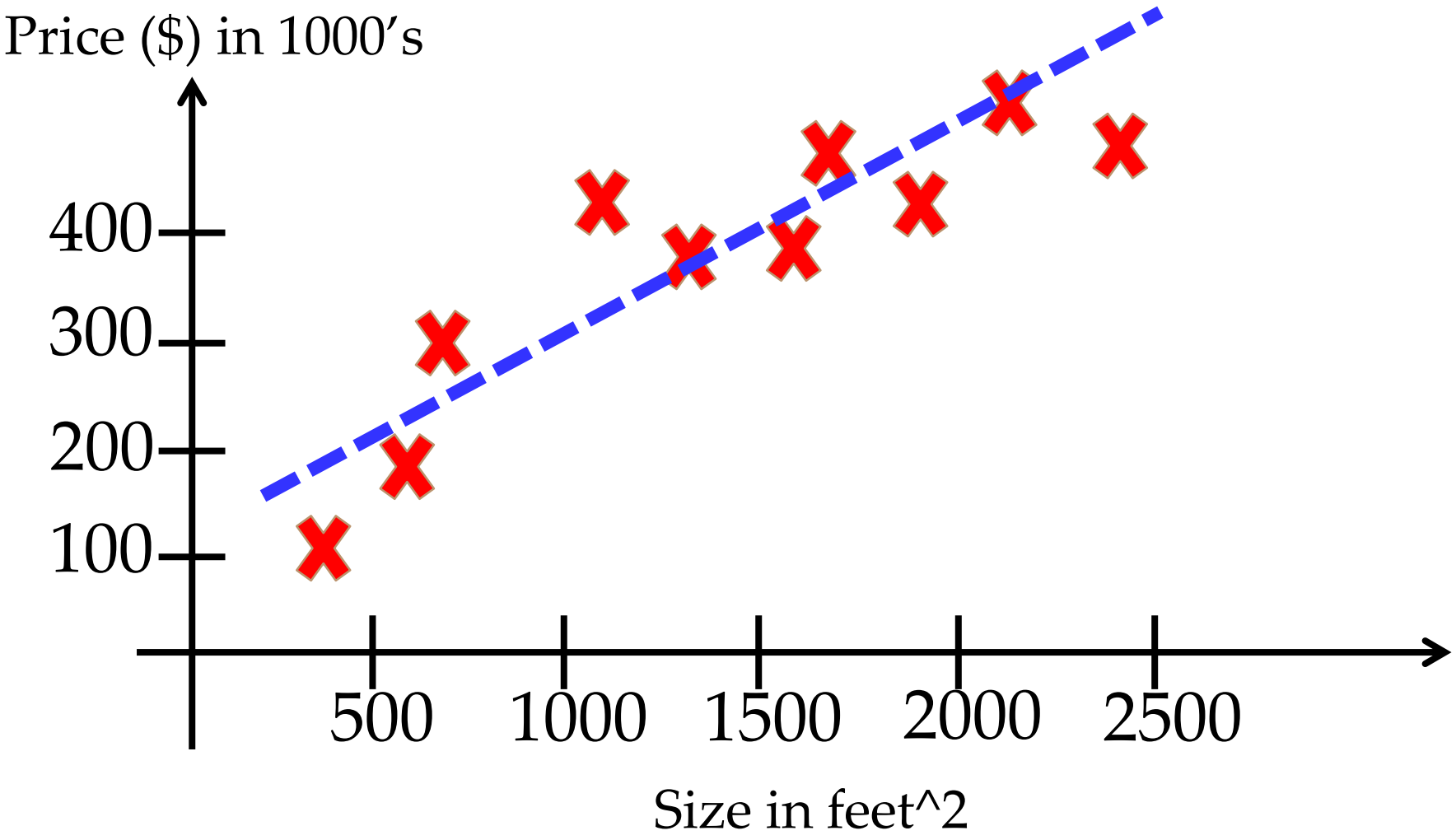


$\theta_0 = 0$
$\theta_1 = 0.5$

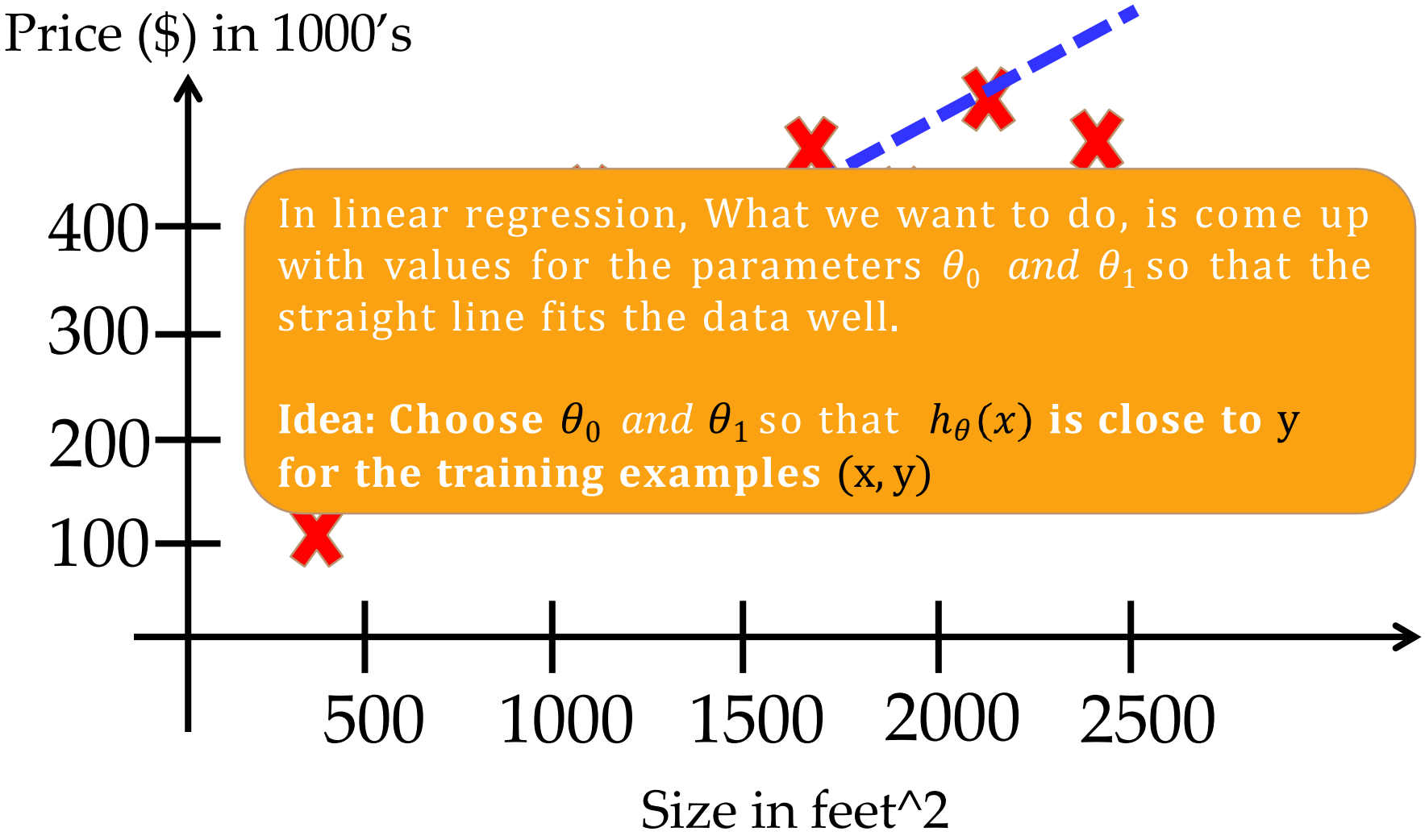


$\theta_0 = 1$
$\theta_1 = 0.5$

Linear regression: model representations



Linear regression: model representations



Linear regression

- Model representation
- Objective function
- Optimization algorithm (parameter learning)
- Gradient descent for linear regression

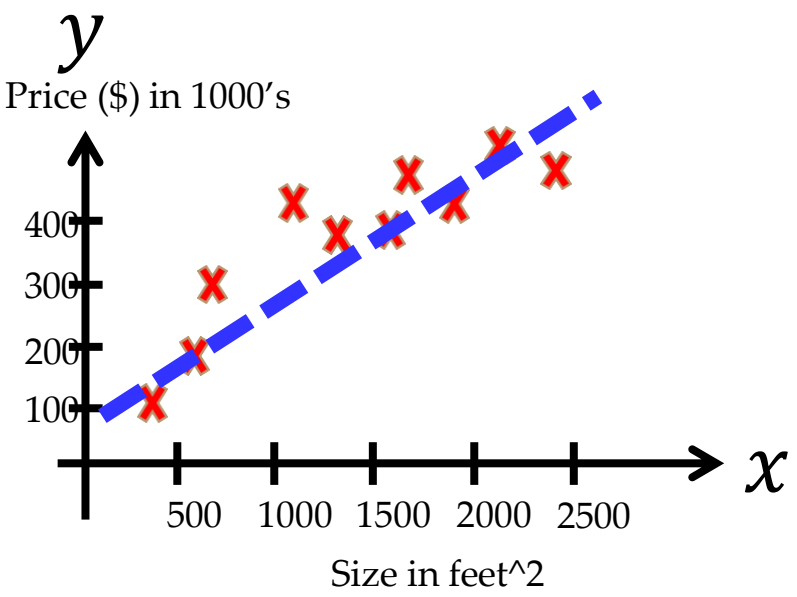
Linear regression: cost function

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

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

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

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



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

Squared error Cost function

Linear regression: cost function example

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

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

$$\theta_1$$

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

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

- Let the training set be (1,1), (2,2), (3,3)
- Calculate the cost function for the following θ values
  - 0.5, 1, 1.5, 0, -0.5

Linear regression: cost function example

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

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

$$\theta_1$$

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

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

- Let
- Ca

s

θ_1	$J(\theta_1)$
1.5	**0.583**
1	**0**
0.5	**0.583**
0	**2.33**
-0.5	**5.25**

Linear regression

- Model representation (hypothesis function)
- Objective function (cost function)
- Optimization algorithm (parameter learning)
- Gradient descent for linear regression

Linear regression: gradient descent

- So we have our hypothesis function and we have a way of measuring how well it fits into the data. Now we need to estimate the parameters in the hypothesis function. That's where gradient descent comes in

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

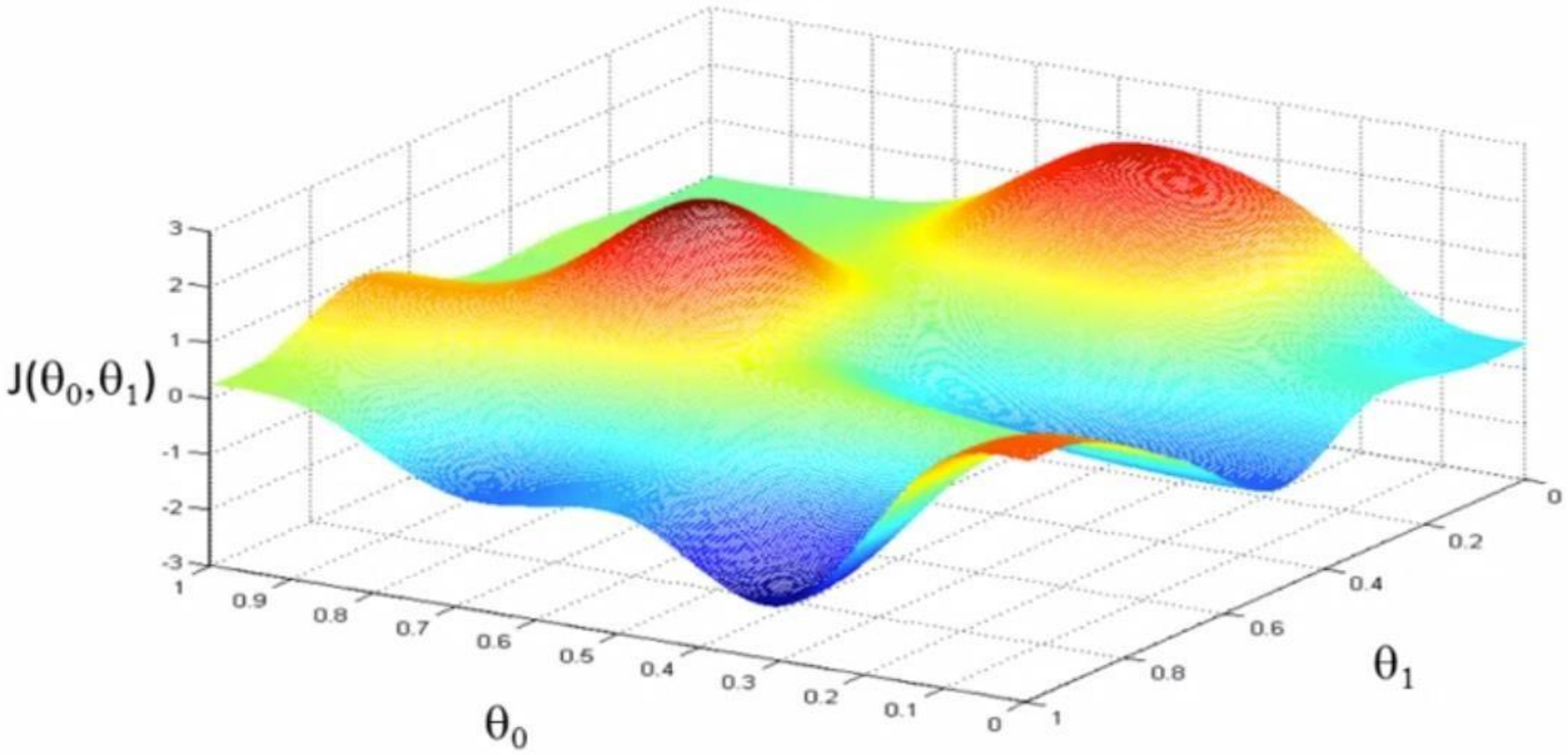
Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Outline:

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$

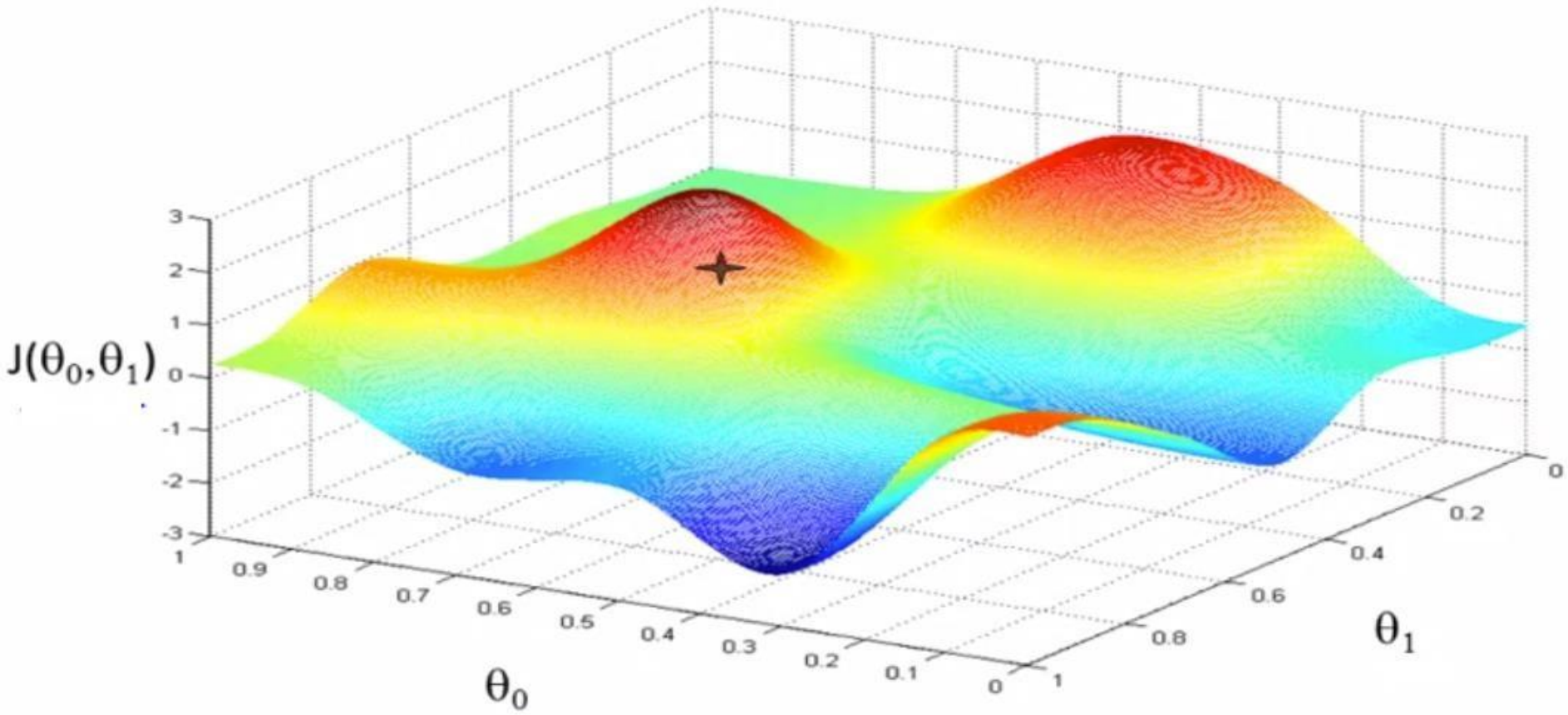
until we hopefully end up at a minimum

Linear regression: gradient descent



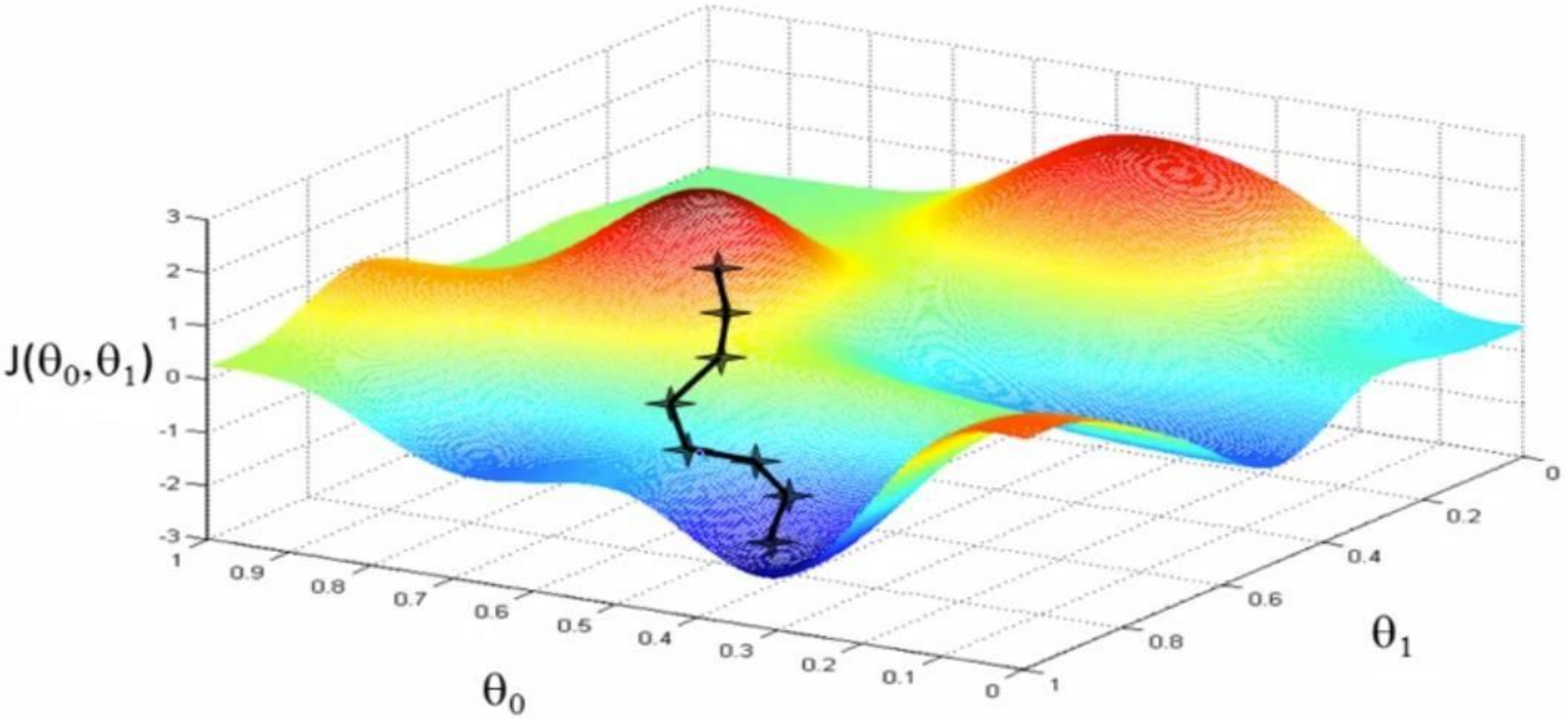
Imagine that we graph the cost function based on the parameters θ_0 and θ_1
The points on our graph will be the result of the cost function using our hypothesis with those specific theta parameters

Linear regression: gradient descent



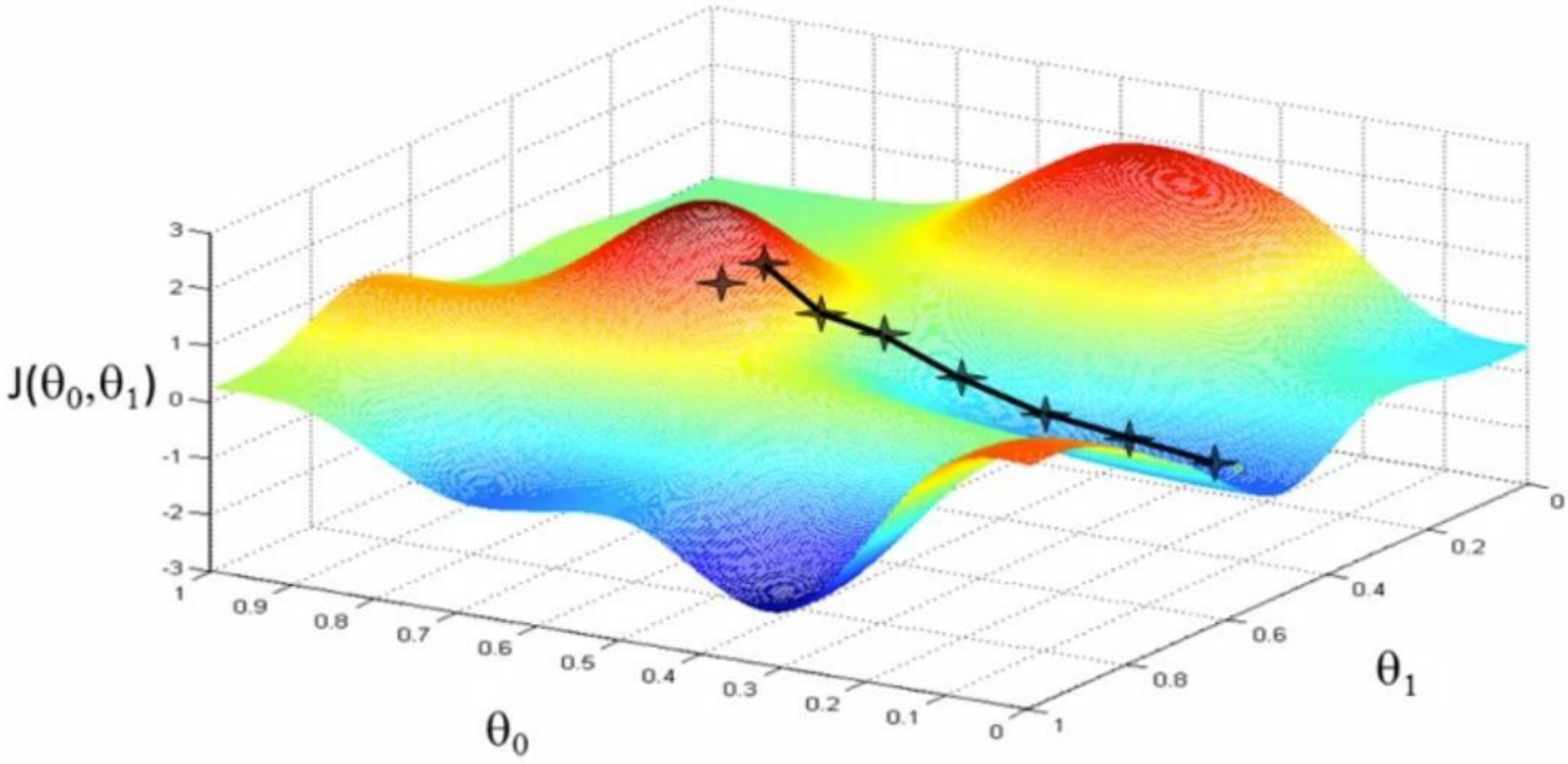
We will know that we have succeeded when our cost function is at the very bottom of the pits in our graph, i.e. when its value is the minimum.

Linear regression: gradient descent



We make steps down the cost function in the direction with the steepest descent. The size of each step is determined by the parameter α, which is called the learning rate.

Linear regression: gradient descent



Depending on where one starts on the graph, one could end up at different points. The image above shows us two different starting points that end up in two different places.

Linear regression: gradient descent algorithm

Gradient descent algorithm

repeat until convergence {

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

}

Correct: Simultaneous update

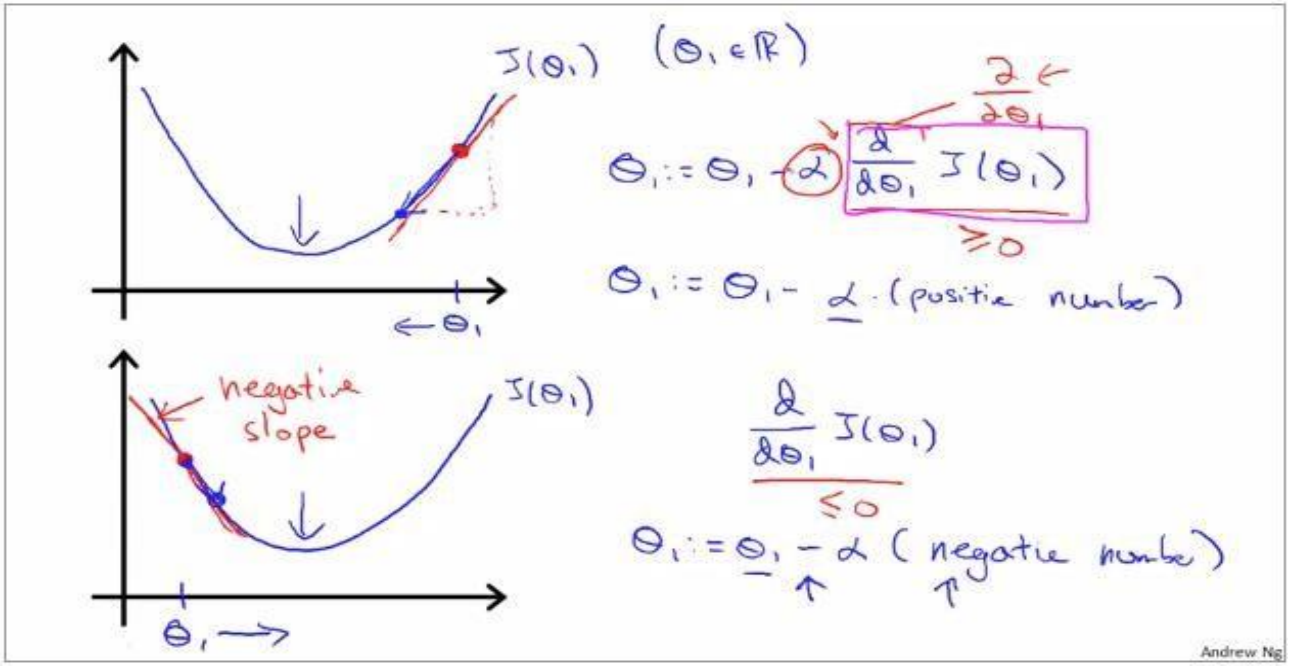
$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

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

$\theta_0 := \text{temp0}$

$\theta_1 := \text{temp1}$

Linear regression: gradient descent algorithm intuition



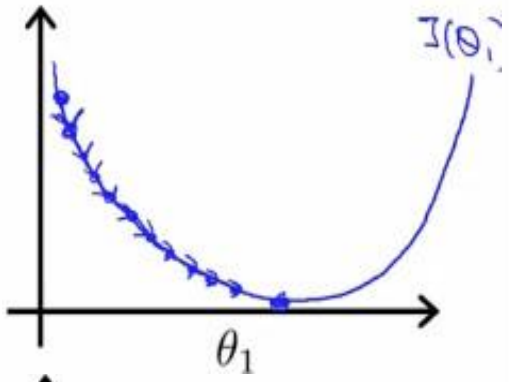
- The intuition behind the convergence of gradient descent is that $\frac{d}{d\theta_1} J(\theta_1)$ approaches 0 as we approach the bottom of our convex function.
- At the minimum, the derivative will always be 0 and thus we get

$$\theta_1 := \theta_1 - \alpha * 0$$

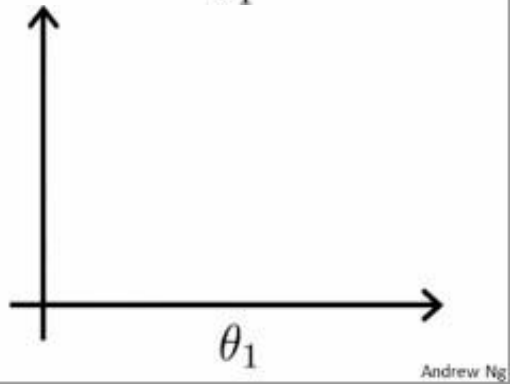
Linear regression: gradient descent algorithm intuition

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

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.



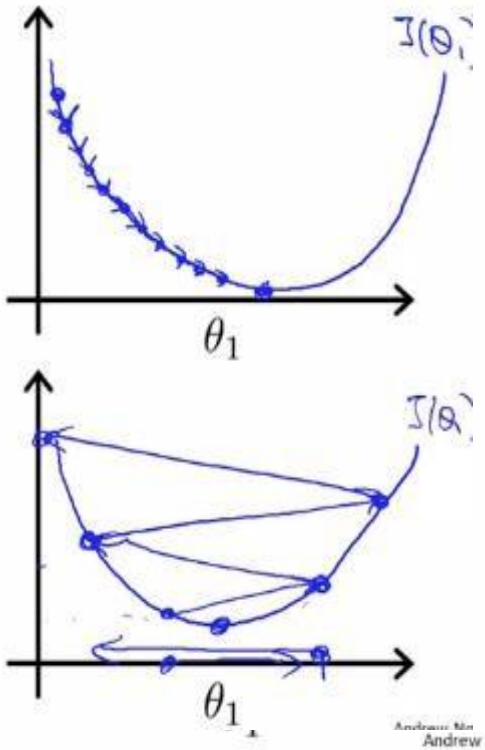
Andrew Ng

Linear regression: gradient descent algorithm intuition

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

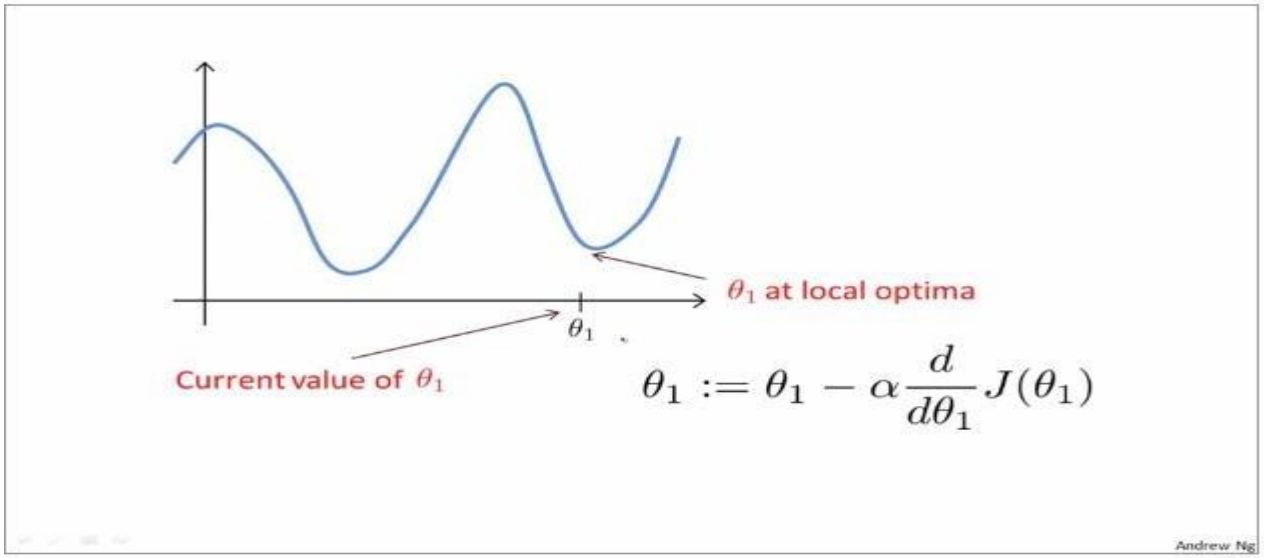
If α is too small, gradient descent can be slow.

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



Linear regression: gradient descent algorithm intuition

Suppose θ_1 is at a local optimum of $J(\theta_1)$, such as shown in the figure below. What will one step of gradient descent $\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$ do?



Gradient descent can converge to a local minimum

Linear regression: 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

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

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

repeat until convergence {

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

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

}

- We want to apply gradient descent to minimize the squared error cost function

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

Linear regression: 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

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

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

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

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

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

- The point of all this is that if we start with a guess for our hypothesis and then repeatedly apply these gradient descent equations, our hypothesis will become more and more accurate.

Univariate Linear Regression: Example

- Consider the problem of predicting how well a student does in her second year of university, given how well she did in her first year.
- Specifically, let x be equal to the number of "A" grades that a student receives in their first year of university. We would like to predict the value of y, which we define as the number of "A" grades they get in their second year.
- Recall that in linear regression, our hypothesis is $h_\theta(x) = \theta_0 + \theta_1 x$, and we use m to denote the number of training examples.

x	y
5	4
3	4
0	1
4	3

- For the training set given above, what is the value of m?
- What is J(0,1)?
- **0.5**
- Suppose we set $\theta_0 = -1, \theta_1 = 0.5$. What is $h_\theta(4)$? $J(\theta_0, \theta_1)$?
- **1 and 3.094**

Linear Regression with multiple variables: multivariate

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

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

Linear Regression with multiple variables: multivariate

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

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

Multiple features (variables).

Size (feet²)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

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

Multivariate Linear Regression: hypothesis

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots\ldots \theta_n x_n$$

If we define $x_0 = 1$, then $h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots\ldots \theta_n x_n$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix} \qquad \Theta = \begin{bmatrix} \Theta_0 \\ \Theta_1 \\ \Theta_2 \\ . \\ . \\ . \\ \Theta_n \end{bmatrix}$$

$$[\Theta_0 \Theta_1 \Theta_1 \ \ldots \ \Theta_n] \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix} = \Theta^T x = h_\theta(x)$$

Multivariate Linear Regression: gradient descent

repeat until convergence {

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

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

}

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:

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

repeat until convergence: {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)}$$

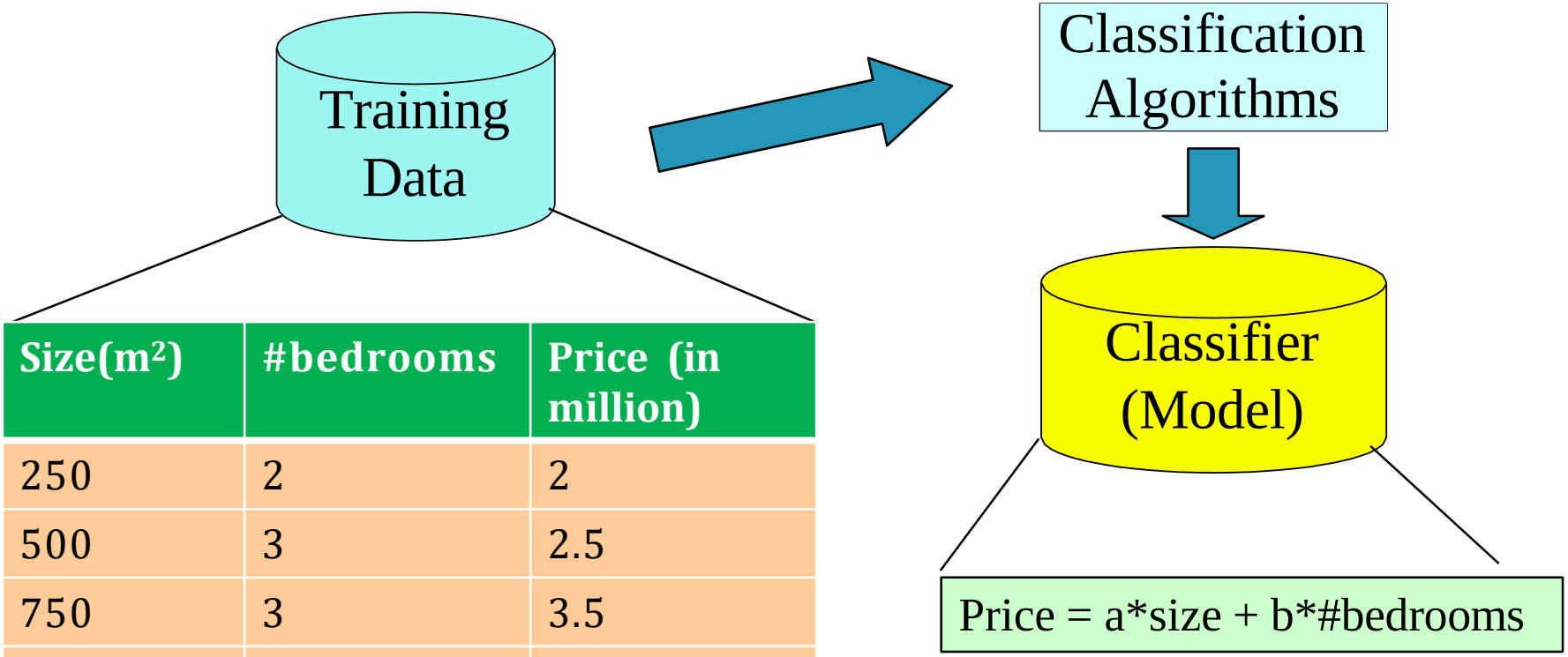
...

}

- The gradient descent equation itself is generally the same form; we just have to repeat it for our 'n' features:

(Summary) Learning – Model Construction

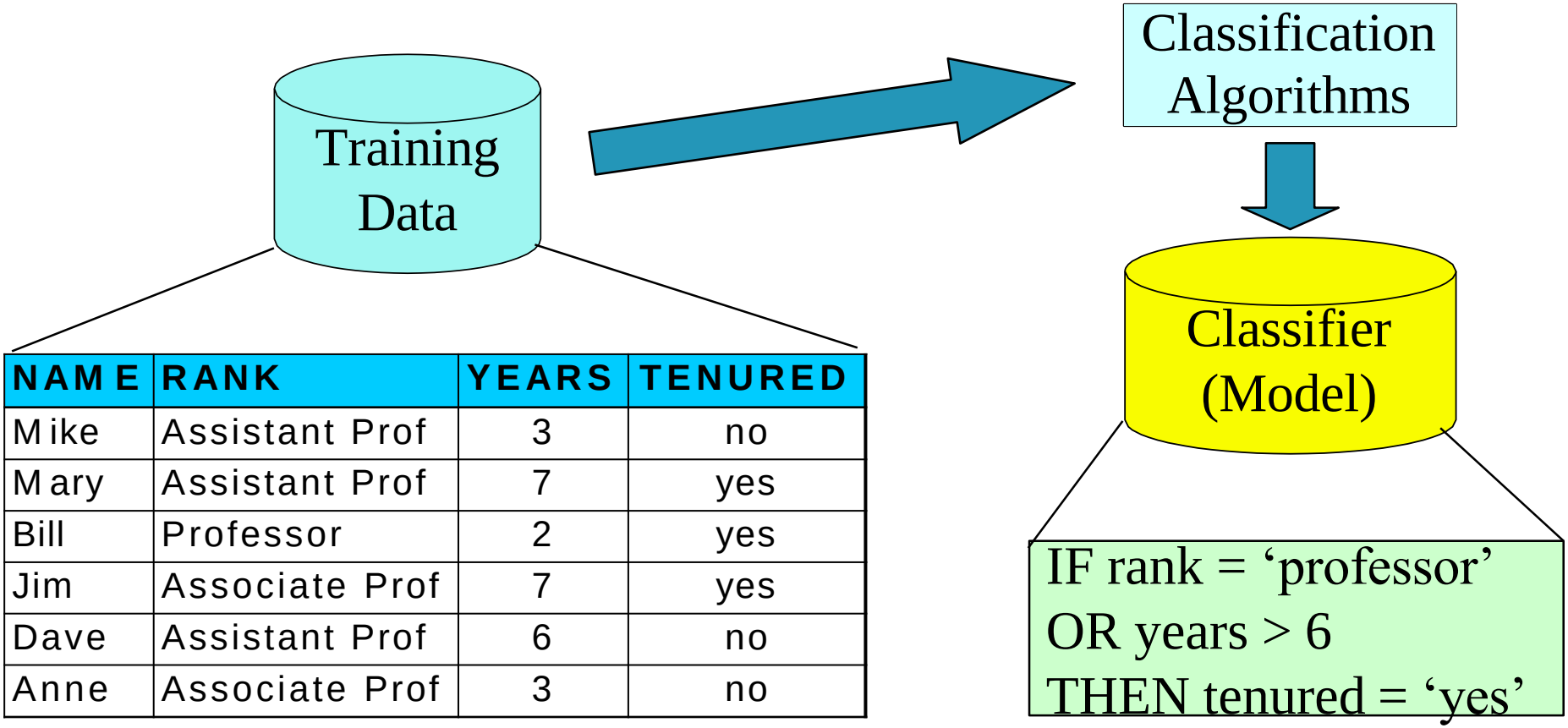
- A training set is used to create the model.
- The model is represented as classification rules, decision trees, or mathematical formula



Size(m^2)	#bedrooms	Price (in million)
250	2	2
500	3	2.5
750	3	3.5
1000	5	5

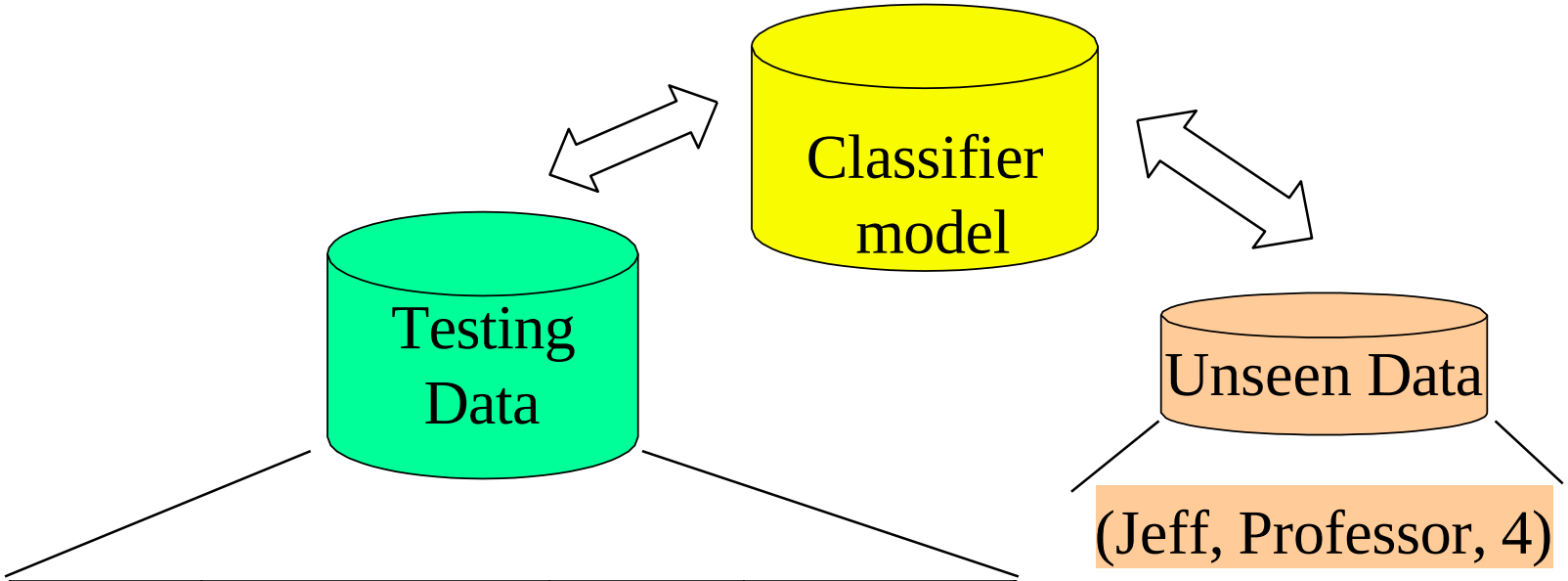
(Summary) Learning – Model Construction

- A training set is used to create the model.
- The model is represented as classification rules, decision trees, or mathematical formula



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

- **Model usage**
  - the test set is used to see how well it works for classifying future or unknown objects



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes