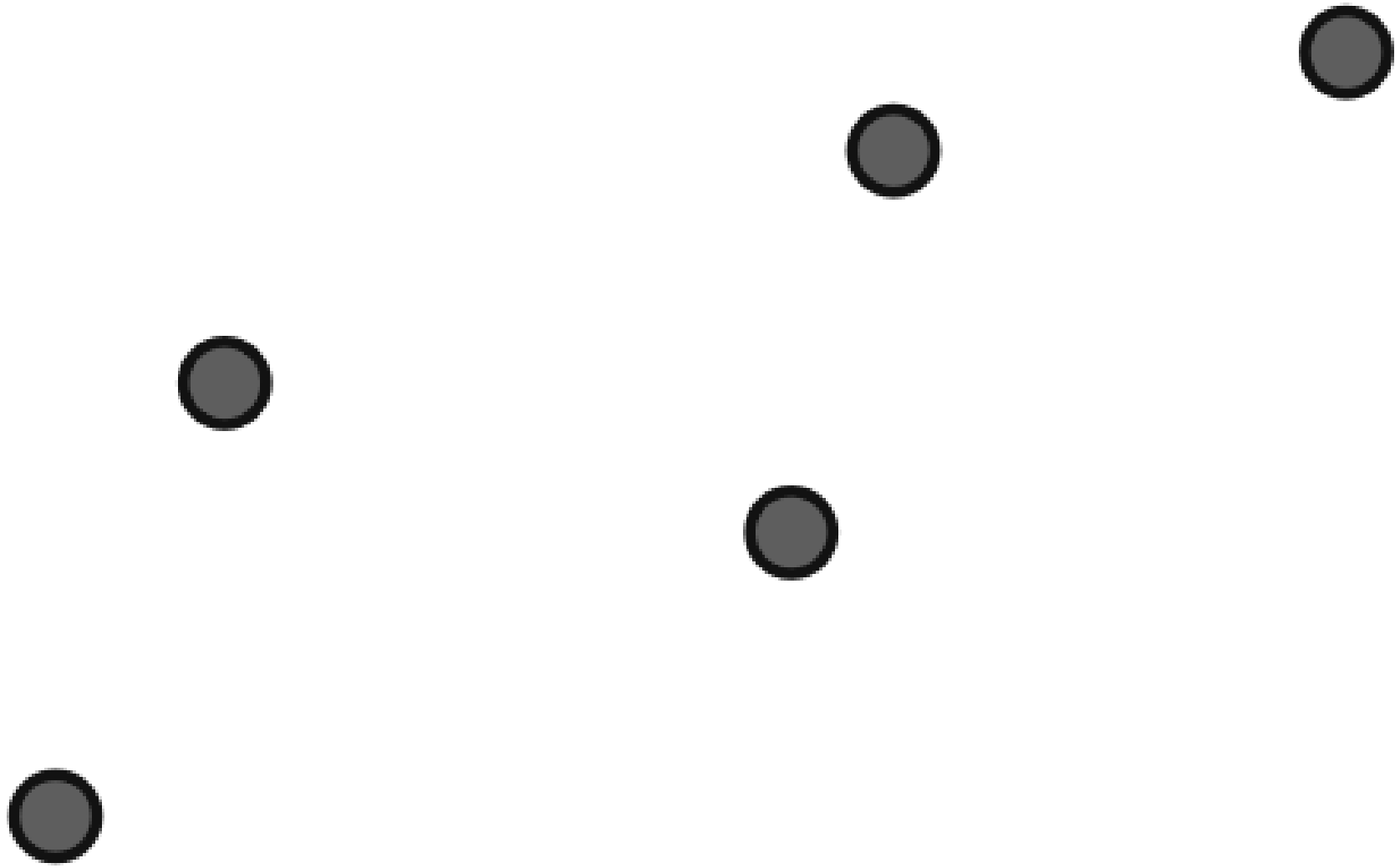
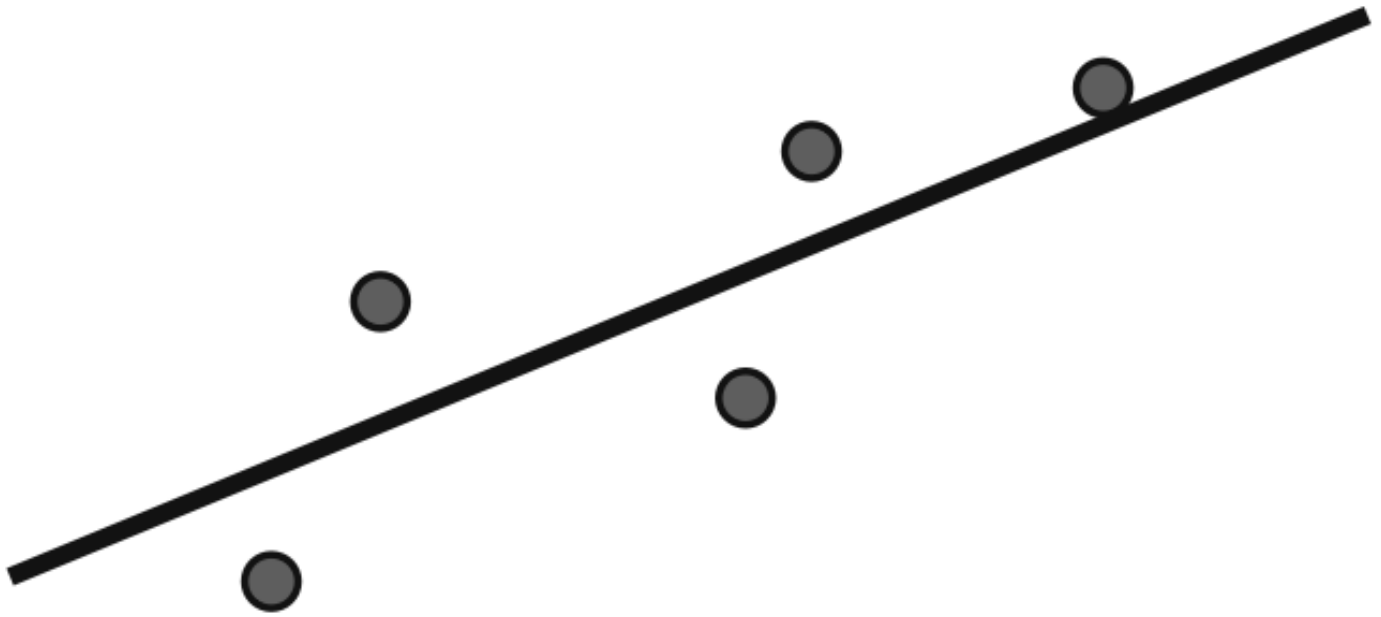
LINEAR REGRESSION

Drawing a line close to our points

What is linear regression?


USER FRIENDLY by Illiad
I DRANK ONE CUP OF COFFEE AND I MANAGED TO WRITE ONE PAGE.
VP SALES
Copyright (c) 2000 Illiad http://www.userfriendly.org/
THE NEXT DAY I DRANK TWO CUPS OF COFFEE AND I MANAGED TO WRITE TWO PAGES.
VP SALES
WHAT DO YOU THINK WOULD HAPPEN IF I DRINK THREE CUPS?
I DON'T KNOW. I'M NO DATA SCIENTIST.
VP SALES

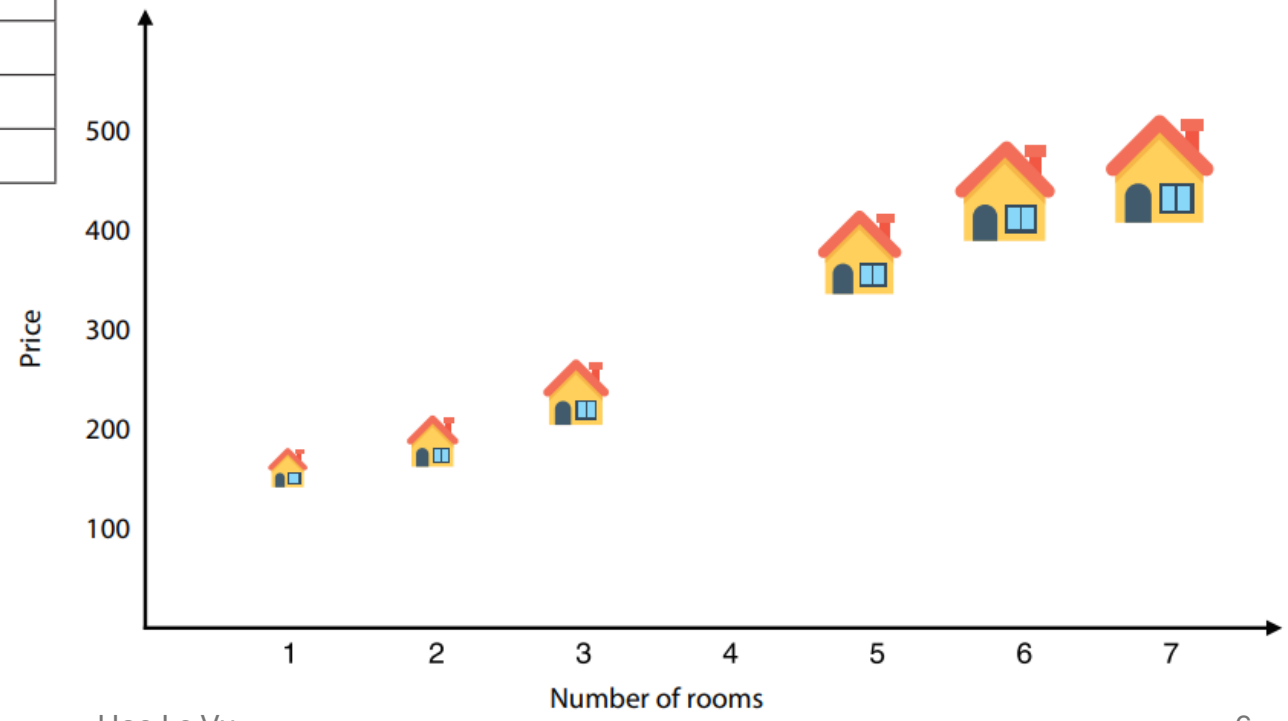




- What do we mean by “points that roughly look like they are forming a line”?
- What do we mean by “a line that passes really close to the points”?
- How do we find such a line?
- Why is this useful in the real world?
- Why is this machine learning?

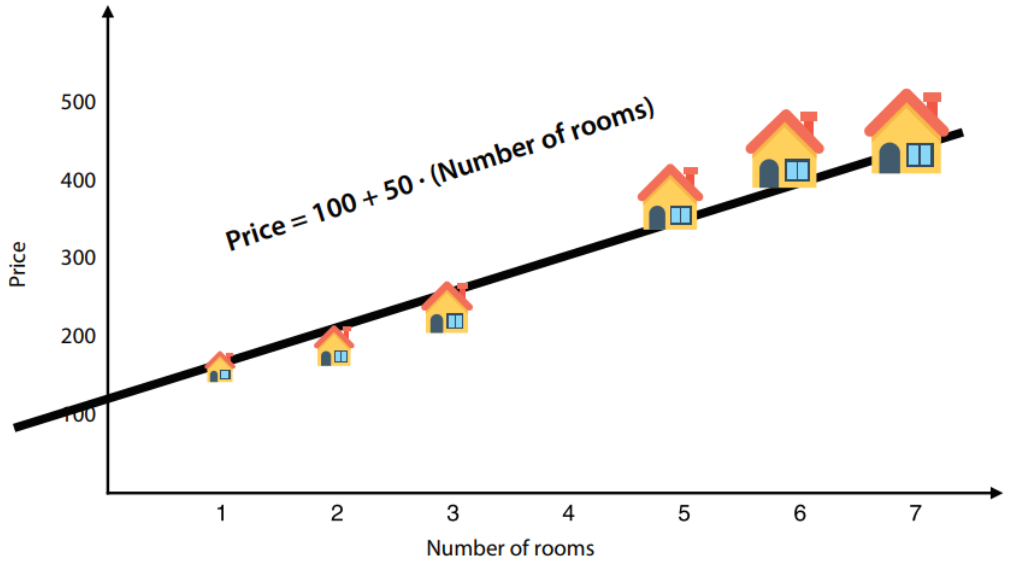
The problem: We need to predict the price of a house

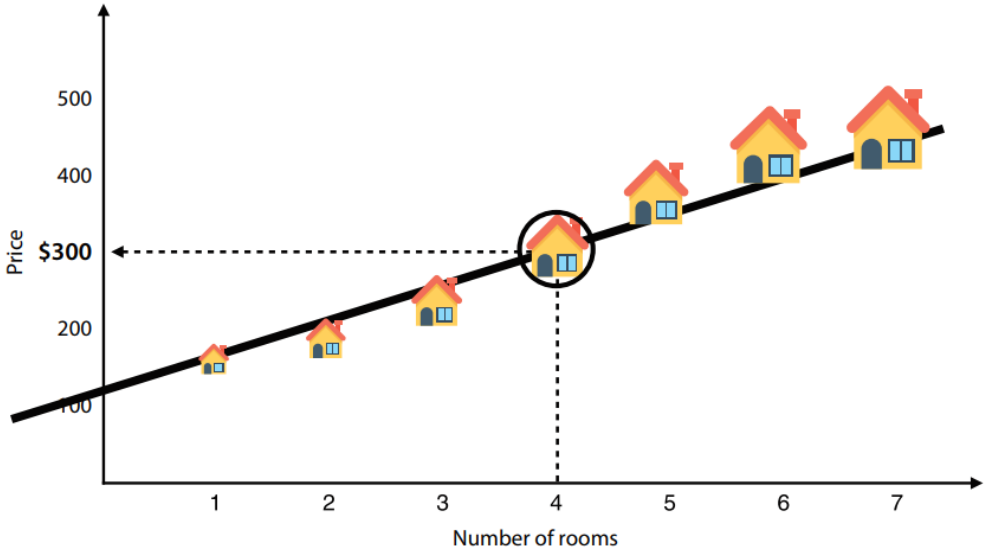
Number of rooms	Price
1	150
2	200
3	250
4	?
5	350
6	400
7	450



The solution: Building a regression model for housing prices

$$\text{Price} = 100 + 50(\text{Number of rooms}) + (\text{Small error})$$





Simple linear regression

- p: The price of a house in the dataset
- $\hat{p}$: The predicted price of a house
- r: The number of rooms
- m: The price per room
- b: The base price for a house

$$\hat{p} = mr + b.$$

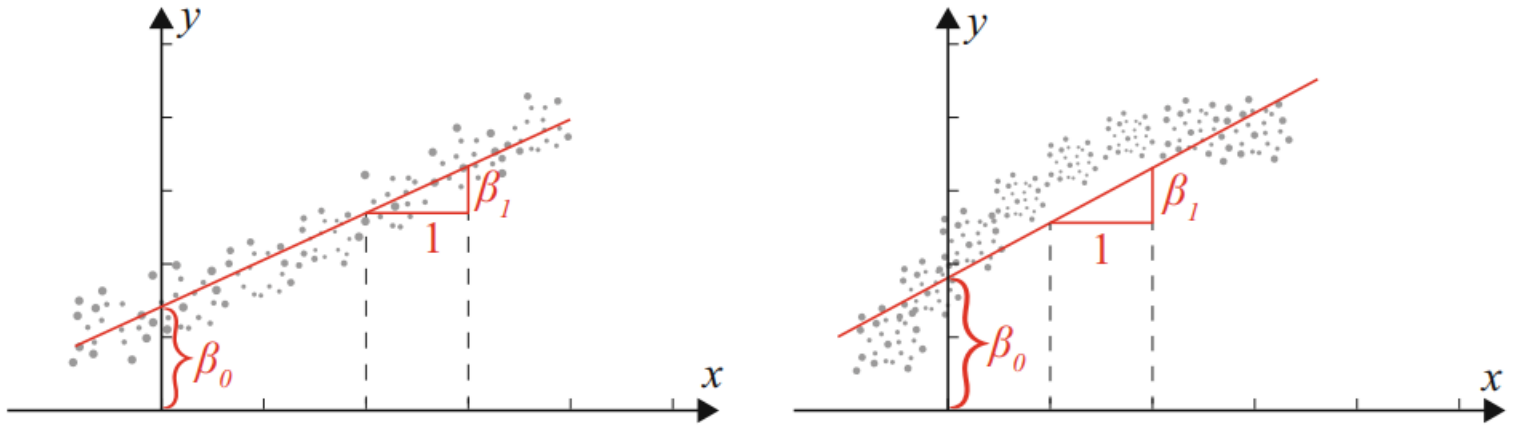
This is a formulaic way of saying

Predicted price = (Price per room)(Number of rooms) + Base price of the house.

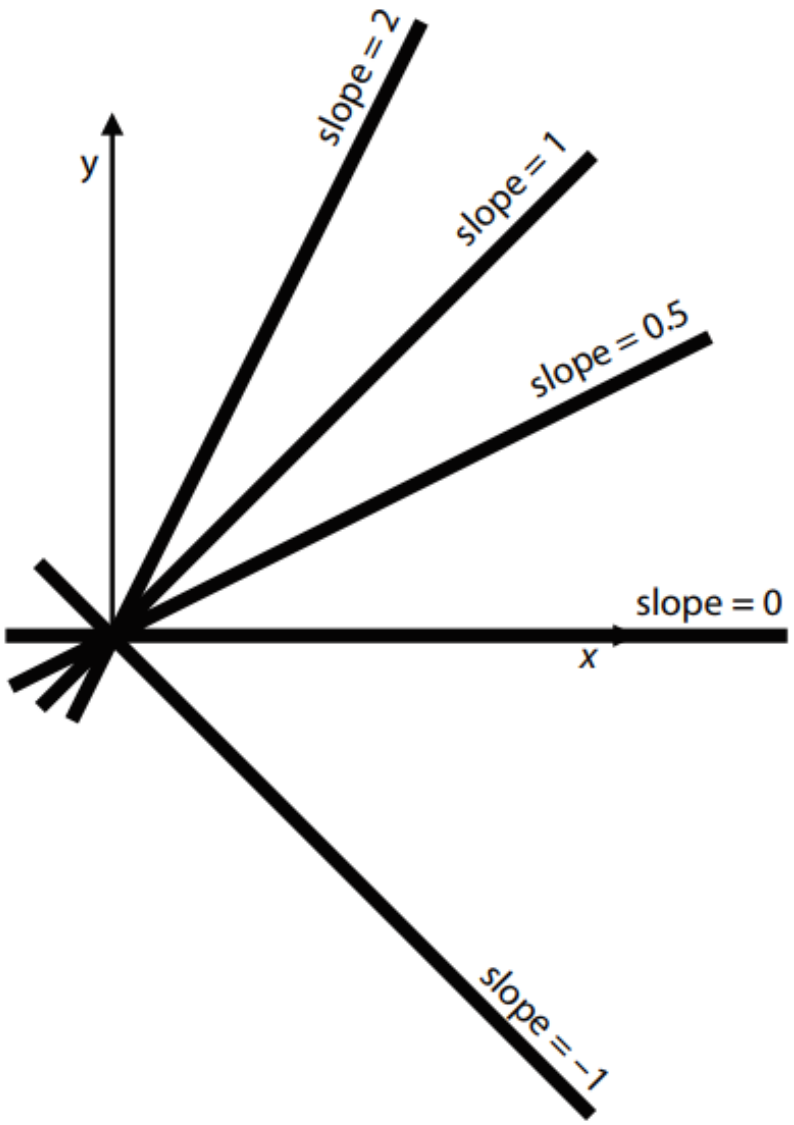
Simple linear regression

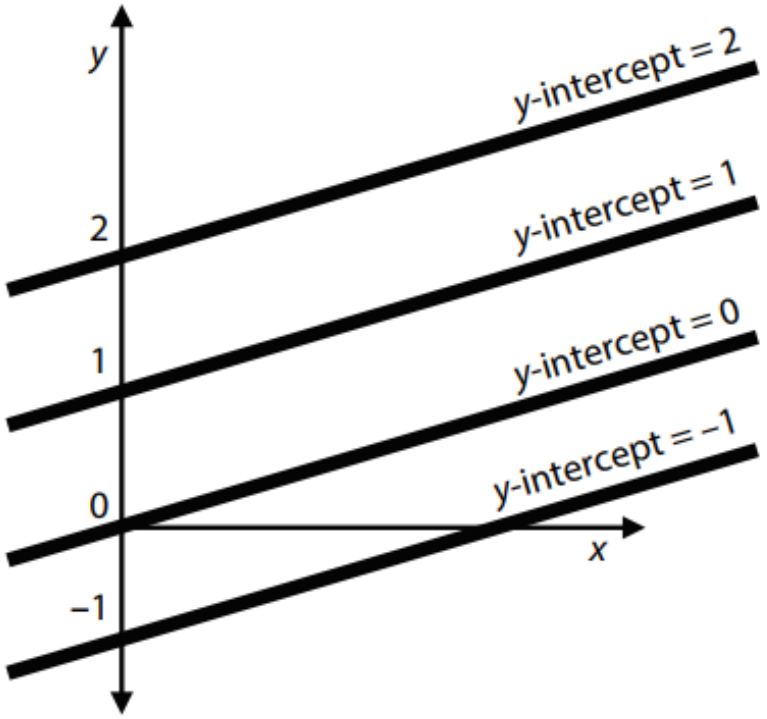
$$y = \beta_0 + \beta_1 x,$$

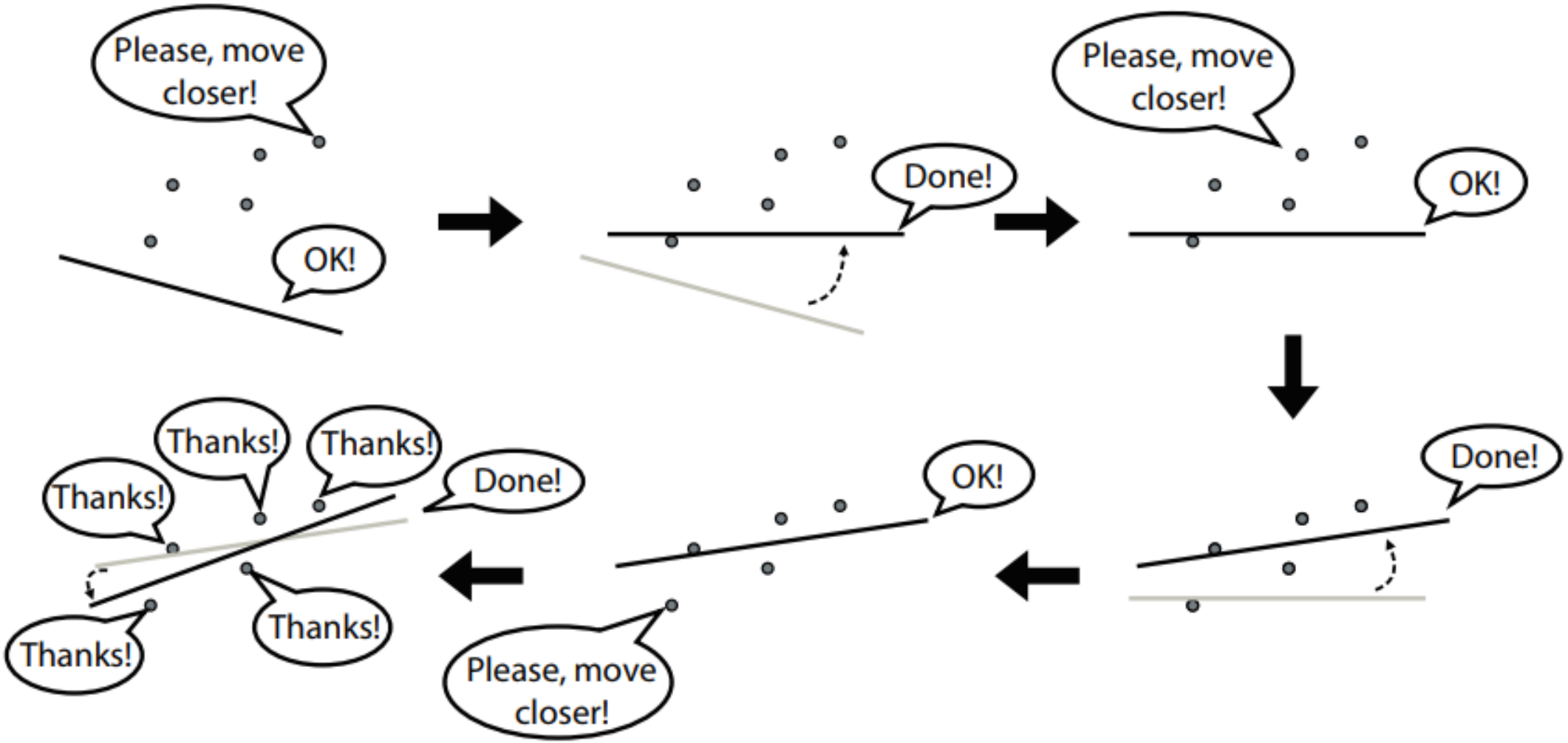
where x and y are the independent and dependent variables, respectively, β_0 is the intercept from the y-axis, and β_1 is the slope.



Fitting straight line (left) and curve line (right) using a simple linear regression model


y
slope = 2
slope = 1
slope = 0.5
slope = 0
x
slope = −1


y
y-intercept = 2
y-intercept = 1
y-intercept = 0
y-intercept = −1
2
1
0
−1
x


Please, move closer!
OK!
Done!
Please, move closer!
OK!
Done!
OK!
Please, move closer!
Thanks!
Thanks!
Thanks!
Thanks!
Thanks!
Done!

Pseudocode for the linear regression algorithm

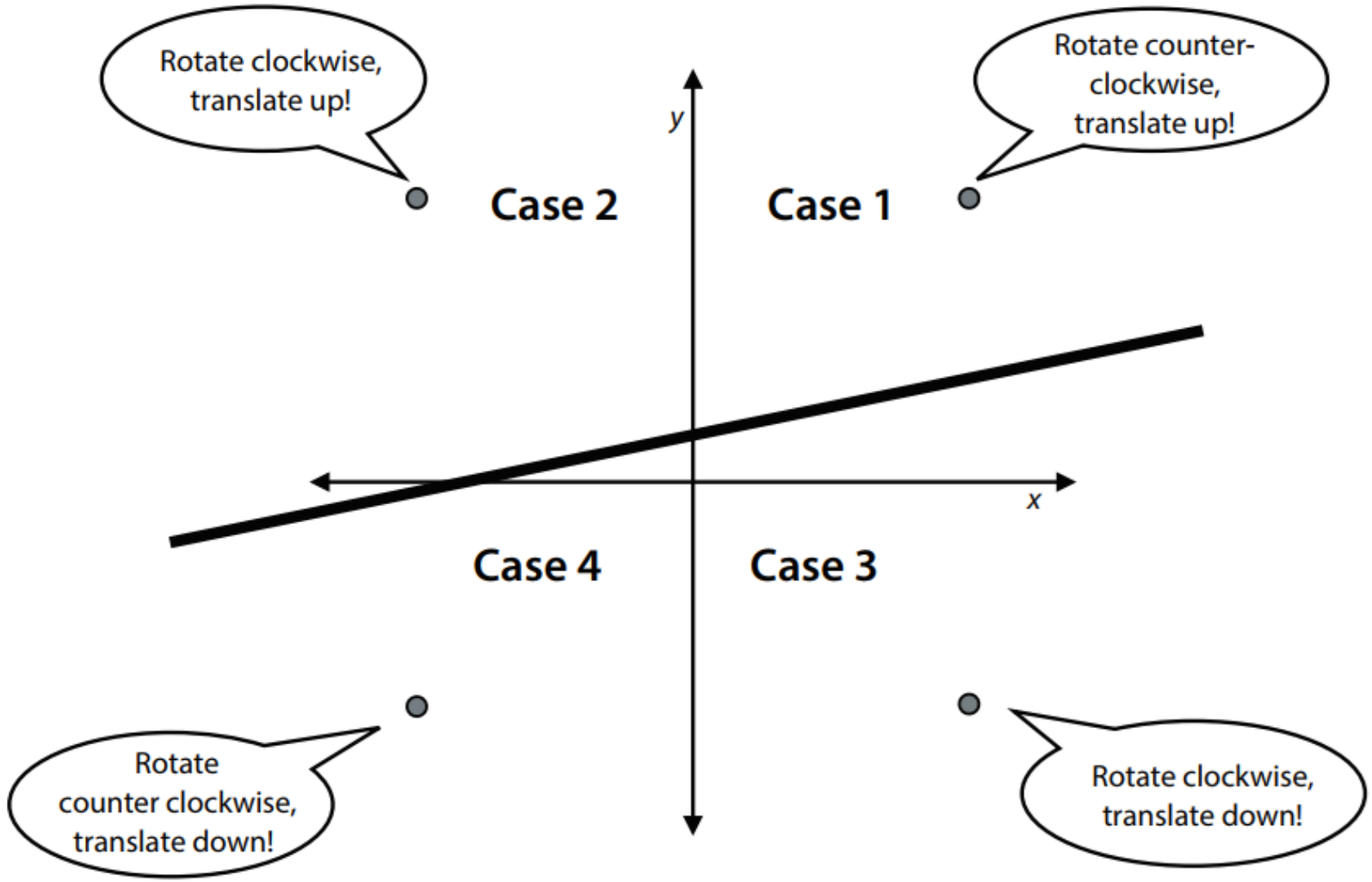
Inputs: A dataset of points

Outputs: A linear regression model that fits that dataset

Procedure:

- Pick a model with random weights and a random bias.
- Repeat many times:
  - Pick a random data point.
  - Slightly adjust the weights and bias to improve the prediction for that particular data point.
- **Return** the model you've obtained.

A simple trick to move a line closer to a set of points, one point at a time



The four cases. In each of these we must rotate the line and translate it in a different way to move the line closer to the corresponding point.

Pseudocode for the simple trick

Inputs:

- A line with slope m, y-intercept b, and equation $\hat{p} = mr + b$
- A point with coordinates (r, p)

Output:

- A line with equation $\hat{p} = m'r + b$ that is closer to the point

Procedure:

Pick two very small random numbers, and call them η_1 and η_2 (the Greek letter *eta*).

Case 1: If the point is above the line and to the right of the y-axis, we rotate the line counterclockwise and translate it upward:

- Add η_1 to the slope m. Obtain $m' = m + \eta_1$.
- Add η_2 to the y-intercept b. Obtain $b' = b + \eta_2$.

Case 2: If the point is above the line and to the left of the y-axis, we rotate the line clockwise and translate it upward:

- Subtract η_1 from the slope m. Obtain $m' = m - \eta_1$.
- Add η_2 to the y-intercept b. Obtain $b' = b + \eta_2$.

Case 3: If the point is below the line and to the right of the y-axis, we rotate the line clockwise and translate it downward:

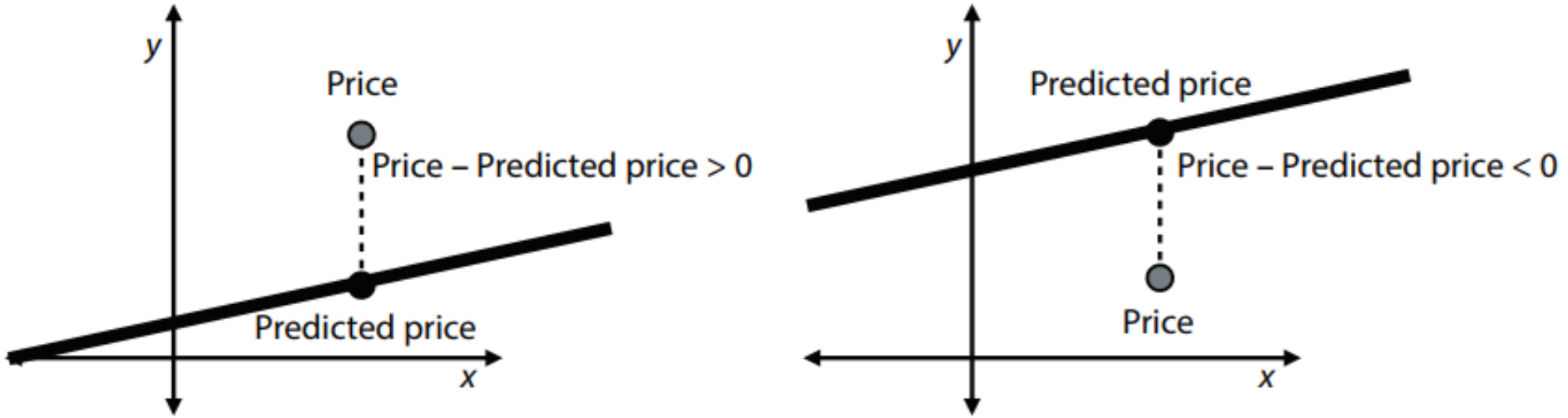
- Subtract η_1 from the slope m. Obtain $m' = m - \eta_1$.
- Subtract η_2 from the y-intercept b. Obtain $b' = b - \eta_2$.

Case 4: If the point is below the line and to the left of the y-axis, we rotate the line counterclockwise and translate it downward:

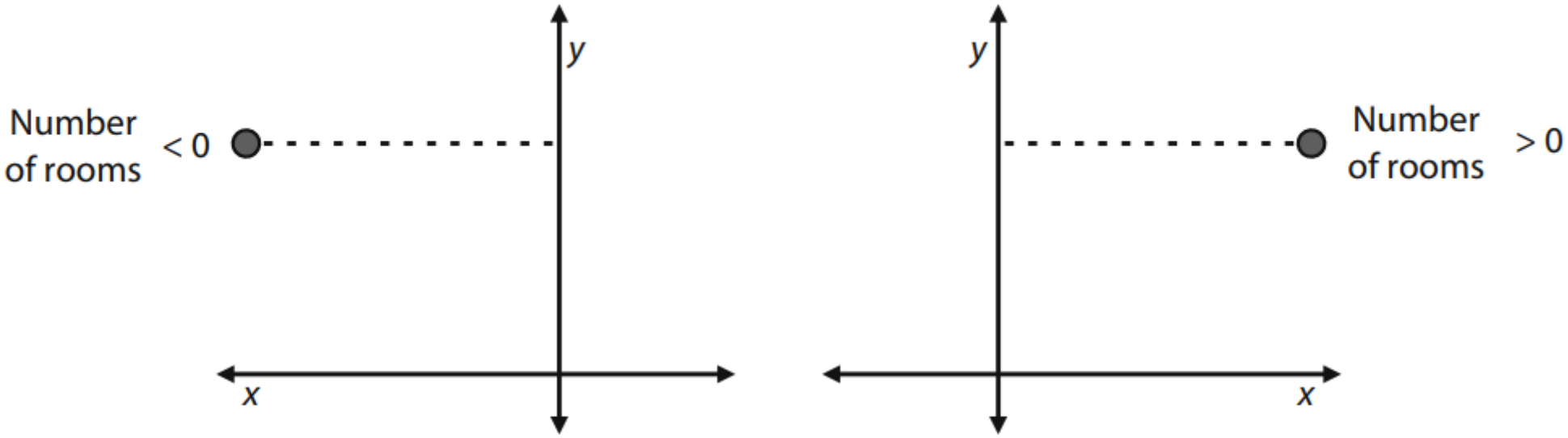
- Add η_1 to the slope m. Obtain $m' = m + \eta_1$.
- Subtract η_2 from the y-intercept b. Obtain $b' = b - \eta_2$.

Return: The line with equation $\hat{p} = m'r + b'$.

The square trick: A much more clever way of moving our line closer to one of the points



Left: When the point is above the line, the price is larger than the predicted price, so the difference is positive. Right: When the point is below the line, the price is smaller than the predicted price, so the difference is negative.



Left: When the point is to the left of the ***y***-axis, the number of rooms is negative. Right: When the point is to the right of the ***y***-axis, the number of rooms is positive.

Pseudocode for the square trick

Inputs:

- A line with slope m, y-intercept b, and equation $\hat{p} = mr + b$
- A point with coordinates (r, p)
- A small positive value η (the learning rate)

Output:

- A line with equation $\hat{p} = m'r + b'$ that is closer to the point

Procedure:

- Add $\eta r(p - \hat{p})$ to the slope m. Obtain $m' = m + \eta r(p - \hat{p})$ (this rotates the line).
- Add $\eta(p - \hat{p})$ to the y-intercept b. Obtain $b' = b + \eta(p - \hat{p})$ (this translates the line).

Return: The line with equation $\hat{p} = m'r + b'$

The absolute trick: Another useful trick to move the line closer to the points

- The square trick is effective, but another useful trick, which we call the absolute trick, is an intermediate between the simple and the square tricks.

> In the square trick, we used the two quantities, $p - \hat{p}$ (price – predicted price) and r (number of rooms), to help us bring the four cases down to one.
>
> In the absolute trick, we use only r to help us bring the four cases down to two.

Pseudocode for the absolute trick

Inputs:

- A line with slope m, y-intercept b, and equation $\hat{p} = mr + b$
- A point with coordinates (r, p)
- A small positive value η (the learning rate)

Output:

- A line with equation $\hat{p} = m'r + b'$ that is closer to the point

Procedure:

Case 1: If the point is above the line (i.e., if $p > \hat{p}$):

- Add ηr to the slope m. Obtain $m' = m + \eta r$ (this rotates the line counterclockwise if the point is to the right of the y-axis, and clockwise if it is to the left of the y-axis).
- Add η to the y-intercept b. Obtain $b' = b + \eta$ (this translates the line up).

Case 2: If the point is below the line (i.e., if $p < \hat{p}$):

- Subtract ηr from the slope m. Obtain $m' = m - \eta r$ (this rotates the line clockwise if the point is to the right of the y-axis, and counterclockwise if it is to the left of the y-axis).
- Subtract η from the y-intercept b. Obtain $b' = b - \eta$ (this translates the line down).

Return: The line with equation $\hat{p} = m'r + b'$

Pseudocode for the linear regression algorithm

Inputs:

- A dataset of houses with number of rooms and prices

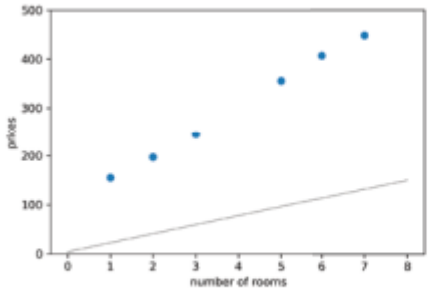
Outputs:

- Model weights: price per room and base price

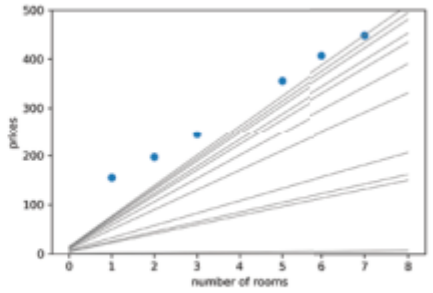
Procedure:

- Start with random values for the slope and *y*-intercept.
- Repeat many times:
  - Pick a random data point.
  - Update the slope and the *y*-intercept using the absolute or the square trick.

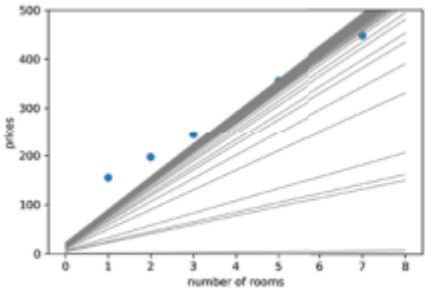
Each iteration of the loop is called an *epoch*, and we set this number at the beginning of our algorithm.



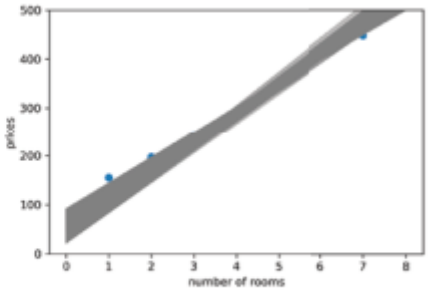
Starting point



Epochs 1–10



Epochs 1–50



Epochs 51–10,000

Drawing some of the lines in our algorithm, as we approach a better solution. The first graphic shows the starting point. The second graphic shows the first 10 epochs of the linear regression algorithm. Notice how the line is moving closer to fitting the points. The third graphic shows the first 50 epochs. The fourth graphic shows epochs 51 to 10,000 (the last epoch).

The general linear regression algorithm

The general case will consist of a dataset of m points and n features. Thus, the model has m weights (think of them as the generalization of the slope) and one bias. The notation follows:

- The data points are $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$. Each point is of the form $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)})$.
- The corresponding labels are $y_1, y_2, \ldots, y_m$.
- The weights of the model are $w_1, w_2, \ldots, w_n$.
- The bias of the model is b.

Pseudocode for the general square trick

Inputs:

- A model with equation $\hat{y} = w_1 x_1 + w_2 x_{2+} + \cdots + w_n x_n + b$
- A point with coordinates (x, y)
- A small positive value η (the learning rate)

Output:

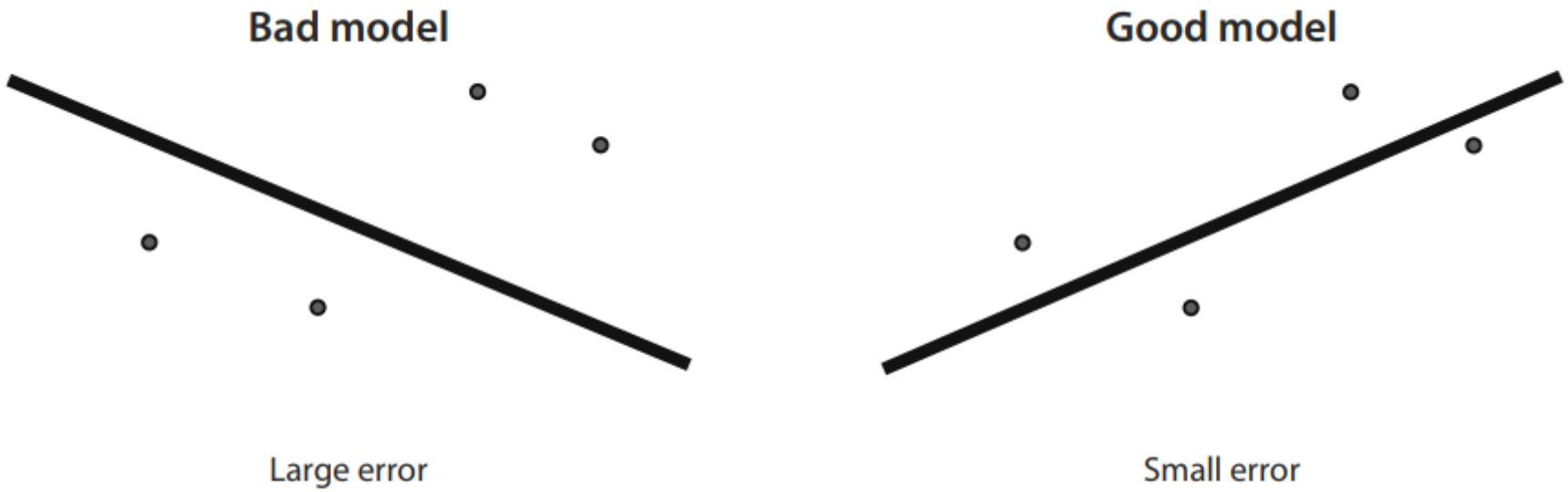
- A model with equation $\hat{y} = w_1' x_1 + w_2' x_2 + \cdots + w_n' x_n + b'$ that is closer to the point

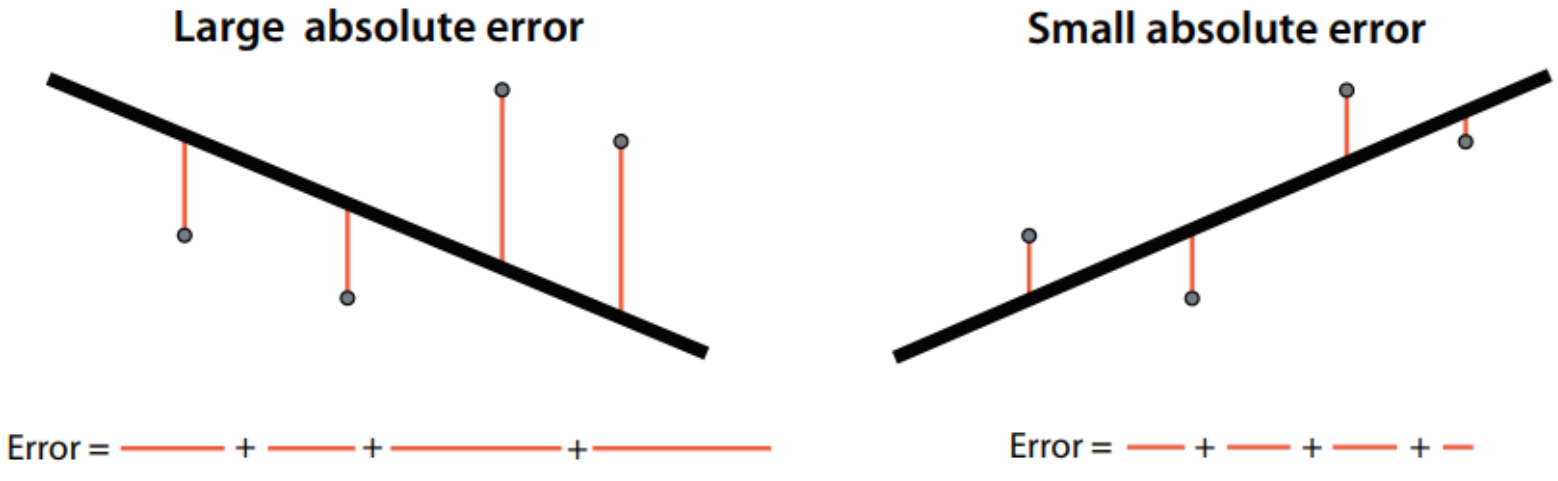
Procedure:

- Add $\eta(y - \hat{y})$ to the y-intercept b. Obtain $b' = b + \eta(y - \hat{y})$.
- For $i = 1, 2, \ldots, n$:
  - Add $\eta x(y - \hat{y})$ to the weight w_i. Obtain $w_i' = w_i + \eta r(y - \hat{y})$.

Return: The model with equation $\hat{y} = w_1' x_1 + w_2' x_2 + \cdots + w_n' x_n + b'$

How do we measure our results? The error function





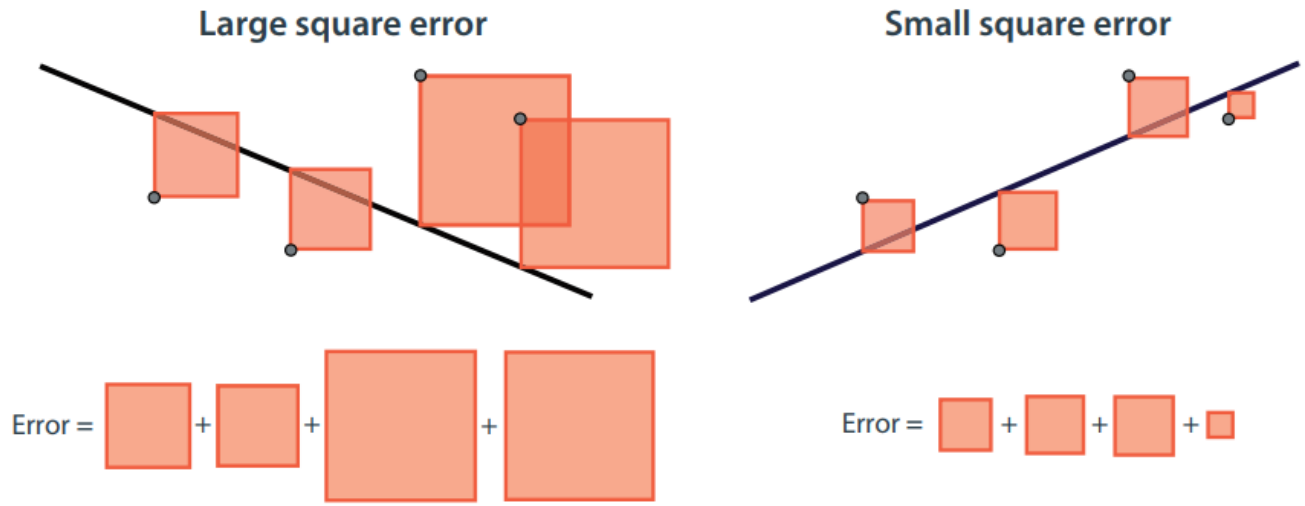
$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y^{(i)} - \hat{y}^{(i)}|.$$

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y^{(i)} - \hat{y}^{(i)}}{\hat{y}^{(i)}}\right|.$$

The absolute error is the sum of the vertical distances from the points to the line. Note that the absolute error is large for the bad model on the left and small for the good model on the right.

mean absolute error (MAE)

mean absolute percentage error (MAPE)



$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2.$$

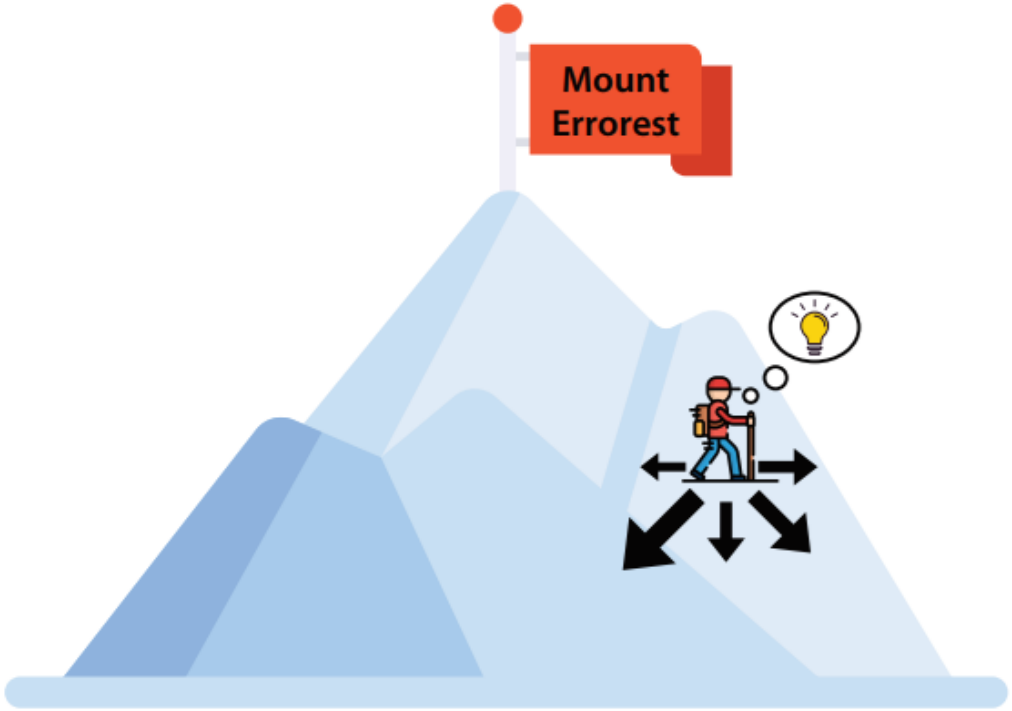
$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2}.$$

The square error is the sum of the squares of the vertical distances from the points to the line. Note that the square error is large for the bad model on the left and small for the good model on the right.

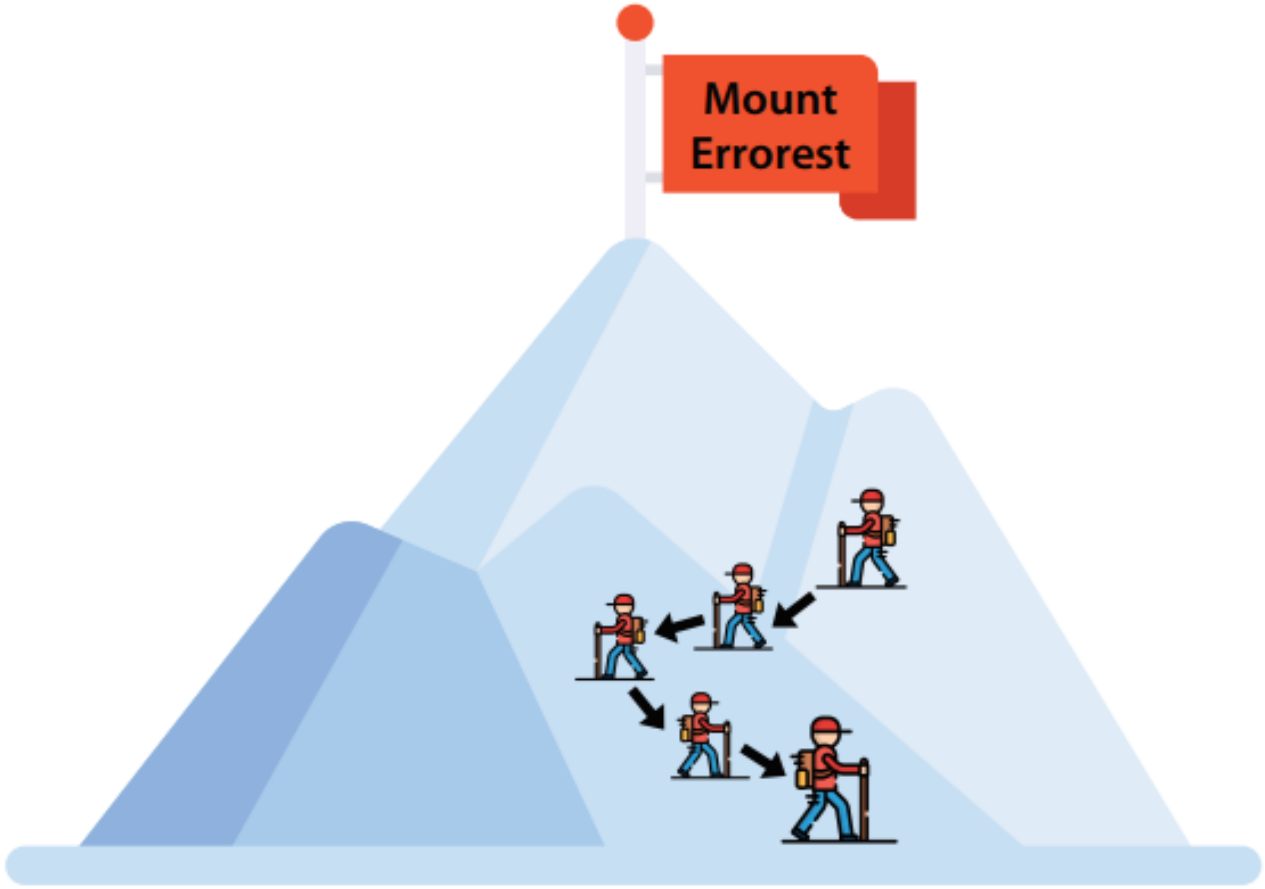
mean squared error (MSE)

root mean squared error (RMSE)

Gradient descent: How to decrease an error function by slowly descending from a mountain

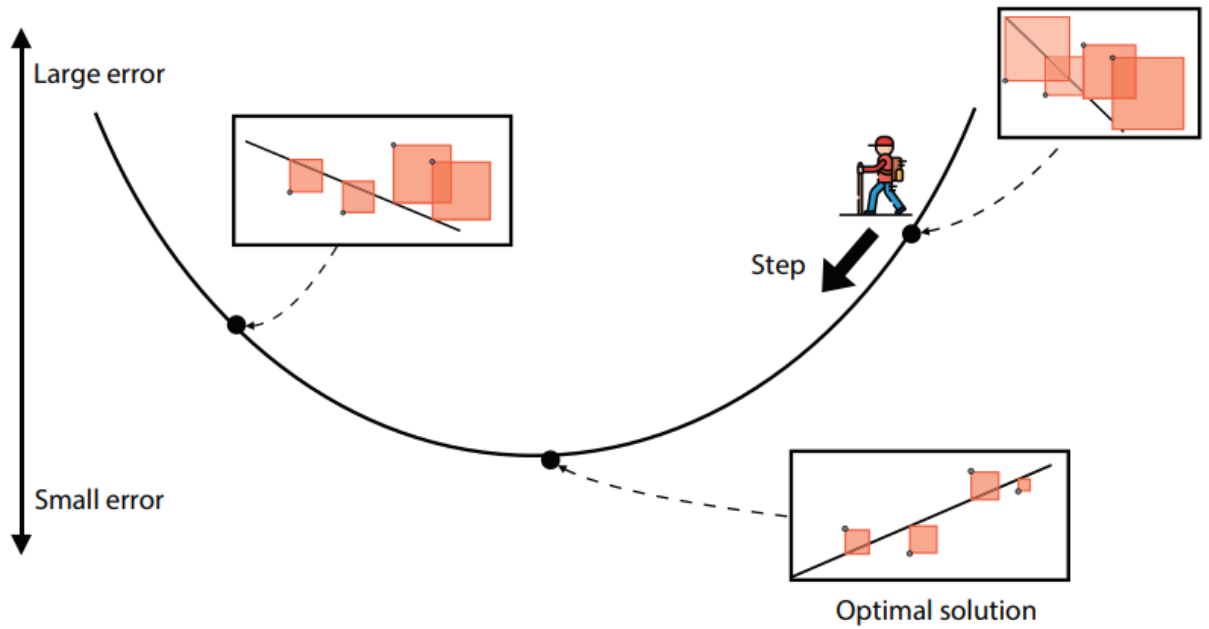


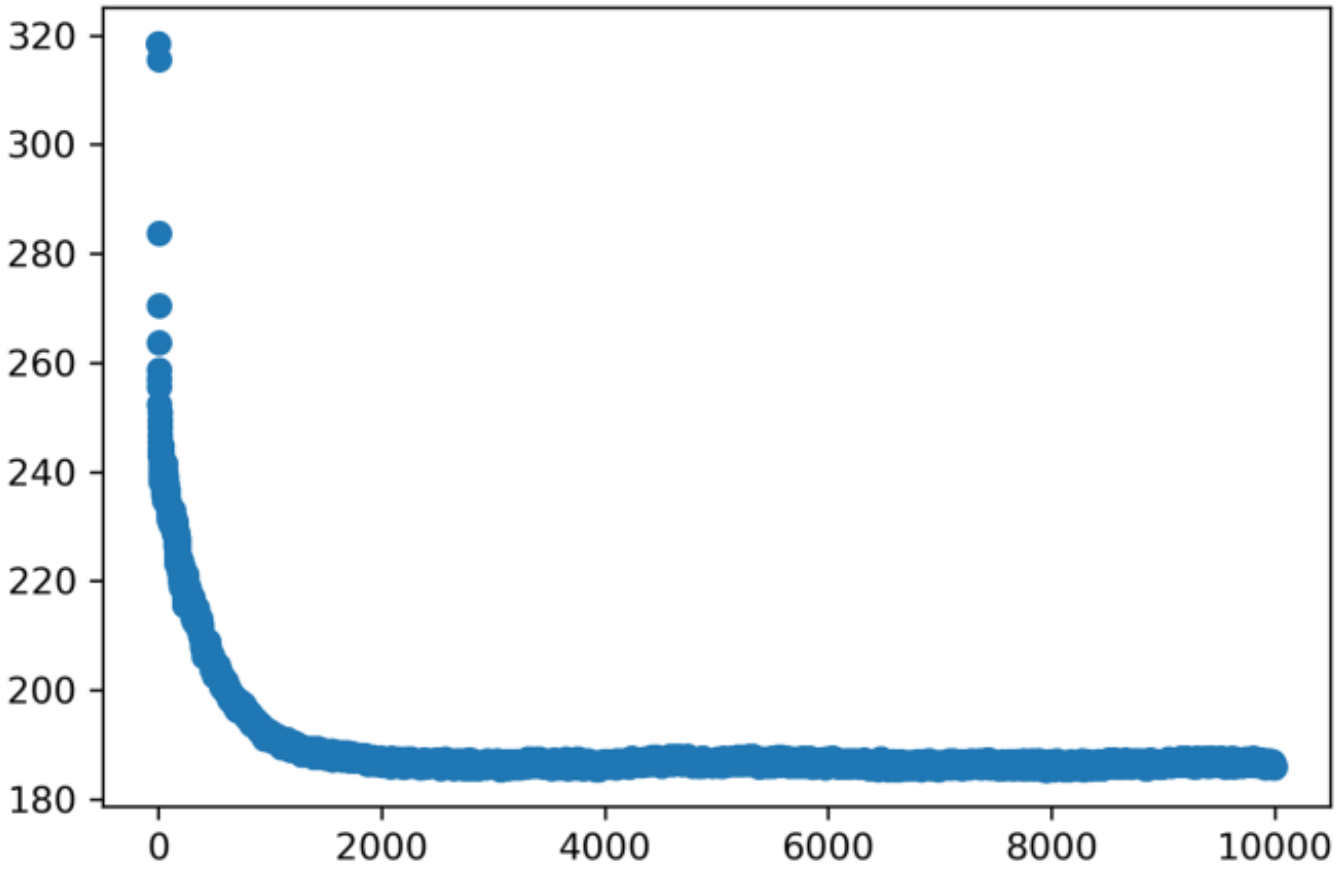
We are on top of Mount Errorest and wish to get to the bottom, but we can't see very far.
A way to go down is to look at all the directions in which we can take one step and figure out which one helps us descend the most. Then we are one step closer to the bottom.



The way to descend from the mountain is to take that one small step in the direction that makes us descend the most and to continue doing this for a long time.

1. Start with any line.
2. Find the best direction to move our line a little bit, using either the absolute or the square trick.
3. Move the line a little bit in this direction.
4. Repeat steps 2 and 3 many times.

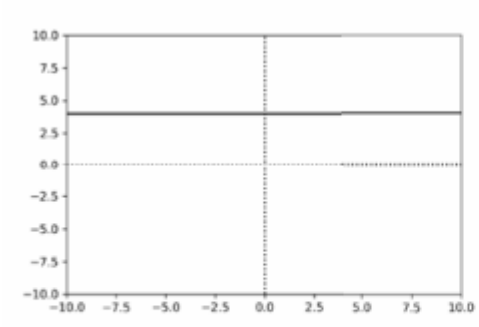




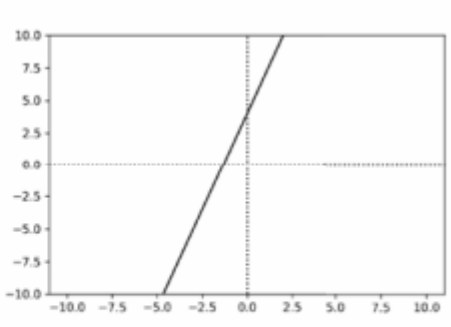
The plot of the root mean square error for our running example. Notice how the algorithm succeeded in reducing this error after a little over 1,000 iterations. This means that we don't need to keep running this algorithm for 10,000 iterations, because around 2,000 of them do the job.

What if the data is not in a line? Polynomial regression

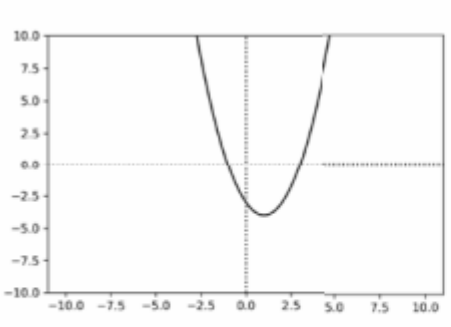
- A special kind of curved functions: Polynomials
- *Polynomials* are a class of functions that are helpful when modeling nonlinear data.
- The *degree* of the polynomial as the exponent of the highest power in the expression of the polynomial.
  - $y = 4$
  - $y = 3x + 2$
  - $y = x^2 - 2x + 5$
  - $y = 2x^3 + 8x^2 - 40$



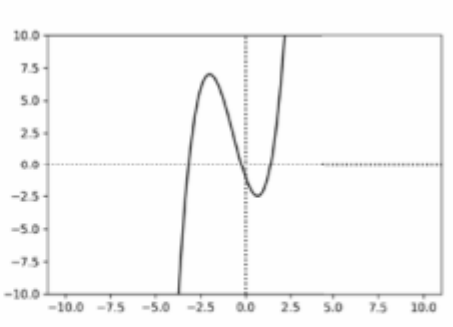
Degree 0
$y = 4$



Degree 1
$y = 3x + 4$



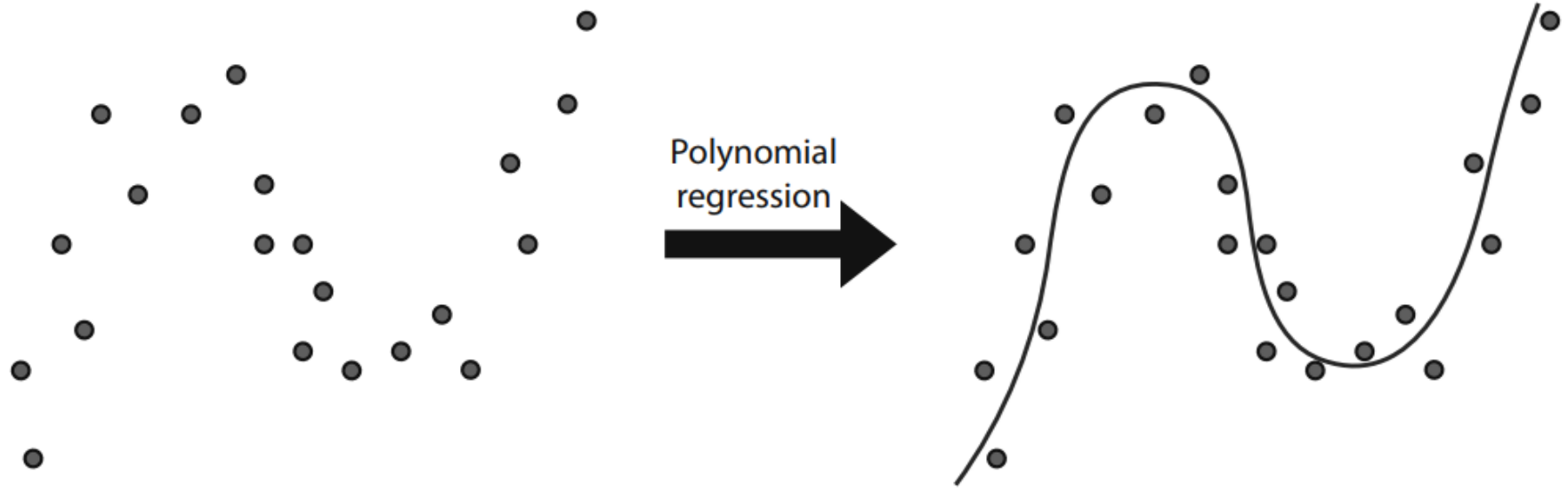
Degree 2
$y = x^2 - 2x - 3$



Degree 3
$y = x^3 + 2x^2 - 4x - 1$

Polynomials are functions that help us model our data better. Here are the plots of four polynomials of degrees 0 to 3. Note that the polynomial of degree 0 is a horizontal line, the polynomial of degree 1 is any line, the polynomial of degree 2 is a parabola, and the polynomial of degree 3 is a curve that oscillates twice.

Nonlinear data? No problem: Let's try to fit a polynomial curve to it



Polynomial regression is useful when it comes to modeling nonlinear data. If our data looks like the left part of the figure, it will be hard to find a line that fits it well. However, a curve will fit the data well, as you can see in the right part of the figure. Polynomial regression helps us find this curve.

Parameters and hyperparameters

- Regression models are defined by their weights and bias—the *parameters* of the model.
- The learning rate, the number of epochs, the degree (if considering a polynomial regression model), and many others are called *hyperparameters*.
- The rule of thumb to avoid confusing:
  - Any quantity that you set *before* the training process is a *hyperparameter*.
  - Any quantity that the model creates or modifies *during* the training process is a *parameter*.

Exercises

Exercise 3.1

A website has trained a linear regression model to predict the amount of minutes that a user will spend on the site. The formula they have obtained is

$$\hat{t} = 0.8d + 0.5m + 0.5y + 0.2a + 1.5$$

where $\hat{t}$ is the predicted time in minutes, and d, m, y, and a are indicator variables (namely, they take only the values 0 or 1) defined as follows:

- d is a variable that indicates if the user is on desktop.
- m is a variable that indicates if the user is on mobile device.
- y is a variable that indicates if the user is young (under 21 years old).
- a is a variable that indicates if the user is an adult (21 years old or older).

Example: If a user is 30 years old and on a desktop, then $d = 1$, $m = 0$, $y = 0$, and $a = 1$.

If a 45-year-old user looks at the website from their phone, what is the expected time they will spend on the site?

Exercise 3.2

Imagine that we trained a linear regression model in a medical dataset. The model predicts the expected life span of a patient. To each of the features in our dataset, the model would assign a weight.

a) For the following quantities, state if you believe the weight attached to this quantity is a positive number, a negative number, or zero. Note: if you believe that the weight is a very small number, whether positive or negative, you can say zero.

1. Number of hours of exercise the patient gets per week
2. Number of cigarettes the patient smokes per week
3. Number of family members with heart problems
4. Number of siblings of the patient
5. Whether or not the patient has been hospitalized

b) The model also has a bias. Do you think the bias is positive, negative, or zero?

Exercise 3.3

The following is a dataset of houses with sizes (in square feet) and prices (in dollars).

	Size (s)	Prize (p)
House 1	100	200
House 2	200	475
House 3	200	400
House 4	250	520
House 5	325	735

Suppose we have trained the model where the prediction for the price of the house based on size is the following:

$$\hat{p} = 2s + 50$$

a. Calculate the predictions that this model makes on the dataset.

b. Calculate the mean absolute error of this model.

c. Calculate the root mean square error of this model.

Exercise 3.4

Our goal is to move the line with equation $\hat{y} = 2x + 3$ closer to the point $(x, y) = (5, 15)$ using the tricks we've learned in this chapter. For the following two problems, use the learning rate $\eta = 0.01$.

a. Apply the absolute trick to modify the line above to be closer to the point.

b. Apply the square trick to modify the line above to be closer to the point.