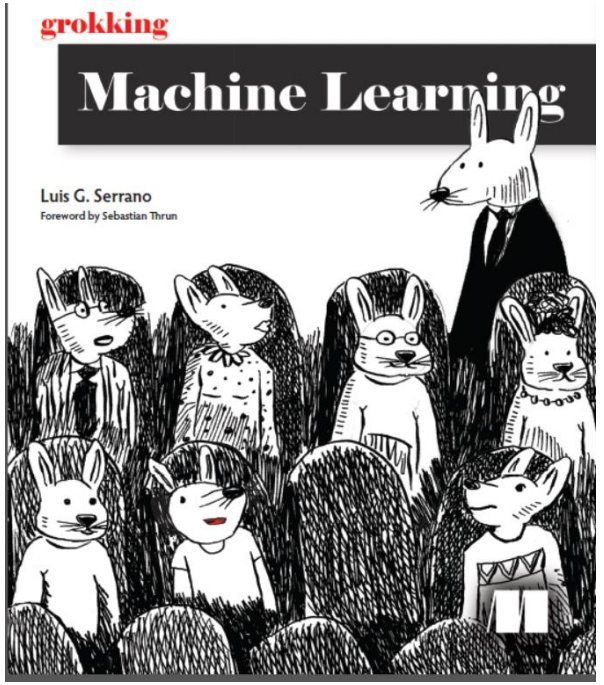
Machine Learning

|Simple – Square – Absolute Trick

Prof. Dr. Mostafa Elhosseini

https://youtube.com/@ProfElhosseiniTechVerse



Agenda

- ~~Linear regression~~
- ~~Def.~~
- ~~Closed-form equation : normal equation~~
- ~~Iterative optimization approach~~
  - ~~Gradient Descent~~
  - ~~Batch~~
  - ~~Stochastic~~
  - ~~Minibatch~~

Predict the price of a house

- **Features**: Properties that we use to make our prediction
  - # rooms in the house, the crime rate, the age of the house, the size, and so on
- **Labels**: This is the target that we try to predict from the features.
- **Model**: A machine learning model is a **rule**, or a **formula**, which predicts a label from the features.
- **Prediction**: The prediction is the **output** of the model

Predict the price of a house

- Weights: In the formula corresponding to the model, **each feature is multiplied** by a corresponding **factor**. These factors are the weights.
  - Price = 100 + **50**(Number of rooms)
  - The price per room
- **bias**: The formula corresponding to the model has a constant that is not attached to any of the features. This constant is called the bias.
  - Price = **100** + 50(Number of rooms)
  - It corresponds to the **base price of a house**.

Equation of a Line

- **Slope**: The slope of a line is a **measure of how steep it is**.
- It is calculated by **dividing the rise over the run**
- How many units it goes up, divided by how many units it goes to the right.
- This ratio is constant over the whole line.
- In a machine learning model, this is the **weight** of the corresponding feature, and it tells us how much we expect the value of the label to go up, when we increase the value of the feature by one unit.
- If the line is **horizontal**, then the slope is **zero**, and if the line goes **down**, the slope is **negative**.

Equation of a Line

- **y-intercept**: The y-intercept of a line is the **height** at which the line **crosses** the vertical (**y**-) axis.
- In a machine learning model, it is the **bias** and tells us what the label would be in a data point where all the **features are precisely zero**.

Equation of a Line

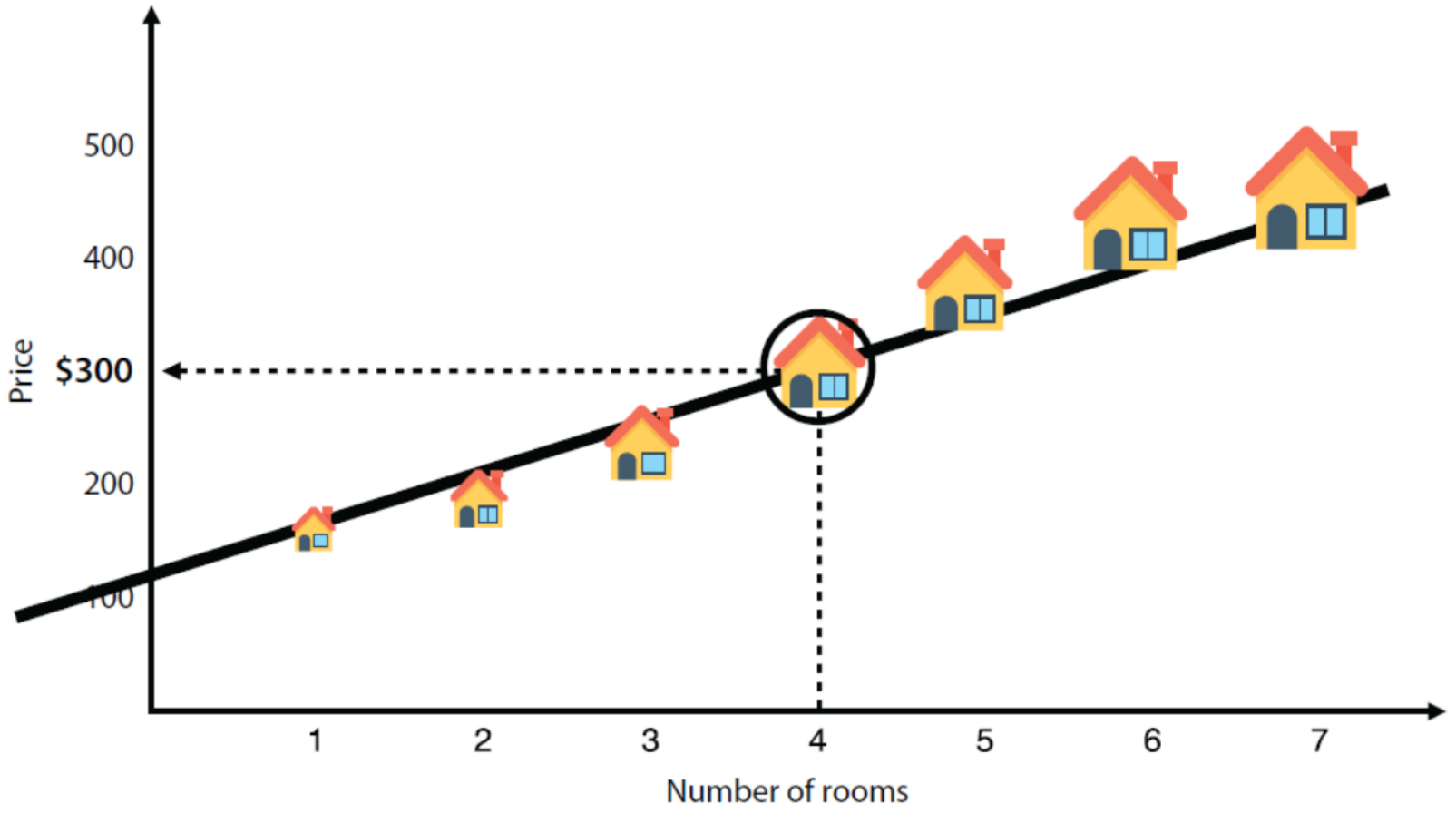
- **linear equation**: This is the **equation of a line**.
- It is given by two parameters: the **slope** and the **y-intercept**.
- If the slope is **m** and the y-intercept is **b**, then the equation of the line is **y = mx + b**, and the line is formed by all the points **(x,y)** that satisfy the equation.
- In a machine learning model, **x** is the value of the **feature** and **y** is the **prediction** for the **label**.
- The **weight** and **bias** of the model are **m** and **b**, respectively.

Equation of a Line

- When we say that the **slope of the line is 50**—this means that each time we **add one room** to the house, we estimate that the **price** of the house **will go up by $50**.
- When we say that the **y-intercept** of the line is **100**, this means that the estimate for the price of a (hypothetical) house with **zero rooms** would be the base price of **$100**

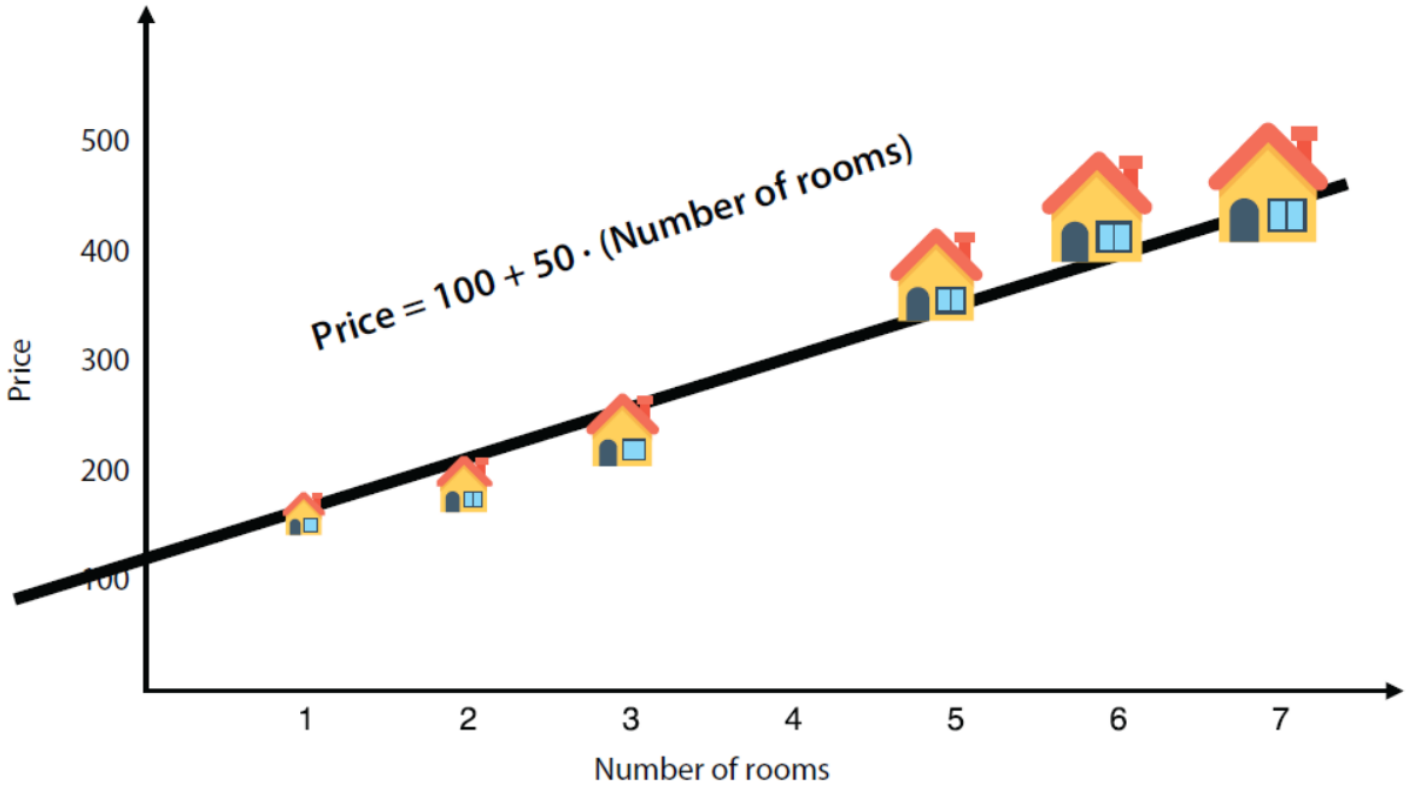
Equation of a Line

Equation of a Line



How to get The Equation of a Line

- Why did we pick this one in particular?
- How do we find this line?



How to get The Equation of a Line

- How did you come up with the formula that predicts the price? And what would we do if instead of six houses, we had thousands of them?
- When we have **six houses**, the problem of **drawing a line** that goes close to them is **simple**, but if we have **thousands of houses**, this task gets hard. What we do in this chapter is **devise an algorithm**, or a **procedure**, for the computer to find a good line

The linear regression algorithm

- Step by step.
- Start with a **random line**, and
  - Figure out a way to **improve** this line a little bit by moving it closer to the points.
- **Repeat** this process many times, we have the desired line.

The linear regression algorithm

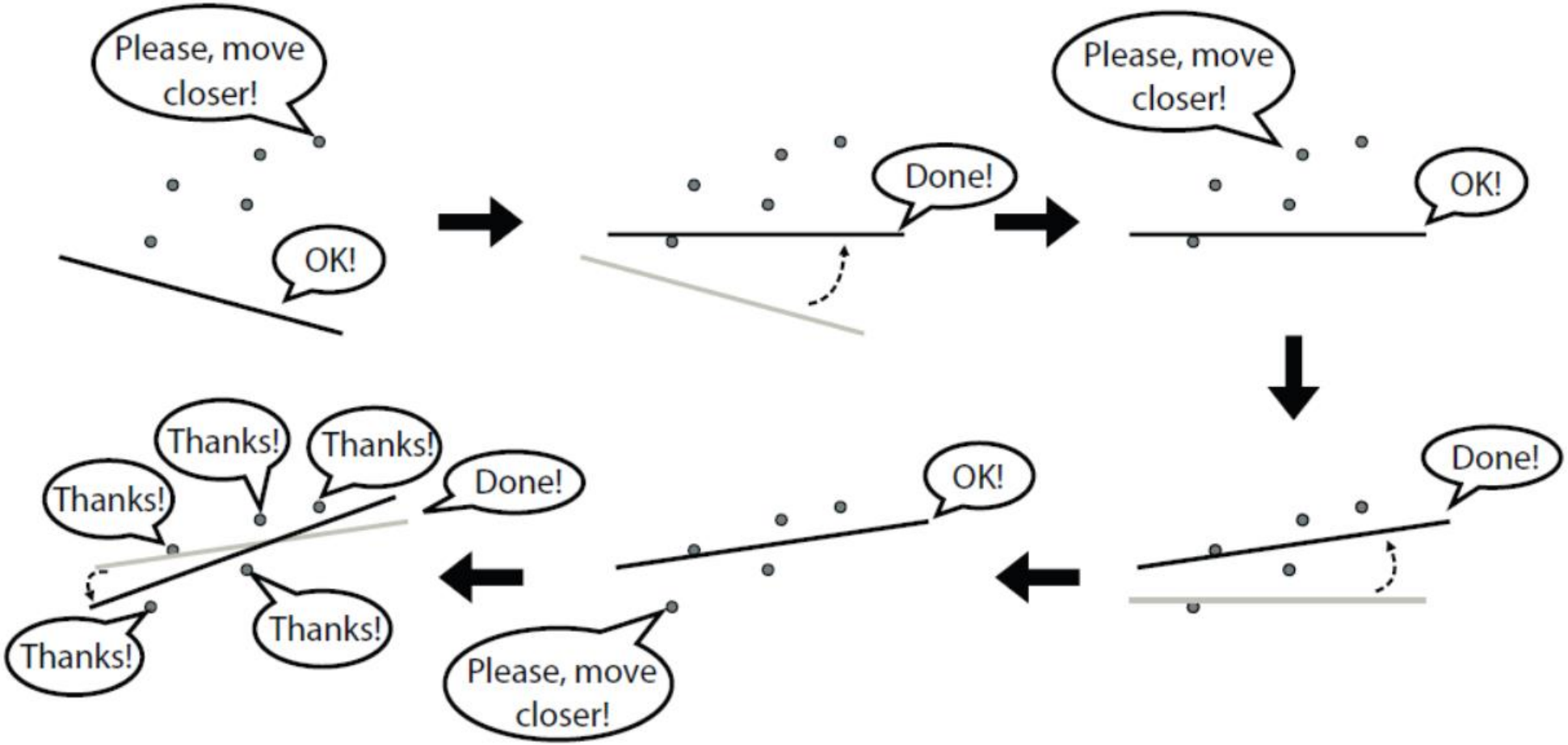
Inputs: A dataset of points in the plane

Outputs: A line that passes close to the points

Procedure:

- Pick a random line.
- Repeat many times:
  - Pick a random data point.
  - Move the line a little closer to that point.
- **Return** the line you've obtained.

The linear regression algorithm



The linear regression algorithm

Let's begin by defining some variables.

- p: The price of a house in the dataset (**Actual** Price)
- $\hat{p}$: The **predicted** price of a house
- r: The number of rooms
- m: The price per room (Slope – Weight)
- b: The base price for a house (Bias)

- $\hat{p} = \mathrm{m} * \mathrm{r} + \mathrm{b}$

The linear regression algorithm

- $\hat{p} = \mathrm{m} * \mathrm{r} + \mathrm{b}$
- To get an idea of the linear regression algorithm, imagine that we have a model in which the **price per room is $40** and the **base price** of the house is **$50**. This model predicts the price of a house using the following formula:
- $\hat{p} = 40 * \mathrm{r} + 50$
- A house with **two rooms** that costs **$150**. This model **predicts** that the price of the house is $50 + 40 \cdot 2 =$ **130**.
- That is **not a bad prediction**, but **it is less than** the price of the house. **How can we improve** the model?

The linear regression algorithm

- $\hat{p} = \mathrm{m} * \mathrm{r} + \mathrm{b}$
- It seems like the model's mistake is thinking that the house is too cheap. **Maybe** the model has a **low base price**, or maybe it has a **low price per room**, or maybe **both**. If we increase both by a small amount, we may get a better estimate. Let's increase the price per room by $0.50 and the base price by $1
- $\hat{p} = 40.5 * \mathrm{r} + 51$
- The new predicted price for the house is $40.5 \cdot r + 51 = 132$
- Therefore, it is a better model for that data point. We don't know if it is a better model for the other data points but let's not worry about that for now. The idea of the linear regression algorithm is to repeat the previous process many times

The linear regression algorithm

Inputs: A dataset of points

Outputs: A linear regression model that fits that dataset

Procedure:

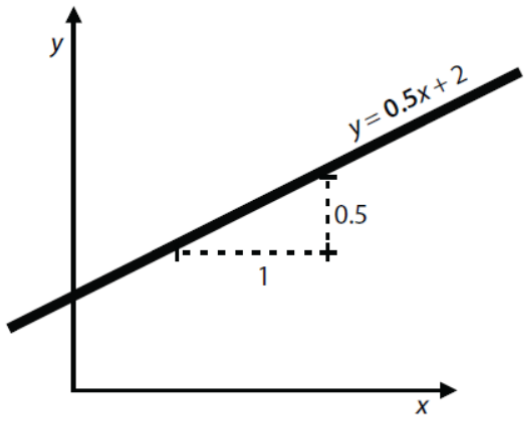
- Pick a model with random weights and a random bias.
- Repeat many times:
  - Pick a random data point.
  - Slightly adjust the weights and bias to improve the prediction for that particular data point.
- Return the model you've obtained.

- By how much should I adjust the weights?
- How many times should I repeat the algorithm? In other words, how do I know when I'm done?
- How do I know that this algorithm works?

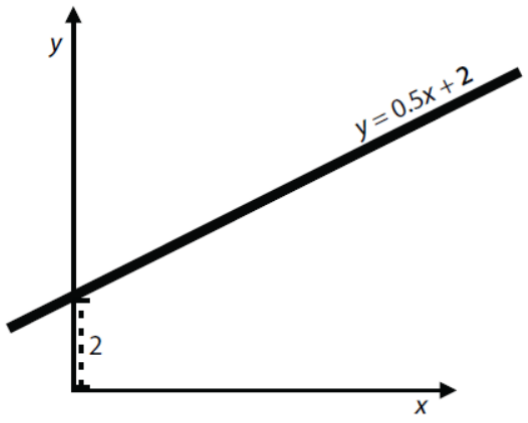
Slope and y-intercept

- How to manipulate this equation $\hat{p} = m * r + b$ to move our line
- When we say that the slope is 0.5, it means that when we walk along this line, for every unit that we move to the right, we are moving 0.5 units up.
- The slope can be zero if we don't move up at all or negative if we move down

$y = 0.5x + 2$



Slope = 0.5

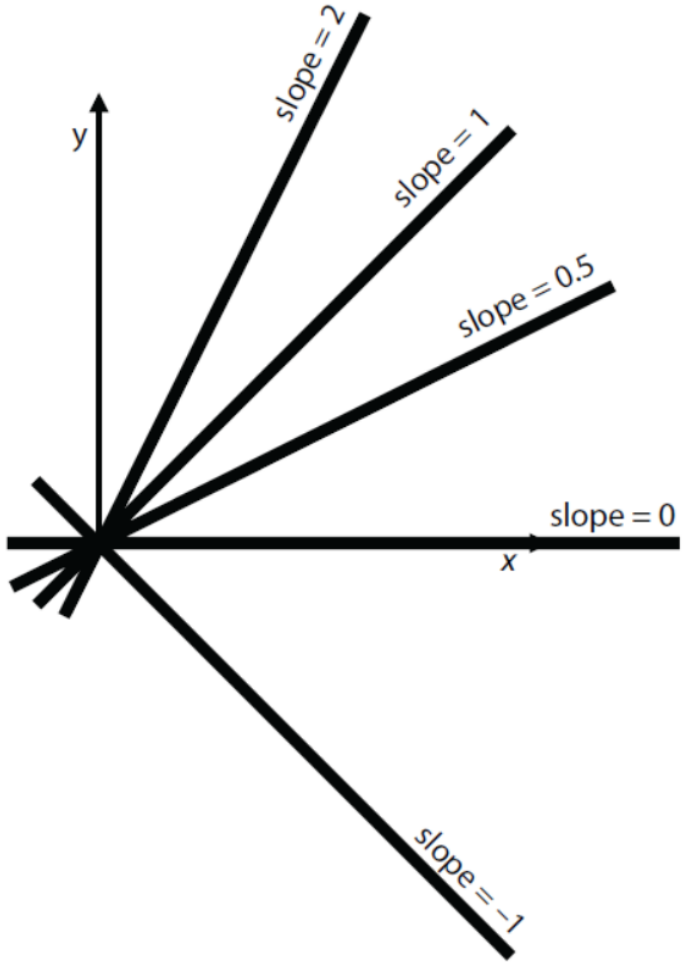


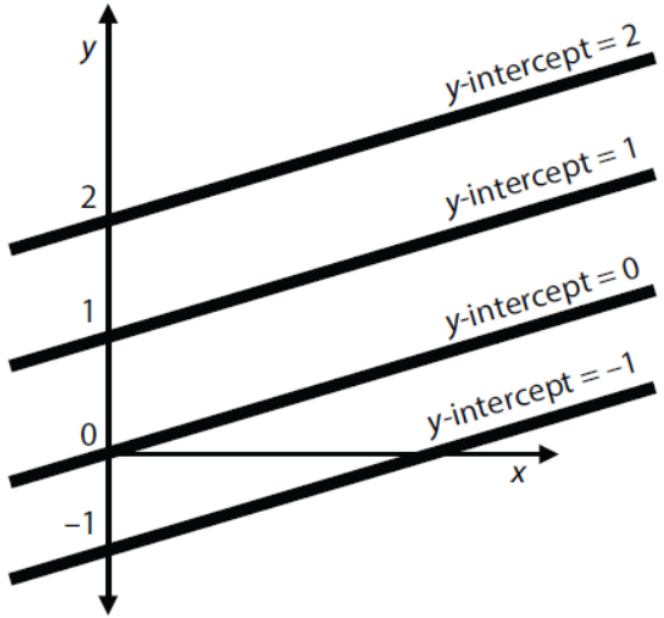
y-intercept = 2

Slope and y-intercept

- The **slope** of the line tells us the **direction in which the line is pointing**, and the **y-intercept** tells us **the location of the line**.
- Notice that by specifying the slope and the y-intercept, the line is **completely specified**

Slope and y-intercept



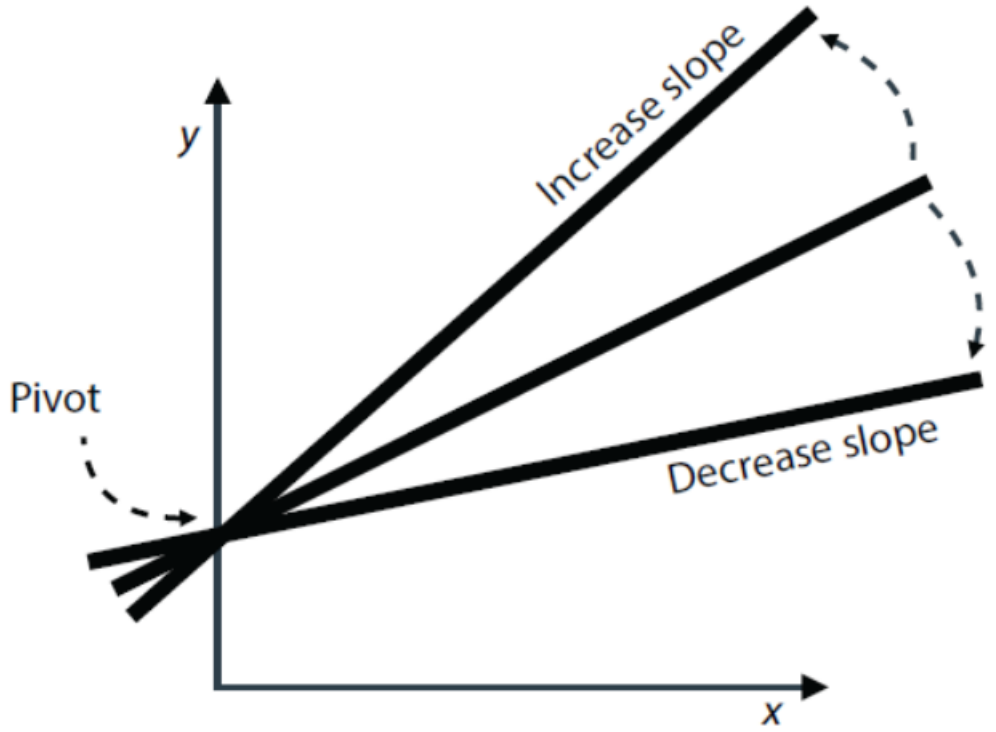


- **slope** represents the price per room,
- **y-intercept** represents the base price of a house

Slope and y-intercept

Changing the slope:

- If we increase the slope of a line, the line will rotate counterclockwise.
- If we decrease the slope of a line, the line will rotate clockwise.

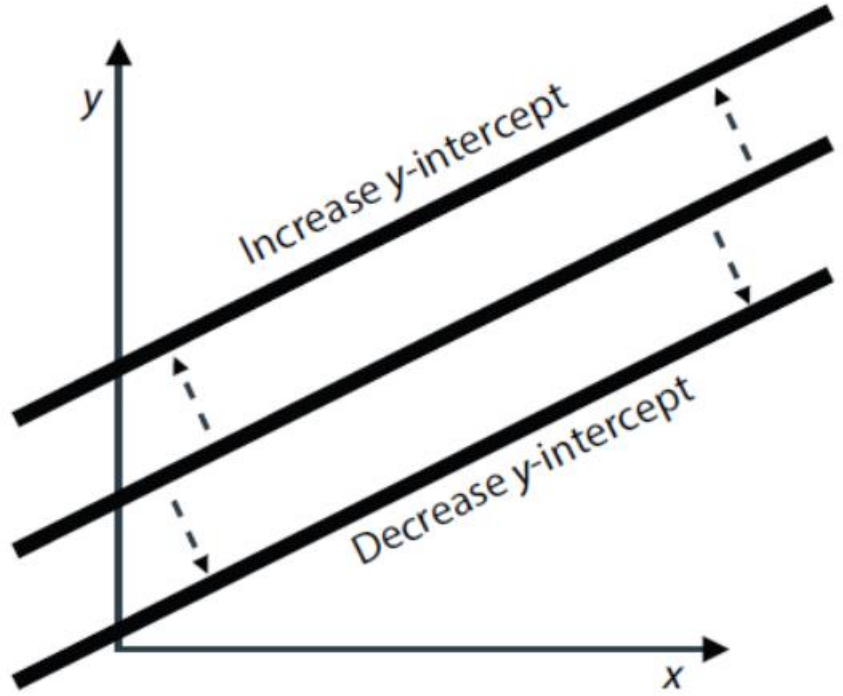


Rotate clockwise and counterclockwise

Slope and y-intercept

Changing the y-intercept:

- If we increase the y-intercept of a line, the line is translated upward.
- If we decrease the y-intercept of a line, the line is translated downward.



Translate up and down

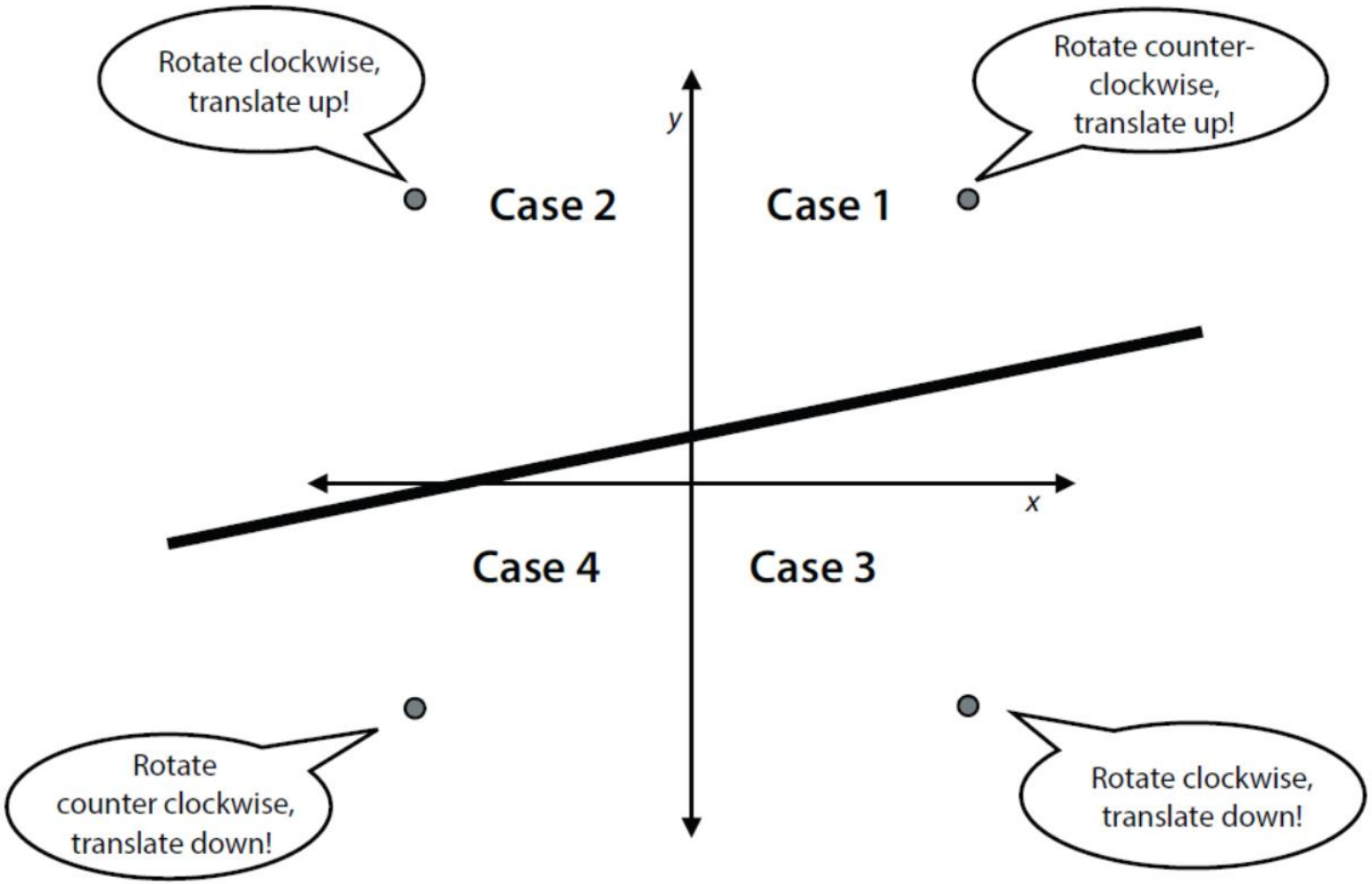
Simple trick

- Consists of slightly **rotating and translating** the line in the direction of the point to move it closer
- If the point is **above the line**, we need to translate the **line up**, and if it is below, we need to translate it down
- if the point is above the line and to the right of the y-axis, or below the line and to the left of the y-axis, we need to rotate the line counterclockwise. In the other two scenarios, we need to rotate the line clockwise

Simple trick

- **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.
- **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.
- **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.
- **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.

Simple trick



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 $b' = b + \eta_2$.

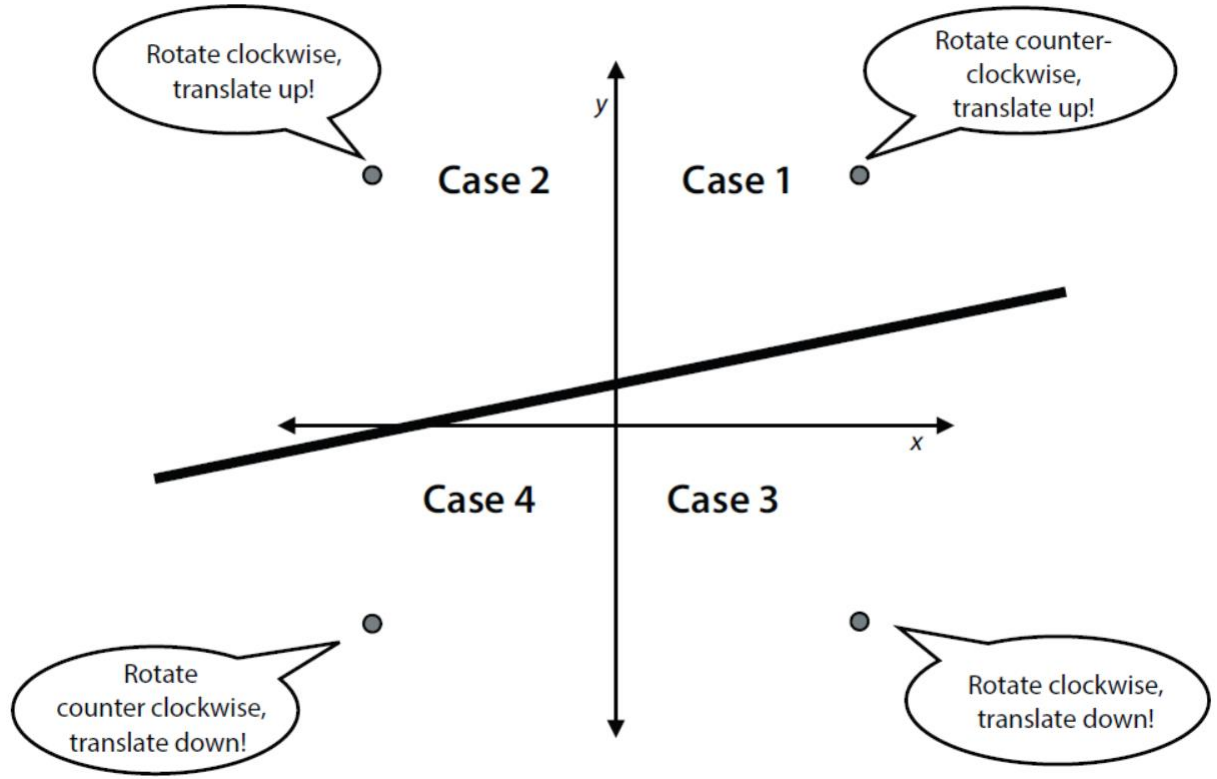
Case 3: If the point is below the line and to the right of the y-axis, we rotate the line clockwise and translate it downward:

- 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 $\hat{p} = m'r + b'$.



Simple trick - Notes

- Adding or Subtracting a small number to the slope means increasing or decreasing the price per room.
- Similarly, adding or subtracting a small number to the y-intercept means increasing or decreasing the base price of the house.
- Furthermore, because the **x-coordinate** is the **number of rooms**, this number is **never negative**. Thus, only **cases 1 and 3 matter** in our example

Simple trick - Notes

- Furthermore, because the **x-coordinate** is the **number of rooms**, this number is **never negative**. Thus, only **cases 1 and 3 matter** in our example
  - If the model gave us a price for the house that is lower than the actual price, add a small random amount to the price per room and to the base price of the house.
  - If the model gave us a price for the house that is higher than the actual price, subtract a small random amount from the price per room and the base price of the house.

Simple trick - Notes

- This trick achieves some success in practice, but it's far from being the best way to move lines.

<u>Some questions may arise, such as the following</u>:

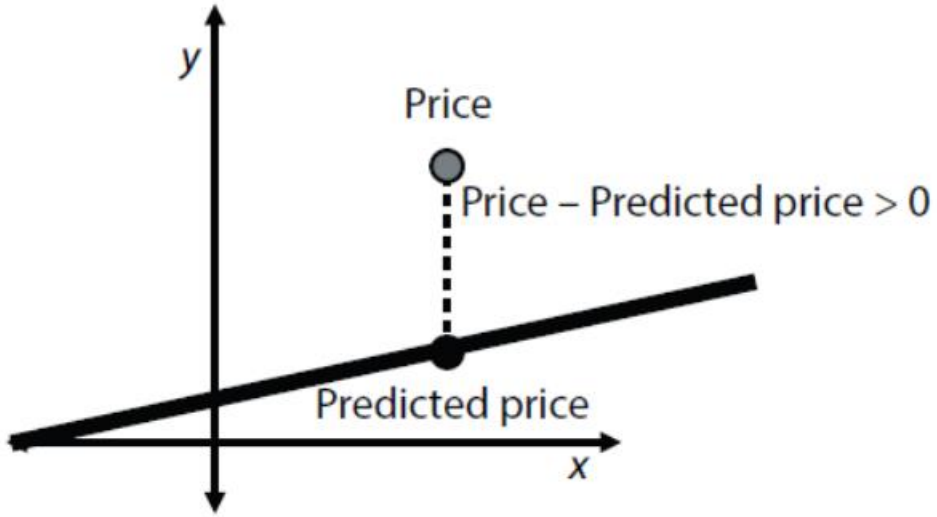
- Can we pick better values for η_1 and η_2 ?
- Can we crunch the four cases into two, or perhaps one?

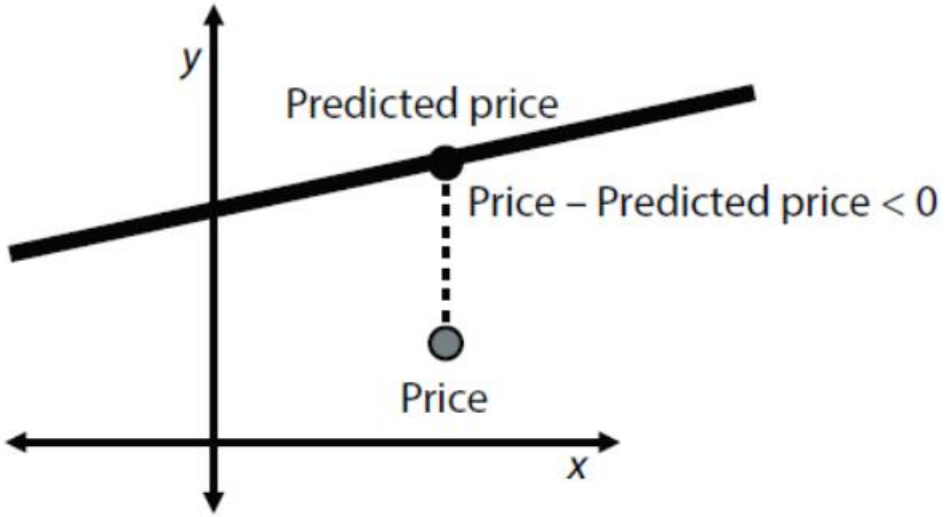
Square trick

- An effective way to move a line closer to a point
- The square trick will bring these four cases down to one by finding values with the correct signs (+ or –) to add to the slope and the y-intercept for the line to always move closer to the point
- In the **simple trick**, when the point is above the line, we add a small amount to the y-intercept. When it is below the line, we subtract a small amount.

Square trick

- If a point is above the line, the value $p - \hat{p}$ (the difference between the Actual price and the predicted price) is positive. If it is below the line, this value is negative





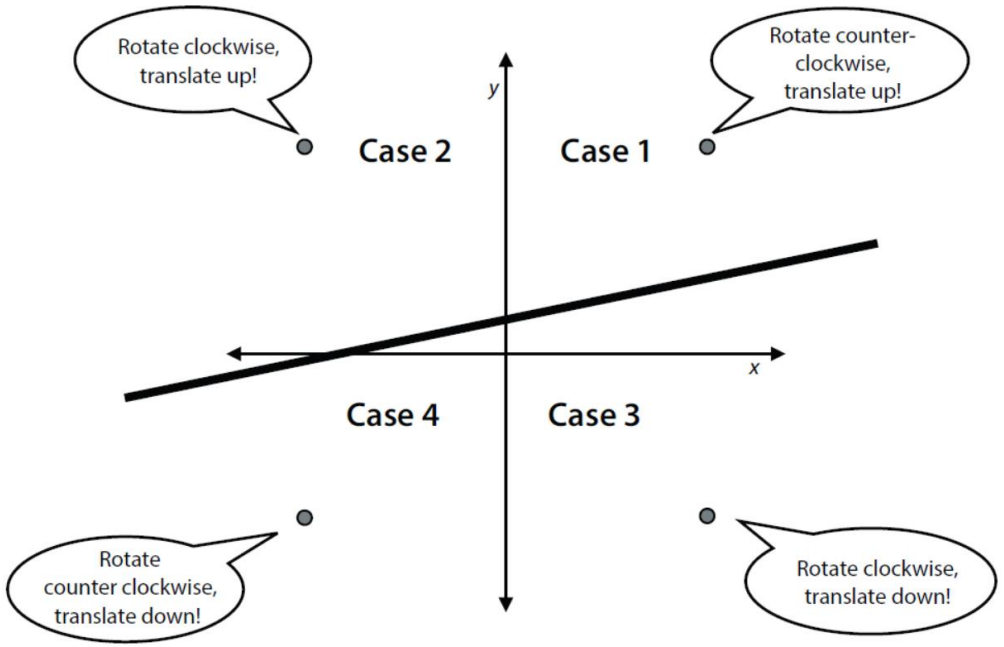
Square trick

- we conclude that if we add the difference $p - \hat{p}$ to the y-intercept, the line will always move toward the point
- However, in machine learning, we always want **to take small steps**. To help us with this, we introduce an important concept in machine learning: the **learning rate**.

- **learning rate:** A very small number that **we pick before training** our model. This number helps us make sure our model changes in very small amounts by training. (η - Greek letter eta).
- Add $\eta(p - \hat{p})$

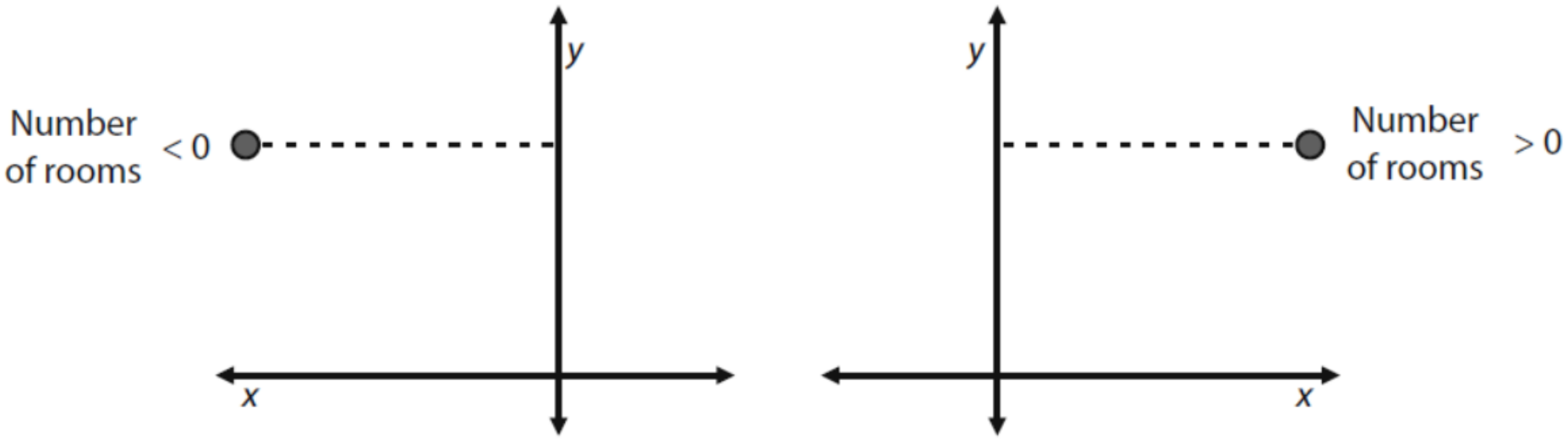
Square trick

- In the **simple trick**, when the point is in **scenario 1 or 4** (above the line and to the right of the vertical axis, or below the line and to the left of the vertical axis), we rotate the line **counterclockwise**. Otherwise (scenario **2 or 3**), we rotate it **clockwise**



Square trick

- If a point (r, p) is to the right of the vertical axis, then r is positive. If the point is to the left of the vertical axis, then r is negative



Square trick

- The value $\boldsymbol{r}(\boldsymbol{p} - \hat{\boldsymbol{p}})$. This value is positive when both r and $\boldsymbol{p} - \hat{\boldsymbol{p}}$ are both positive or both negative. This is precisely what occurs in scenarios **1 and 4**.
- Similarly, $\boldsymbol{r}(\boldsymbol{p} - \hat{\boldsymbol{p}})$ is negative in scenarios **2 and 3**.
- Therefore, due to observation 4, this is the quantity that we need to add to the slope
- Adding $\boldsymbol{\eta r}(\boldsymbol{p} - \hat{\boldsymbol{p}})$ to the slope will always move the line in the direction of the point

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 $\hat{p} = m'r + b'$

Square trick - Code

- https://colab.research.google.com/drive/1GvRGWKezxzUFov8mz3X8RYFjW8IglbhA

Square trick - Code

- https://colab.research.google.com/drive/1GvRGWKezxzUFov8mz3X8RYFjW8IglbhA

```
def square_trick(base_price, price_per_room, num_rooms, price, learning_rate):
    predicted_price = base_price + price_per_room*num_rooms
    price_per_room += learning_rate*num_rooms*(price-predicted_price)
    base_price += learning_rate*(price-predicted_price)
    return price_per_room, base_price
```

Absolute trick

- 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

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 $\hat{p} = m'r + b'$

Linear Regression Algorithm

- Repeating the absolute or square trick many times to move the line closer to the points

Inputs:

- A dataset of houses with number of rooms and prices

Outputs:

- Model weights: price per room and base price

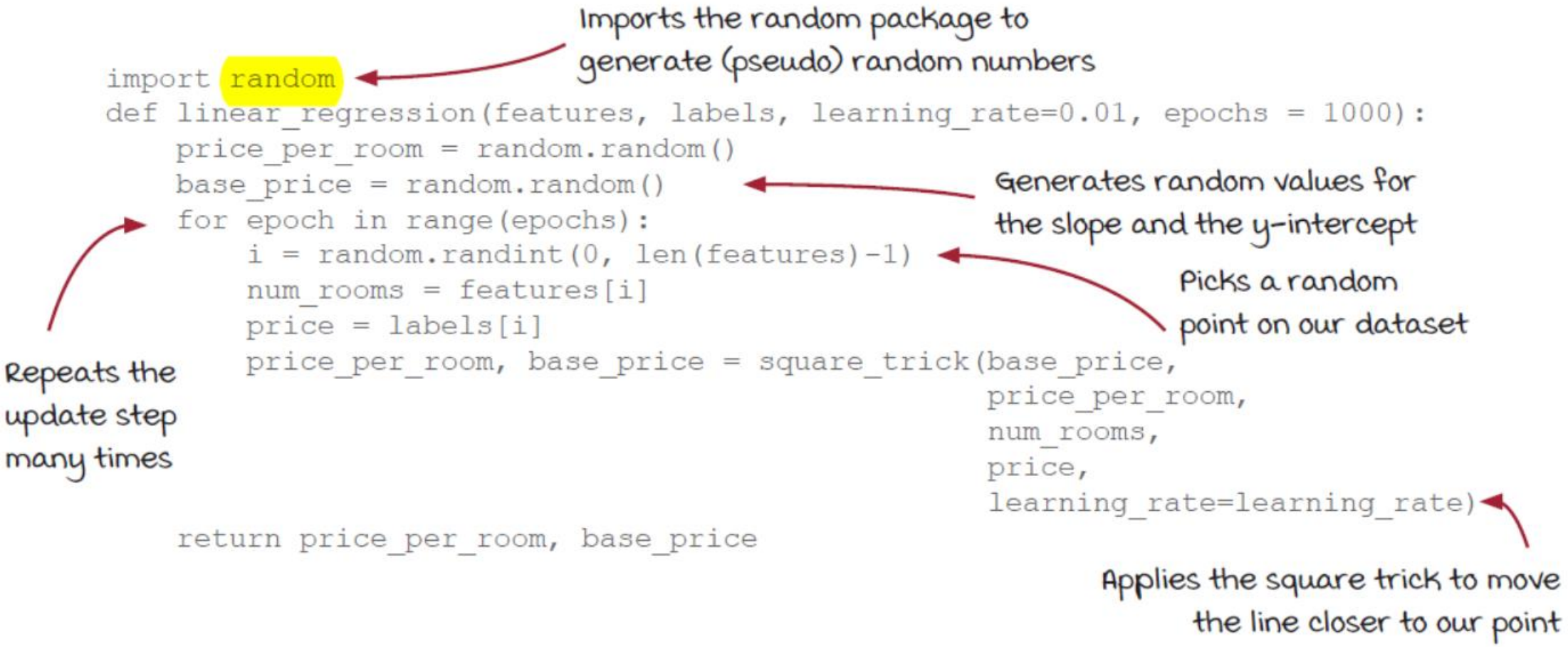
Procedure:

- 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.

Linear Regression Algorithm

- Each iteration of the loop is called an epoch, and we set this number at the beginning of our algorithm.
- The simple trick was mostly used for illustration. it doesn’t work very well.
- In real life, we **use the absolute or square trick**, which works a lot better.
- In fact, although both are commonly used, the **square trick is more popular**. Therefore, we’ll use that one for our algorithm, but feel free to use the absolute trick if you prefer.

Linear Regression Algorithm



The general linear regression algorithm

- In real life, we will be working with datasets with many features
- The only difference is that **each of the features** is updated in the **same way that the slope** was updated.
- In the **housing example**, we had **one slope and one y-intercept**. In the **general case**, think of **many slopes** and **still one y-intercept**.

The general linear regression algorithm

- Dataset of m points and n features. Thus, the model has n weights (think of them as the generalization of the slope) and one bias
- The data points are $x^{(1)}, x^{(2)}, \dots, x^{(m)}$.
- Each point is of the form $\boldsymbol{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)})$
- The corresponding labels are $y_1, y_2, \dots, y_m$.
- The weights of the model are $w_1, w_2, \dots, w_n$.
- The bias of the model is b.

The general linear regression algorithm

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 $\hat{y} = w_1' x_1 + w_2' x_2 + \cdots + w_n' x_n + b'$

Next Lecture

- ~~Polynomial regression~~