Gradient descent: intuition

on the example of linear regression

Lecture 15

by Marina Barsky

Iterative solution to Linear regression

Why do we need another algorithm? We already have a closed-form solution

- The gradient descent is used in many other Machine Learning algorithms
- It is useful to look at it at the very basic example of linear regression

Gradient is *a partial derivative* of a function that shows how fast the function grows and in which direction (*descent*)

Fitting the best line as optimization problem

1. We start with the random line

2. We compute *SSR*

3. We adjust the parameters of the line **into the direction of gradient descent**

Example: optimize intercept only (b)

We set the slope to a constant: say 0.65

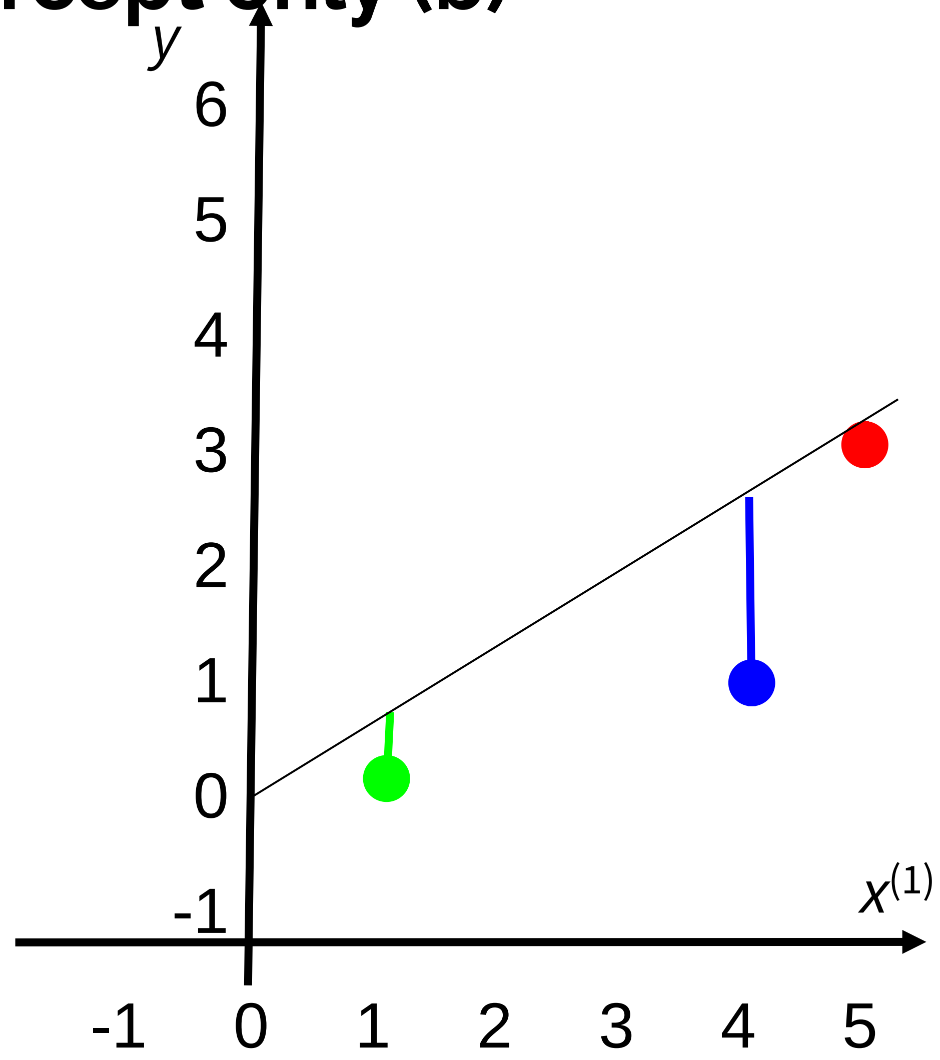
Let's say our first guess is the line:

f(x) = 0.65x + 0

We evaluate how well the line fits the data with SSR as before

The sum of squared errors is called a *loss function*

The optimization task: *minimize the loss function* by learning a better intercept



Data:{[1,0], [4,1], [5,3]}

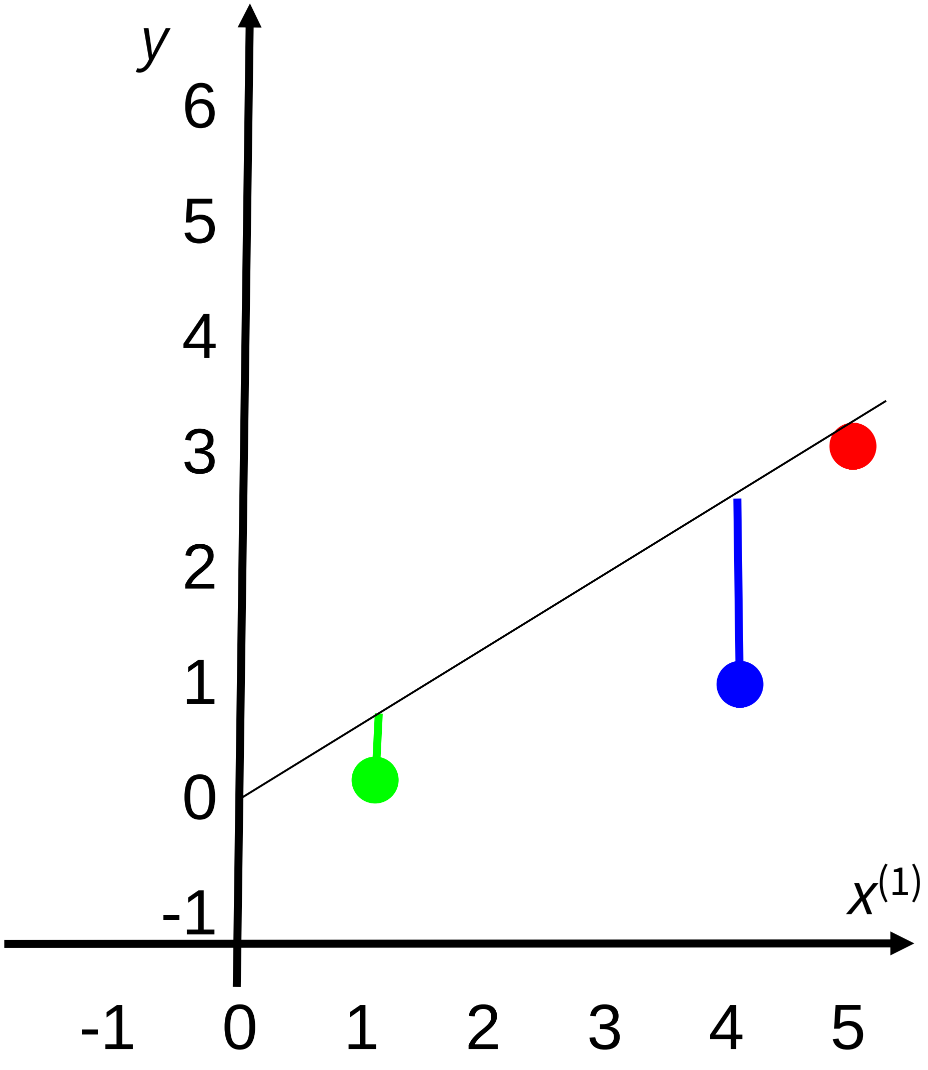
Compute the error

$f(x) = 0.65x + 0$

Point x_1 error:

Predicted: 0.65, actual: 0

error $(0-0.65)^2 = 0.1225$



E: 0.1225+

Data:{[1,0], [4,1], [5,3]}

Compute the error

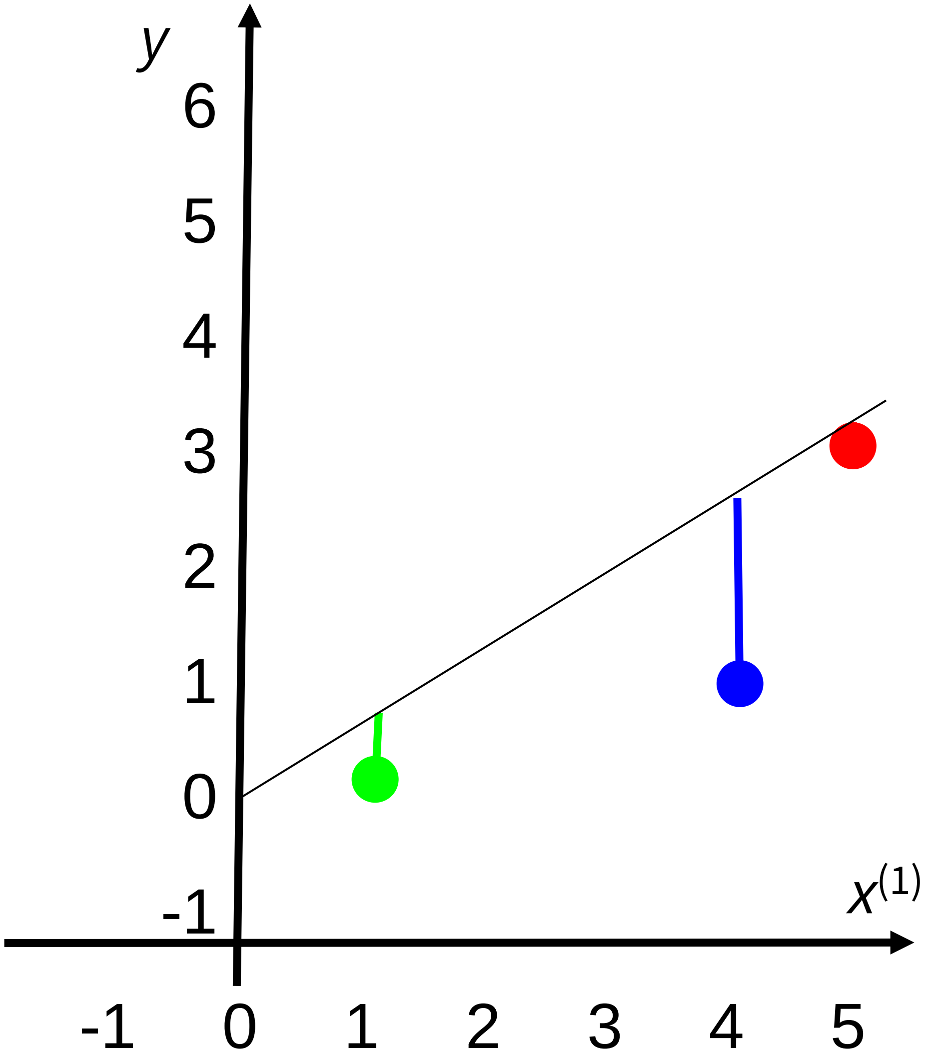
f(x) = 0.65x + 0

Point x_2 error:

Predicted: 2.6, actual: 1

error $(1-2.6)^2$ =2.56

E: 0.1225+2.56



Data:{[1,0], [4,1], [5,3]}

Compute the error

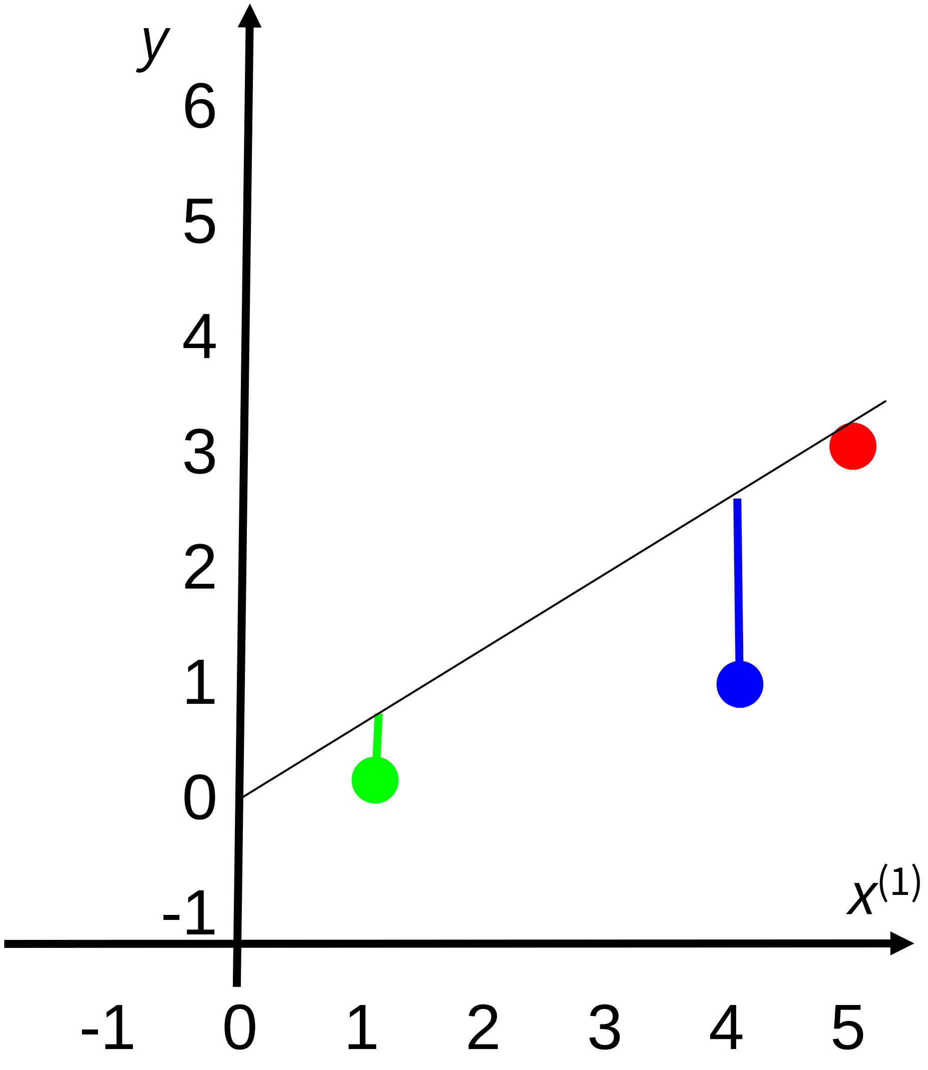
$f(x) = 0.65x + 0$

Point x_3 error:

Predicted: 3.25, actual: 3

error $(3-3.25)^2 = 0.0625$

E: 0.1225+2.56+0.0625=**2.745**



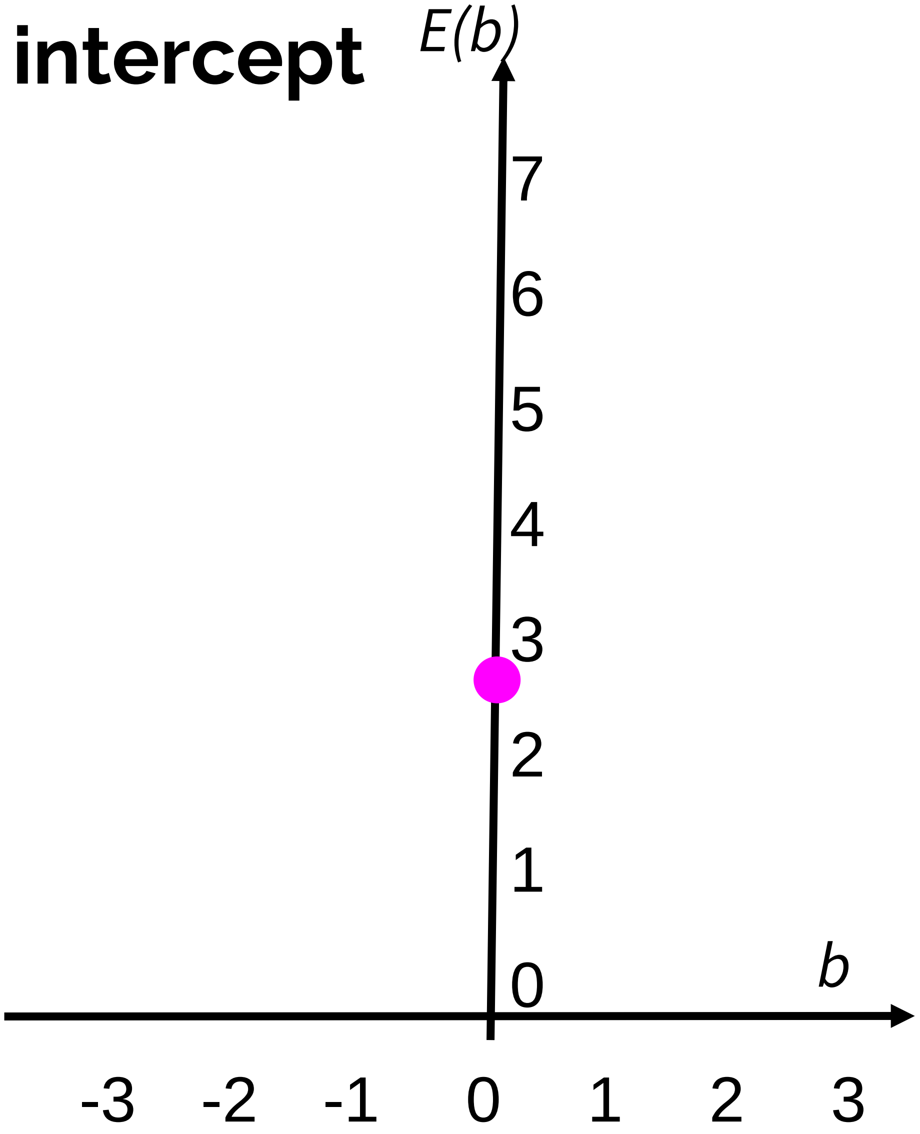
Data:{[1,0], [4,1], [5,3]}

Error as function of the intercept

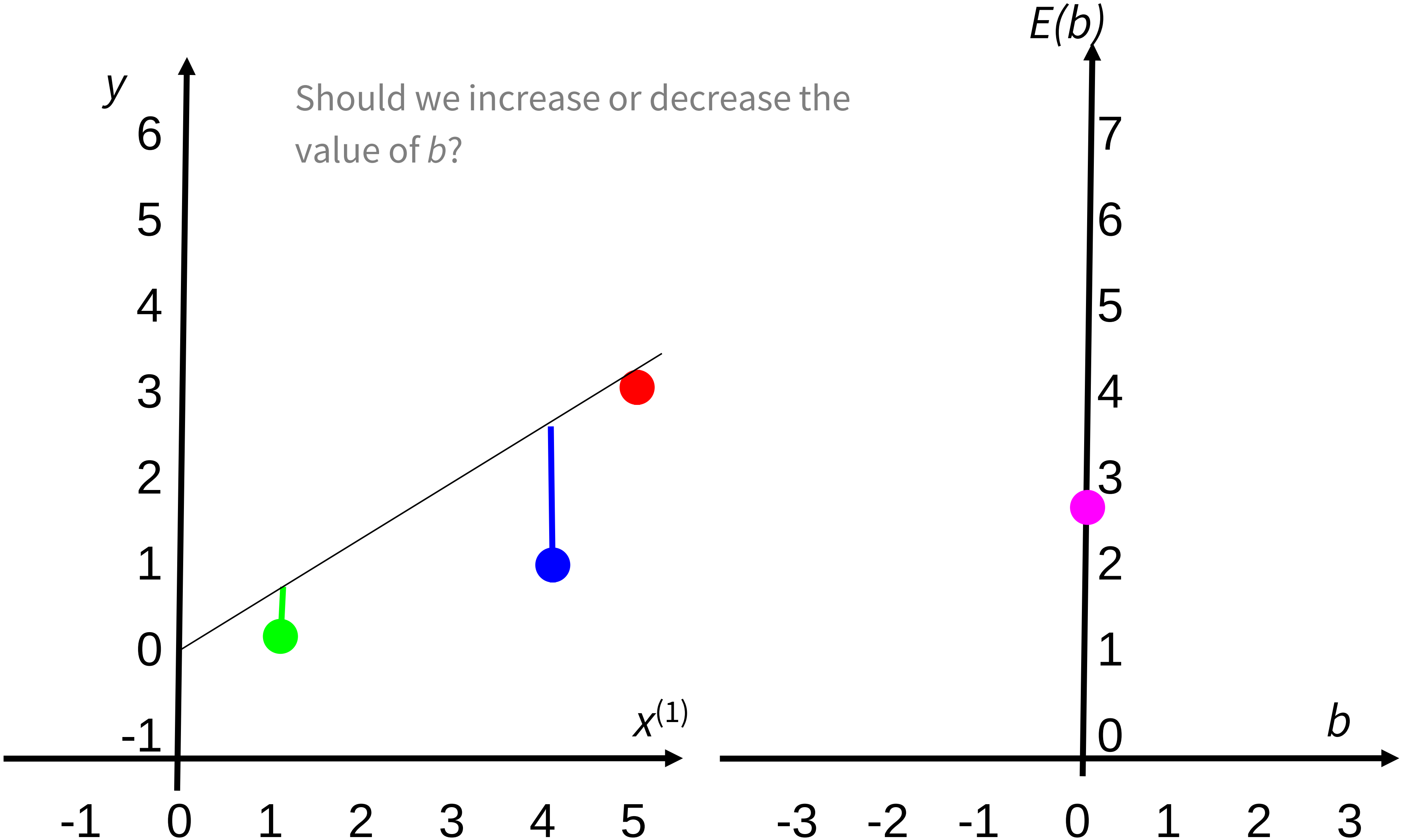
Let's plot the value of *E* as a function of the intercept b:

$E = f(b)$

E(**0**): 0.1225+2.56+0.0625=**2.745**

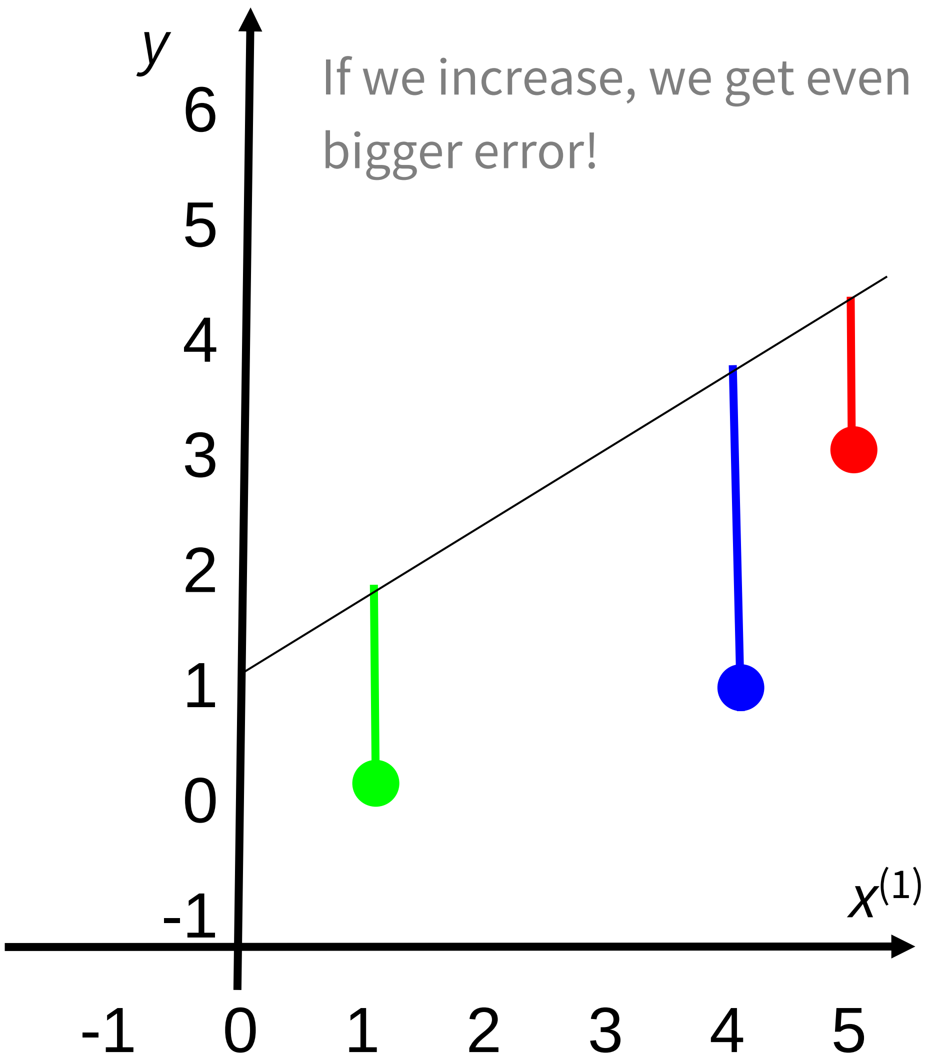


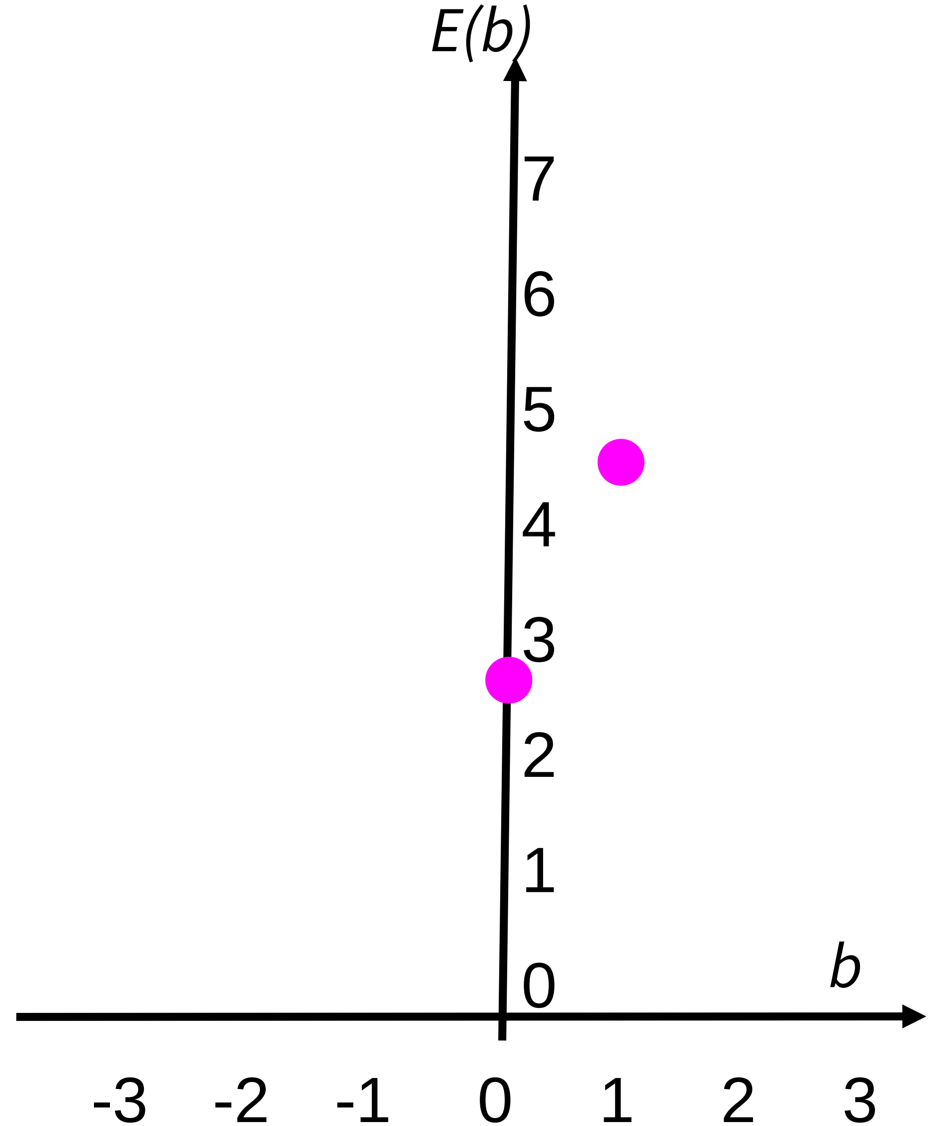
Where to go from here to make *E* smaller?



Data:{[1,0], [4,1], [5,3]}

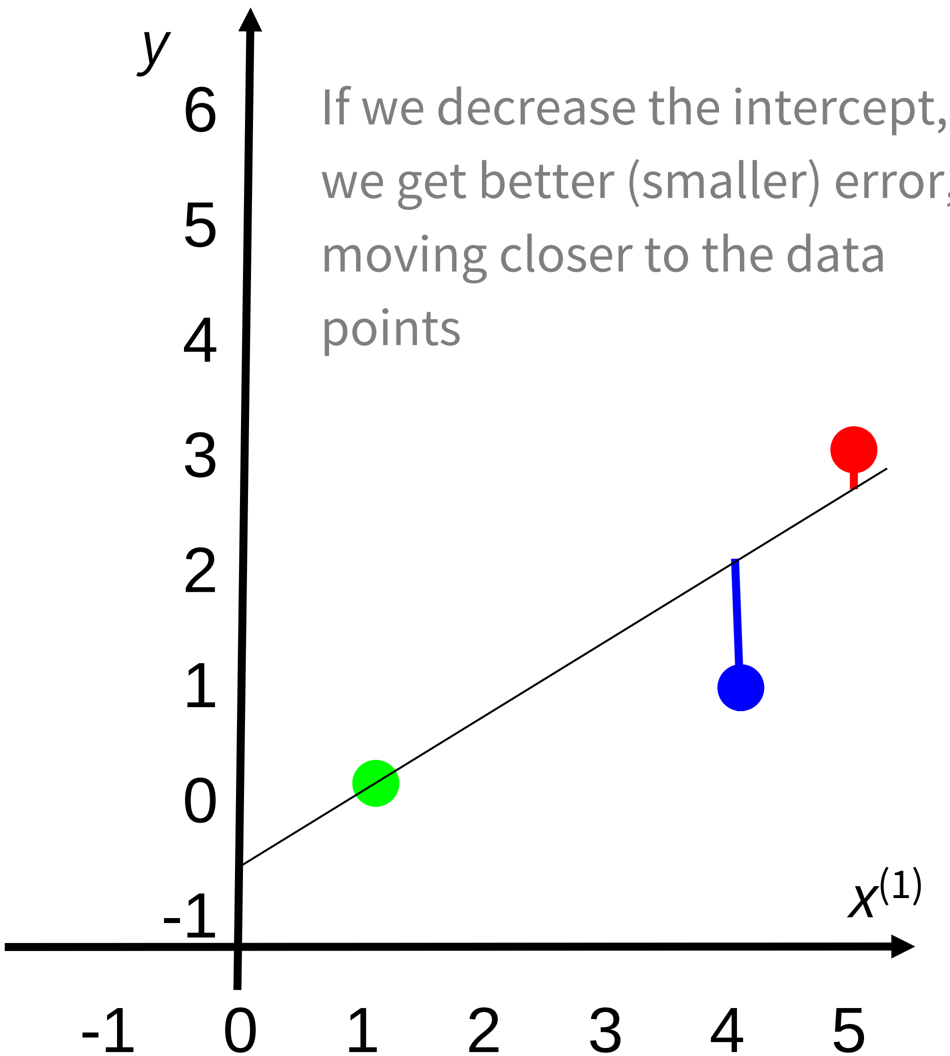
Where to go from here?



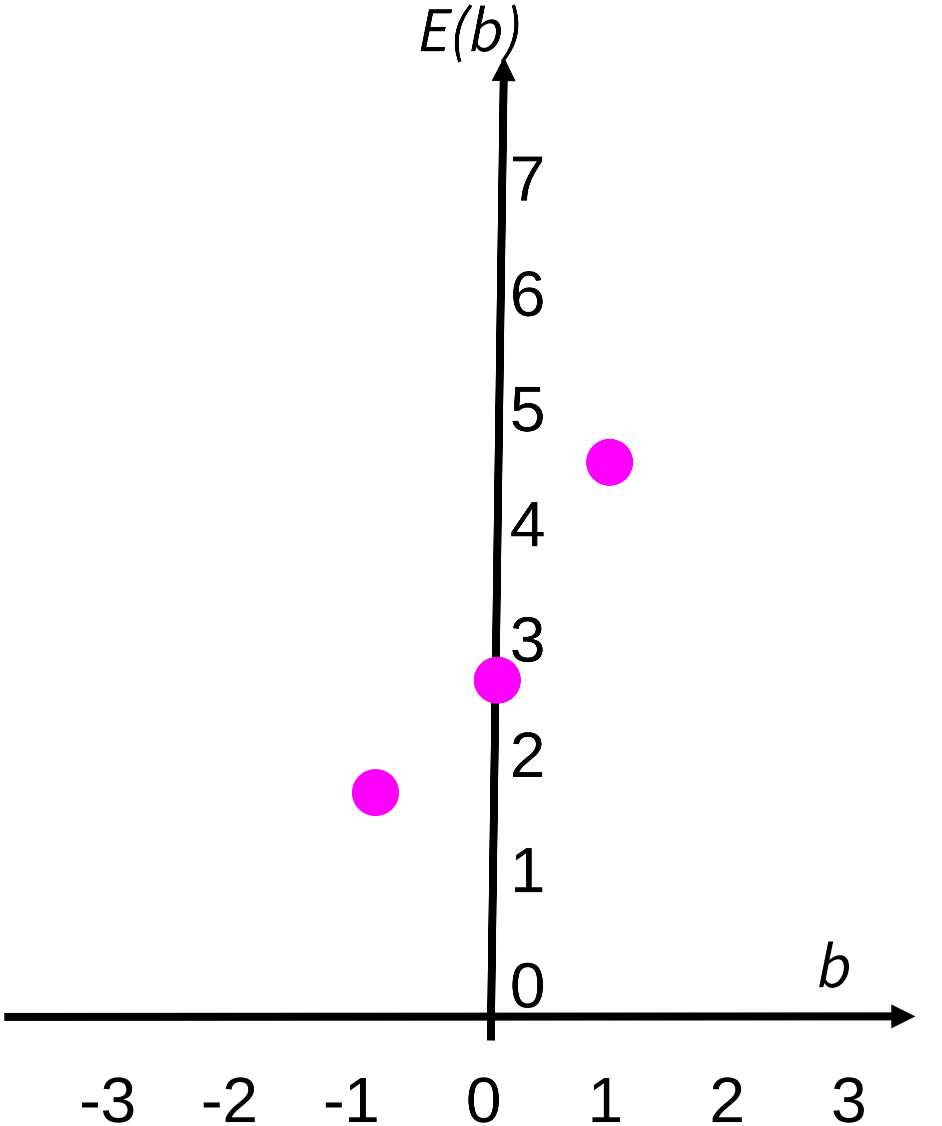


Data:{[1,0], [4,1], [5,3]}

Where to go from here?



If we decrease the intercept, we get better (smaller) error, moving closer to the data points

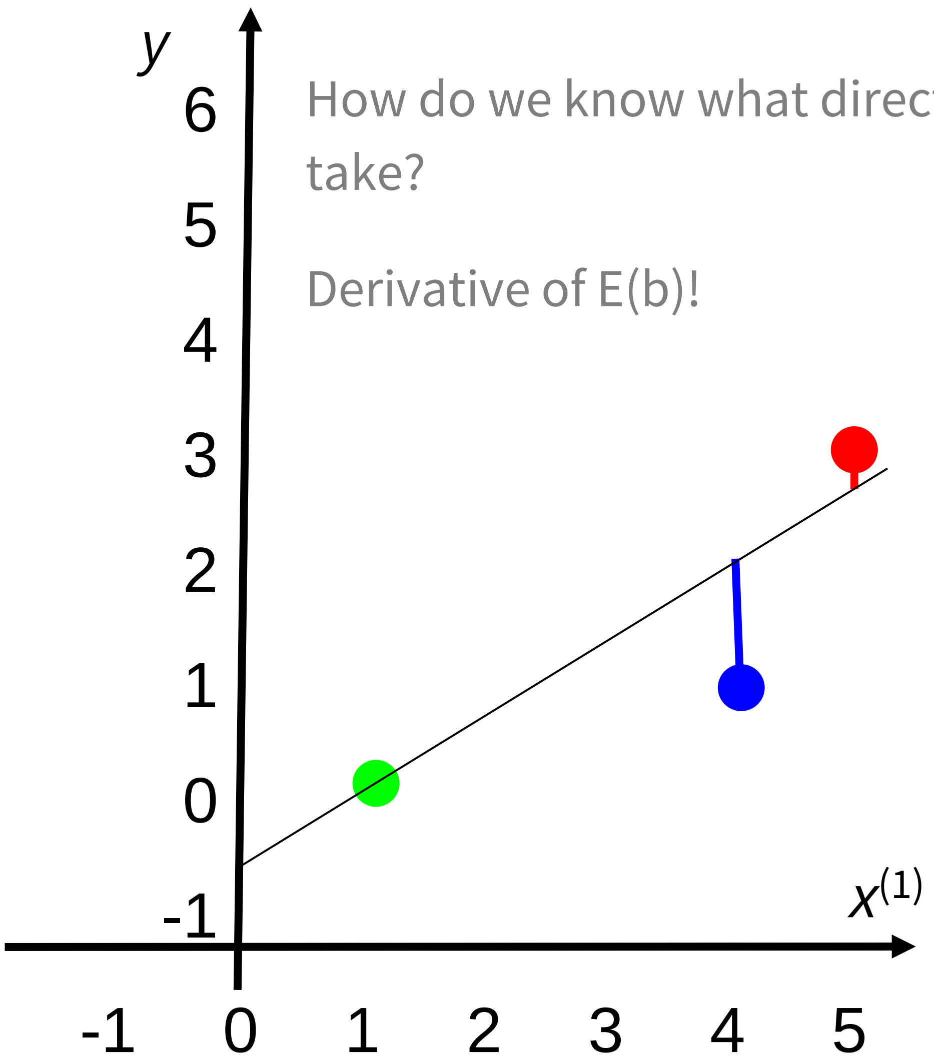


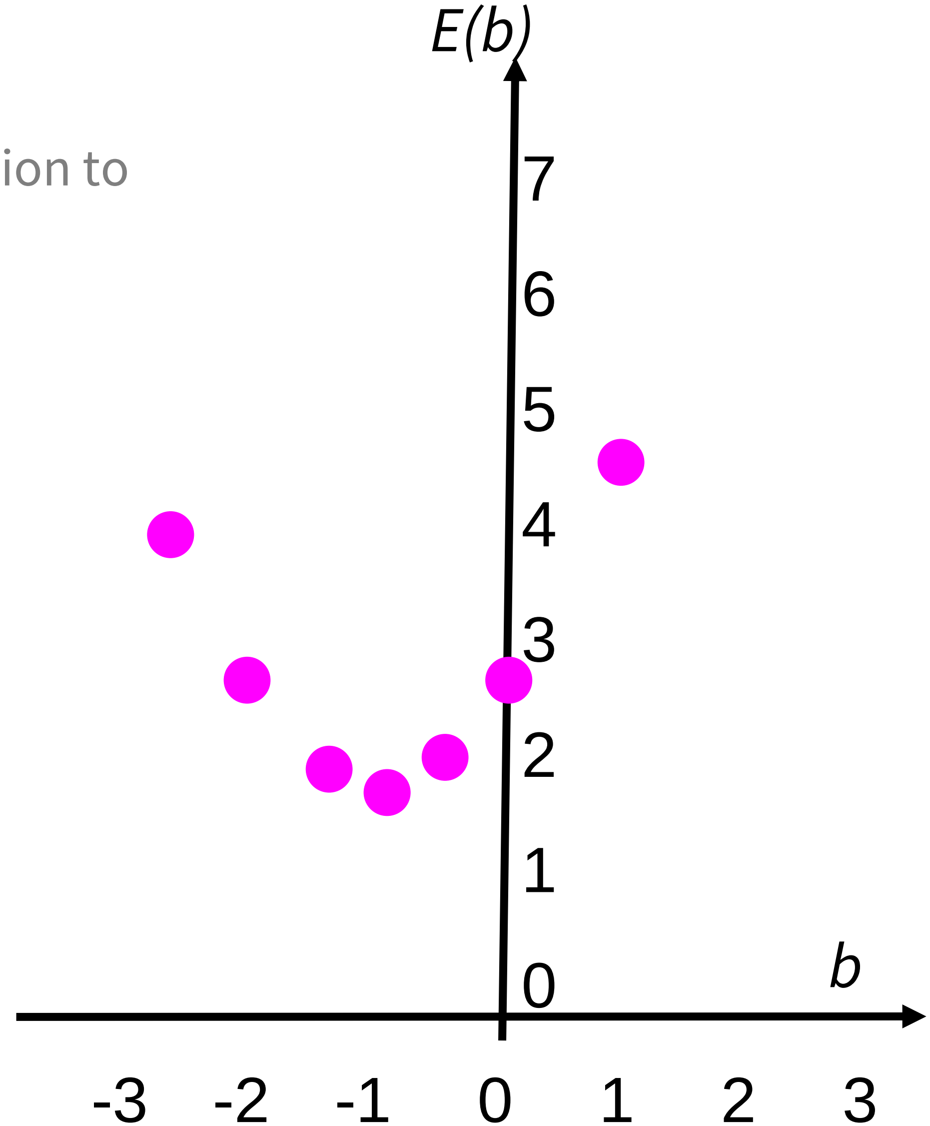
Data:{[1,0], [4,1], [5,3]}

Where to go from here?

How do we know what direction to take?

Derivative of E(b)!





Data:{[1,0], [4,1], [5,3]}

Derivative tells us how the function grows

Data:{[1,0], [4,1], [5,3]}

Derivative of E(b) at point b=0:

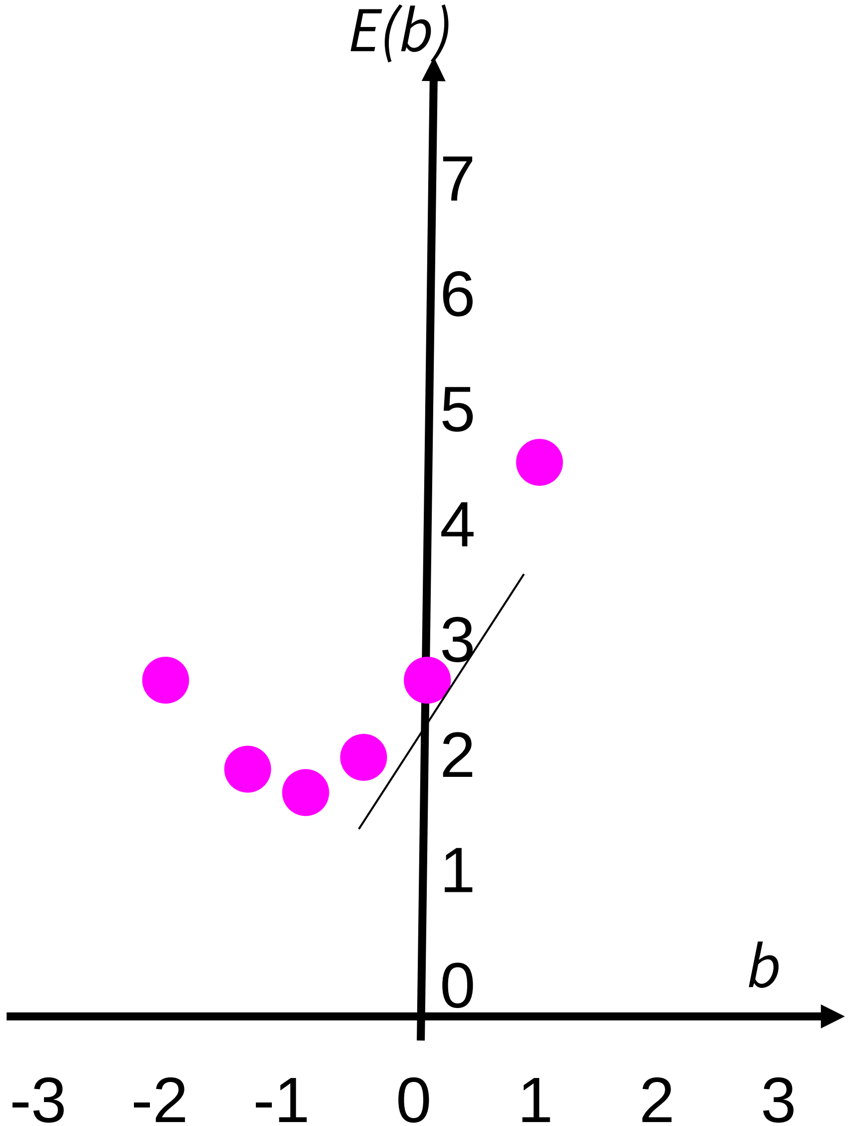
$E(b) = \sum_{i \text{ from } 1 \text{ to } n}(y_i - (0.65x_i+b))^2$

$\partial E/\partial b = \sum_{i \text{ from } 1 \text{ to } n} 2(y_i - (0.65x_i+b))^* (-1)$

If we substitute values of all (x_i,y_i), we find that the derivative of E(b) at point b=0 is positive

That means the function grows and we need to move into the opposite direction (decrease *b*)

By how much?
Derivative tells us how fast the function grows, so we can decrease by the value proportional to the rate of growth: the steeper the line at this point the more we need to change the value of *b*



Derivative tells us how the function grows

Data:{[1,0], [4,1], [5,3]}

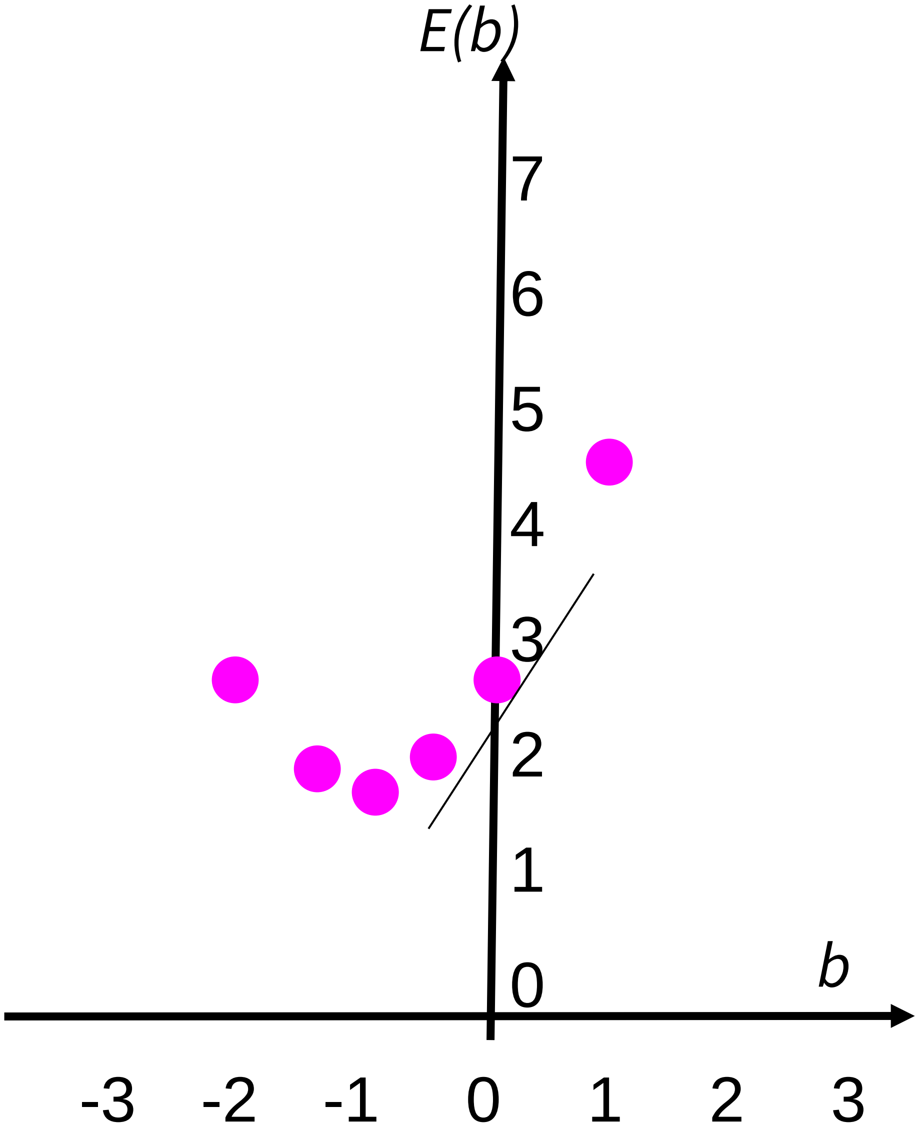
Derivative of E(b) at point b=0:

$E(b) = \sum_{i \text{ from } 1 \text{ to } n}(y_i - (0.65x_i+b))^2$

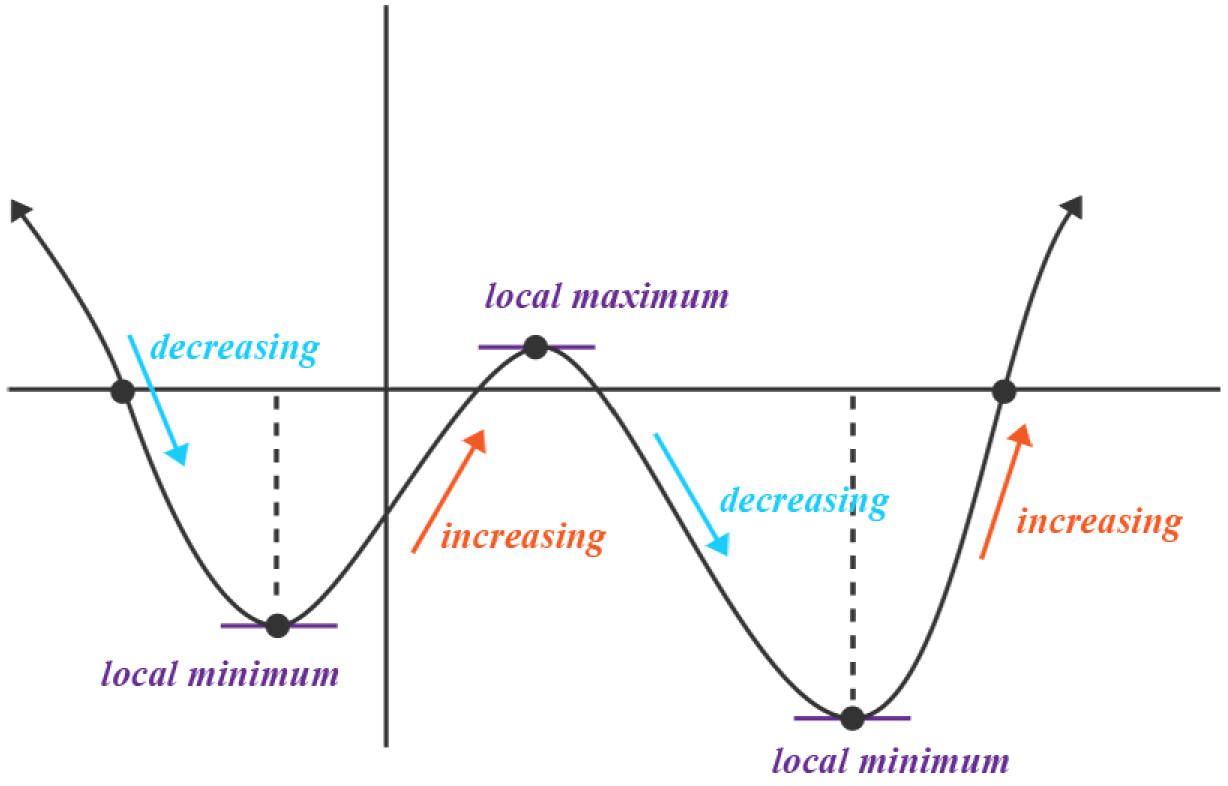
$\partial E/\partial b = \sum_{i \text{ from } 1 \text{ to } n} 2(y_i - (0.65x_i+b))^* (-1)$

In order to move slowly towards the minimum (derivative=0), we multiply the rate of growth by a constant called ***learning rate η***

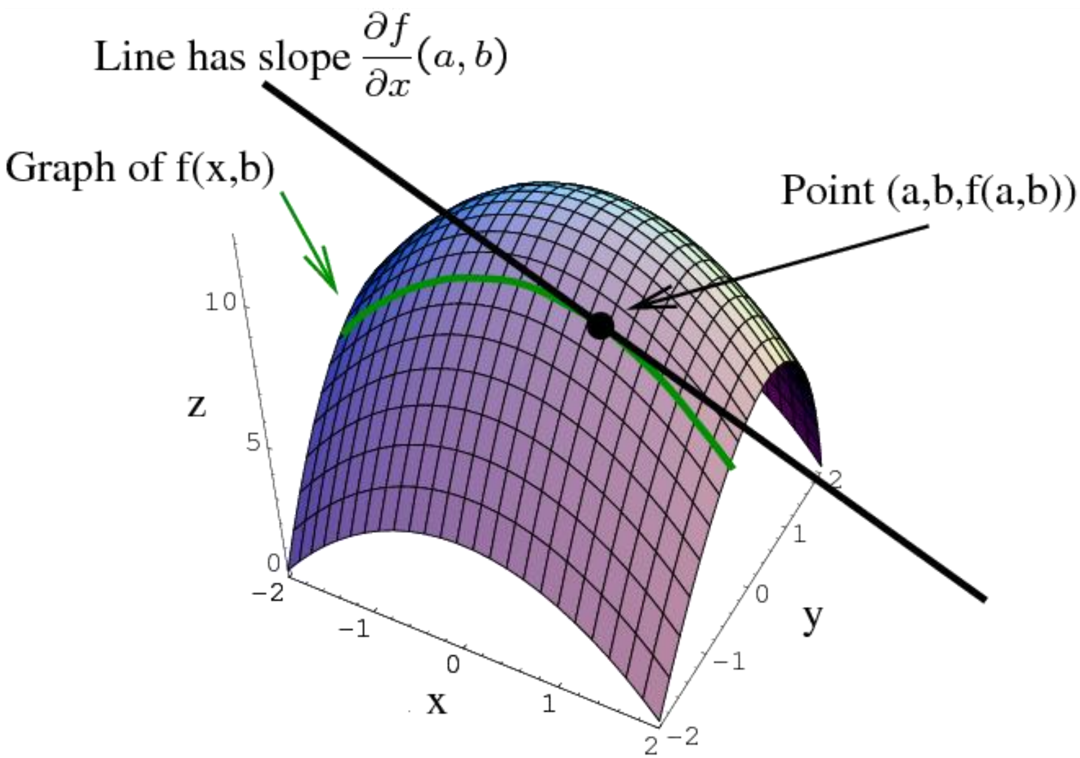
Traditional default value for the learning rate is 0.1 or 0.01. These values can be adjusted depending on the problem



Will we always reach the optimal solution with this method?



To learn both *a* and *b* at the same time



- We take a derivative of the error function *E*(*x*,*y*) at some randomly selected initial point (*a*,*b*)
- We differentiate with respect to *x* and with respect to *y* separately (partial derivatives)
- We find how to change current values of *a* and *b* - in which direction and by how much

Very detailed video about gradient descent

[LINK]

By Statquest