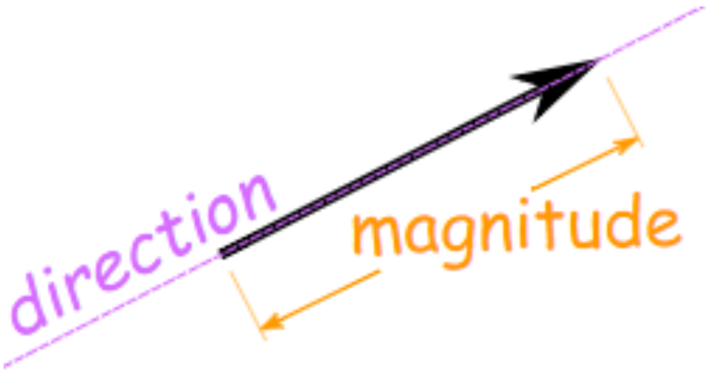
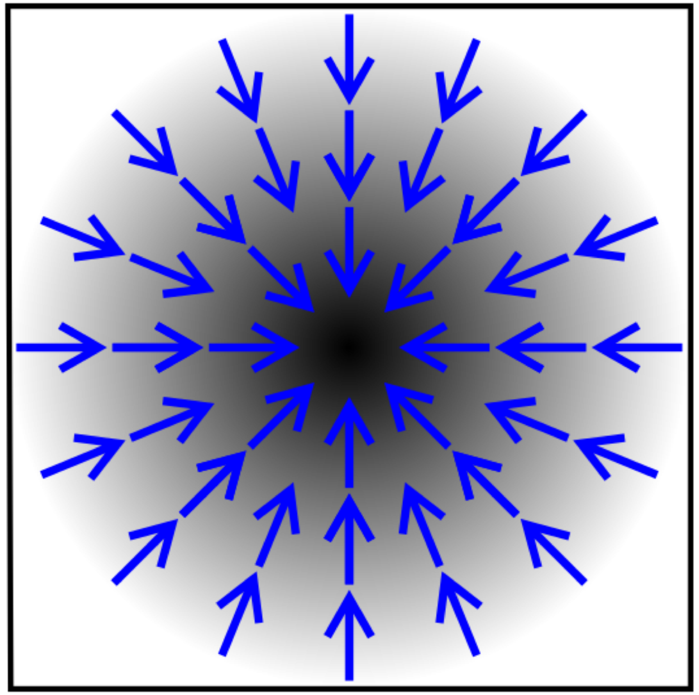
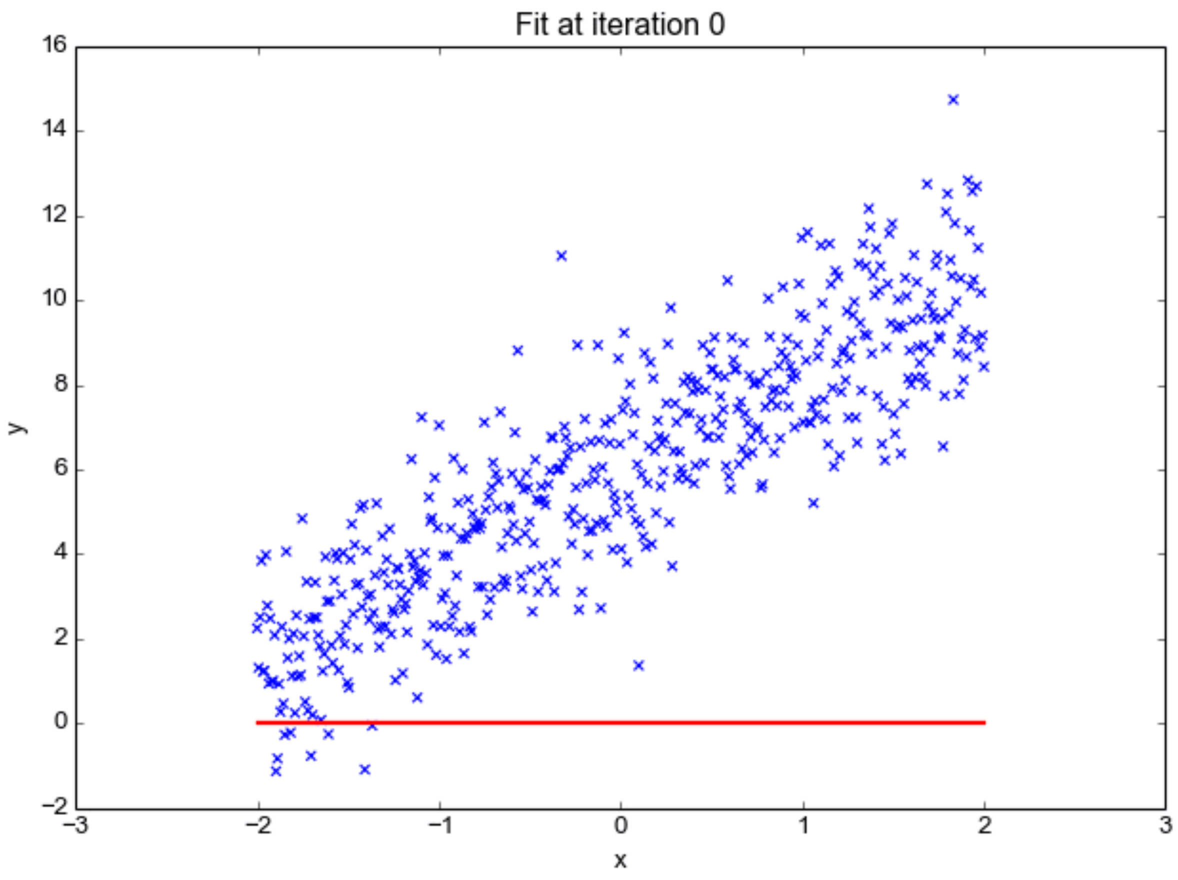
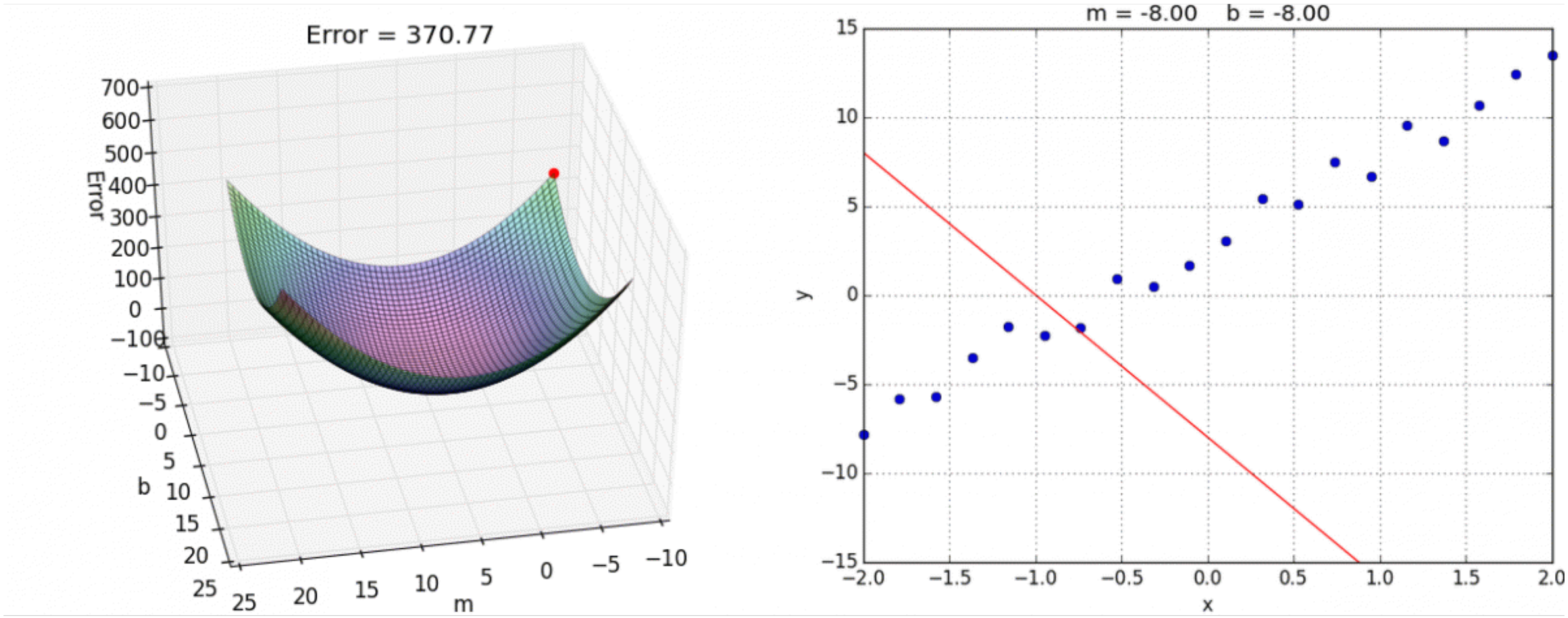

direction
magnitude



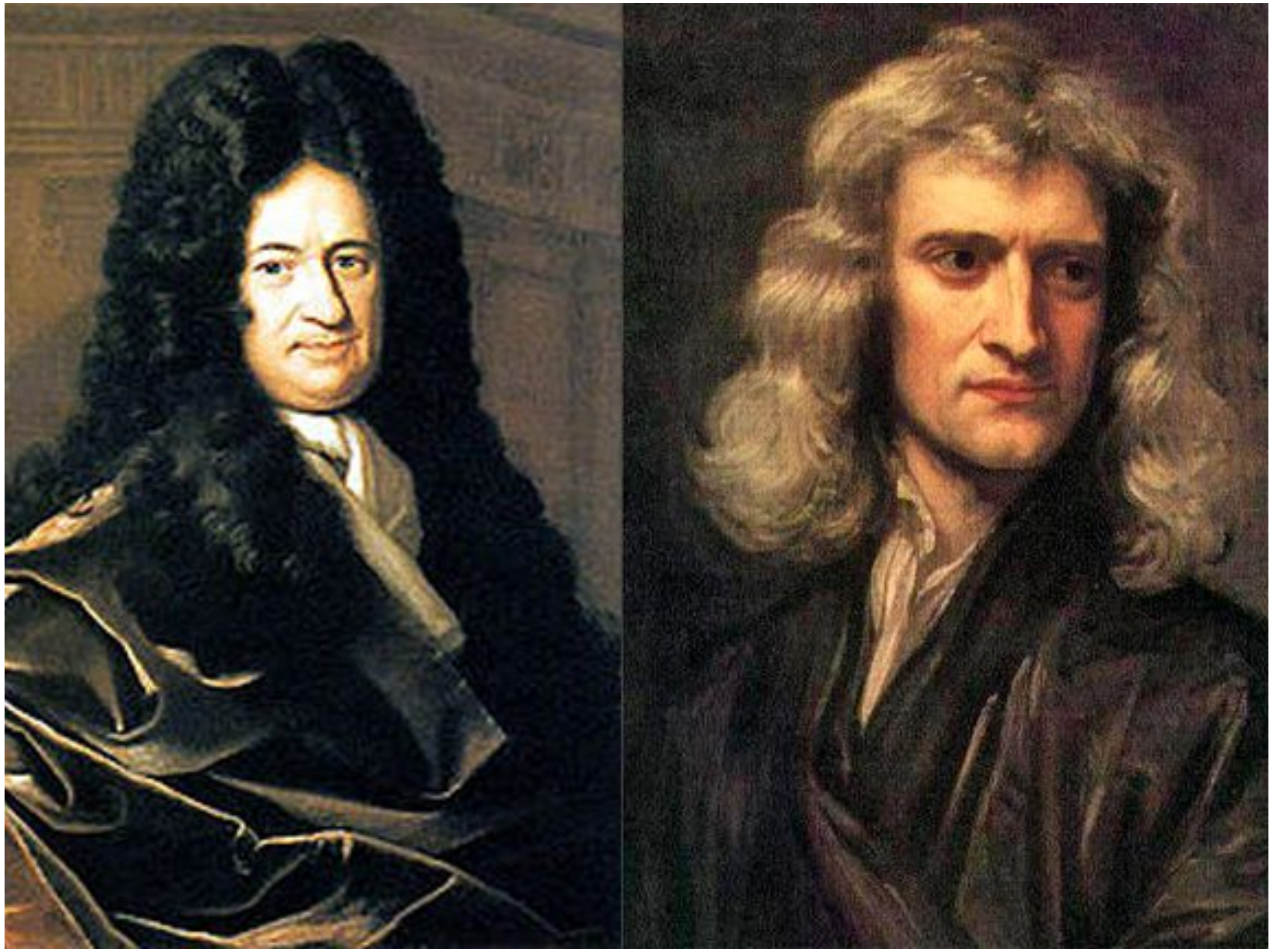
How do we find the best linear regression line?



How do we find the best linear regression line with multiple variables?



Partial Differential Equations!



Revisit SSE

$$\text{Error}_{(\text{m,b})} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (mx_i + b))^2$$

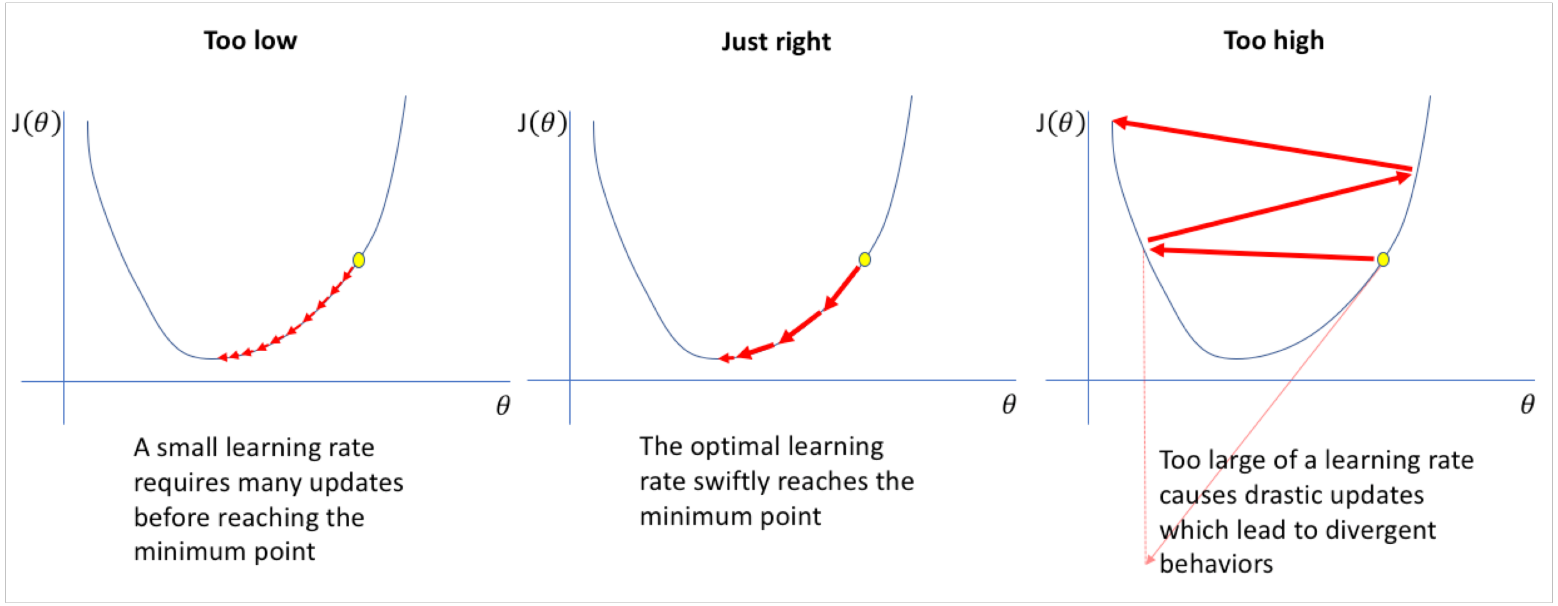
$$\frac{\partial}{\partial \text{m}} = \frac{2}{N}\sum_{i=1}^{N} -x_i(y_i - (mx_i + b))$$

$$\frac{\partial}{\partial \text{b}} = \frac{2}{N}\sum_{i=1}^{N} -(y_i - (mx_i + b))$$

Both are basic applications of the chain rule ☺

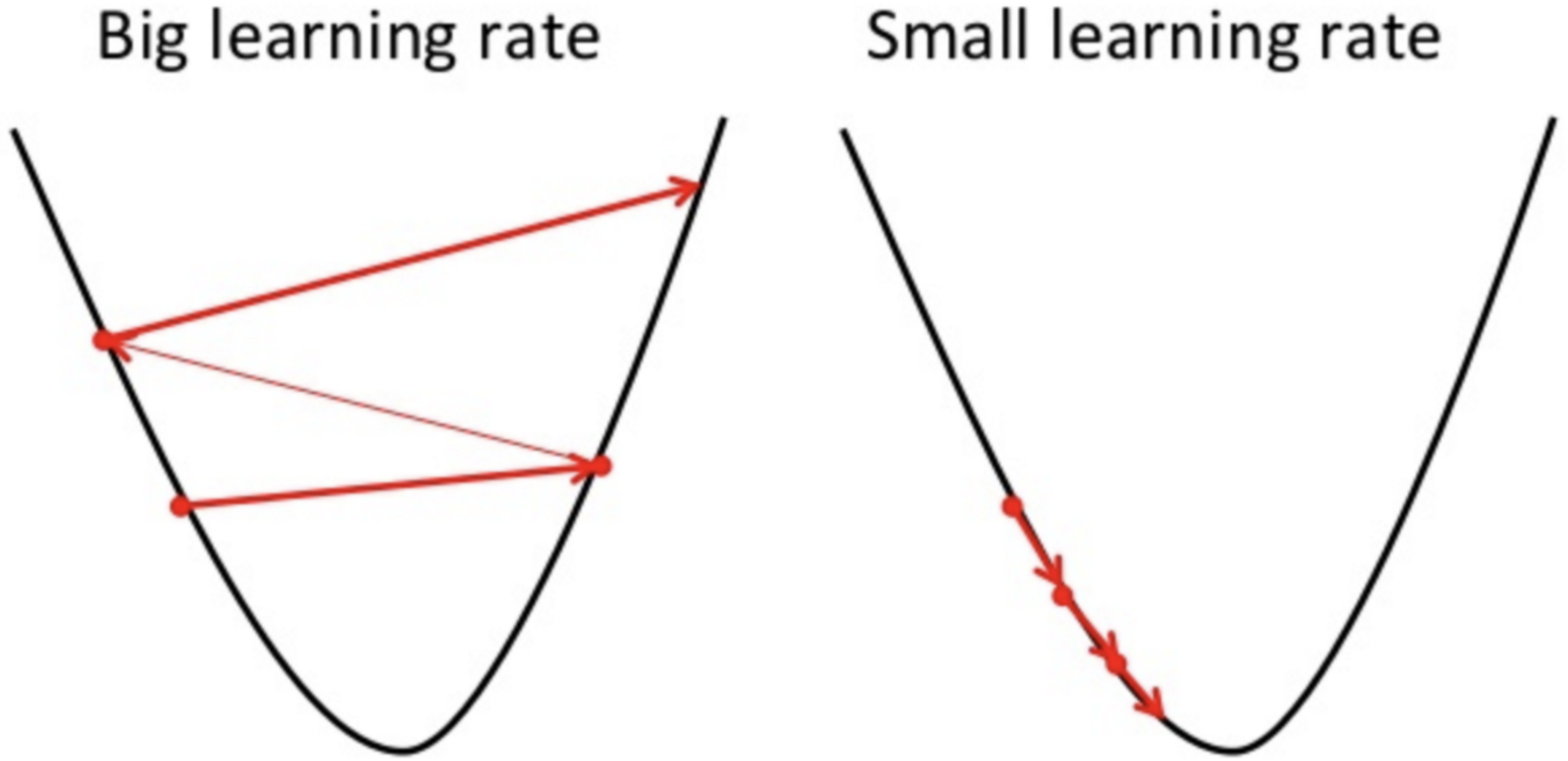
Gradient Descent

- **Gradient descent** is an optimization algorithm for finding the minimum of a function. To find a local minimum of a function using **gradient descent**, one takes steps proportional to the negative of the **gradient** of the function at the current point. These steps are governed by a learning rate.



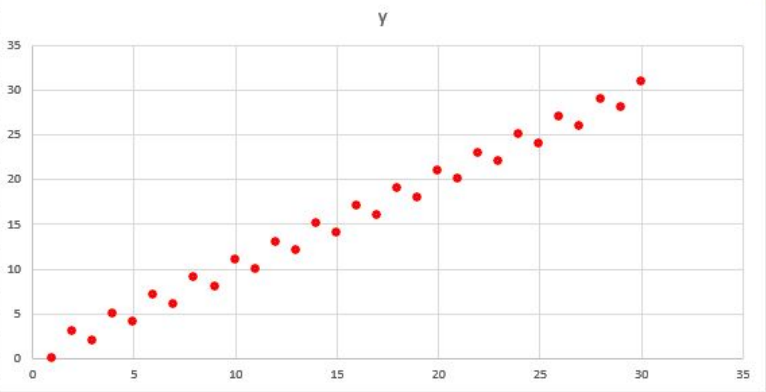
Learning Rate

How do we set the learning rate?



Excel example

alpha		x	y	m	b	y predicted	(y predicted -y)^2	pdm	pdb	m	b	y predicted	(y predicted -y)^2	pdm	pdb	m	b	y predicte
0.001		1	0	0.1	0.1	0.2	0.04	0.2	0.2	0.6652	2.87	3.5352	12.49763904	3.5352	3.5352	0.78827	3.33388	4.12215
		2	3			0.3	7.29	-5.4	-2.7			4.2004	1.44096016	2.4008	1.2004			4.91041
	q7;l;lr`lnm	3	2			0.4	2.56	-4.8	-1.6			4.8656	8.21166336	8.5968	2.8656			5.69868
		4	5			0.5	20.25	-18	-4.5			5.5308	0.28174864	2.1232	0.5308			6.48694
		5	4			0.6	11.56	-17	-3.4			6.196	4.822416	10.98	2.196			7.27521
		6	7			0.7	39.69	-37.8	-6.3			6.8612	0.01926544	-0.8328	-0.1388			8.06347
		7	6			0.8	27.04	-36.4	-5.2			7.5264	2.32989696	10.6848	1.5264			8.85174
		8	9			0.9	65.61	-64.8	-8.1			8.1916	0.65351056	-6.4672	-0.8084			9.64
		9	8			1	49	-63	-7			8.8568	0.73410624	7.7112	0.8568			10.4283
		10	11			1.1	98.01	-99	-9.9			9.522	2.184484	-14.78	-1.478			11.2165
		11	10			1.2	77.44	-96.8	-8.8			10.1872	0.03504384	2.0592	0.1872			12.0048
		12	13			1.3	136.89	-140.4	-11.7			10.8524	4.61218576	-25.7712	-2.1476			12.7931
		13	12			1.4	112.36	-137.8	-10.6			11.5176	0.23270976	-6.2712	-0.4824			13.5813
		14	15			1.5	182.25	-189	-13.5			12.1828	7.93661584	-39.4408	-2.8172			14.3696
		15	14			1.6	153.76	-186	-12.4			12.848	1.327104	-17.28	-1.152			15.1579
		16	17			1.7	234.09	-244.8	-15.3			13.5132	12.15777424	-55.7888	-3.4868			15.9461
		17	16			1.8	201.64	-241.4	-14.2			14.1784	3.31822656	-30.9672	-1.8216			16.7344
		18	19			1.9	292.41	-307.8	-17.1			14.8436	17.27566096	-74.8152	-4.1564			17.5227
		19	18			2	256	-304	-16			15.5088	6.20607744	-47.3328	-2.4912			18.3109
		20	21			2.1	357.21	-378	-18.9			16.174	23.290276	-96.52	-4.826			19.0992
		21	20			2.2	316.84	-373.8	-17.8			16.8392	9.99065664	-66.3768	-3.1608			19.8875
		22	23			2.3	428.49	-455.4	-20.7			17.5044	30.20161936	-120.903	-5.4956			20.6757
		23	22			2.4	384.16	-450.8	-19.6			18.1696	14.67196416	-88.0992	-3.8304			21.464
		24	25			2.5	506.25	-540	-22.5			18.8348	38.00969104	-147.965	-6.1652			22.2523
		25	24			2.6	457.96	-535	-21.4			19.5	20.25	-112.5	-4.5			23.0405
		26	27			2.7	590.49	-631.8	-24.3			20.1652	46.71449104	-177.705	-6.8348			23.8288
		27	26			2.8	538.24	-626.4	-23.2			20.8304	26.72476416	-139.579	-5.1696			24.6171
		28	29			2.9	681.21	-730.8	-26.1			21.4956	56.31601936	-210.123	-7.5044			25.4053
		29	28			3	625	-725	-25			22.1608	34.09625664	-169.337	-5.8392			26.1936
		30	31			3.1	778.41	-837	-27.9			22.826	66.814276	-245.22	-8.174			26.9818
						error	**254.405**	-565.2	-27.7			error	**15.11190344**	-123.066	-4.6388			error
												error % diff	-94%					error % di



We need to find a best fit line which means minimze the Error = SSE. The new line will be of form y' = mx+b and we'll subtract every y' from y to see what the error is. Error is given by Error = 1/n*sigma(y-y')^2 = 1/n*sigma(y-(mx+b)^2 = 1/n*sigma(y-mx-b)^2. So Error is a function of m and b. We know n=30, and have all the y points. Therefore

$$\text{Error}_{(\text{m,b})} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (mx_i + b))^2$$

$$\frac{\partial}{\partial \text{m}} = \frac{2}{N}\sum_{i=1}^{N} - x_i(y_i - (mx_i + b))$$

$$\frac{\partial}{\partial \text{b}} = \frac{2}{N}\sum_{i=1}^{N} - (y_i - (mx_i + b))$$

Java Example!

```java
import java.util.function.Function;
import static java.lang.Math.*;
import static java.lang.System.out;

double gamma = 0.01;
double precision = 0.00001;

Function<Double,Double> df = x ->  4 * pow(x, 3) - 9 * pow(x, 2);

double gradientDescent(Function<Double,Double> f) {

    double curX = 6.0;
    double previousStepSize = 1.0;

    while (previousStepSize > precision) {
        double prevX = curX;
        curX -= gamma * f.apply(prevX);
        previousStepSize = abs(curX - prevX);
    }
    return curX;
}

double res = gradientDescent(df);
out.printf("The local minimum occurs at %f", res);
```

Python Example

```
cur_x = 6 # The algorithm starts at x=6
gamma = 0.01 # step size multiplier
precision = 0.00001
previous_step_size = 1
max_iters = 10000 # maximum number of iterations
iters = 0 #iteration counter

df = lambda x: 4 * x**3 - 9 * x**2

while previous_step_size > precision and iters < max_iters:
    prev_x = cur_x
    cur_x -= gamma * df(prev_x)
    previous_step_size = abs(cur_x - prev_x)
    iters+=1

print("The local minimum occurs at", cur_x)
#The output for the above will be: ('The local minimum occurs at', 2.2499646074278457)
```

Gradient Descent Applications

Multi-variable Regression