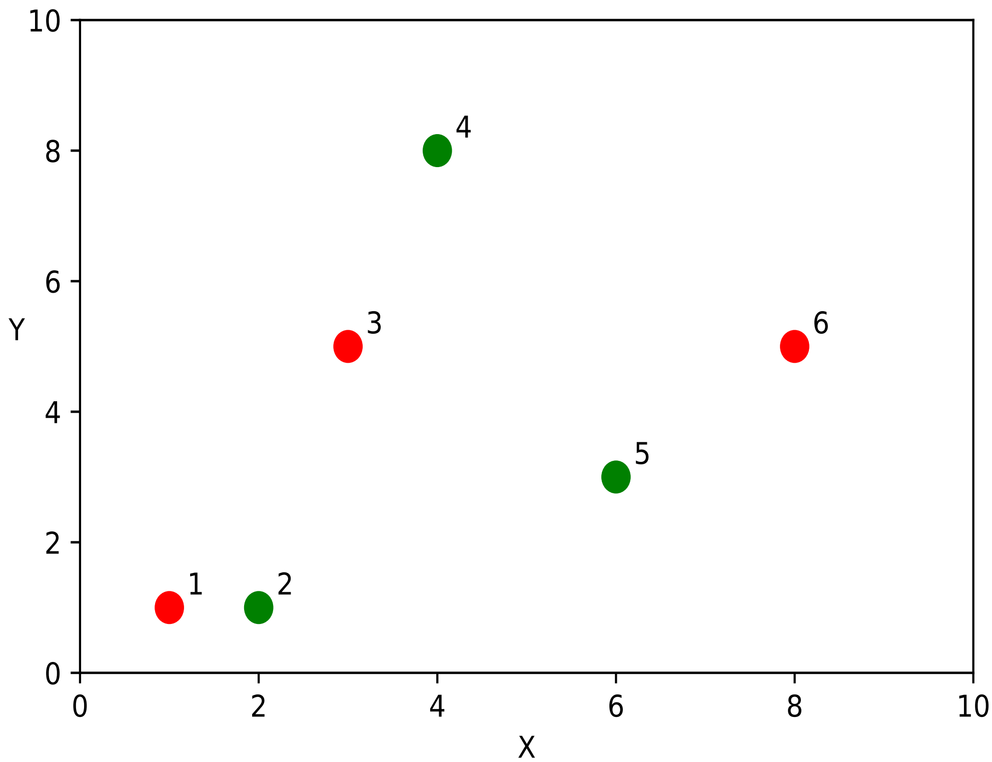
LINEAR

REGRESSION

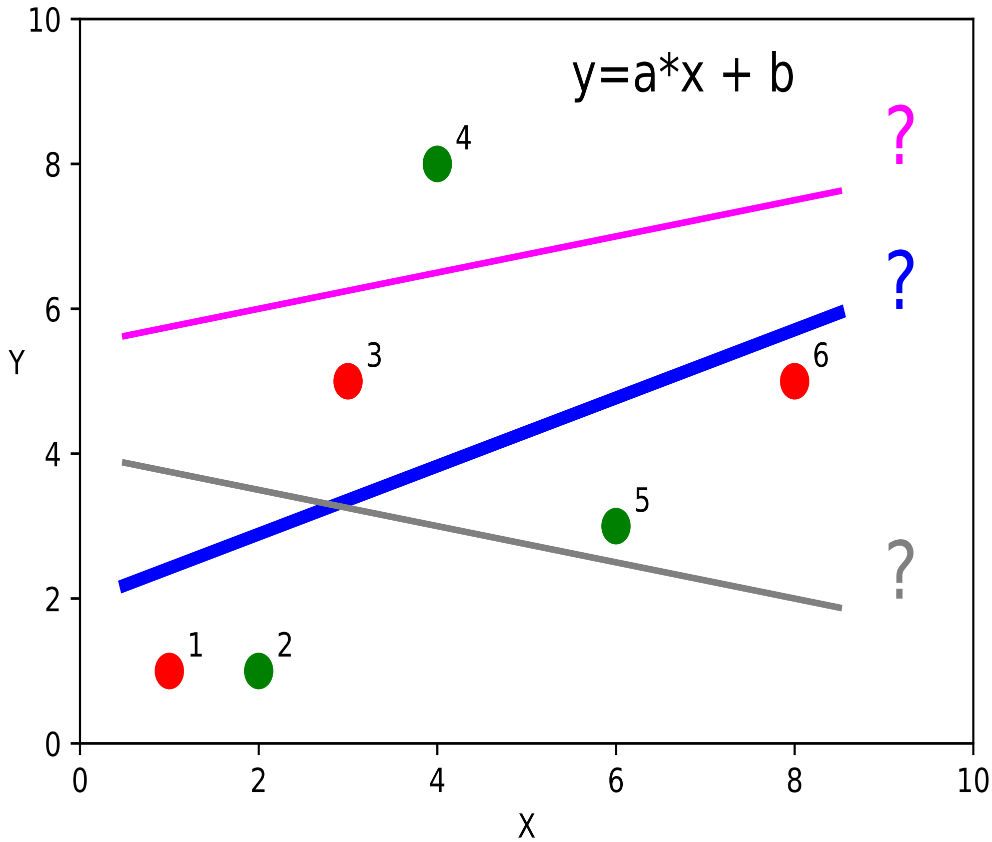
Linear Relationship



- want to establish a linear relationship:

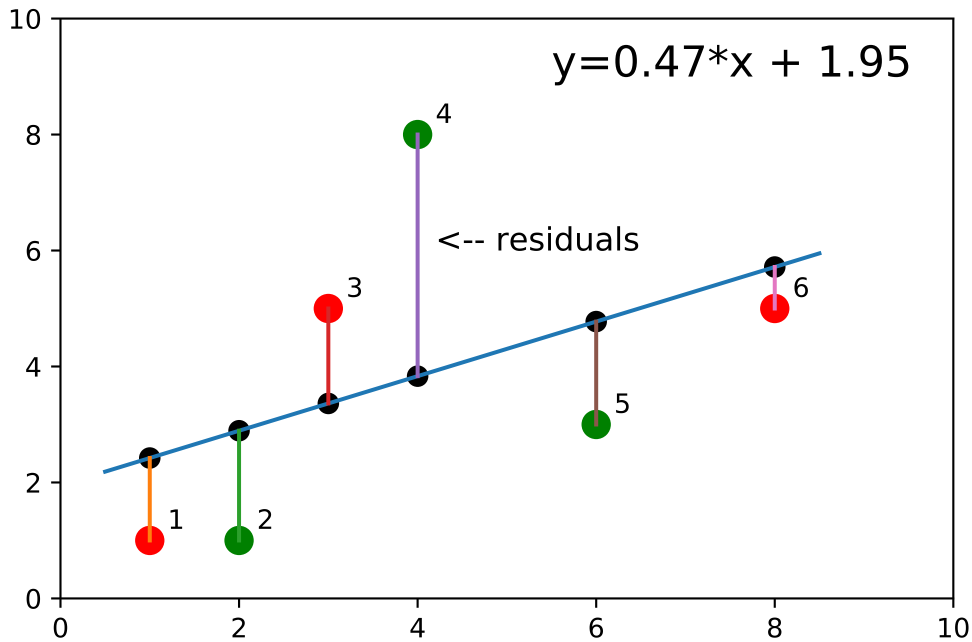
$$y = ax + b + e$$

Where is the difficulty?



- need a criteria to compute slope a and intercept b

How do we choose?



- choose line to minimize sum of squared residuals

$$Q = \sum_{i=1}^{n} e_i^2$$

Python Code

```
import numpy as np
from sklearn.linear_model import LinearRegression

x = np.array([1,2,3,4,6,8])
y = np.array([1,1,5,8,3,5])
x_2 = x_2 = x[:,np.newaxis]
lin_reg = LinearRegression(fit_intercept=True)
lin_reg.fit(x_2, y)
```

```
> x
array([1, 2, 3, 4, 6, 8])
> x_2
array([[1],
       [2],
       [3],
       [4],
       [6],
       [8]])
```

Python Code

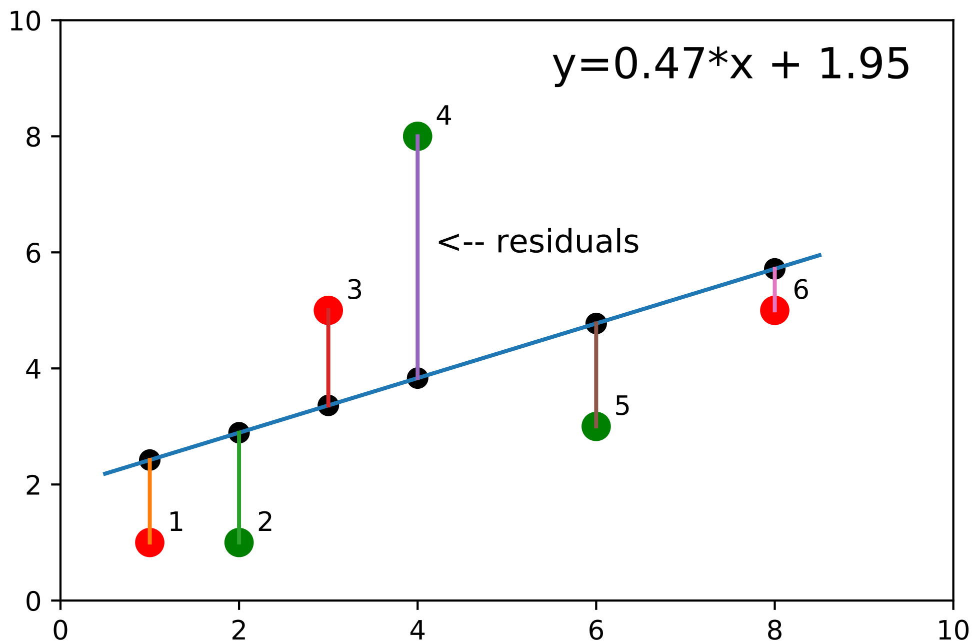
```
import numpy as np
from sklearn.linear_model import LinearRegression

x = np.array([1,2,3,4,6,8])
y = np.array([1,1,5,8,3,5])

x_2 = x_2 = x[:,np.newaxis]
lin_reg = LinearRegression(fit_intercept=True)
lin_reg.fit(x_2, y)
```

```
> lin_reg.score(x_2,y)
 0.20441841895129076
> lin_reg.coef_
array([ 0.47058824])
> lin_reg.intercept_
1.9509803921568625
```

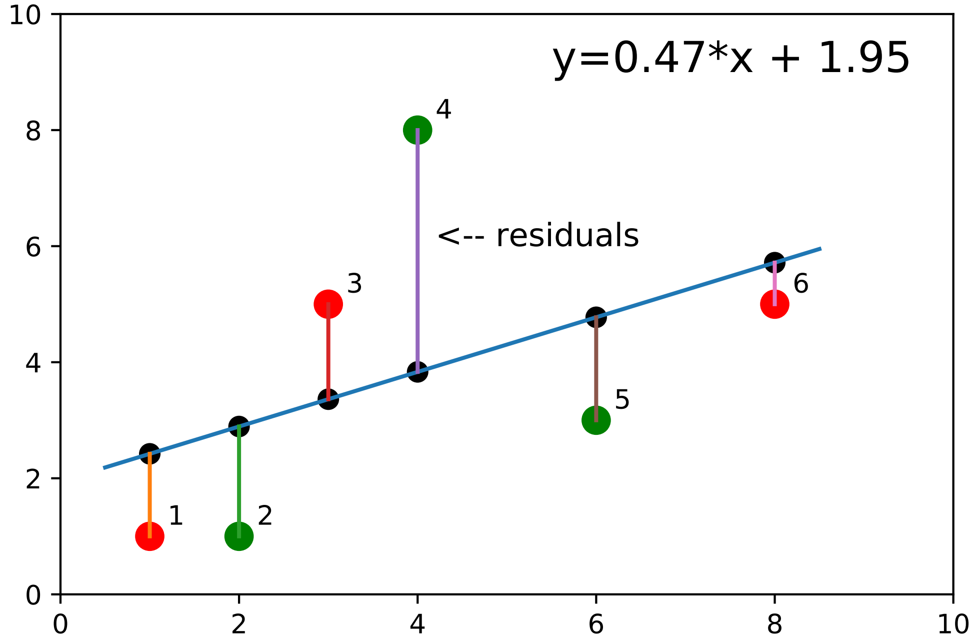
Derivation



- want to minimize loss function (sum of squares)

$$Q = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} \left(y_i - \left(ax_i + b\right)\right)^2$$

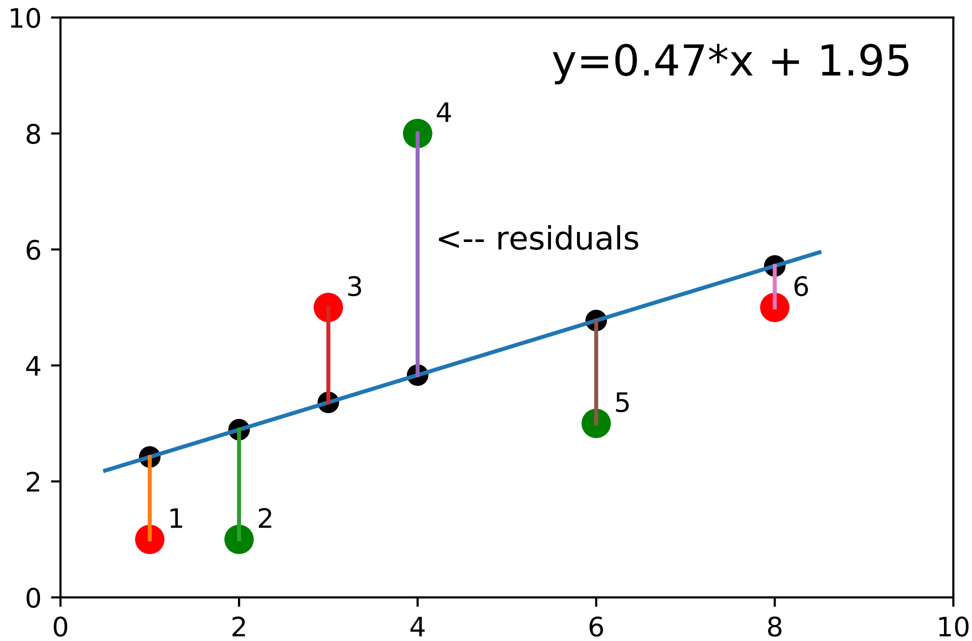
Derivation



- need to solve

$$\frac{\partial Q}{\partial a} = 0, \quad \frac{\partial Q}{\partial b} = 0$$

Derivation



- define means μ_x and μ_y
- define (co)variances σ_x^2 and σ_{xy}^2

Derivation

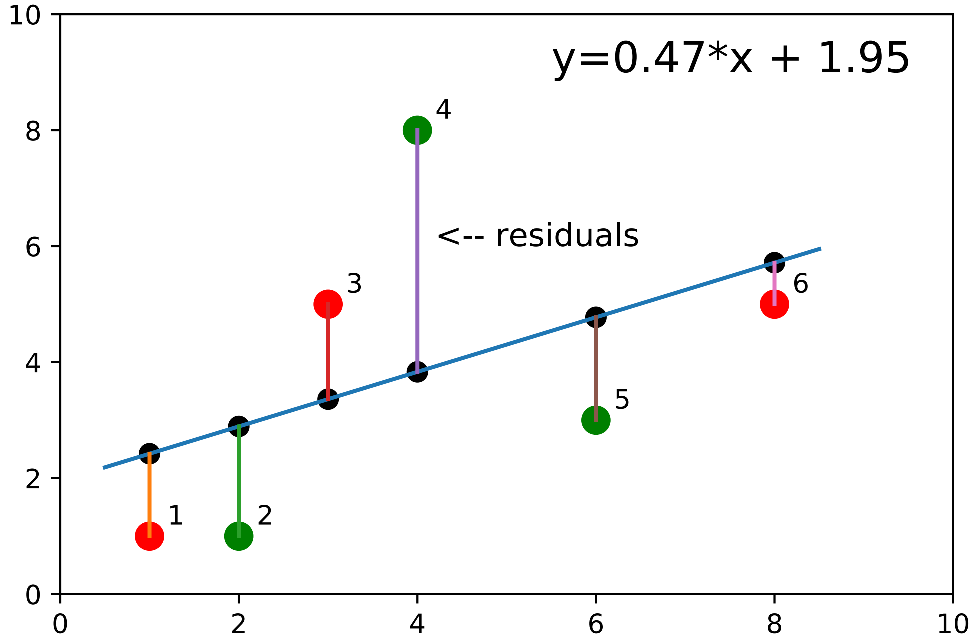
- define means:

$$\mu_x = \frac{x_1 + \cdots + x_n}{n}$$
$$\mu_y = \frac{y_1 + \cdots + y_n}{n}$$

and (co)variances

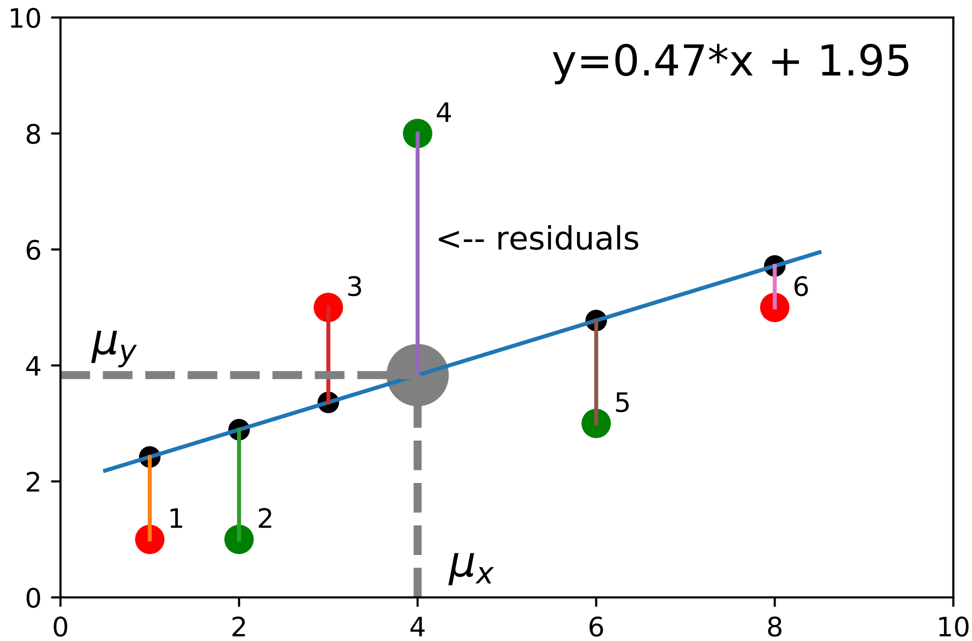
$$\sigma_{xy}^2 = \frac{\left(x_1 y_1 + \cdots + x_n y_n\right)}{n} - \mu_x \mu_y$$
$$\sigma_x^2 = \frac{\left(x_1^2 + \cdots + x_n^2\right)}{n} - \mu_x^2$$

Derivation for b



$$\frac{\partial Q}{\partial b} = -2\sum_{i=1}^{n}\left(y_i - \left(ax_i + b\right)\right.$$
$$= 2n\left(b + a\mu_x - \mu_y\right) = 0$$

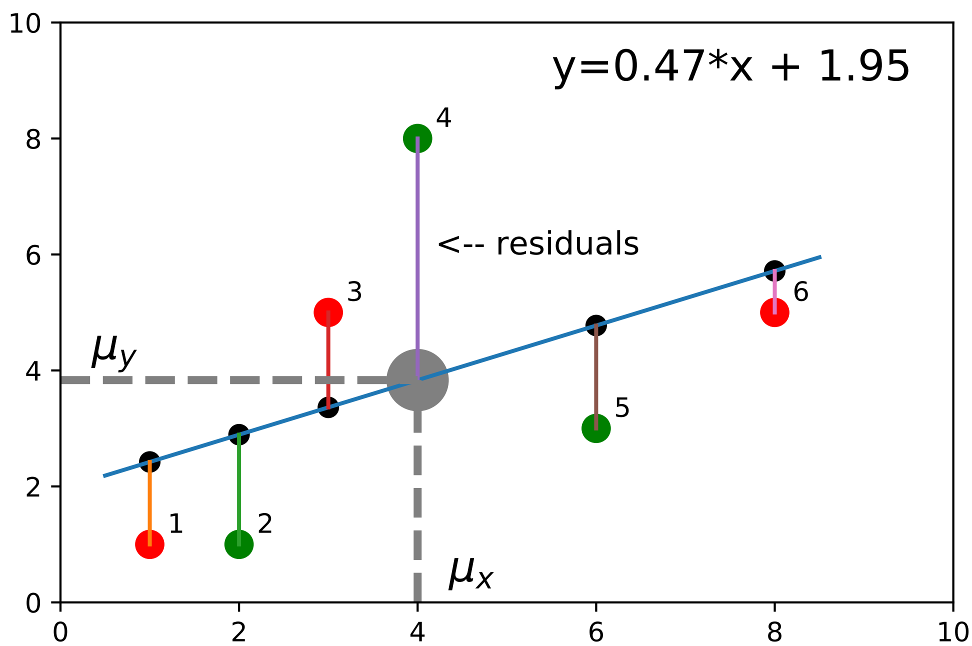
Derivation for b



- line must go through μ_x amd μ_y

$$b = \mu_y - a\mu_x$$

Derivation for a



- solve for b from

$$\frac{\partial Q}{\partial b} = 0$$

Derivation for a

$$\begin{aligned}
\frac{\partial Q}{\partial b} &= -2\sum_{i=1}^{n} x_i\Big(y_i - \big(ax_i + b\big)\Big) \\
&= -2\sum_{i=1}^{n}\big(x_i y_i - x_i\mu_y + a x_i\mu_x - a x_i^2\big) \\
&= -2\left[\sum_{i=1}^{n}\big(x_i y_i - x_i\mu_y\big) + a\sum_{i=1}^{n}\big(x_i^2 - \mu_x x_i\right. \\
&= -2\big(\sigma_{xy}^2 - a\cdot\sigma_x^2\big) = 0
\end{aligned}$$

- we obtain:

$$a = \sigma_{xy}^2 \cdot \left(\sigma_x^2\right)^{-1}$$

Summary of Derivation

- define:

$$\mu_x = \frac{x_1 + \cdots + x_n}{n}$$

$$\mu_y = \frac{y_1 + \cdots + y_n}{n}$$

$$\sigma_{xy}^2 = \frac{\left(x_1 y_1 + \cdots + x_n y_n\right)}{n} - \mu_x \mu_y$$

$$\sigma_x^2 = \frac{\left(x_1^2 + \cdots + x_n^2\right)}{n} - \mu_x^2$$

- compute slope and intercept:

$$a = \left[\sigma_{xy}^2 \cdot \left(\sigma_x^2\right)^{-1}\right]$$

$$b = \mu_y - \left[\sigma_{xy}^2 \cdot \left(\sigma_x^2\right)^{-1}\right] \mu_x$$

Python Code

```python
import numpy as np
import matplotlib.pyplot as plt

def estimate_coef(x, y):
    n = np.size(x)
    mu_x, mu_y = np.mean(x), np.mean(y)
    SS_xy = np.sum(y*x) - n* mu_y * mu_x
    SS_xx = np.sum(x*x) - n *mu_x* mu_x
    slope = SS_xy / SS_xx
    intercept = mu_y - slope*mu_x
    return(slope, intercept)

def plot_regression(x, y, slope, intercept):
    plt.scatter(x, y, color = "blue",
                marker = "o", s = 100)
    y_pred = slope * x + intercept
    plt.plot(x, y_pred, color = "green", lw = 3)

    plt.xlabel('x')
    plt.ylabel('y')
    plt.show()


x = np.array([1,2,3,4,6,8])
y = np.array([1,1,5,8,3,5])
slope, intercept = estimate_coef(x,y)
plot_regression(x,y,slope, intercept)
```

Equivalent Derivation

$$\frac{\partial Q}{\partial a} = -2\sum_{i=1}^{n} x_i\big(y_i - \big(ax_i + b\big)\big)$$
$$= -2\sum_{i=1}^{n} x_i\big(y_i - e_i\big) = 0$$
$$\frac{\partial Q}{\partial b} = -2\sum_{i=1}^{n}\big(y_i - \big(ax_i + b\big)\big)$$
$$= -2\sum_{i=1}^{n} e_i = 0$$

- these are equivalenet to

$$E\big(Xe\big) = 0 \quad \text{and} \quad E\big(e\big) = 0$$

Numerical Computation

- use gradient descent algorithm
- we have two equations:

$$\frac{\partial Q}{\partial a} = -2\sum_{i=1}^{n} x_i(y_i - e_i) = 0$$

$$\frac{\partial Q}{\partial b} = -2\sum_{i=1}^{n} e_i = 0$$

- choose learning rate L

Numerical Computation

```
x = np.array([1,2,3,4,6,8])
y = np.array([1,1,5,8,3,5])

a = 0; b= 0; n=len(x);
epochs = 100; error=[]
for i in range(epochs):
    y_pred = slope * x + intercept
    error = sum((y-y_pred)*(y-y_pred))
    D_slope = (-2.0/n)* sum(x*(y-y_pred))
    D_intercept = (-2.0/n) * sum(y - y_pred)
    a = a - L * D_slope
    b = b - L * D_intercept
```

- use gradient descent algorithm
- we have two equations:

$$\frac{\partial Q}{\partial a} = -2\sum_{i=1}^{n} x_i(y_i - e_i) = 0$$

$$\frac{\partial Q}{\partial b} = -2\sum_{i=1}^{n} e_i = 0$$

- choose learning rate L

Illustration of Gradient Descent

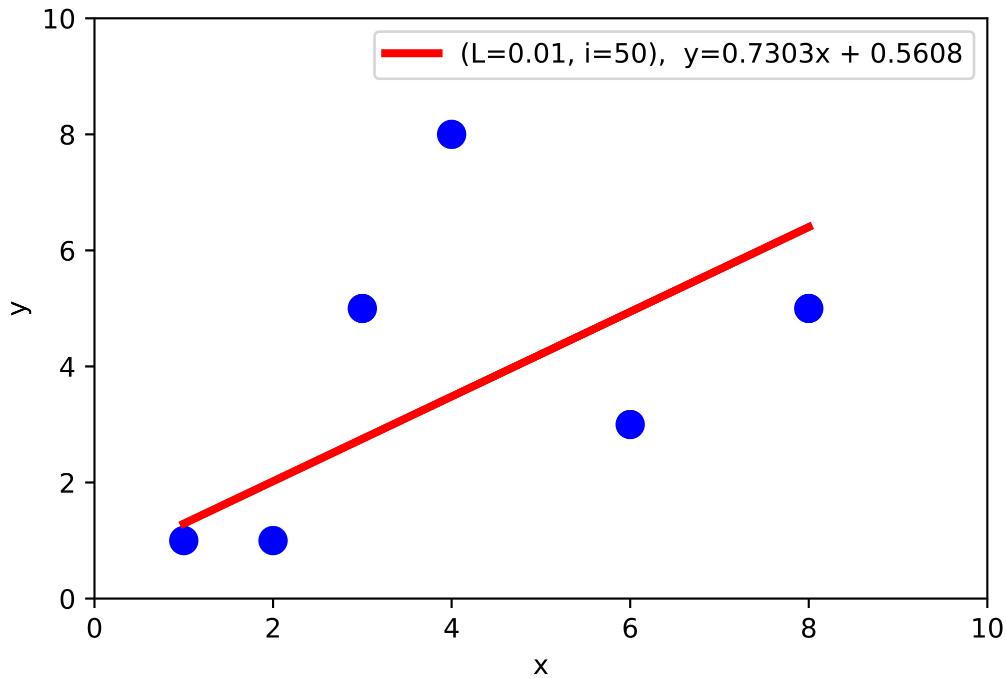


Illustration of Gradient Descent

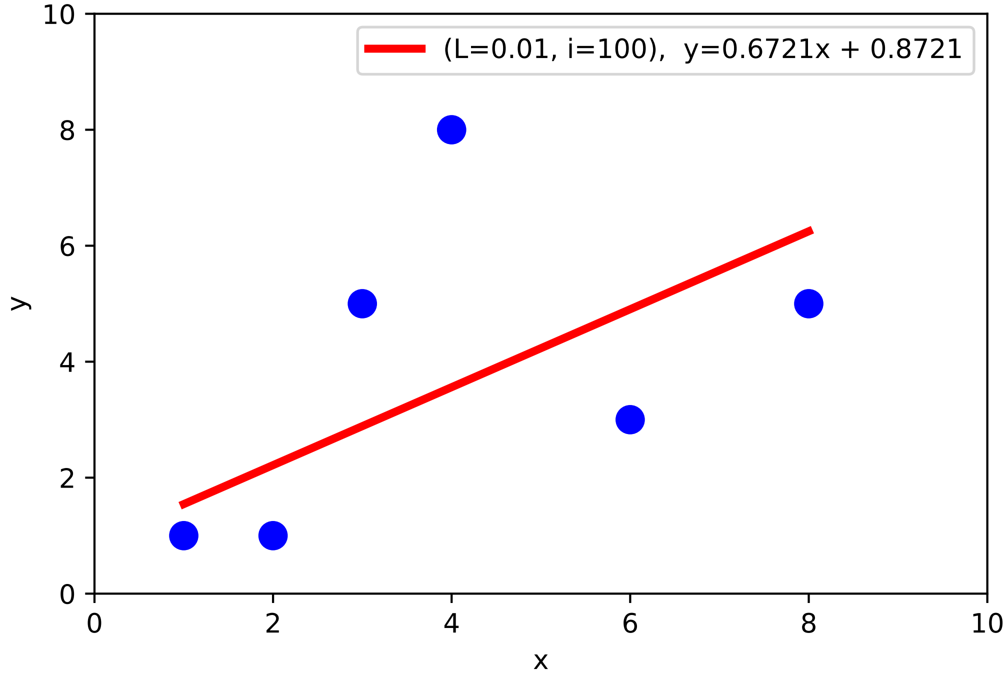


Illustration of Gradient Descent

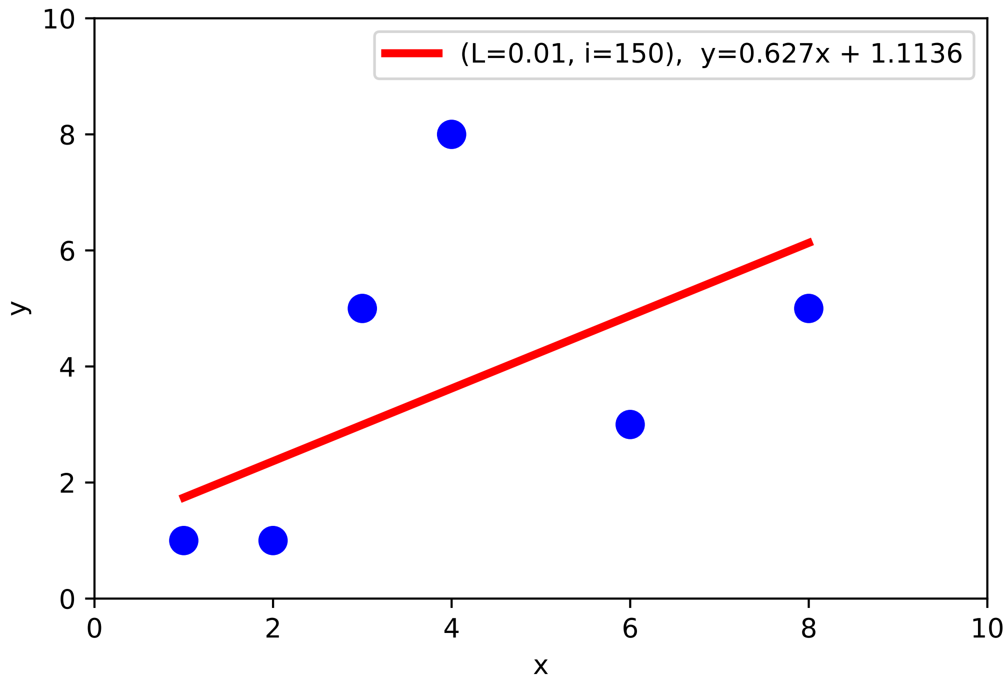


Illustration of Gradient Descent

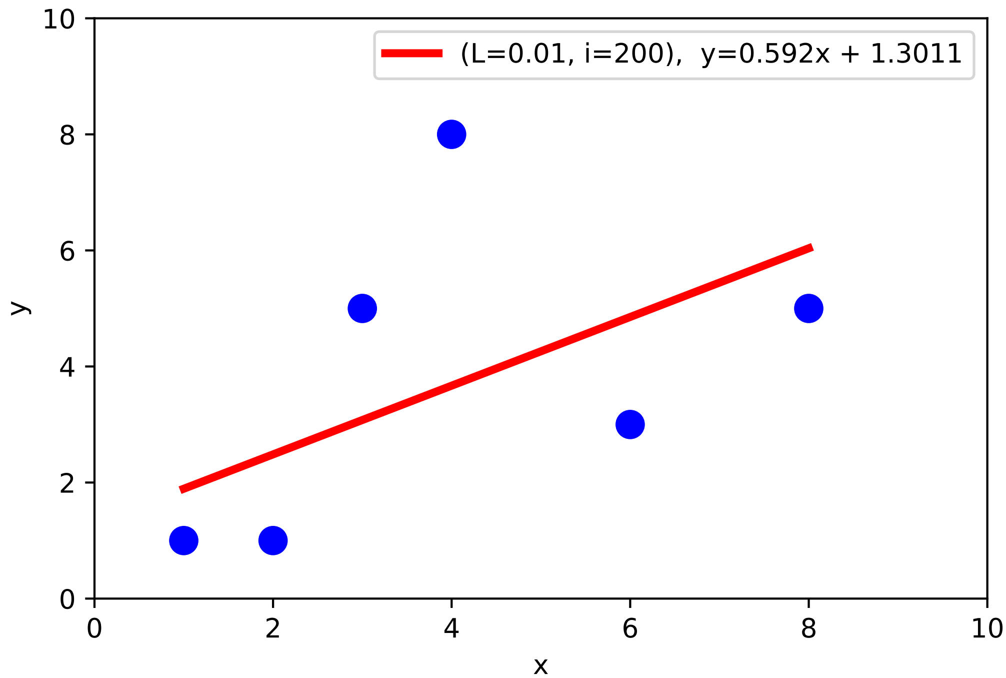


Illustration of Gradient Descent

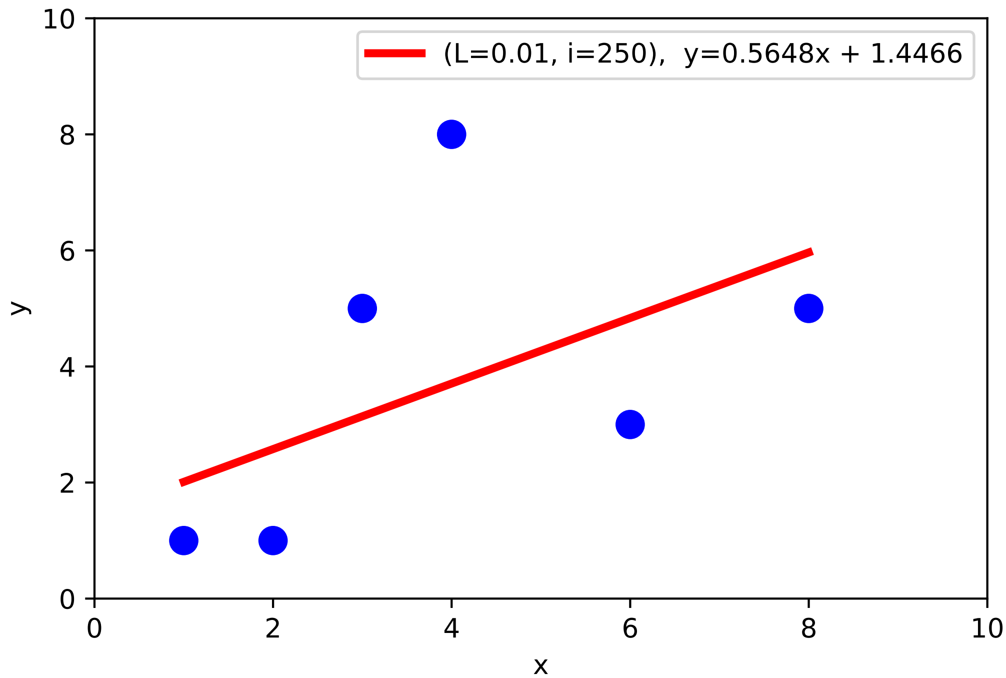


Illustration of Gradient Descent

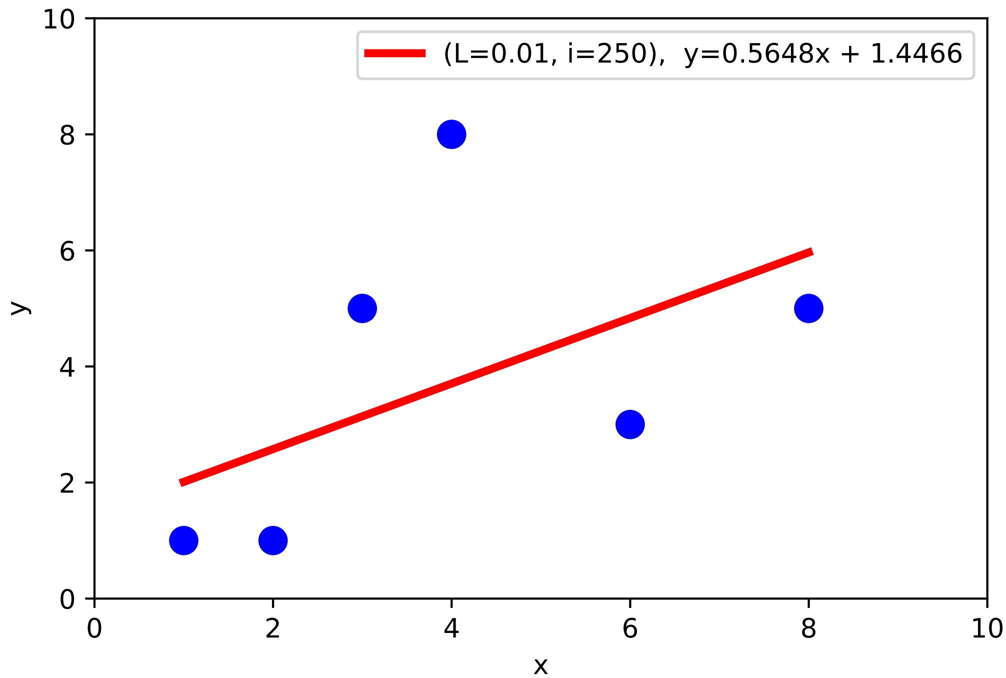


Illustration of Gradient Descent

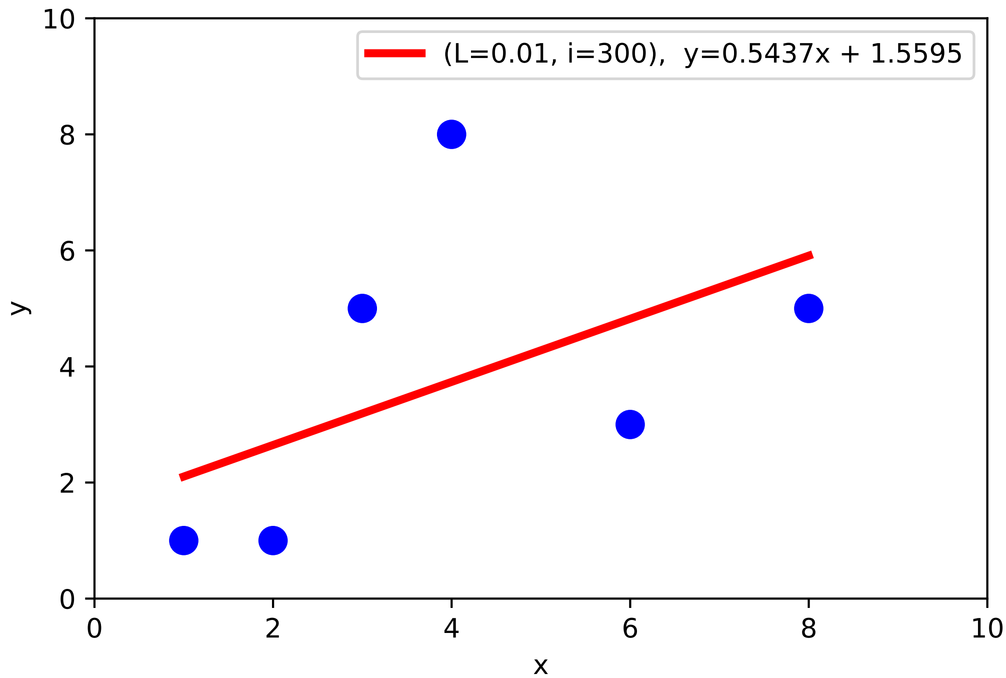


Illustration of Gradient Descent

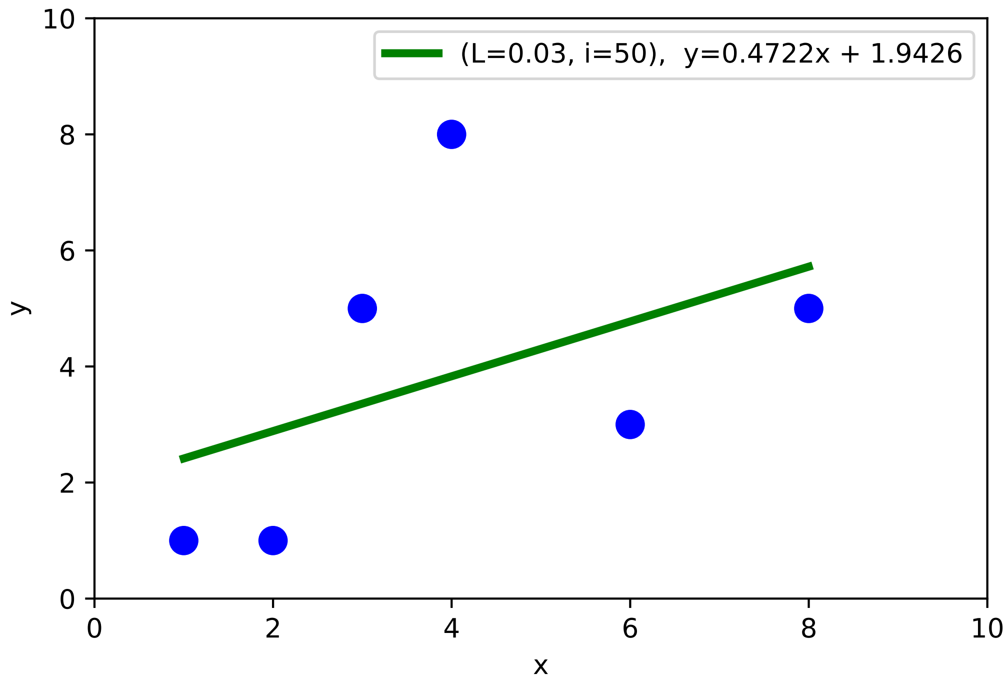


Illustration of Gradient Descent

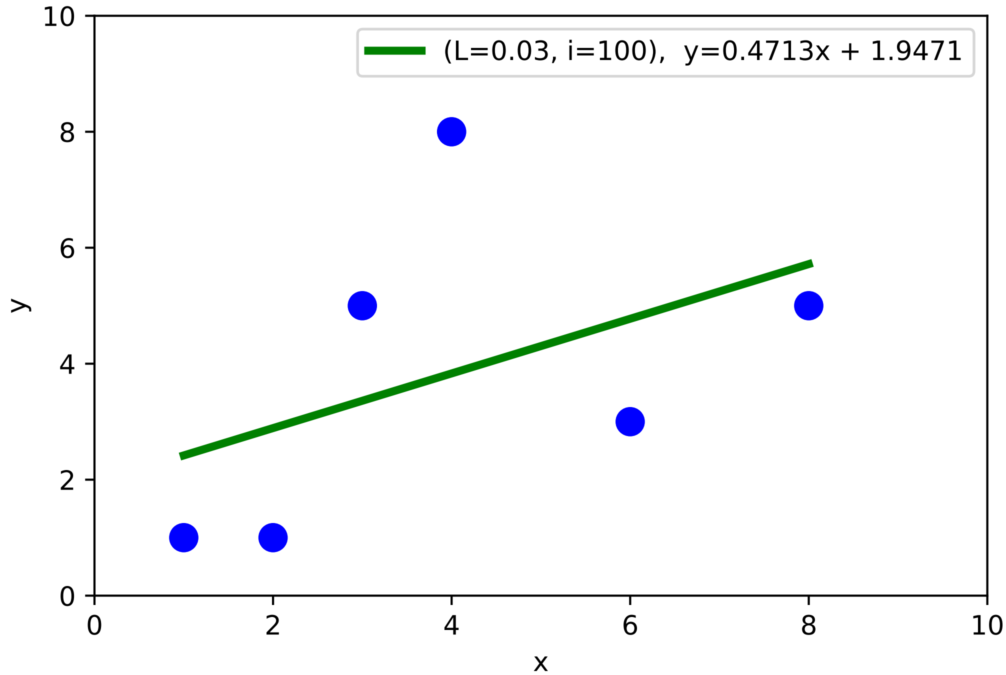


Illustration of Gradient Descent

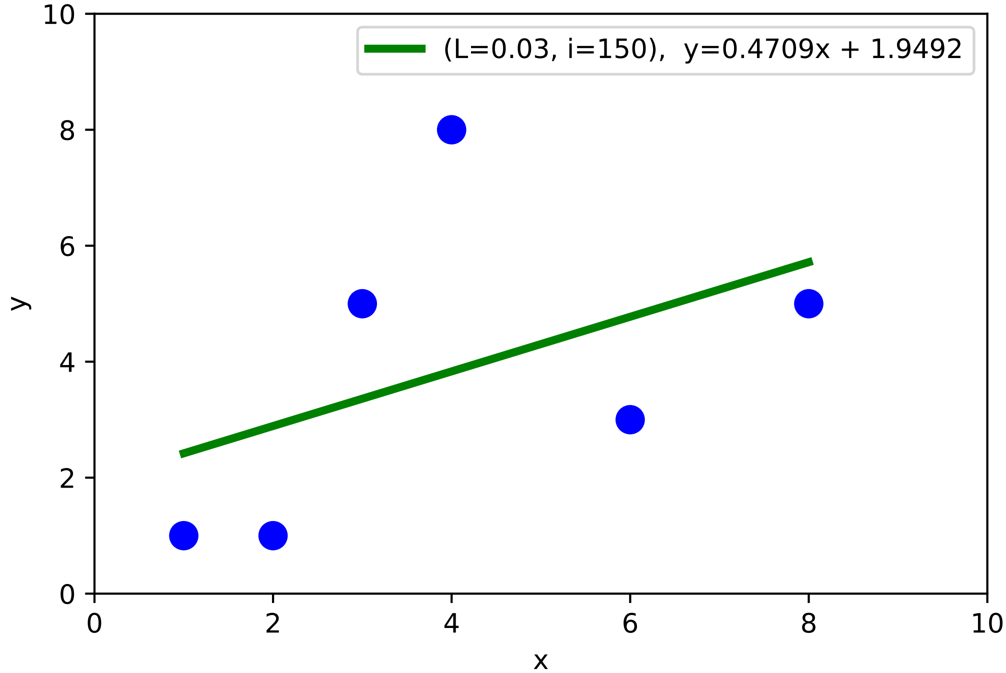


Illustration of Gradient Descent

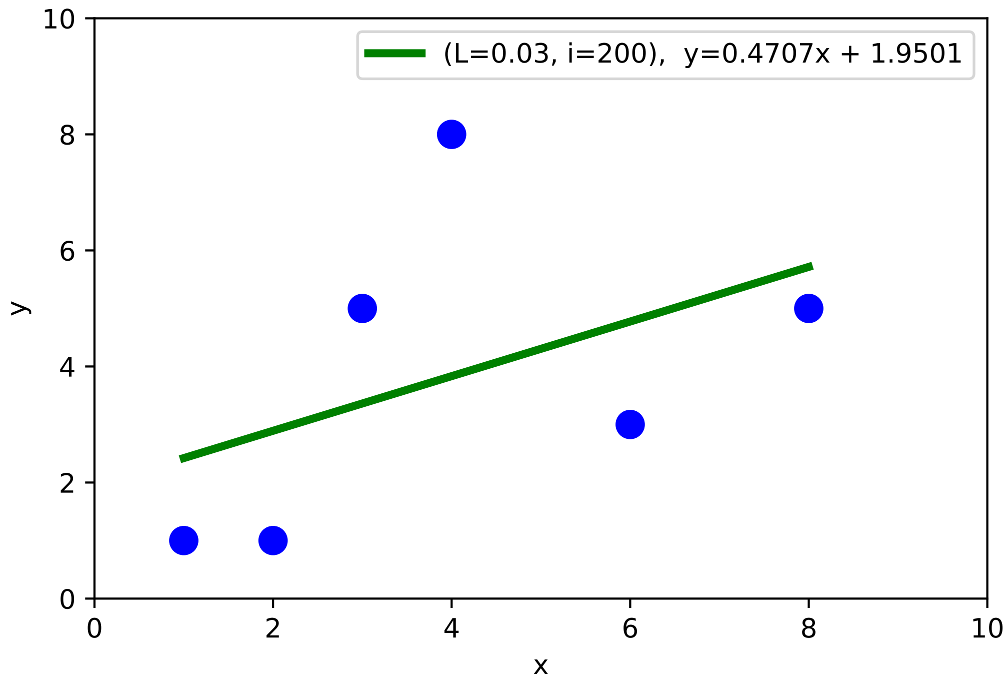


Illustration of Gradient Descent

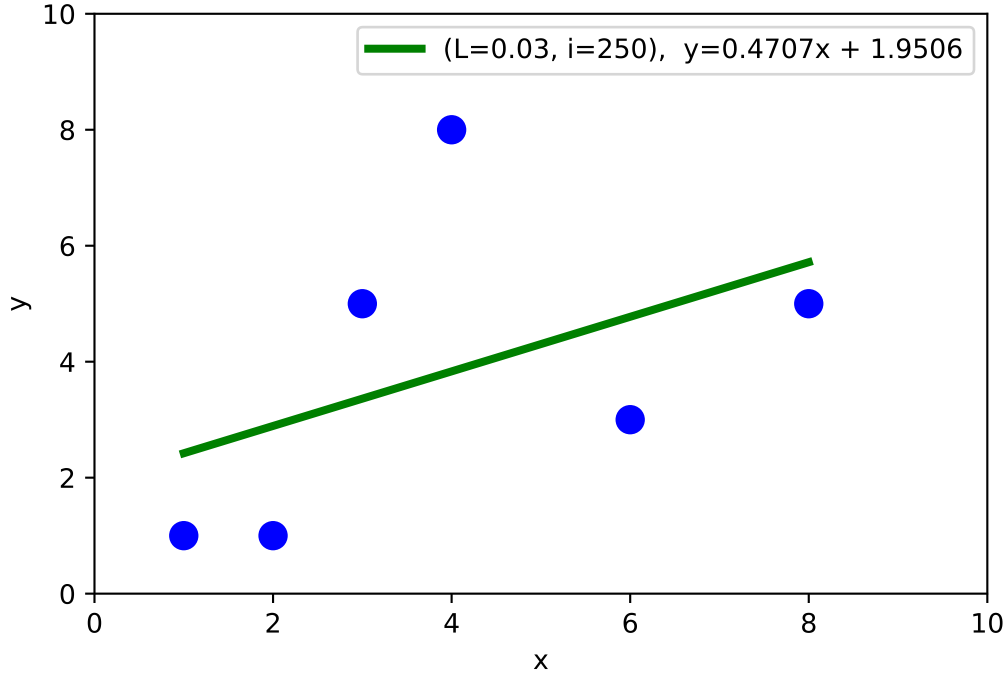


Illustration of Gradient Descent

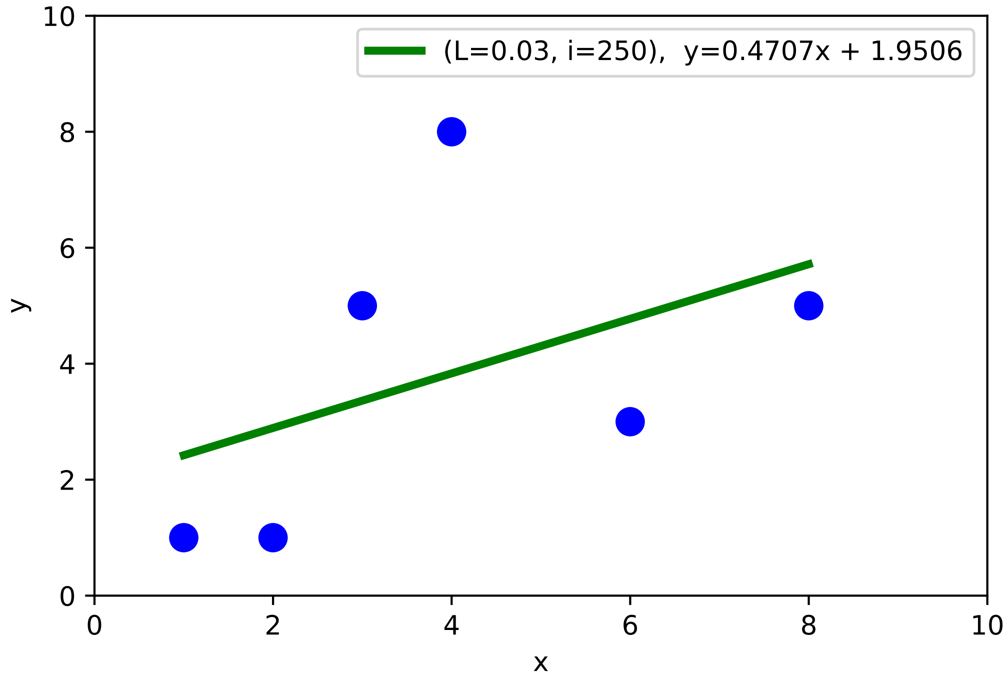


Illustration of Gradient Descent

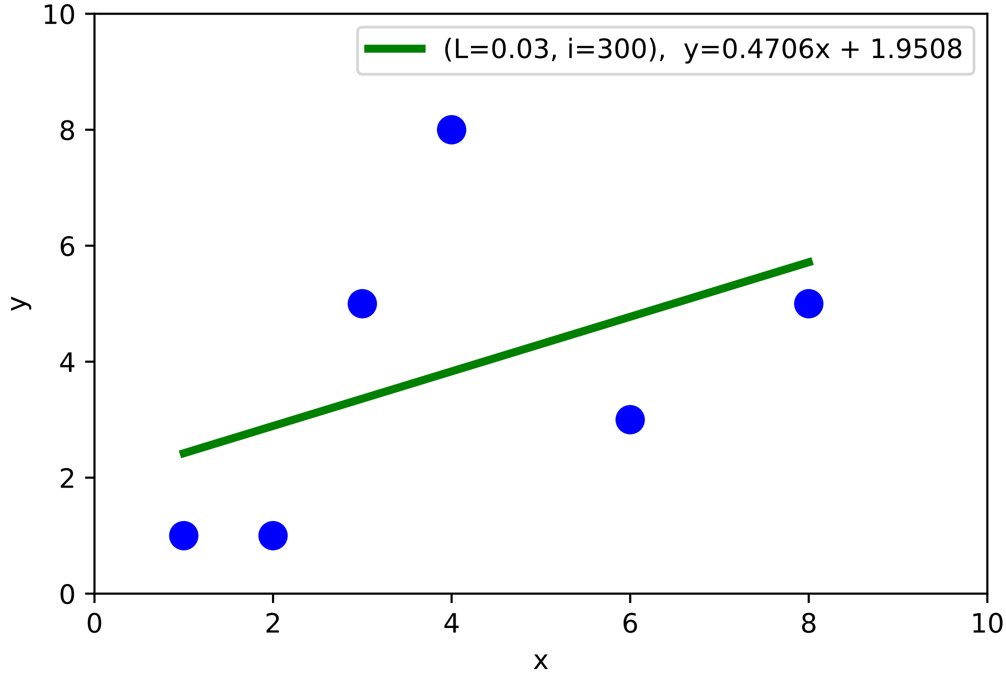


Illustration of Gradient Descent

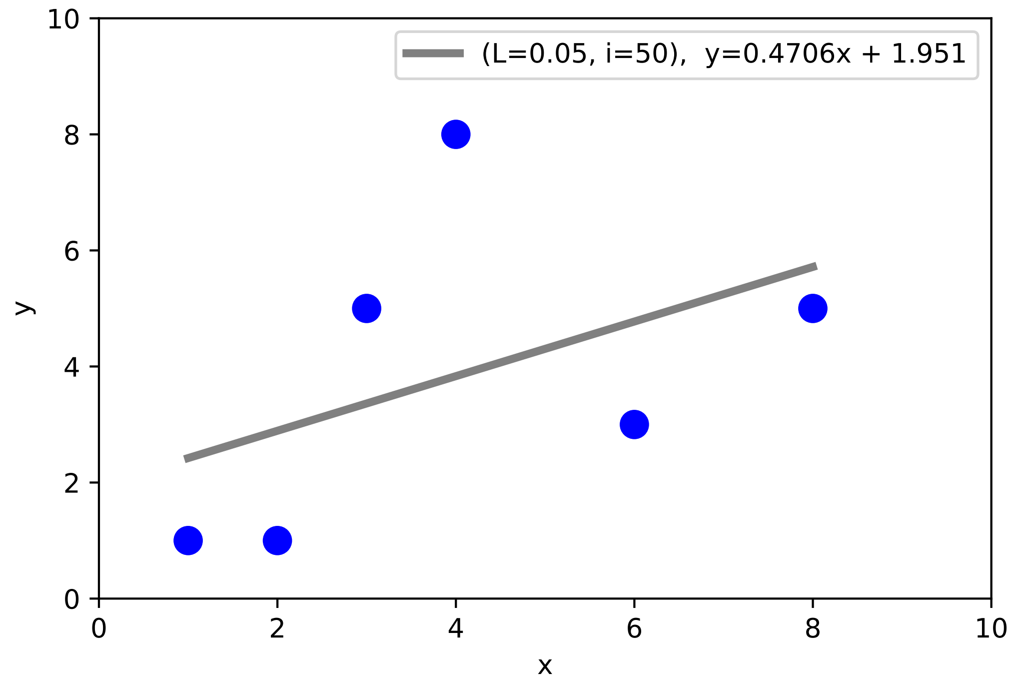


Illustration of Gradient Descent

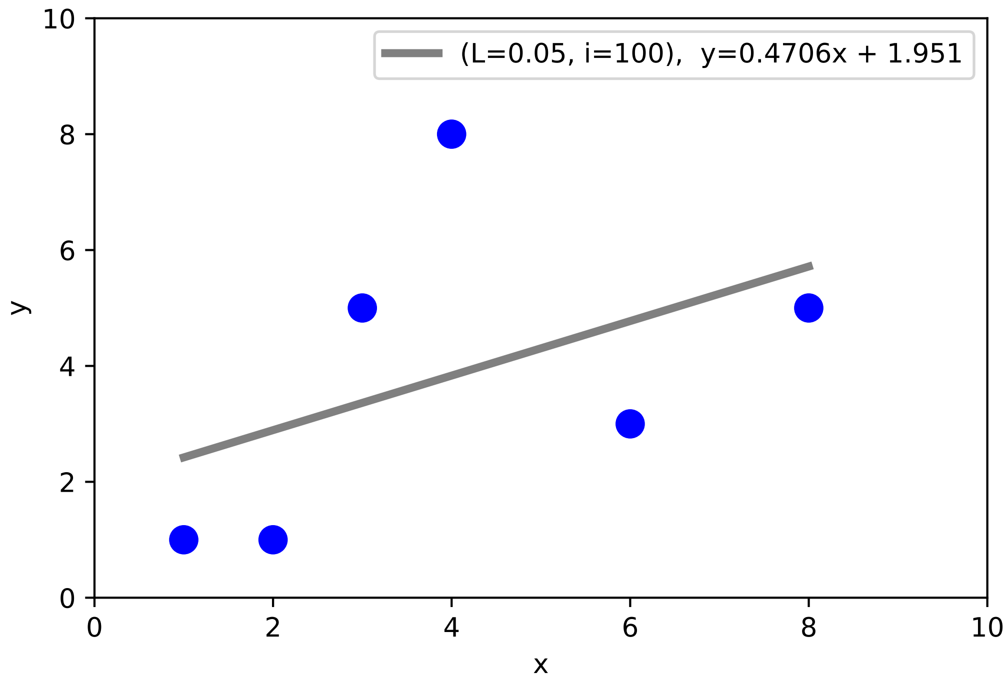


Illustration of Gradient Descent

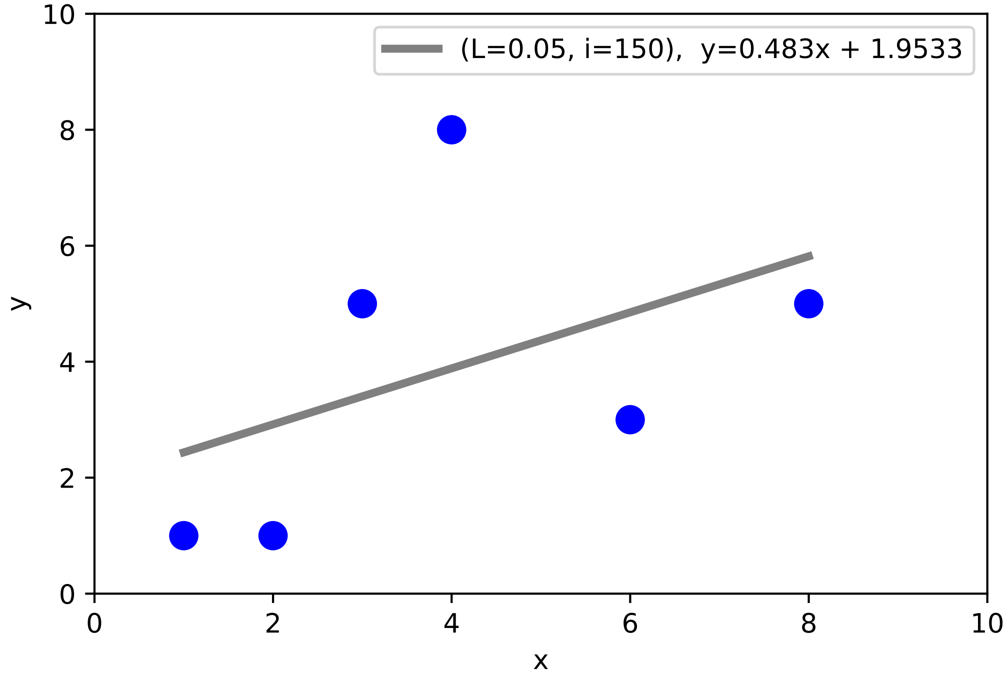


Illustration of Gradient Descent

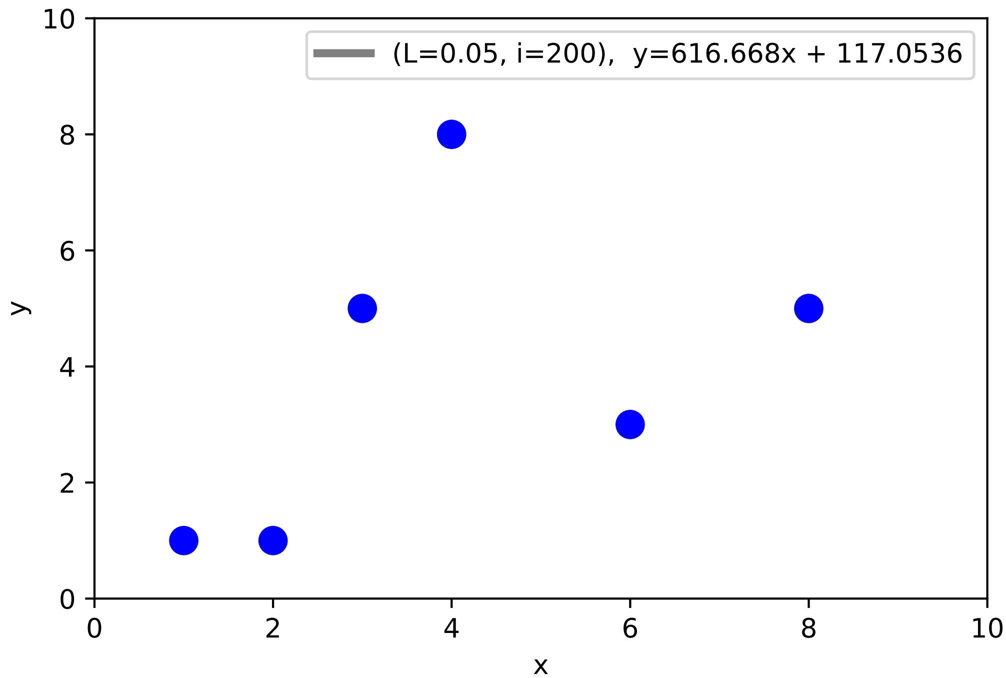


Illustration of Gradient Descent

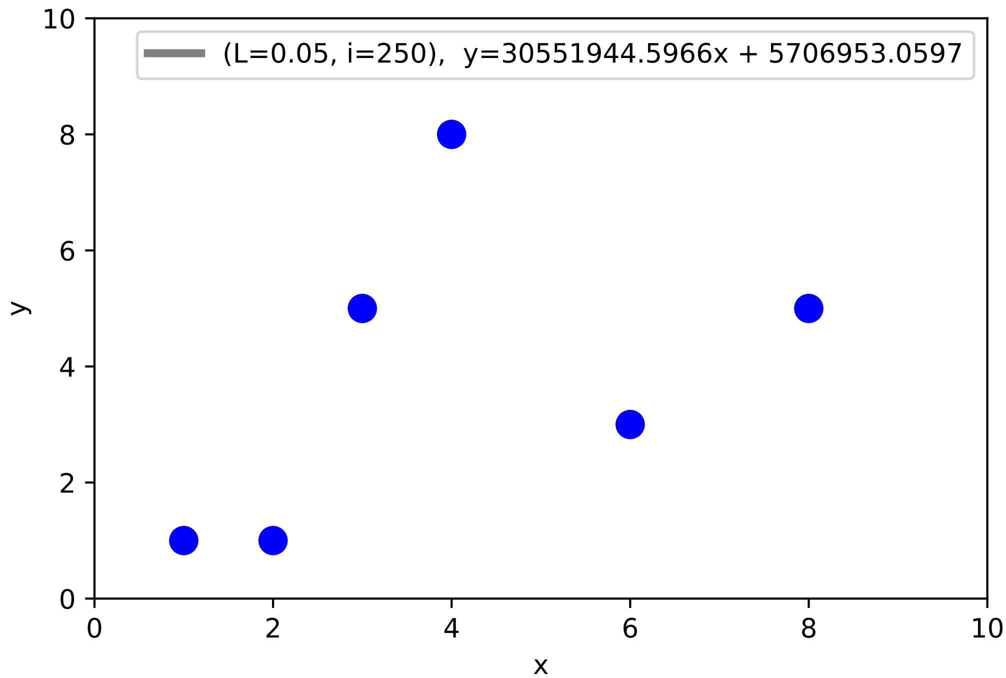


Illustration of Gradient Descent

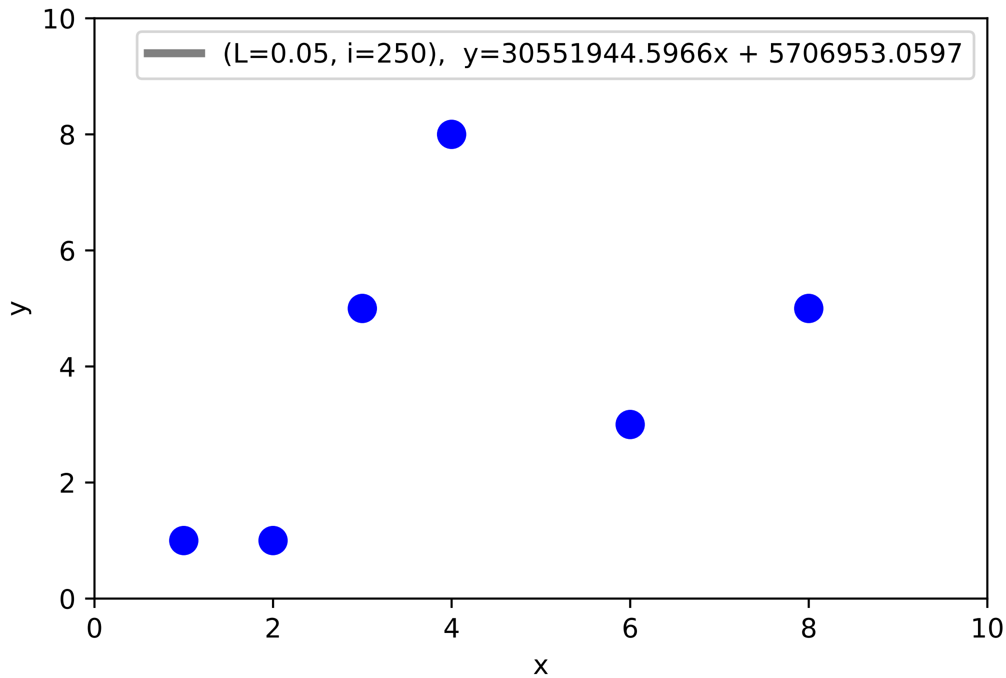


Illustration of Gradient Descent