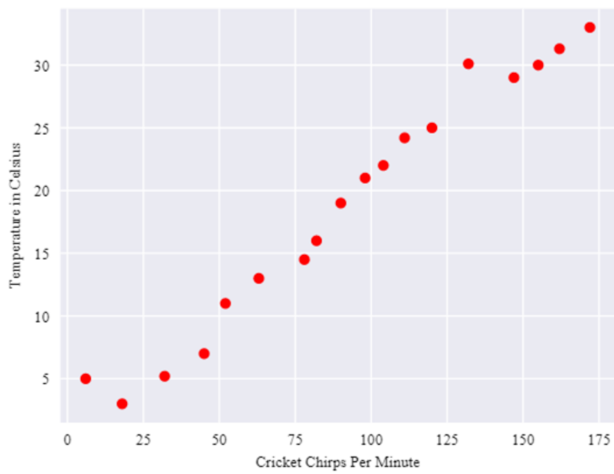
Supervised learning. Prediction

Lecture 14

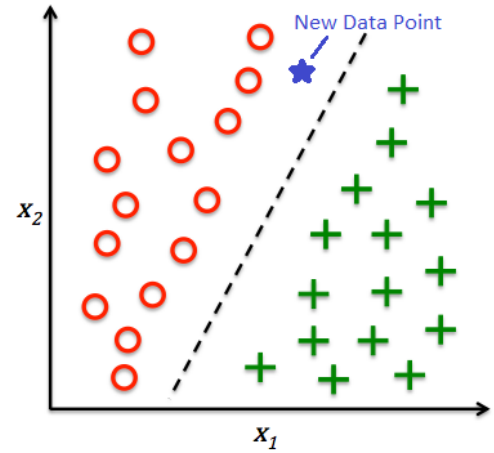
By Marina Barsky

Types of learning tasks

Supervised learning

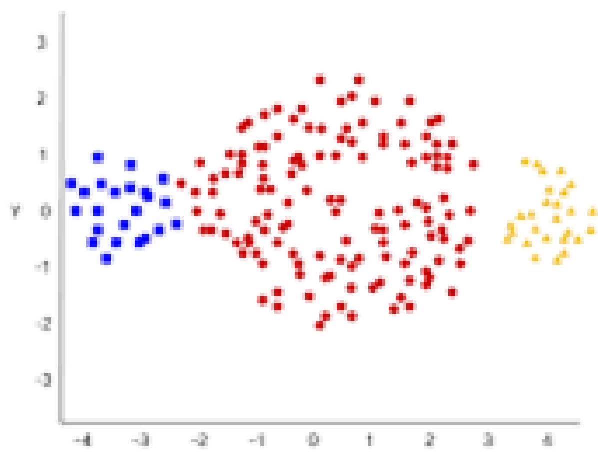


Prediction ■■



Classification ■■

Unsupervised learning



Clustering ■■■

TransactionId	Items
1	{A,C,D}
2	{B,C,D}
3	{A,B,C,D}
4	{B,D}
5	{A,B,C,D}

Associations ■■

The task of numeric prediction

- We are given a dataset containing all numeric attributes (features)
- We select one of the features and call it a target attribute y
- The task is to predict a value of the target attribute given the values of all the other attributes
- To do that we need to discover (learn) a function $y = f(x)$ which models the relationships between x and y in the given dataset

Sample problem: predicting housing prices

1 room	2 rooms	3 rooms	4 rooms	5 rooms
$150K	$200K	???	$300K	$350K

Sample problem: predicting housing prices

1 room	2 rooms	3 rooms	4 rooms	5 rooms
$150K	$200K	???	$300K	$350K

For a sample dataset above the input data looks like that:

Target attribute

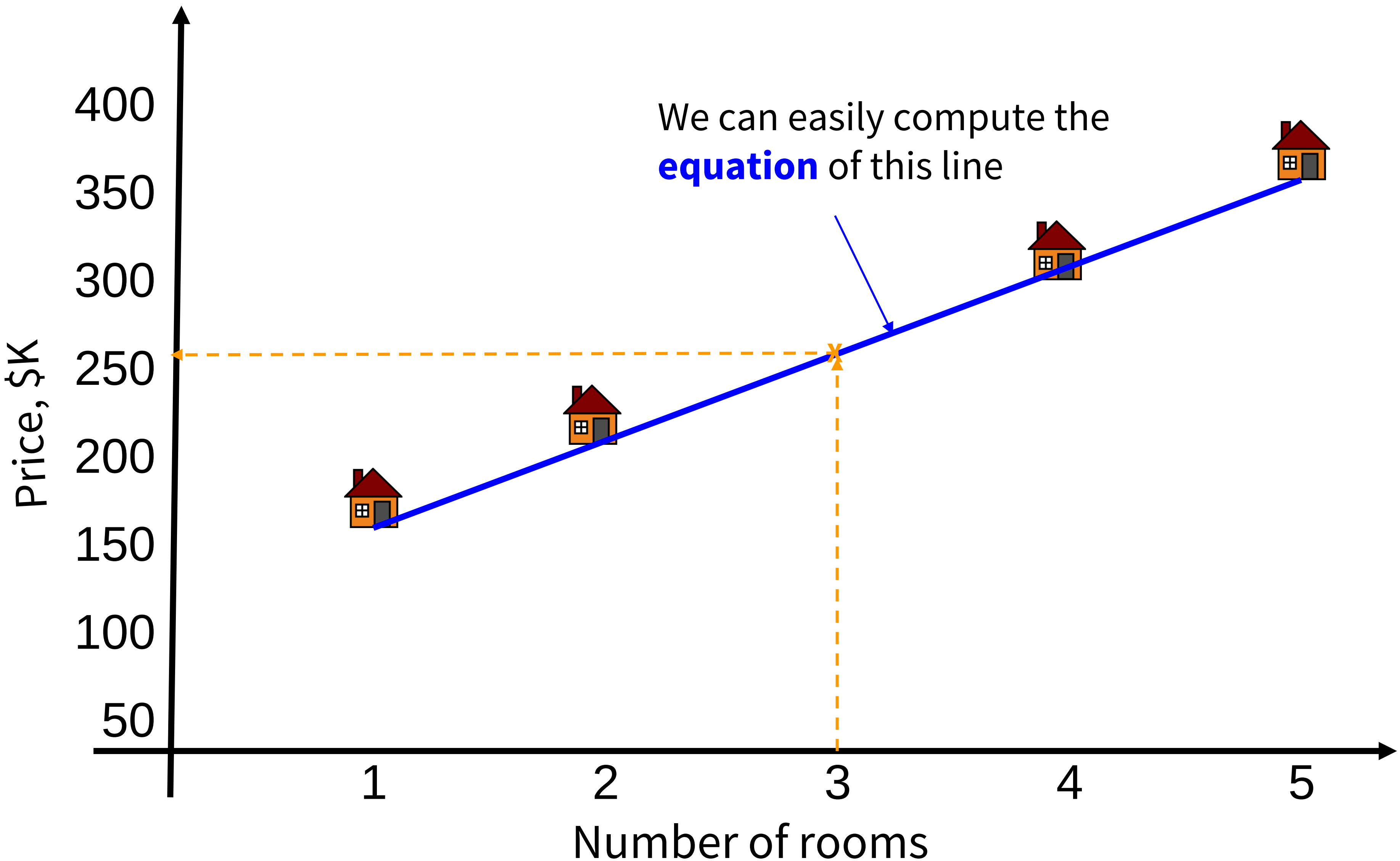
Attributes→	# rooms	price , K
Point 1	1	150
Point 2	2	200
Point 3	4	300
Point 4	5	350

The set of points is called a ***training set***

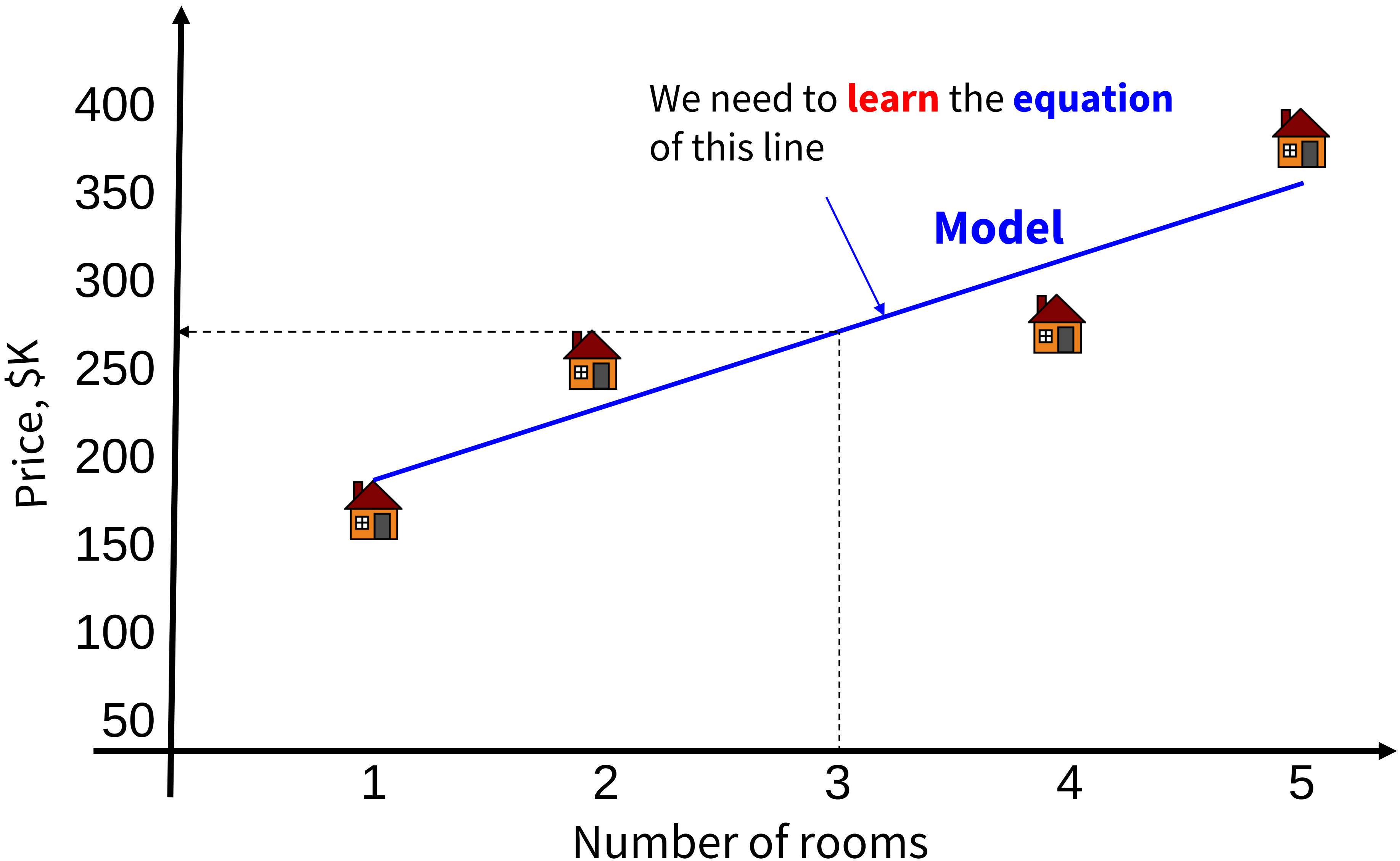
Task: given # rooms = 3, predict price

Linear regression

Price of 3-bedroom home: ideal world



Real world: line that fits the data points best



Simple Linear Regression

We have a set of data points

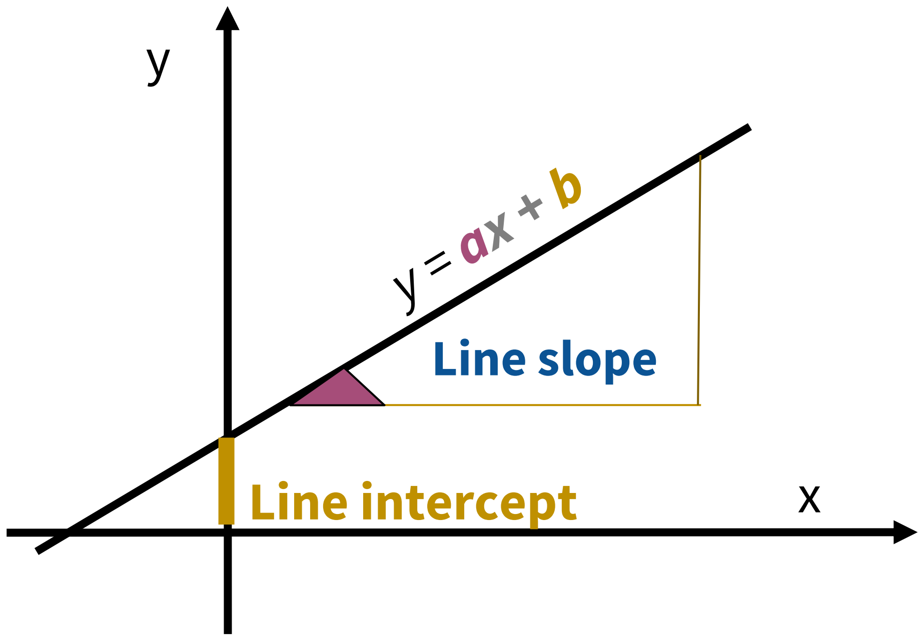
$$\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

Learning task:

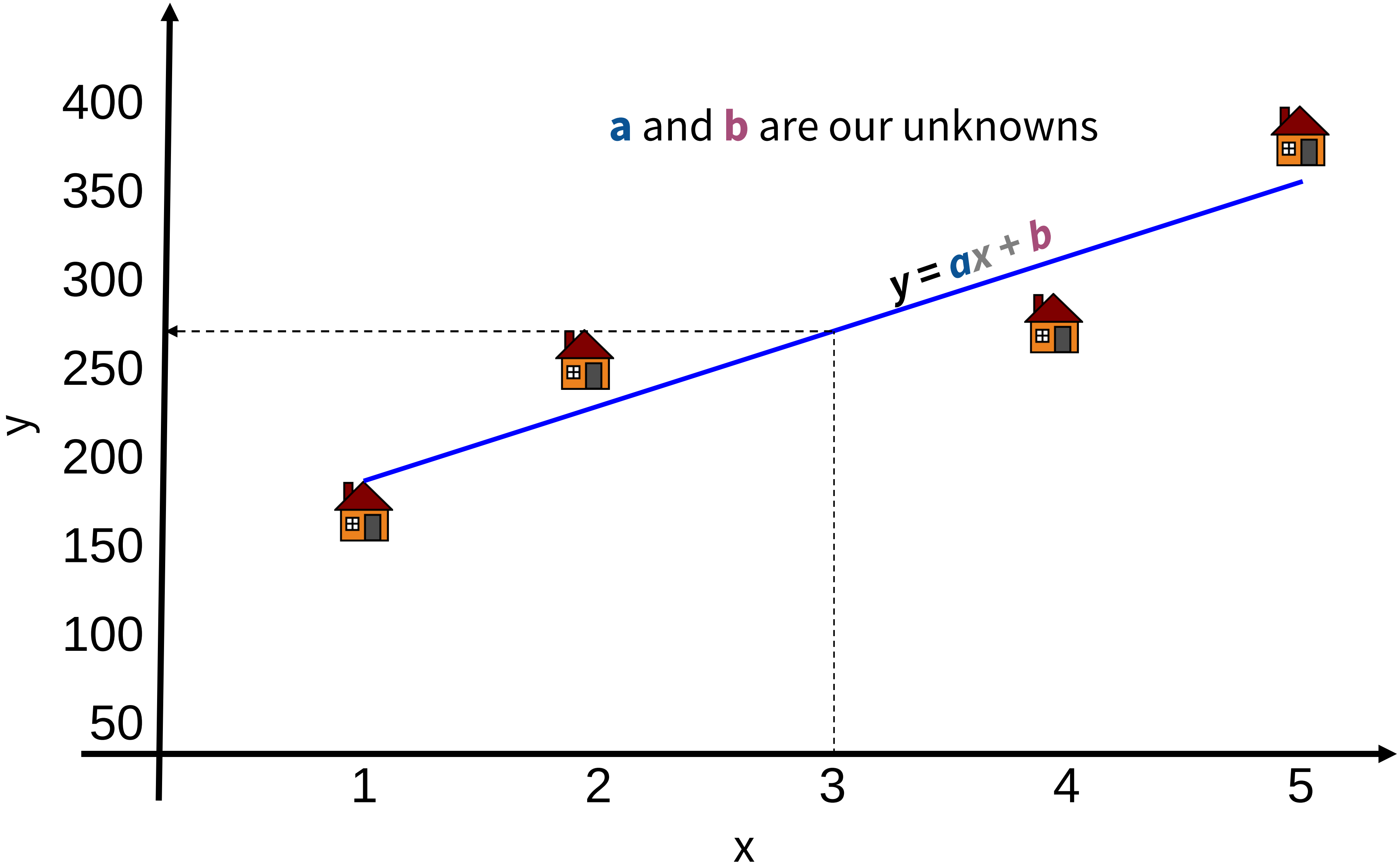
Learn the linear function f(*x*) =***a****x* + ***b***

which best describes linear relationship

between *x* and *y*



Line closest to a set of points



This problem has a closed-form mathematical solution

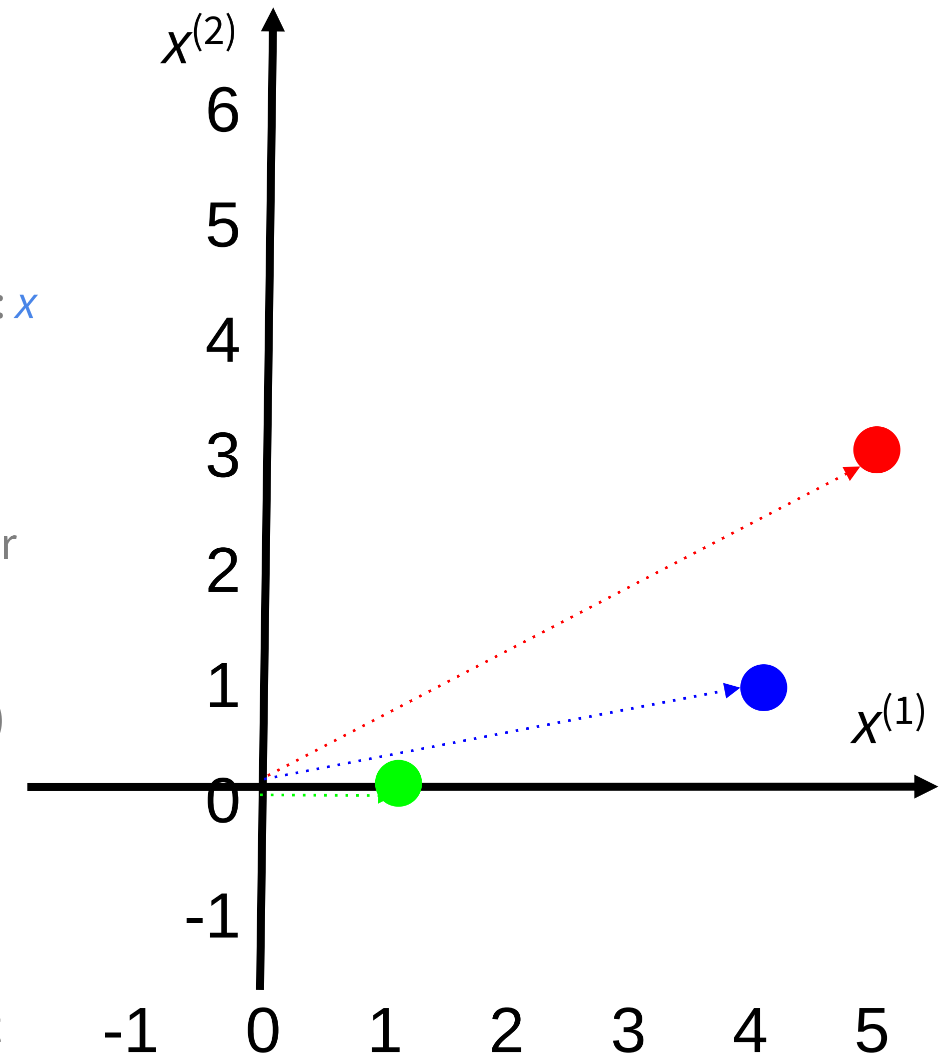
We need vectors

A *scalar* is a simple numerical value, like 15 or -3.25. Variables or constants that take scalar values are denoted by an *italic* letter: x or a.

A **vector** is an ordered list of scalar values. We denote a vector as a **bold** character: $\mathbf{x}$ or $\mathbf{w}$.

We represent each data point (observation) as a vector. Each vector dimension represents a value of some attribute.

We choose one of the attributes to be a ***target*** variable y: that is we want to predict it given a vector of all other attributes.



Vectors: [1,0], [4,1], [5,3]

We need operations on vectors (and matrices)

- Sum of 2 vectors
- Multiplication by scalar
- Dot product
- Multiplication of matrix by vector
- Matrix inverse: https://www.mathsisfun.com/algebra/matrix-inverse.html

Vector multiplication

Let: $\mathbf{a} = (a_1, a_2, a_3, \ldots a_n)$

& $\mathbf{b} = (b_1, b_2, b_3, \ldots b_n)$

$$\boxed{\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + a_3 b_3 + \ldots + a_n b_n}$$

Scalar product or “dot product”

Matrix vector multiplication

$$\begin{bmatrix} A & B \\ C & D \\ E & F \end{bmatrix} \times \begin{bmatrix} G \\ H \end{bmatrix} = \begin{bmatrix} A \times G + B \times H \\ C \times G + D \times H \\ E \times G + F \times H \end{bmatrix}$$

We make use of capital Sigma and capital Pi

$$\sum_{i=1}^{n} x_i = x_1 + x_2 + \cdots + x_n$$

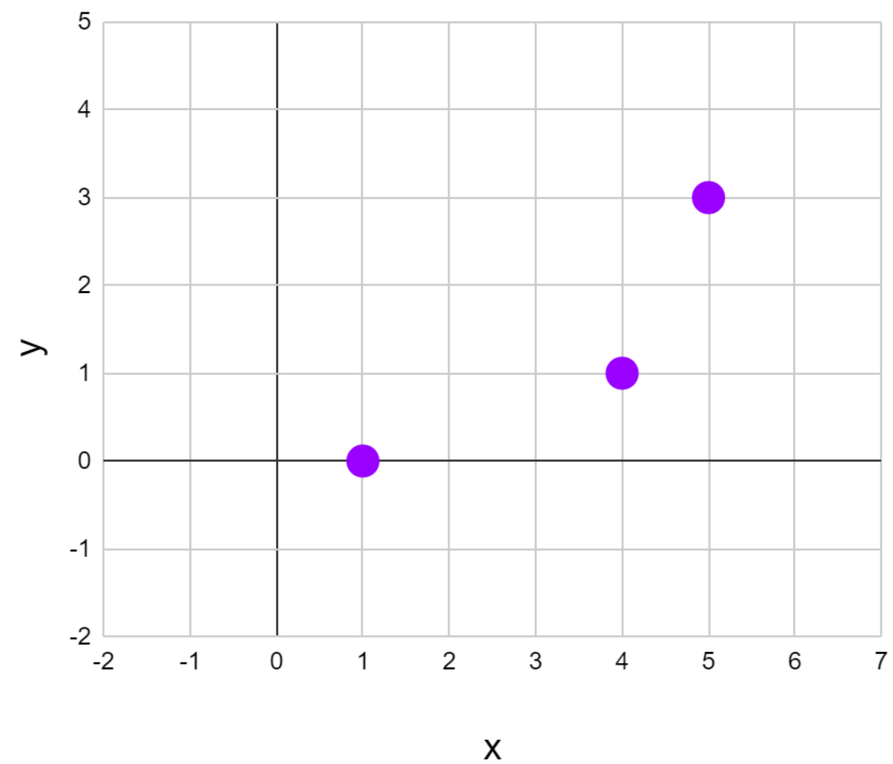
$$\prod_{i=1}^{n} x_i = x_1 \cdot x_2 \dots x_n$$

Example

Find an equation of a line which best describes (fits) data in our training set:

$y = ax + b$

This line is going to serve as a *model* used to predict values of y given x.



Training set: {[1,0], [4,1], [5,3]}

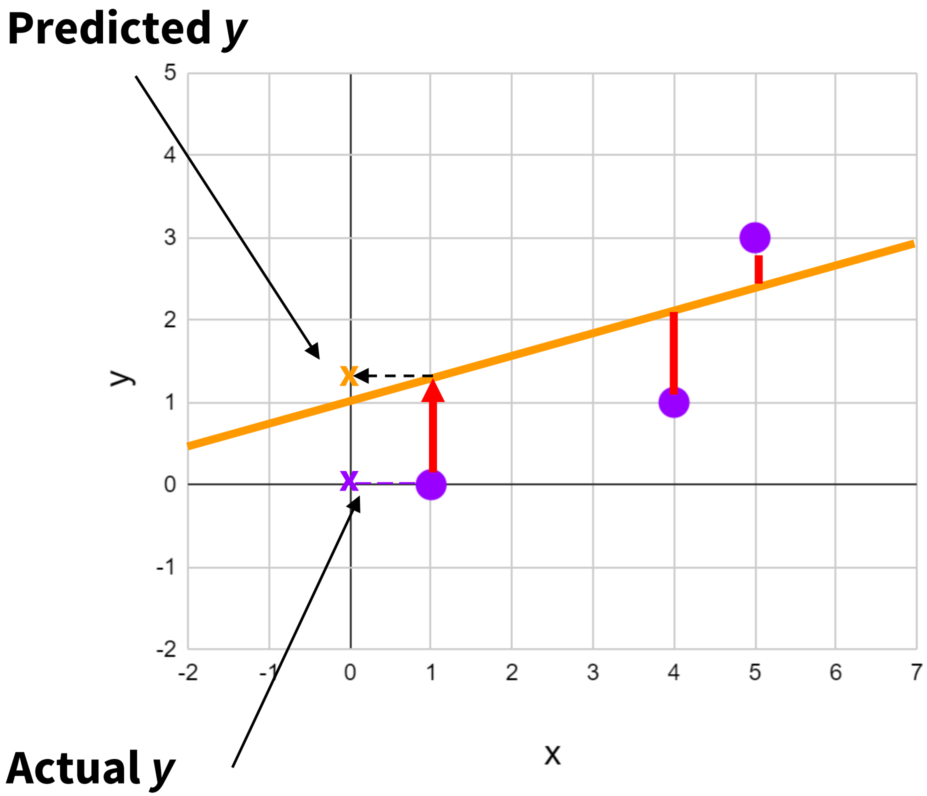
Fitting the line to data

f(x) = $ax + b$

For each value of x we can compare the y value predicted by our model (line) to the actual value of y.

Let's consider an arbitrary line.

The difference between the predicted and actual value is an error of the prediction



Training set: {[1,0], [4,1], [5,3]}

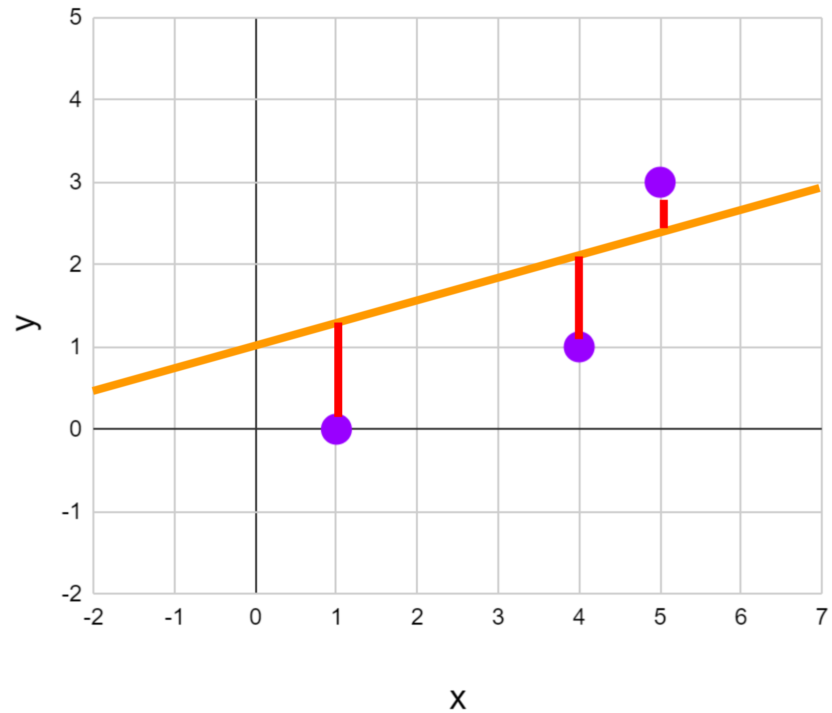
Error of the entire model: SSR

f(x) =*ax* + *b*

The overall error of the model can be defined as a sum over errors for all data points (all red lines).

If the overall error is big, we want to decrease it, by moving the line closer to the data points.

Instead of using a sum of errors we use the **sum of the squared errors** as a measure of success. This makes big errors much bigger, penalizes the overall score more, and we want our line to move to these distant points first



Training set: {[1,0], [4,1], [5,3]}

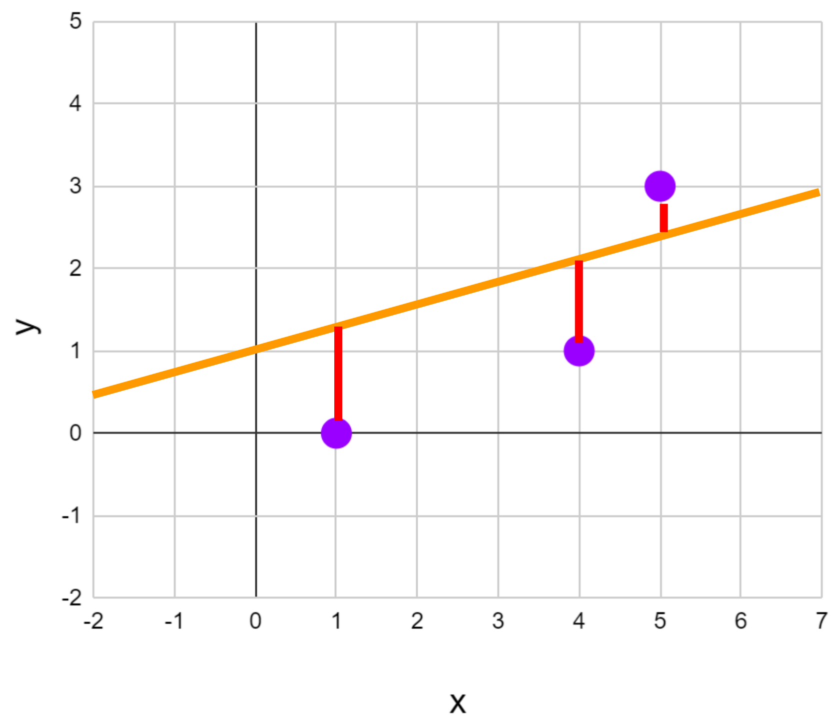
Error of the entire model: SSR

f(x) =$ax + b$

Instead of using a sum of errors we use the **sum of the squared errors**

This sum is called ***SSR*** - Sum of Squared Residuals. It represents an example of a *loss function*: how much information do we lose if we abstract real data by the model

SSR is also an example of the *objective function* for the case of linear regression: our object is to minimize SSR



Training set: {[1,0], [4,1], [5,3]}

Math: problem of least squares

Given data vectors: $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$

we define an error associated with the model f(*x*) = *ax* + *b* by:

$$E(a, b) = \sum_{i=1}^{n} (y_i - (ax_i + b))^2$$

The error is a function of 2 variables: *a* and *b*.

The goal is to find the combination of *a* and *b* that minimizes the overall error

Using calculus

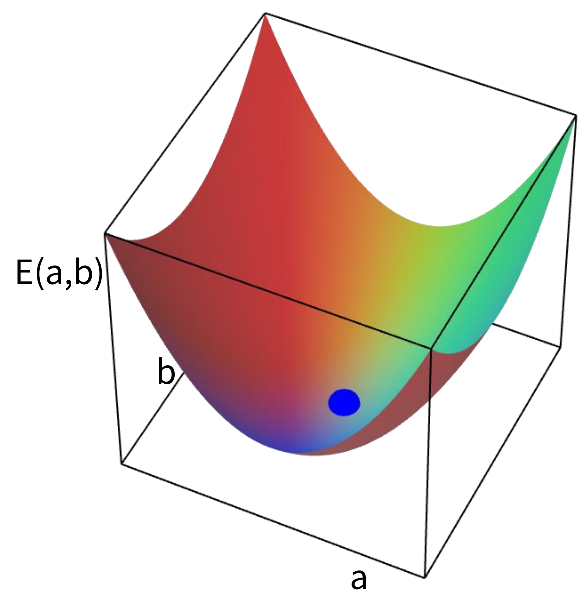
The error is a function of 2 variables: *a* and *b*

$$E(a,b) = \sum_{i=1}^{n} (y_i - (ax_i + b))^2$$

The goal is to find the combination of *a* and *b* that minimize the overall error.

From multivariate calculus we know that this requires us to find the extrema of function E:

$$\frac{\partial E}{\partial a} = 0, \frac{\partial E}{\partial b} = 0$$



Differentiating $E(a,b) = \sum_{i=1}^{n} (y_i - (ax_i + b))^2$

$$\frac{\partial E}{\partial a} = \sum_{i=1}^{n} 2(y_i - (ax_i + b)) \cdot (-x_i)$$

$$\frac{\partial E}{\partial b} = \sum_{i=1}^{n} 2(y_i - (ax_i + b)) \cdot 1$$

Setting this differentials to 0 and dividing by 2 yields:

$$\sum_{i=1}^{n} \big(y_i - (ax_i + b)\big) \cdot x_i = 0$$

$$\sum_{i=1}^{n} \big(y_i - (ax_i + b)\big) = 0$$

System of 2 linear equations with 2 unknowns

$$\begin{cases} \sum_{i=1}^{n} \big(y_i - (ax_i + b)\big) \cdot x_i = 0 \\ \sum_{i=1}^{n} \big(y_i - (ax_i + b)\big) = 0 \end{cases}$$

We may rewrite these equations as:

$$\begin{pmatrix} \sum_{i=1}^{n} {x_i}^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} y_i \end{pmatrix}$$

Thus we need to solve a system of linear equations represented by this matrix

Solution: best values for *a* and *b*

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} {x_i}^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} 1 \end{pmatrix}^{-1} \begin{pmatrix} \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} y_i \end{pmatrix}$$

This is the closed mathematical solution to the least squares problem

More math details: here
Another step-by-step example: here

Step-by-step example 1/3

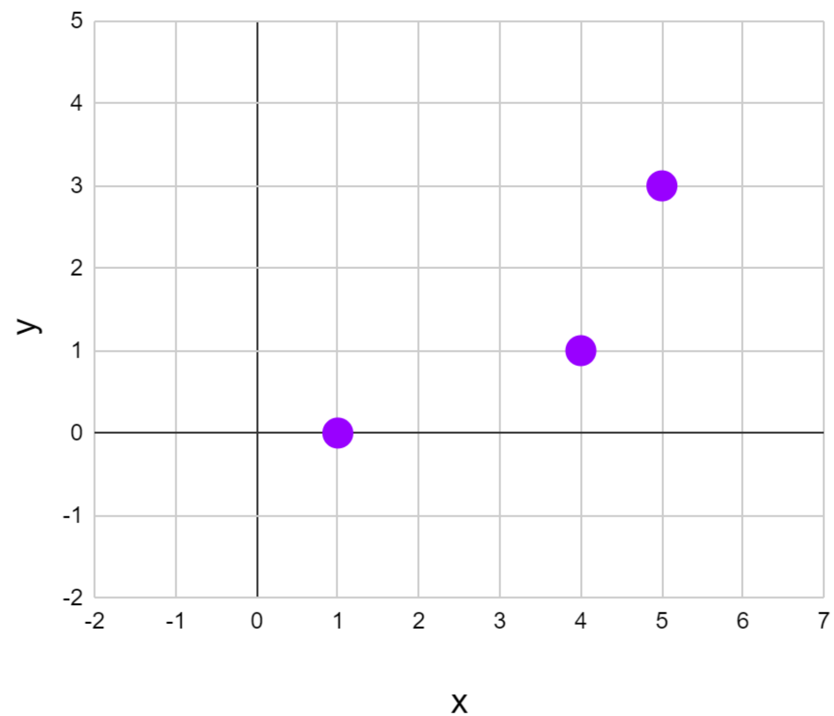
$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} {x_i}^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} 1 \end{pmatrix}^{-1} \begin{pmatrix} \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} y_i \end{pmatrix}$$

$$\sum_{i=1}^{n} {x_i}^2 = 1^2 + 4^2 + 5^2 = 42$$

$$\sum_{i=1}^{n} x_i = 1 + 4 + 5 = 10$$

$$\sum_{i=1}^{n} x_i \, y_i = 1 \cdot 0 + 4 \cdot 1 + 5 \cdot 3 = 19$$

$$\sum_{i=1}^{n} y_i = 0 + 1 + 3 = 4$$



Training set: {[1,0], [4,1], [5,3]}

Step-by-step example 2/3

Inverse of a matrix

$$\begin{bmatrix} 4 & 7 \\ 2 & 6 \end{bmatrix}^{-1} = \frac{1}{4\text{x}6-7\text{x}2}\begin{bmatrix} 6 & -7 \\ -2 & 4 \end{bmatrix} = \frac{1}{10}\begin{bmatrix} 6 & -7 \\ -2 & 4 \end{bmatrix} = \begin{bmatrix} 0.6 & -0.7 \\ -0.2 & 0.4 \end{bmatrix}$$

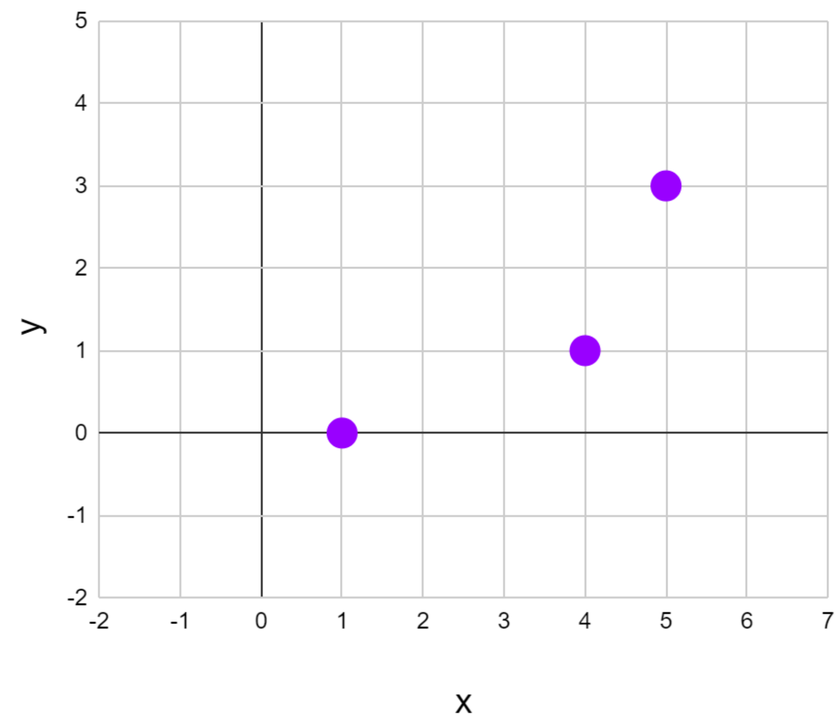
$$\binom{a}{b} = \begin{pmatrix} \sum_{i=1}^{n} {x_i}^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} 1 \end{pmatrix}^{-1} \begin{pmatrix} \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} y_i \end{pmatrix}$$

$$\binom{a}{b} = \begin{pmatrix} 42 & 10 \\ 10 & 3 \end{pmatrix}^{-1} \begin{pmatrix} 19 \\ 4 \end{pmatrix}$$

$$det = 42 * 3 - 10 * 10 = 26$$

$$\begin{pmatrix} 42 & 10 \\ 10 & 3 \end{pmatrix}^{-1} = \frac{1}{26}\begin{pmatrix} 3 & -10 \\ -10 & 42 \end{pmatrix}$$

$$\binom{a}{b} = \frac{1}{26}\begin{pmatrix} 3 & -10 \\ -10 & 42 \end{pmatrix} \begin{pmatrix} 19 \\ 4 \end{pmatrix}$$



Training set: {[1,0], [4,1], [5,3]}

Step-by-step example 3/3

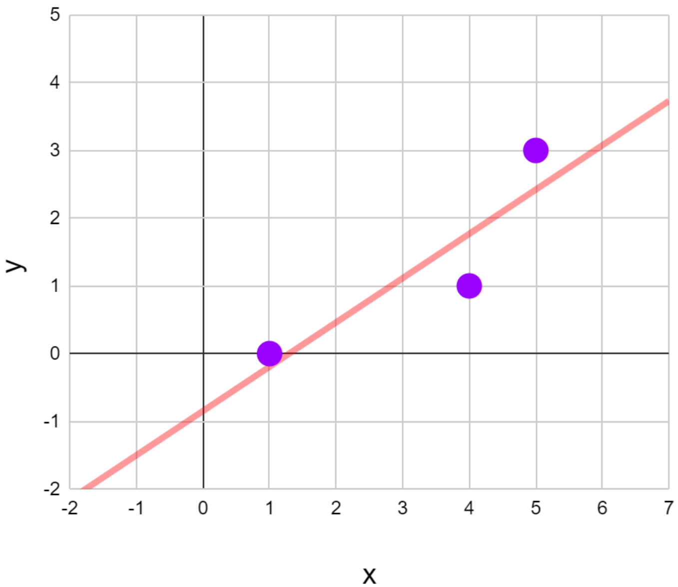
$$\binom{a}{b} = \begin{pmatrix} \sum_{i=1}^{n} {x_i}^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} 1 \end{pmatrix}^{-1} \begin{pmatrix} \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} y_i \end{pmatrix}$$

$$\binom{a}{b} = \frac{1}{26}\begin{pmatrix} 3 & -10 \\ -10 & 42 \end{pmatrix}\begin{pmatrix} 19 \\ 4 \end{pmatrix}$$

$$\binom{a}{b} = \frac{1}{26}\begin{pmatrix} 3 * 19 - 10 * 4 \\ -10 * 19 + 42 * 4 \end{pmatrix} = \frac{1}{26}\begin{pmatrix} 17 \\ -22 \end{pmatrix} = \begin{pmatrix} 17/26 \\ -22/26 \end{pmatrix} \approx \begin{pmatrix} 0.65 \\ -0.85 \end{pmatrix}$$

Best fitting line (model):

f(x) = **0.65**x **- 0.85**



Training set: {[1,0], [4,1], [5,3]}

Linear regression: summary

The equation of the line can be computed from the formula, thus learning the simple linear regression model is very fast

The same solution can be applied to the case when data vectors have D dimensions. This is called multiple linear regression: we find an equation of a hyperplane of form:

$f(\mathbf{x}) = b + w_1x_1 + w_2x_2 + \ldots + w_Dx_D$

We can also construct polynomial features and find the equation of a best-fitting polynomial function: polynomial regression

Is our model good?

How good is the model?
Coefficient of determination

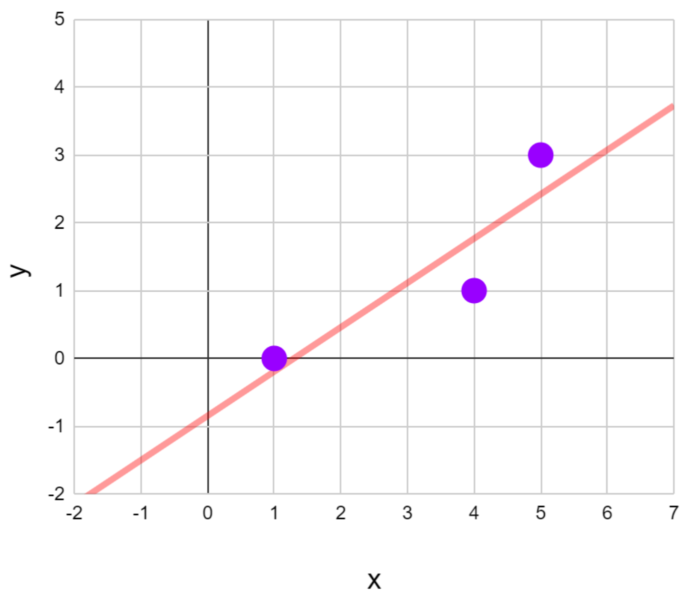
We did the best we can, but is it a good line for modeling the given data?

To tell how good the model describes the data, we use the ***coefficient of determination*** R^2, which tells how much variation in y is explained by the variance in $\mathbf{x}$, according to our regression model.

Larger R^2 indicates a better model. The value $R^2=1.0$ corresponds to $SSR=0$: the model explained all the variance in y by the changes in $\mathbf{x}$, perfect fit.

Our model:

$f(x) = 0.65x - 0.85$

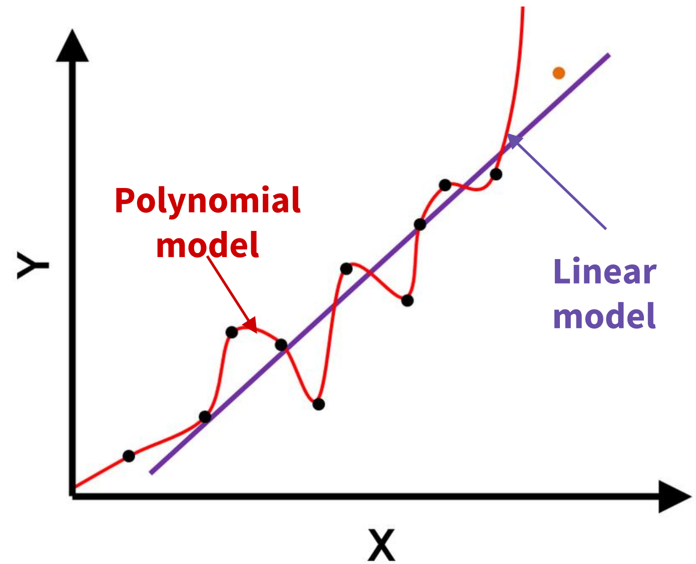


Training set: {[1,0], [4,1], [5,3]}

How good is the model? New data

If we find that the model describes the training data well, we still want to know how would it perform on a new data?

- We need another ***testing datataset*** which was not used for model building
- We break data into 2 sets: **training** and **testing**
- We build the model using **only training** dataset
- Then we run testing data through the model and compute coefficient of determination R^2
- If the model performs much worse on the testing than on the training set - we conclude that the model ***overfitted*** the training data - it ***failed to generalize***



Example of overfitting

Regression demo with sklearn: Predicting house prices (for real)

https://github.com/mgbarsky/demos_ml_regression