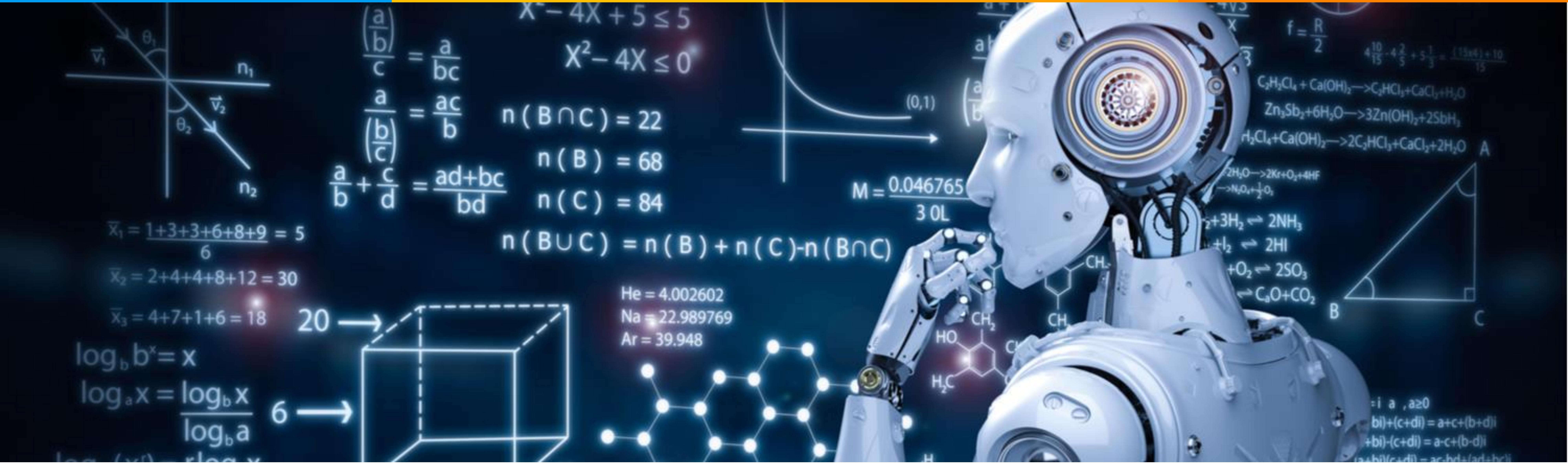
Duration: 60 min | Srikanth Shenoy | AML 5102 | M.E. AI & ML



Lecture 21 & 22: Linear Regression Part 1

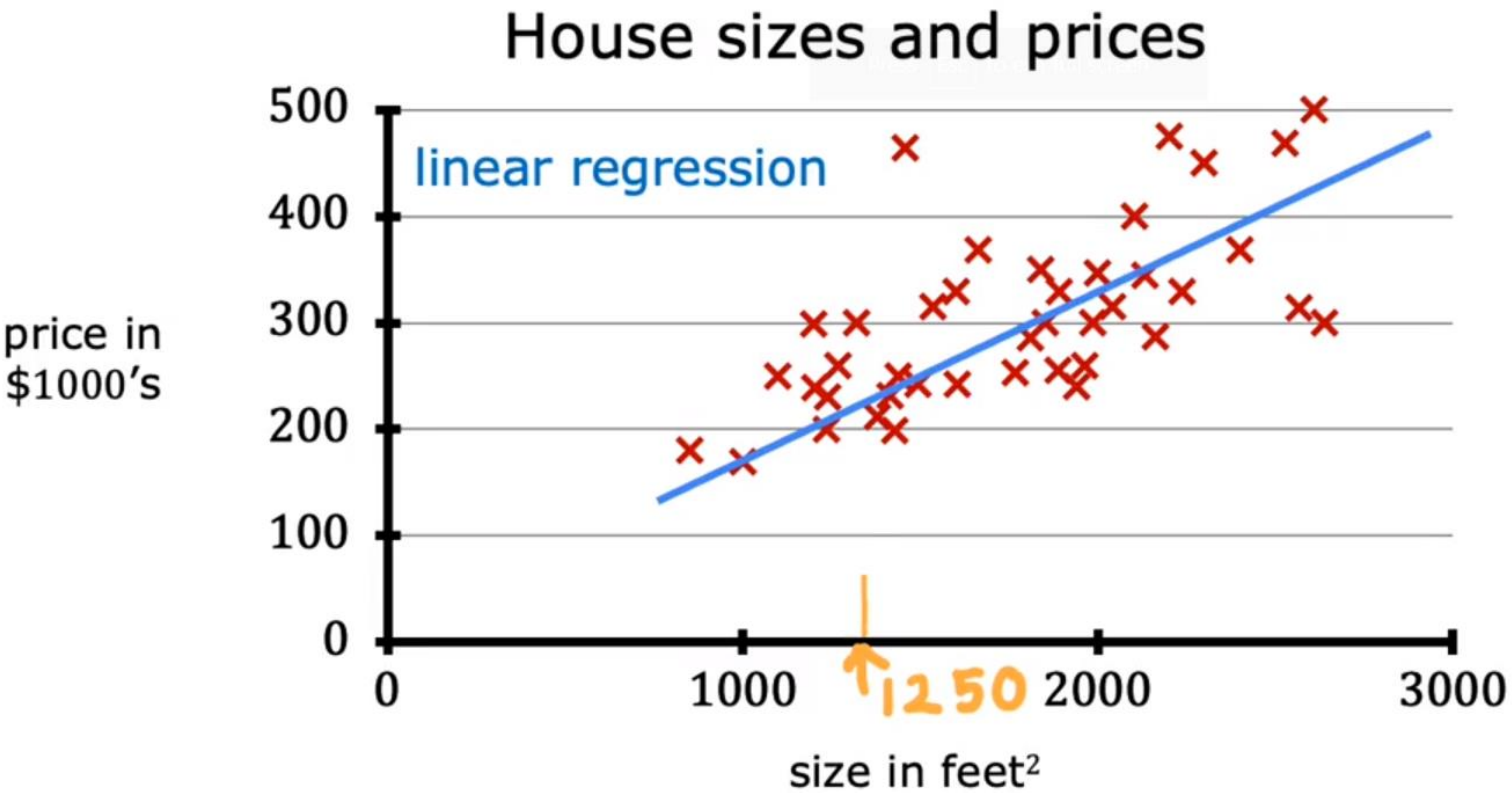
Recap

- Evaluation metrics
- Entropy
- Join Entropy
- Conditional Entropy
- Mutual Information (Information Gain)
- Use in Decision Trees

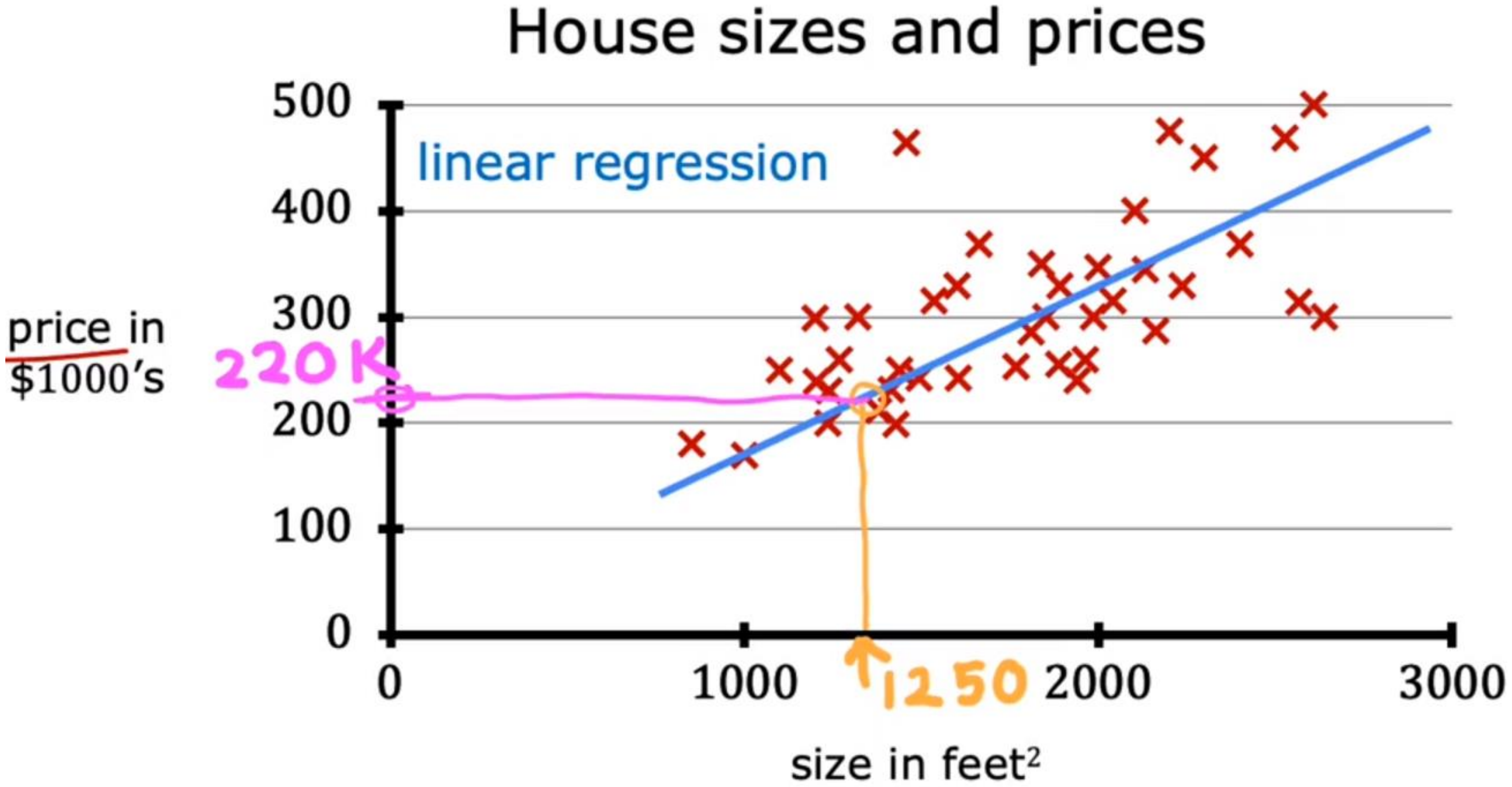
Simple Linear Regression

Univariate Linear Regression

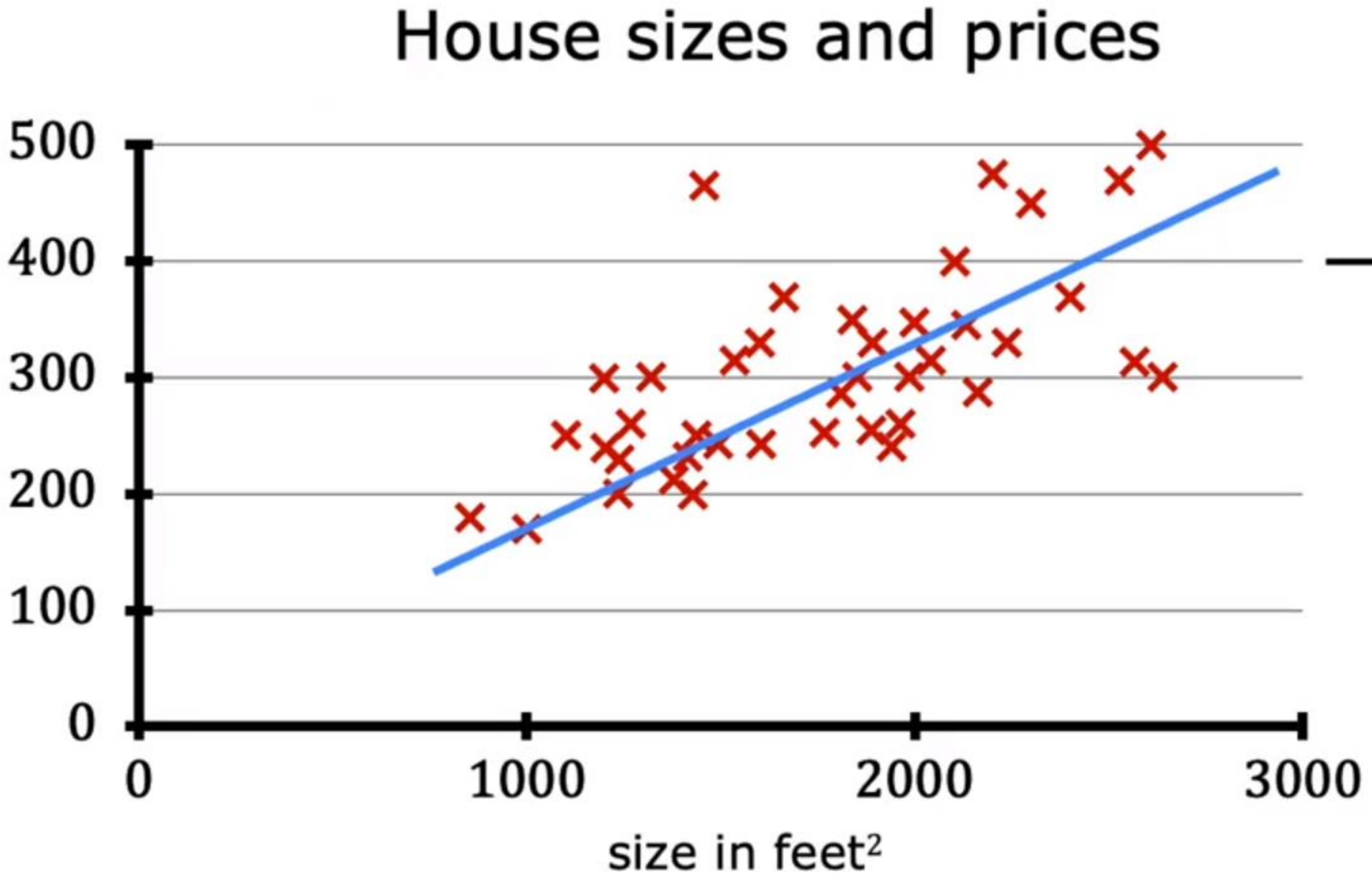
Univariate Linear Regression



Univariate Linear Regression



Univariate Linear Regression



Data table

size in feet^2	price in \$1000's
2104	400
1416	232
1534	315
852	178
...	...
3210	870

Population versus Sample

- Population Regression Function
  - Deterministic Component $y = f(x)$
  - Stochastic Component $y = f(x) + \epsilon$
- Normally distributed error component
- Univariate function $y = wx + b + \epsilon$
- Multivariate function

$$y = w_n x_n + \ldots . w_1 x_1 + w_0 + \epsilon$$
$$= \mathbf{w}^T \mathbf{x} + \epsilon$$

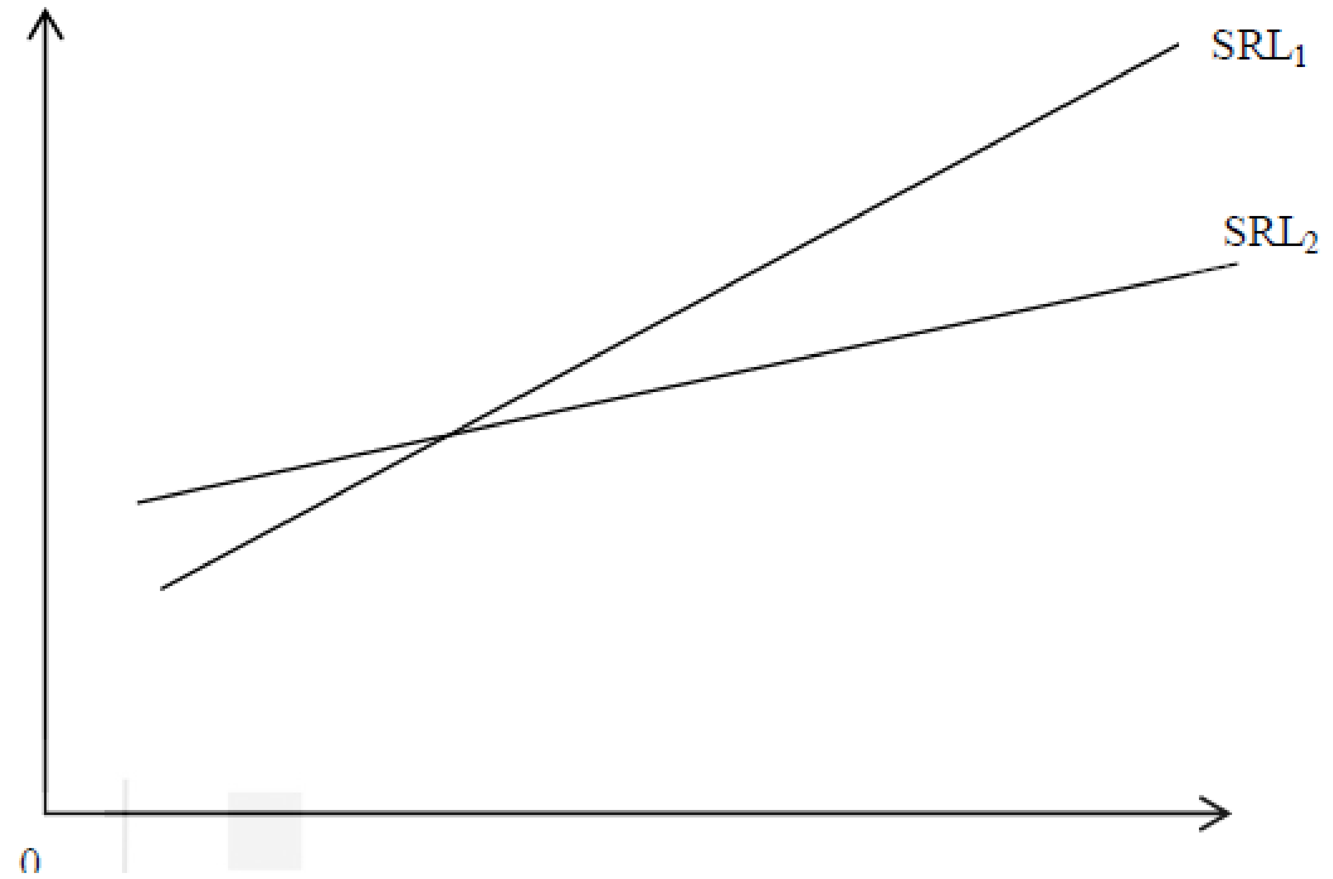
- Population Regression Line/Plane/Hyperplane

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ .. \\ w_n \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ .. \\ x_n \end{bmatrix}$$

Population versus Sample

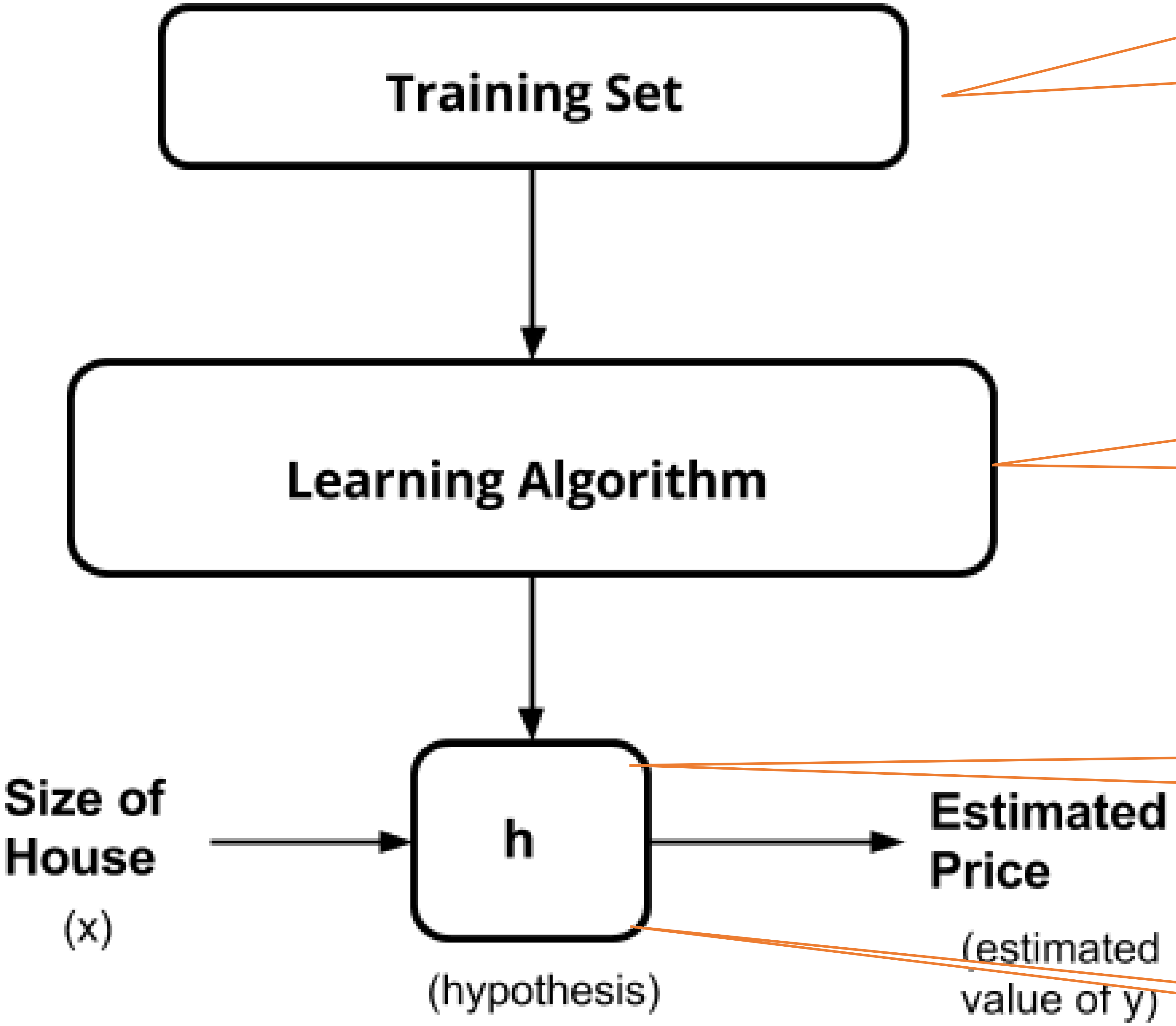
- Sample Regression Functions
  - Different Regression Line/Plane/Hyperplane



Hypothesis function(s)

- Univariate function $\hat{y} = h(x) = wx + b$
- Multivariate function $\hat{y} = h(x) = \mathbf{w}^T\mathbf{x}$

Hypothesis function(s)



Different training sets results in different parameters of the hypothesis function

Parametric form of hypothesis function determined before learning

Parameters are learnt during training

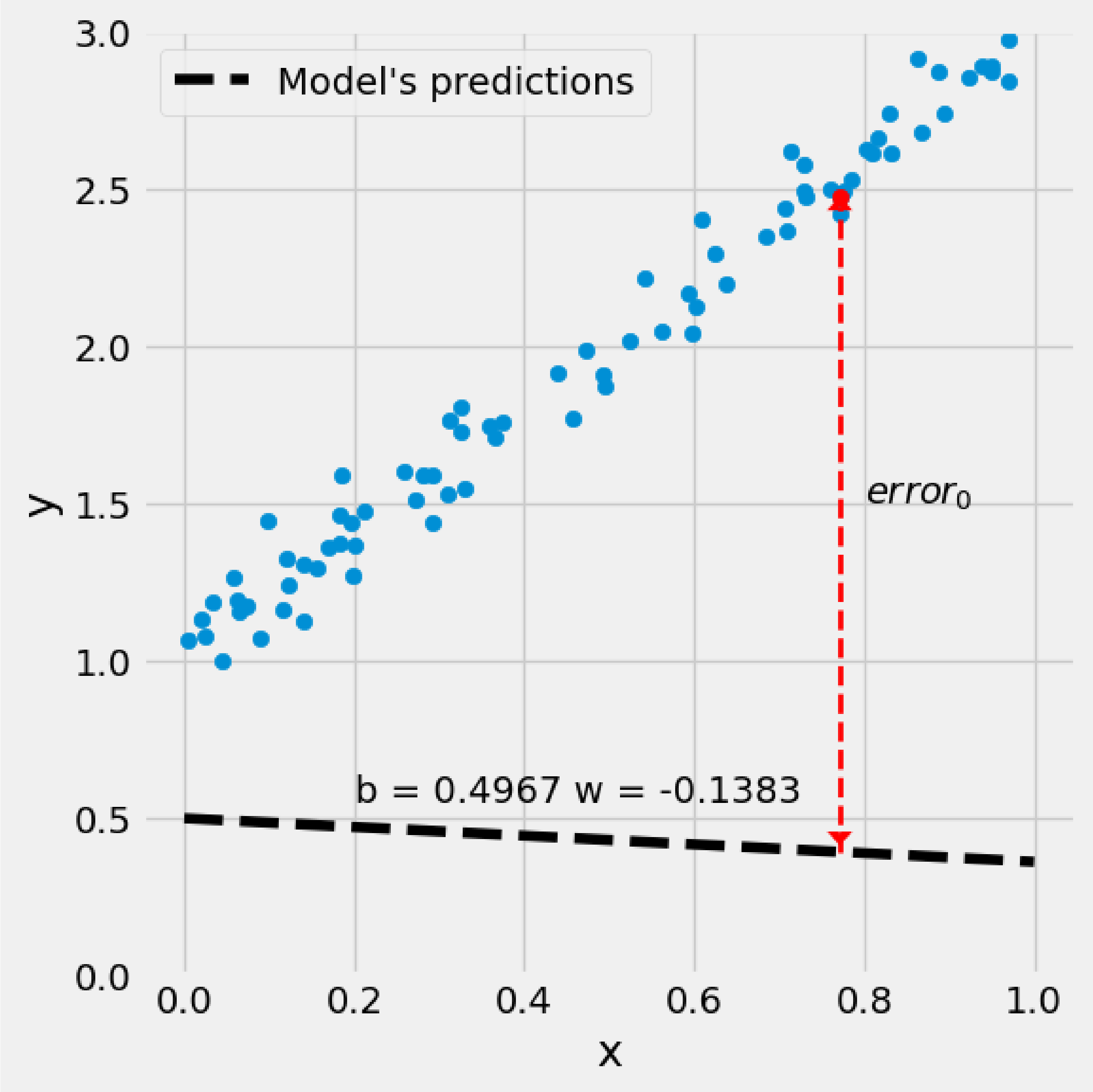
Parameters itself is model

Linear Regression algorithm

Step 1. Initialization

- Assume parametric form $\hat{y} = wx + b$
- Assign random values for w and b

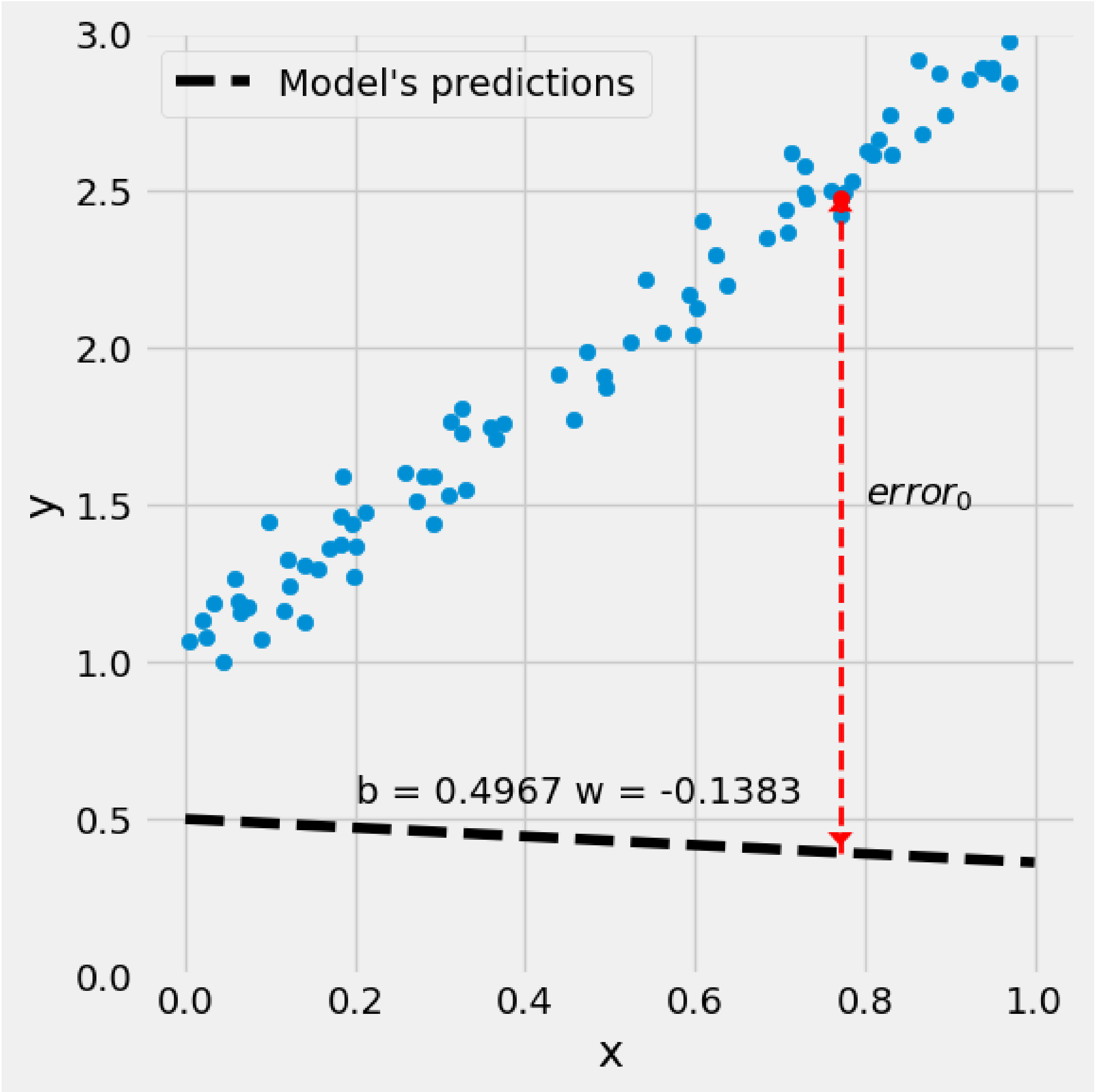
Parametric form of the Hypothesis function



- Calculate error using the initial w & b

$$error^{(i)} = \hat{y}^{(i)} - y^{(i)}$$

Formulate Objective function



$$error^{(i)} = \hat{y}^{(i)} - y^{(i)}$$

Also called cost / loss function

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{error}^{(i)^2}$$

How do I quantify my unhappiness

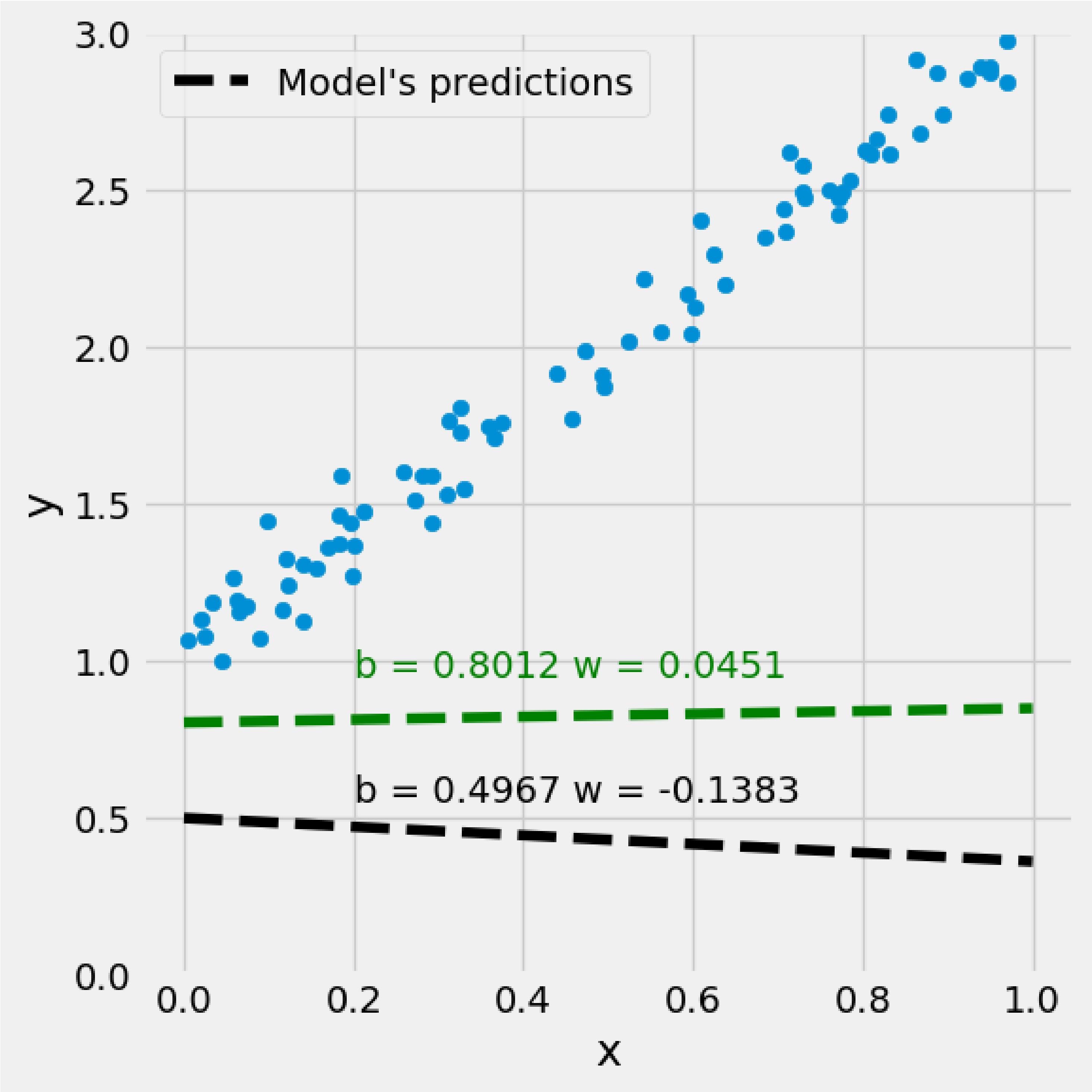
$$= \frac{1}{n}\sum_{i=1}^{n} \left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

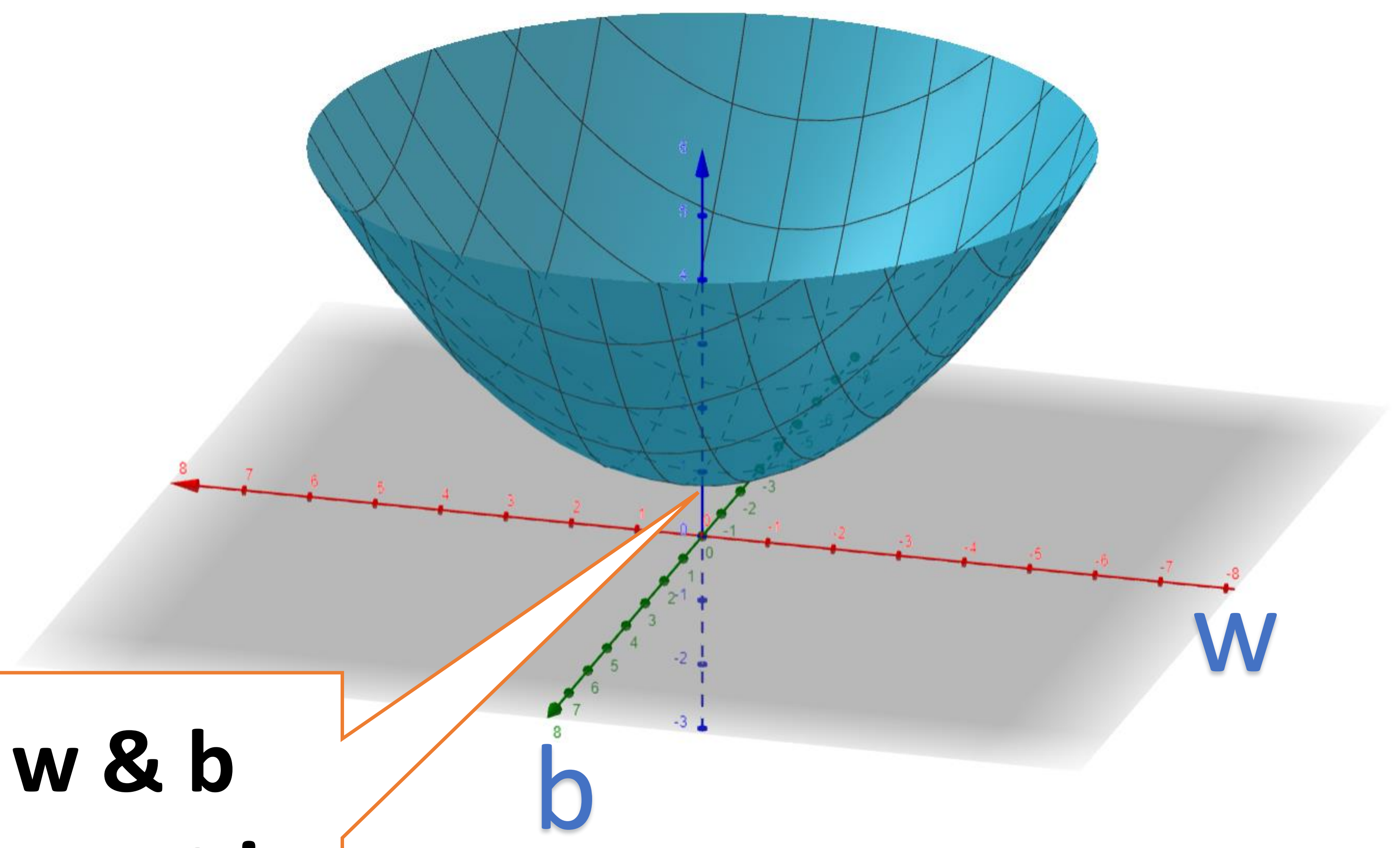
$$\mathcal{J}(w, b) = \frac{1}{n}\sum_{i=1}^{n} \left(b + wx^{(i)} - y^{(i)}\right)^2$$

Feature space versus parameter space

$$\hat{y} = wx + b$$

$$\mathcal{J}(w,b) = \frac{1}{n}\sum_{i=1}^{n}\left(b + wx^{(i)} - y^{(i)}\right)^2$$



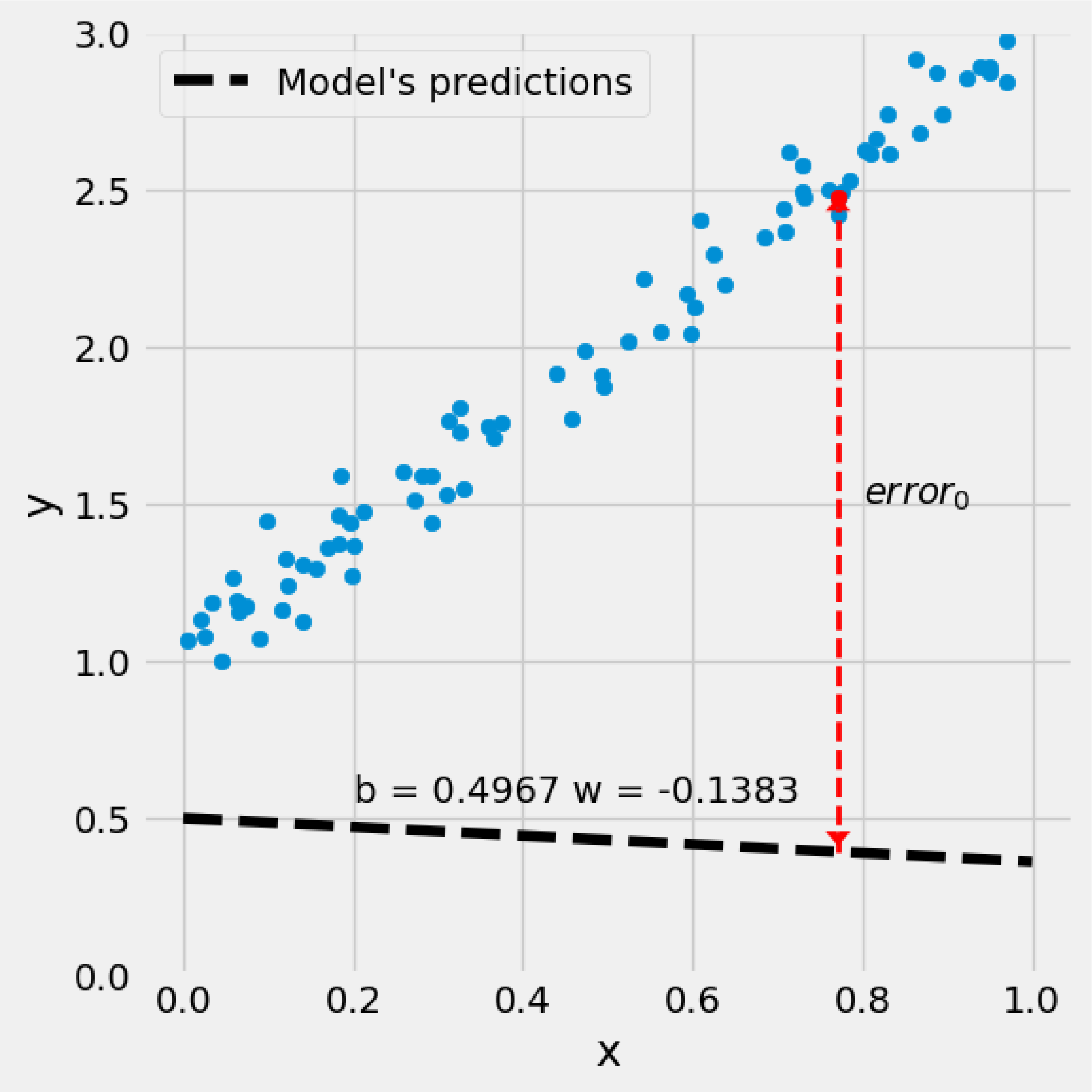


Find the w & b when the cost is minimum

Step 2. Evaluate y-hat & evaluate objective function

- Evaluate y-hat $\hat{y} = wx + b$
- Evaluate Objective function
- Also known as forward pass

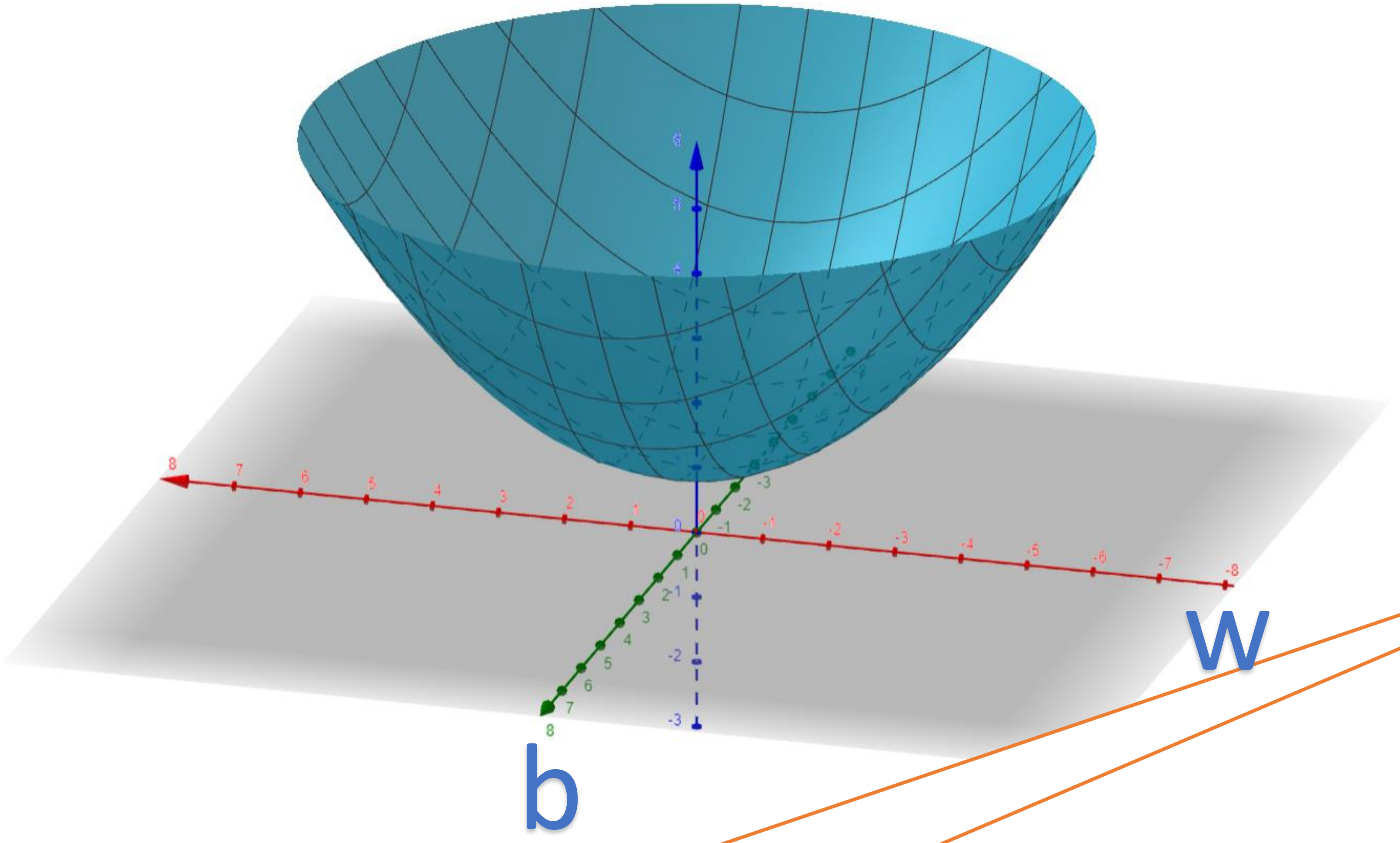
$$\mathcal{J}(w,b) = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

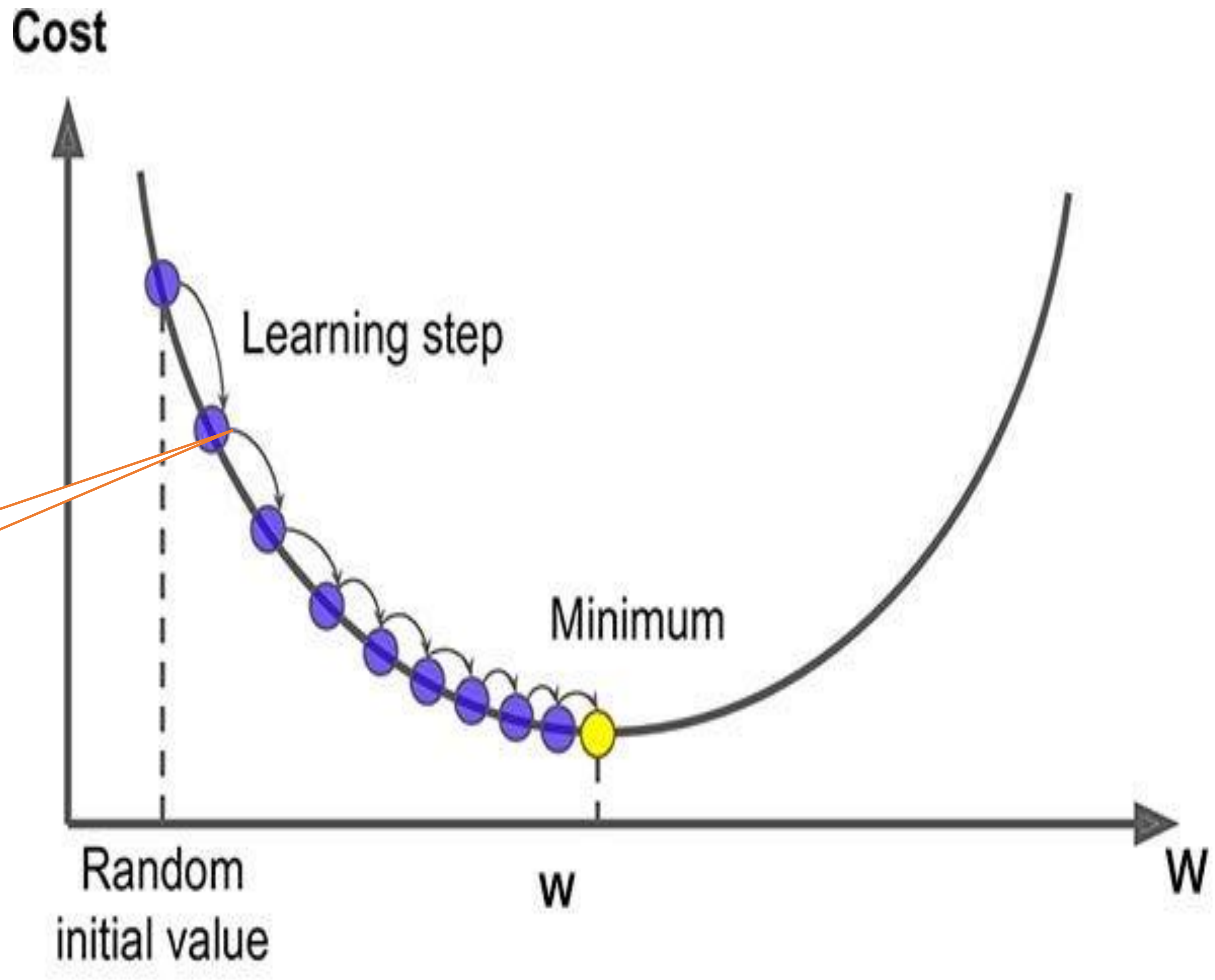


$$\mathcal{J}(w,b) = \frac{1}{n}\sum_{i=1}^{n}\left(b + wx^{(i)} - y^{(i)}\right)^2$$

Objective function plot

- https://www.geogebra.org/calculator/ua52fqtr





Take a little step towards lowest cost.

1. **How little?**
2. **Which direction?**

Step 3. Calculate analytical gradients

$$\mathcal{J}(w,b) = \frac{1}{n}\sum_{i=1}^{n}\left(b + wx^{(i)} - y^{(i)}\right)^2$$

$$\mathcal{J}(w,b) = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

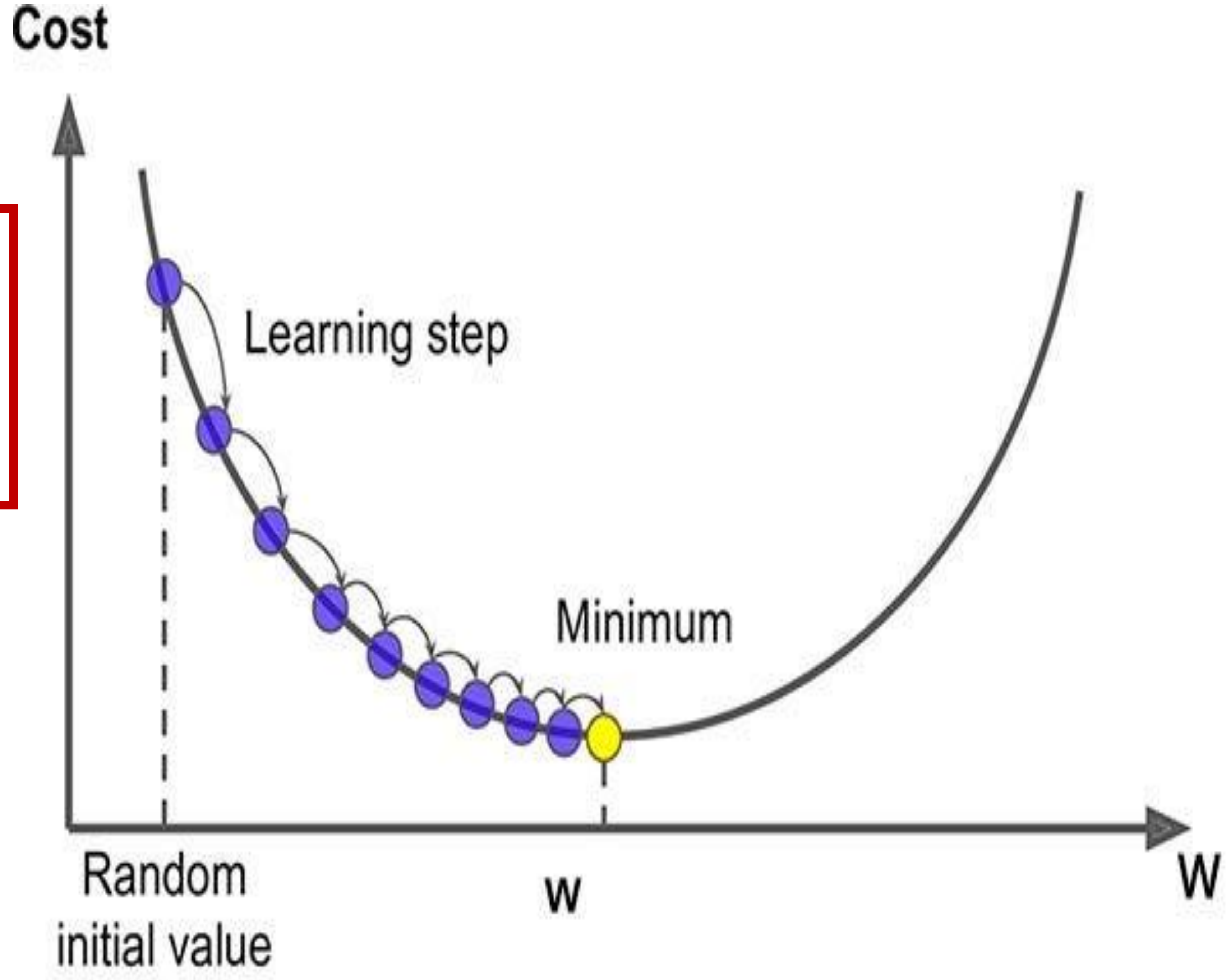
- Calculate gradient

$$\boxed{\frac{\partial \mathcal{J}}{\partial b} = \frac{\partial \mathcal{J}}{\partial \hat{y}^{(i)}}\frac{\partial \hat{y}^{(i)}}{\partial b}}$$

$$\boxed{\frac{\partial \mathcal{J}}{\partial w} = \frac{\partial \mathcal{J}}{\partial \hat{y}^{(i)}}\frac{\partial \hat{y}^{(i)}}{\partial w}}$$

$$\frac{\partial \mathcal{J}}{\partial b} = \frac{1}{m}\sum_{i=1}^{m} 2\left(b + wx^{(i)} - y^{(i)}\right)$$

$$\frac{\partial \mathcal{J}}{\partial w} = \frac{1}{m}\sum_{i=1}^{m} 2x^{(i)}\left(b + wx^{(i)} - y^{(i)}\right)$$



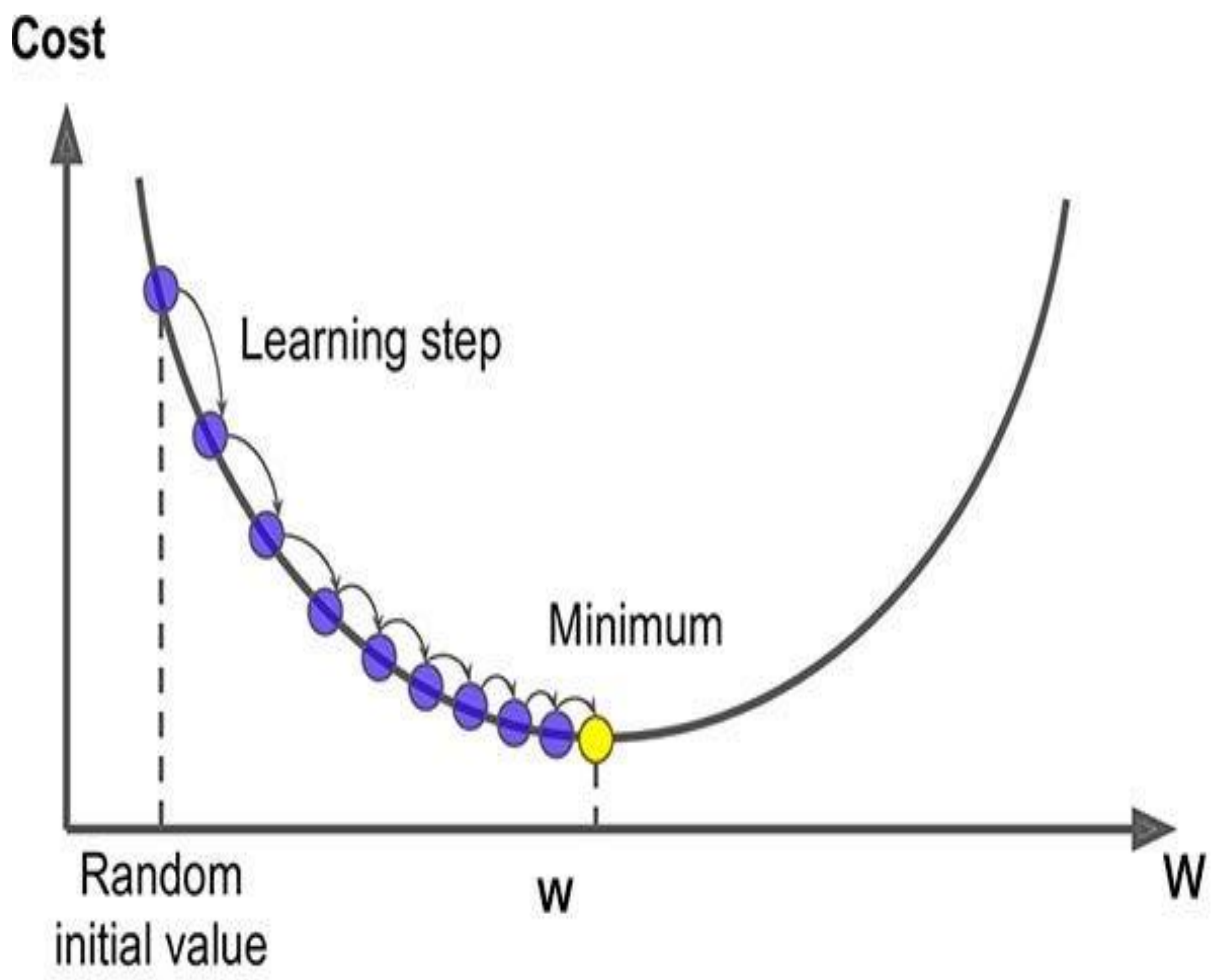
Step 4. Perform numerical gradient descent

- Also known as backward pass

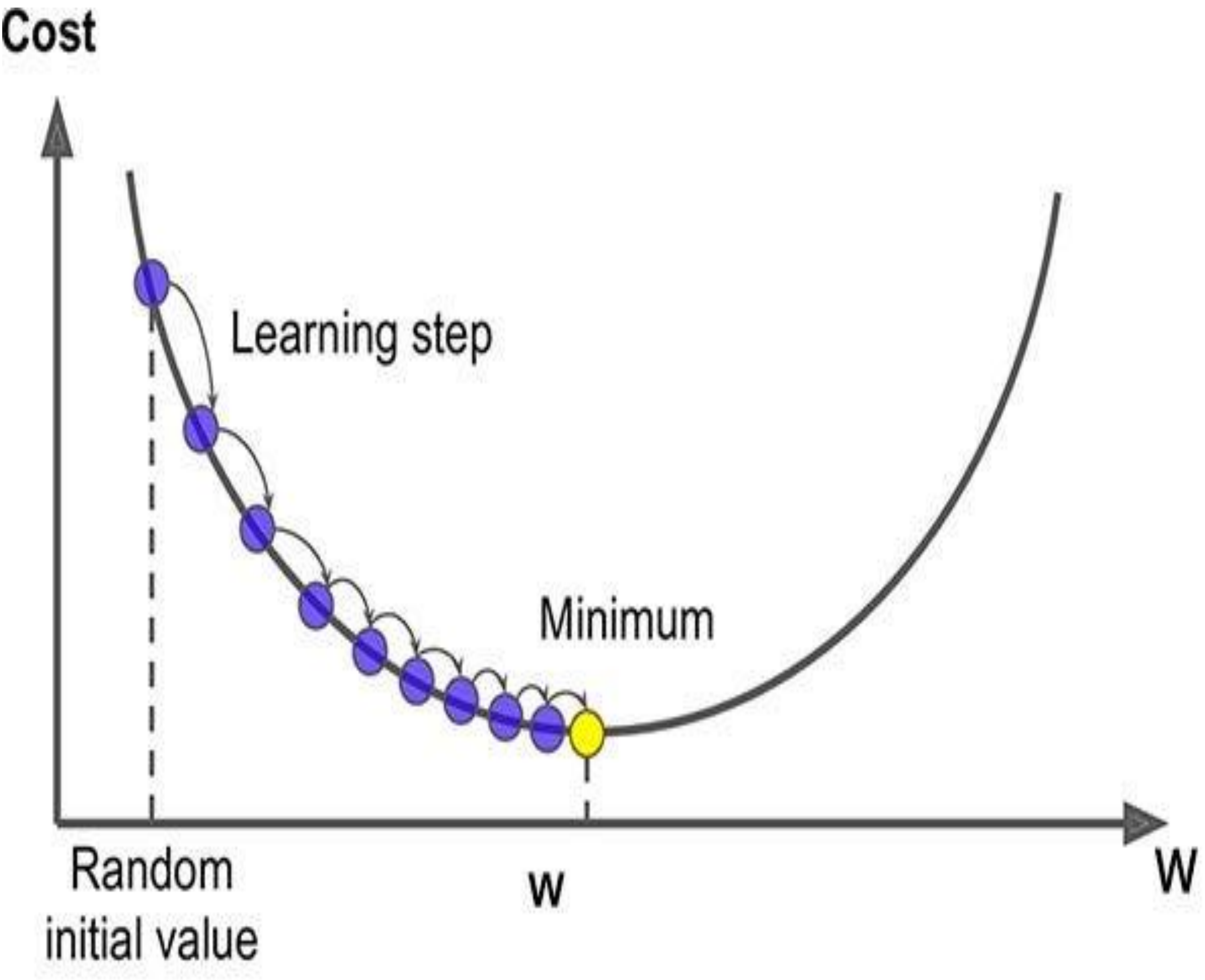
$$b = b - \eta \frac{\partial \mathcal{J}}{\partial b}$$

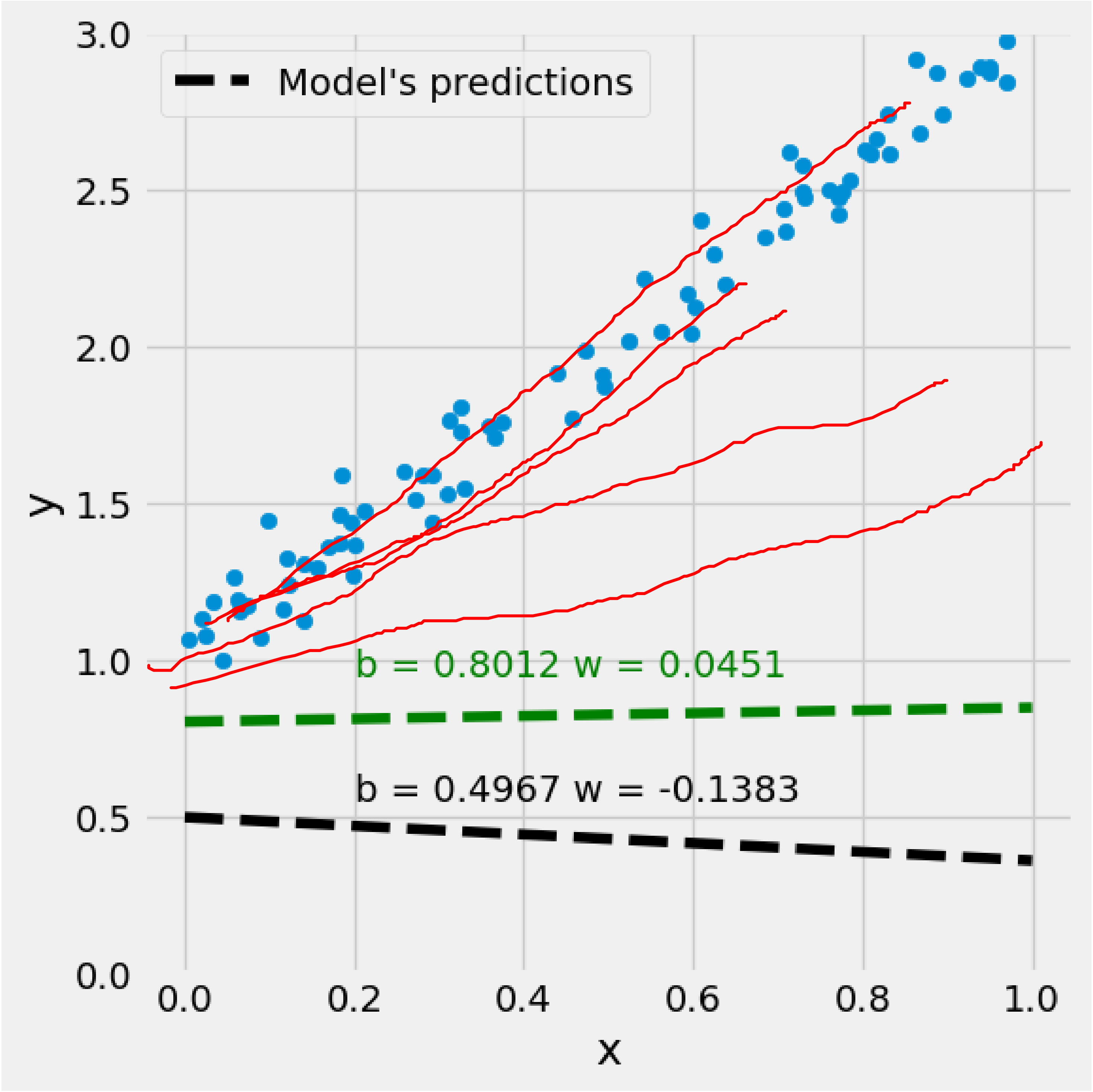
$$w = w - \eta \frac{\partial \mathcal{J}}{\partial w}$$

- Repeat Step 2, 3, 4



Change in w and b with gradient descent





Review Linear Regression

- Initialization: Select Random w & b, choose learning rate η
- Loop
  - Calculate new-cost for given w & b $\mathcal{J}(w,b) = \frac{1}{n}\sum_{i=1}^{n}\left(b + wx^{(i)} - y^{(i)}\right)^2$
  - Break If iter == max or new-cost – old cost < threshold
  - Calculate gradients wrt w and b

$$\frac{\partial \mathcal{J}}{\partial b} = \frac{1}{m}\sum_{i=1}^{m} 2\left(b + wx^{(i)} - y^{(i)}\right)$$

$$\frac{\partial \mathcal{J}}{\partial w} = \frac{1}{m}\sum_{i=1}^{m} 2x^{(i)}\left(b + wx^{(i)} - y^{(i)}\right)$$

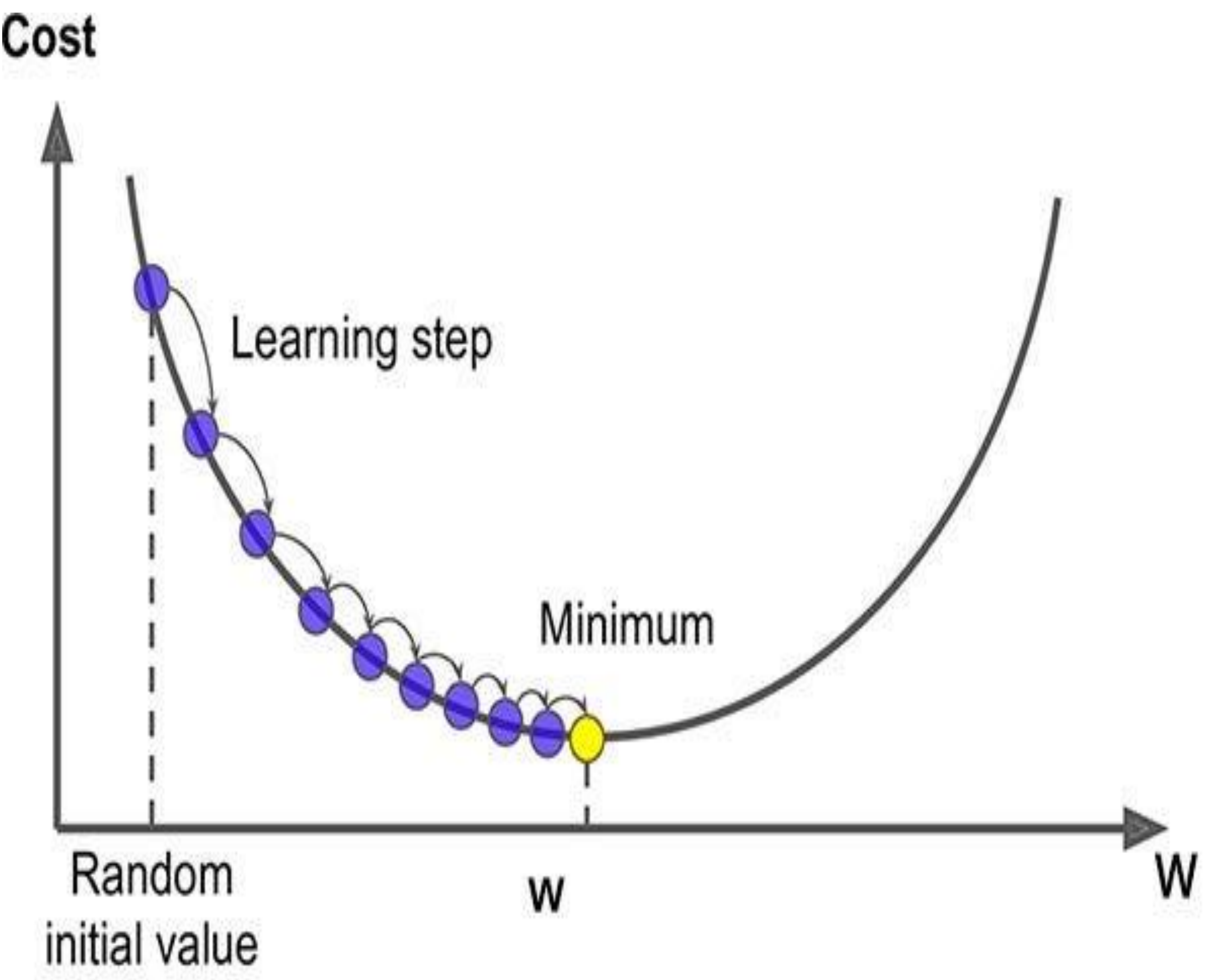
  - Do gradient descent

$$b = b - \eta\frac{\partial \mathcal{J}}{\partial b}$$

$$w = w - \eta\frac{\partial \mathcal{J}}{\partial w}$$

  - Old Cost = new cost

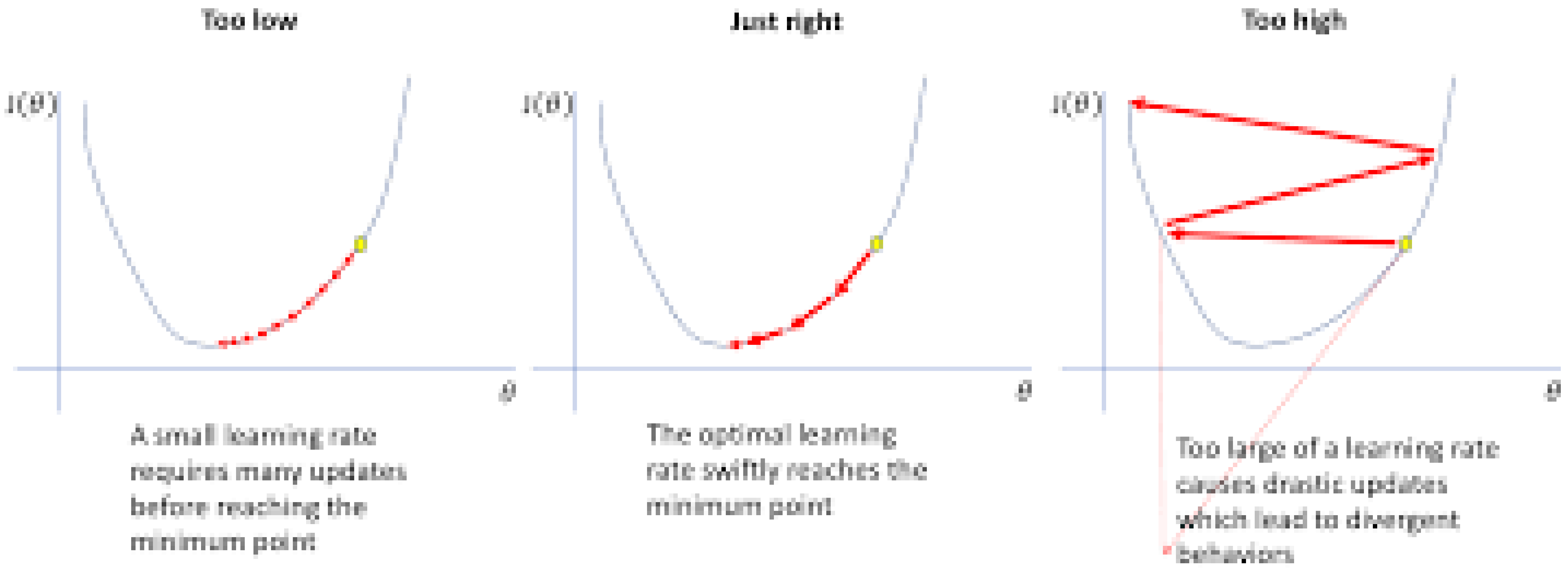
Gradient descent summary



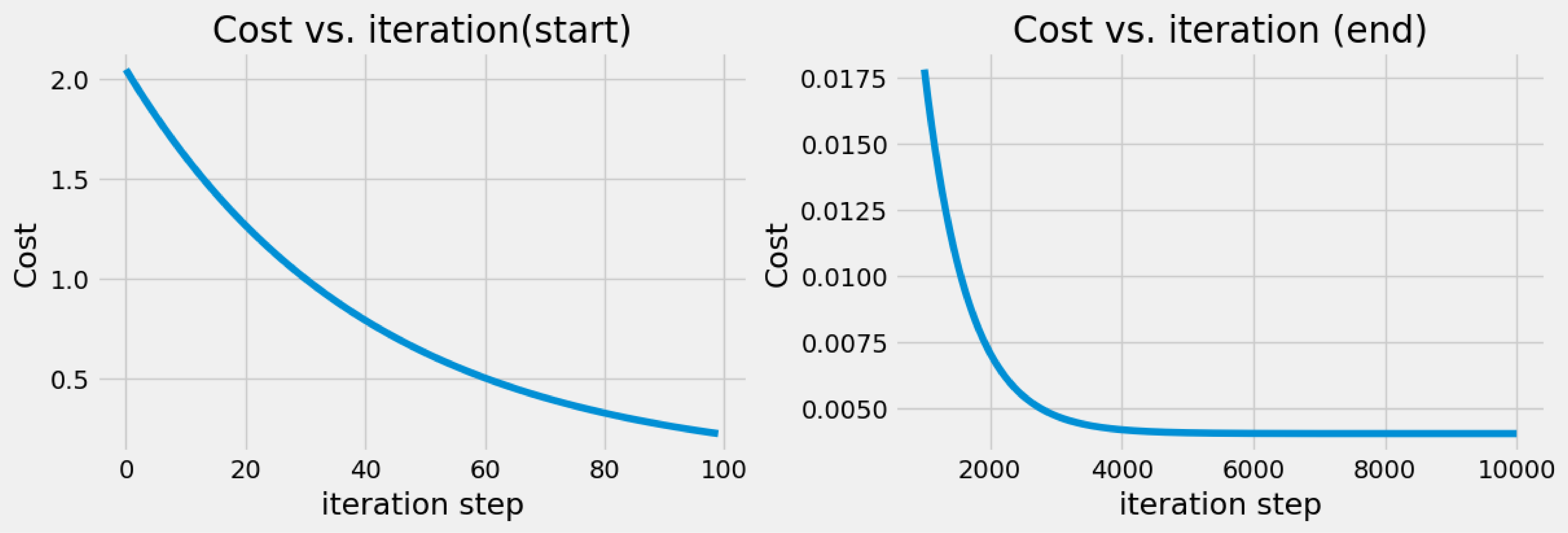
Iteration	w	b	Cost	dJ/dw	dJ/db
0	0.0110	0.0195	2.0443	-1.1077	-1.9538
1000	1.4309	1.2985	0.0178	-0.0407	0.02078
2000	1.7162	1.1527	0.0071	-0.0191	0.00976
3000	1.8502	1.0842	0.0047	-0.0090	0.00459
4000	1.9132	1.0520	0.0042	-0.0042	0.00215
5000	1.9430	1.0369	0.0041	-0.0020	0.00101
6000	1.9567	1.0298	0.0040	-0.0009	0.00047
7000	1.9632	1.0265	0.0040	-0.0004	0.00022
8000	1.9663	1.0249	0.0040	-0.0002	0.00010
9000	1.9677	1.0242	0.0040	-9.637e-05	4.925e-05

Demo in Jupyter Notebook

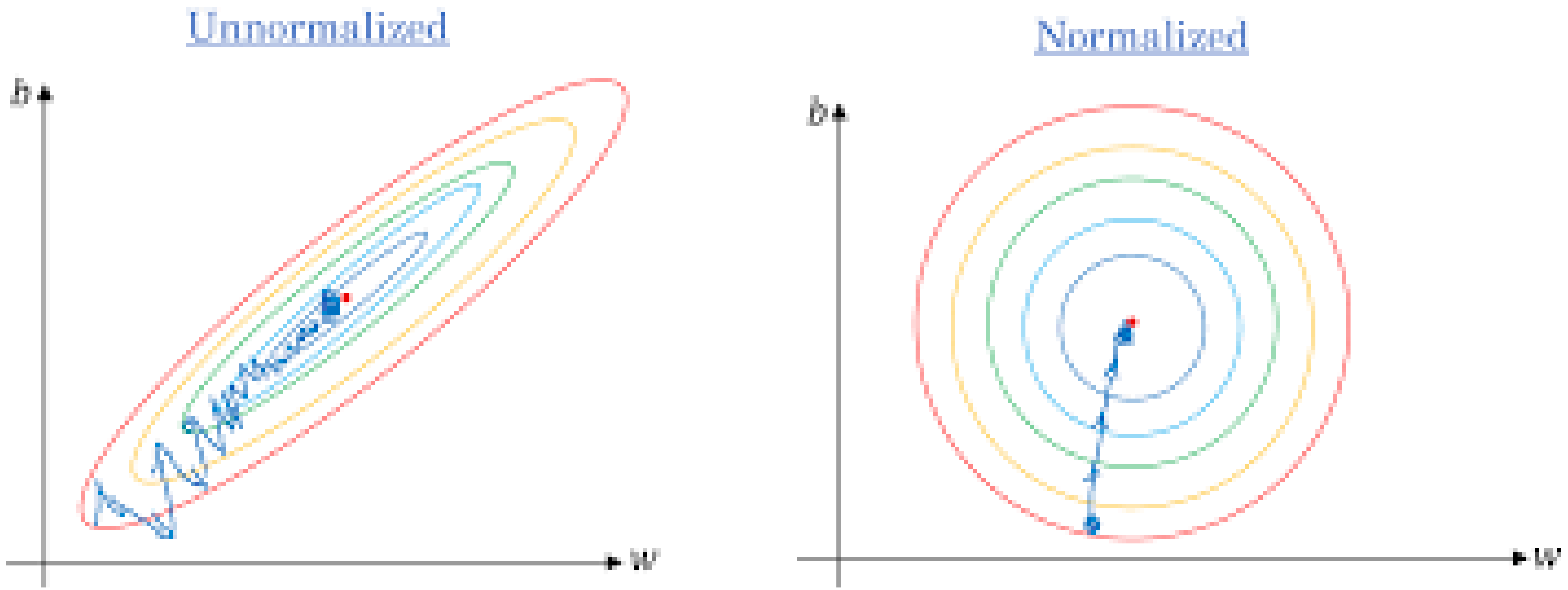
Impact of Learning rate on gradient descent



Impact of Learning rate on gradient descent



Importance of Scaling/Normalization



- Faster convergence (Less steps to minima)
- Robust convergence (will not wander away)

Closed form analytical solution

Closed form analytical solution

$$y = w_1 x + w_0 + \epsilon \qquad \hat{y} = w_1 x + w_0$$

$$\mathcal{J}(w_1, w_0) = \frac{1}{n}\sum_{i=1}^{n}\left(w_0 + w_1 x^{(i)} - y^{(i)}\right)^2$$

$$\frac{\partial \mathcal{J}}{\partial w_0} = \frac{\partial \mathcal{J}}{\partial \hat{y}^{(i)}}\frac{\partial \hat{y}^{(i)}}{\partial w_0} = \frac{1}{n}\sum_{i=1}^{n} 2(w_0 + w_1 x^{(i)} - y^{(i)}) = 0$$

$$\frac{\partial \mathcal{J}}{\partial w_1} = \frac{\partial \mathcal{J}}{\partial \hat{y}^{(i)}}\frac{\partial \hat{y}^{(i)}}{\partial w_1} = \frac{1}{n}\sum_{i=1}^{n} 2(w_0 + w_1 x^{(i)} - y^{(i)})x^{(i)} = 0$$

Closed form analytical solution

$$w_1 = \frac{n\sum_{i=1}^{m} x^{(i)} y^{(i)} - \sum_{i=1}^{m} x^{(i)} \sum_{i=1}^{m} y^{(i)}}{n\sum_{i=1}^{m} {x^{(i)}}^2 - (\sum_{i=1}^{m} x^{(i)})^2}$$

$$w_0 = \frac{\sum_{i=1}^{m} y^{(i)} - w_1 \sum_{i=1}^{m} x^{(i)}}{n}$$

- Formula starts getting complicated with inter-dependencies
- Needs to load all data at once
  - What happens when there are million+ records?

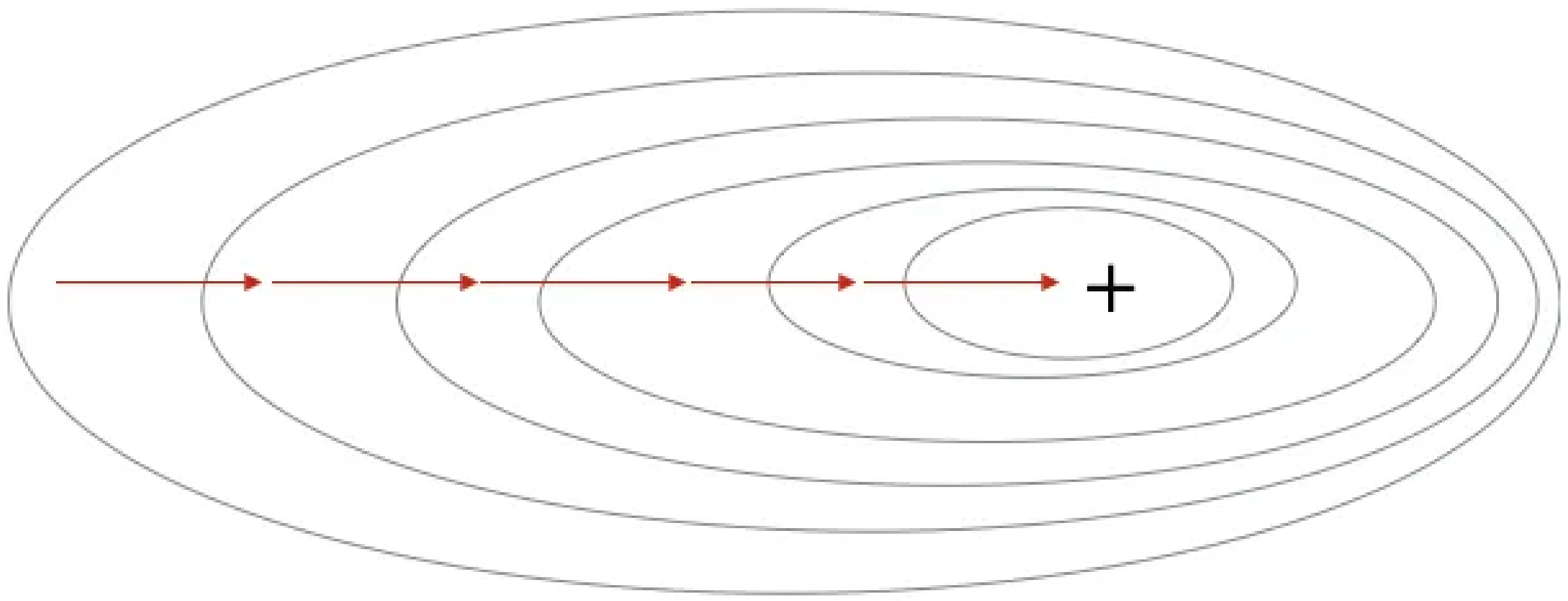
Types of Gradient Descent

Gradient descent - Batch, mini batch & stochastic

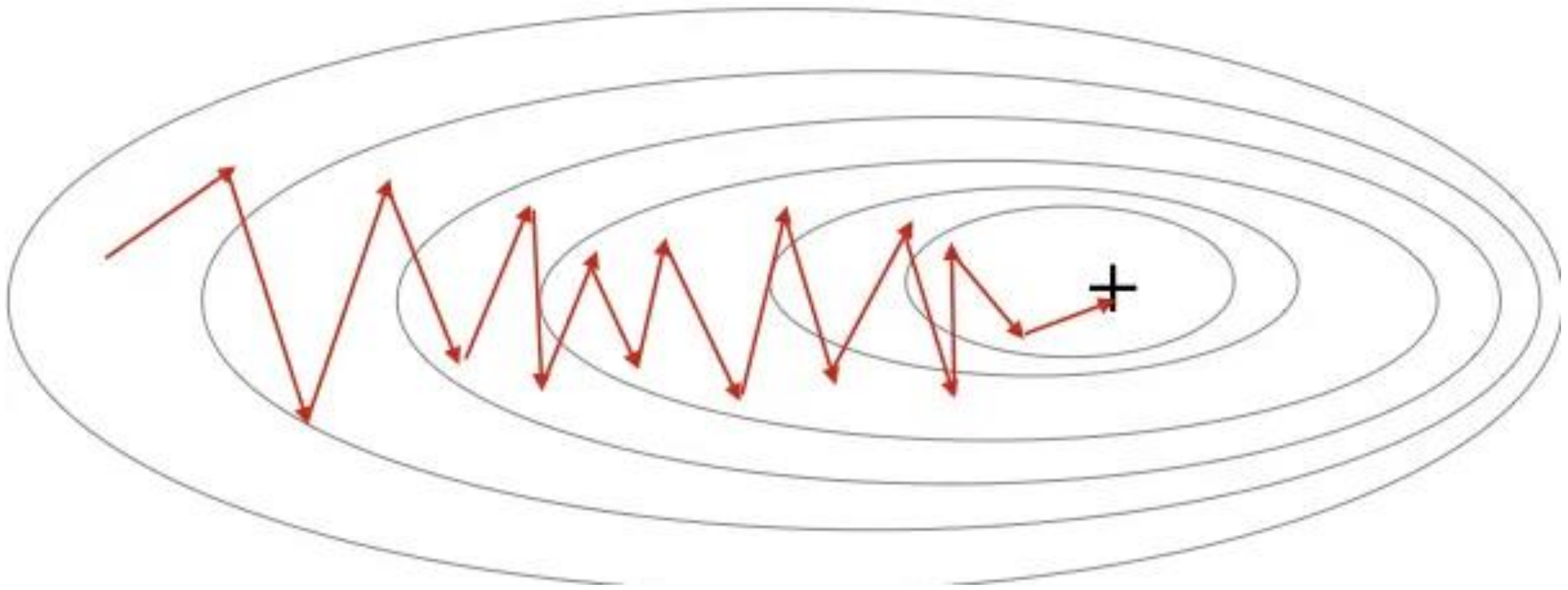
- Batch gradient descent
  - Entire dataset used, Offline training
  - Good for small data set
- Stochastic gradient descent
  - One record used at a time, Online training
  - Good for streaming data
- Mini-batch gradient descent
  - Large dataset is cut into chunks, Calc J on each chunk
  - Iterate over entire dataset many times progressively reducing cost

Gradient descent comparison

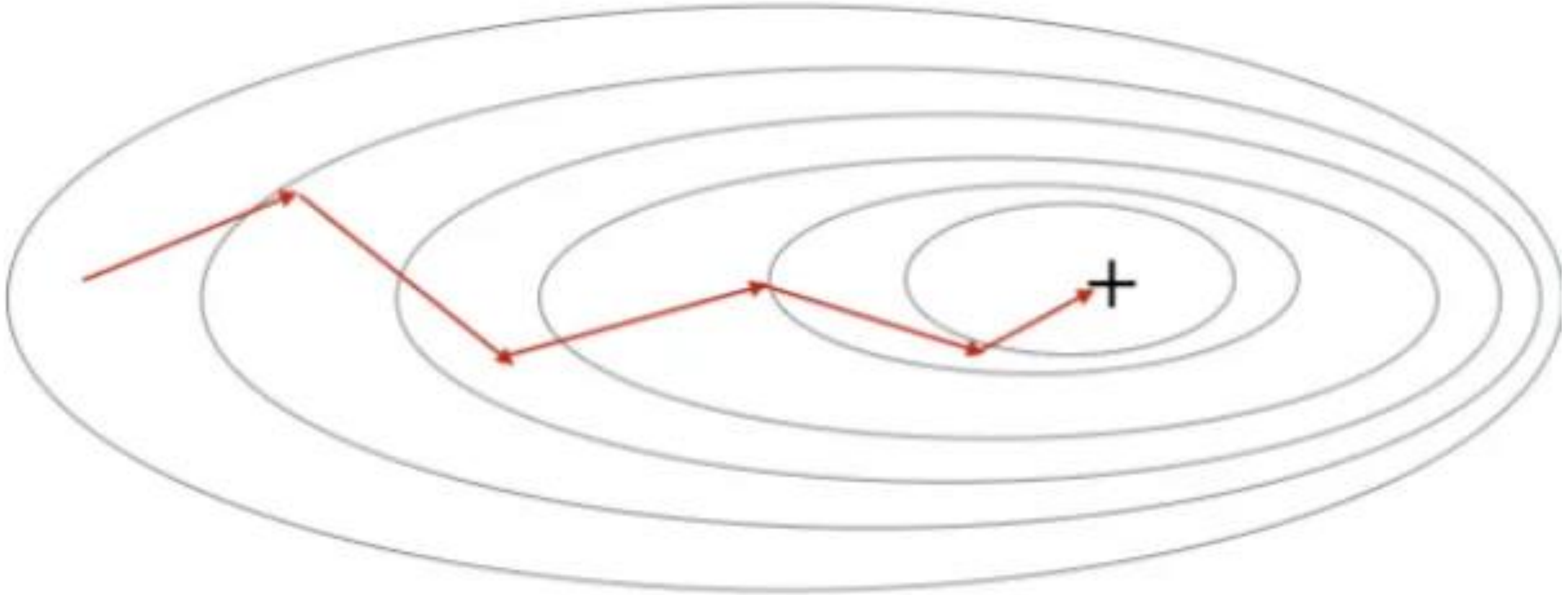
Batch Gradient Descent



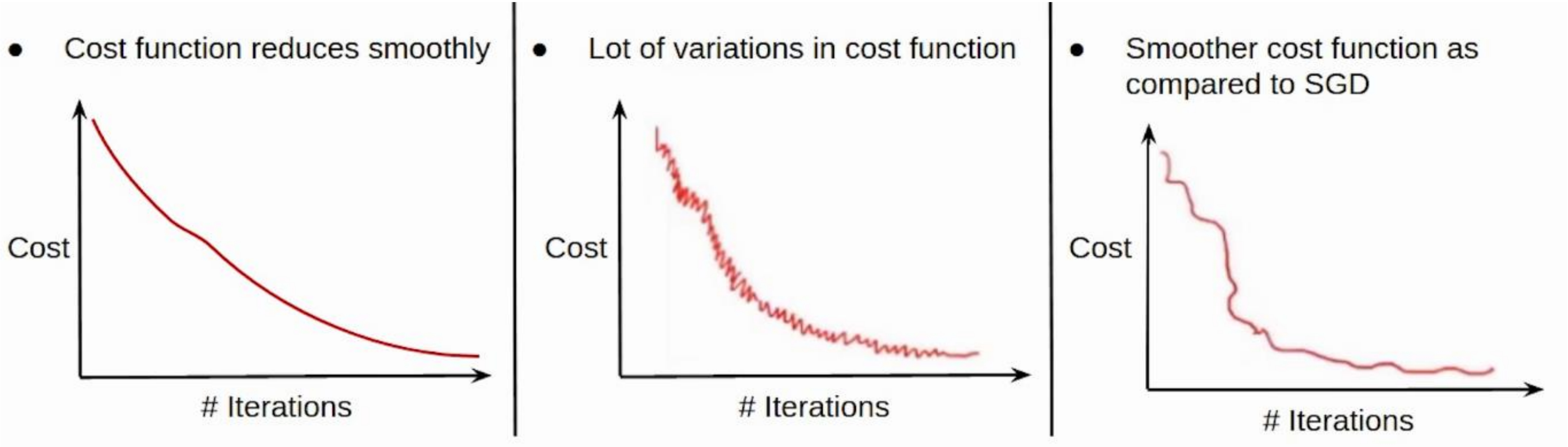
Stochastic Gradient Descent



Mini-batch Gradient Descent



Cost function comparison



Gradient descent comparison

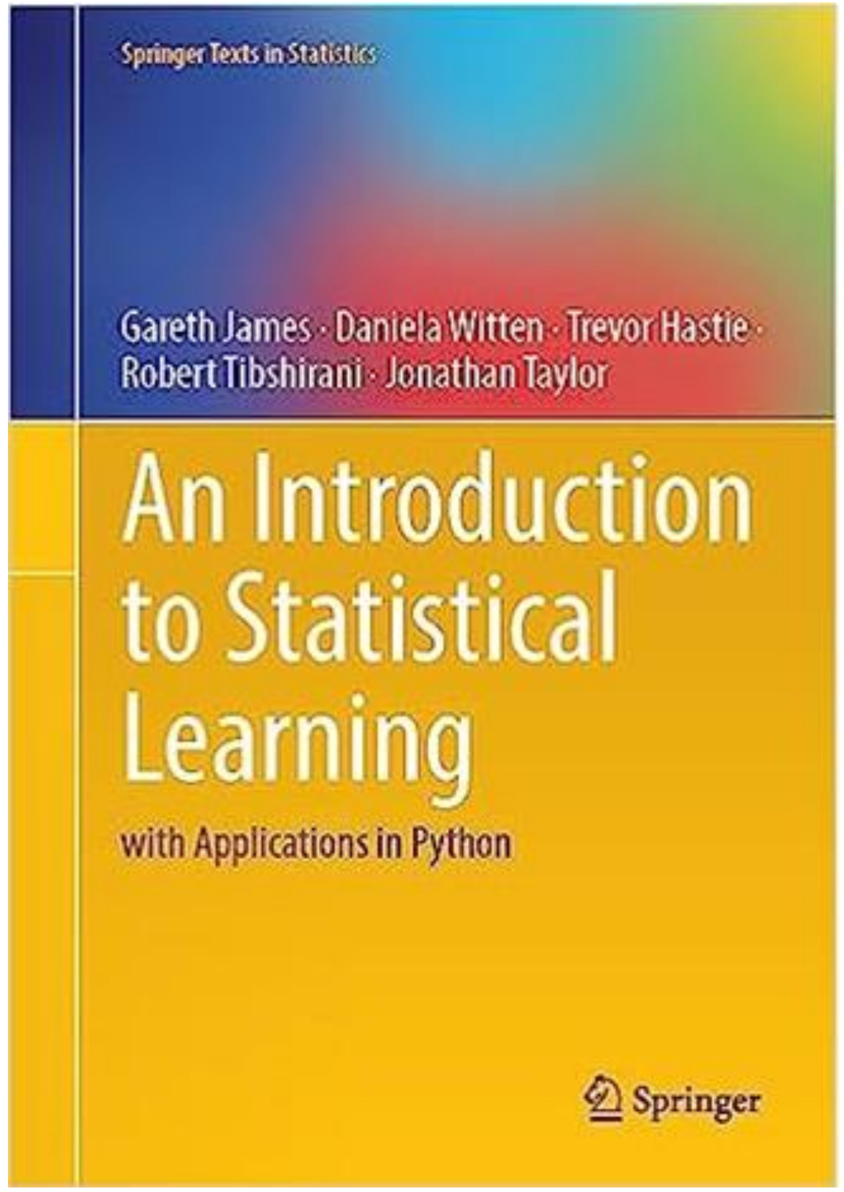
Batch Gradient Descent	Stochastic Gradient Descent (SGD)	Mini-Batch Gradient Descent
• Entire dataset for updation	• Single observation for updation	• Subset of data for updation
• Cost function reduces smoothly	• Lot of variations in cost function	• Smoother cost function as compared to SGD
• Computation cost is very high	• Computation time is more	• Computation time is lesser than SGD
		• Computation cost is lesser than Batch Gradient Descent

Coding Linear Regression

Linear Regression Dataset

- Advertising.csv

TV	Radio	Newspaper	Sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5
180.8	10.8	58.4	12.9



Coding with statsmodels

```
import statsmodels.api as sm
lm = sm.OLS(y, X)
model = lm.fit()
```

```
model.summary()
```

OLS Regression Results

Dep. Variable:	sales	**R-squared (uncentered):**	0.982
Model:	OLS	**Adj. R-squared (uncentered):**	0.982
Method:	Least Squares	**F-statistic:**	3566.
Date:	Sun, 28 Mar 2021	**Prob (F-statistic):**	2.43e-171
Time:	13:42:33	**Log-Likelihood:**	-423.54
No. Observations:	200	**AIC:**	853.1
Df Residuals:	197	**BIC:**	863.0
Df Model:	3		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
TV	0.0538	0.001	40.507	0.000	0.051	0.056
radio	0.2222	0.009	23.595	0.000	0.204	0.241
newspaper	0.0168	0.007	2.517	0.013	0.004	0.030

Omnibus:	5.982	**Durbin-Watson:**	2.038
Prob(Omnibus):	0.050	**Jarque-Bera (JB):**	7.039
Skew:	-0.232	**Prob(JB):**	0.0296
Kurtosis:	3.794	**Cond. No.**	12.6

Coding with sklearn

```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
model = lm.fit(X,y)
```

```
model.intercept_
```

```
array([2.93888937])
```

```
model.coef_
```

```
array([[ 0.04576465,  0.18853002, -0.00103749]])
```

```
model.predict(new_data)
```

```
array([[6.15044172]])
```

Different types of gradient descent in sklearn

- Batch gradient descent
  - `sklearn.linear_model.LinearRegression`
- Stochastic gradient descent
  - `sklearn.linear_model.SGDRegressor`
- Mini-batch gradient descent
  - `sklearn.linear_model.SGDRegressor`
    - `partial_fit()`
    - Pass each mini batch into `partial_fit()`
    - Cannot use `partial_fit()` in Pipeline!

Evaluation metrics for Regression

```
model.score(X_train, y_train)
```

```
0.910413637900632
```

```
model.score(X_test,y_test)
```

```
0.8495077592917368
```

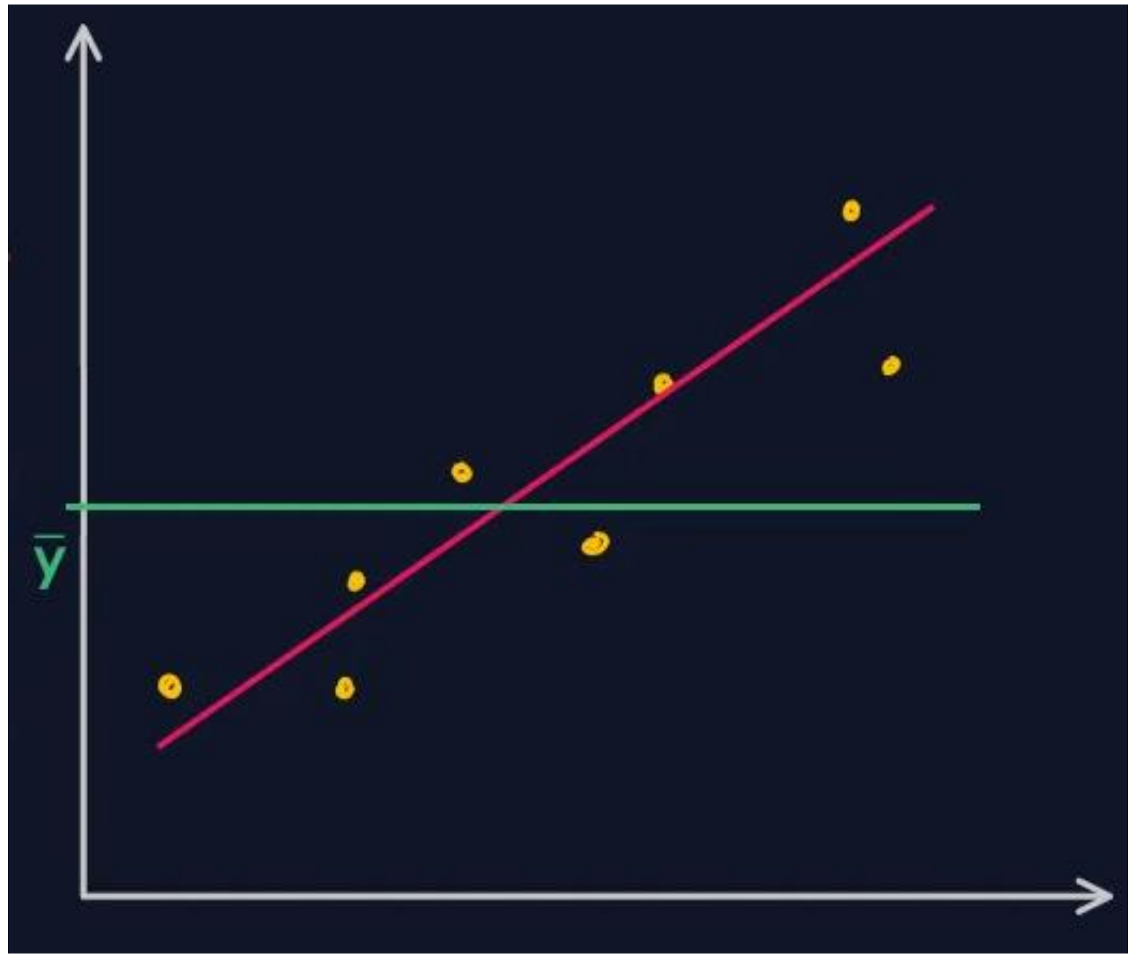
- What does score mean in Regression?

Evaluating Linear Regression

Evaluation metrics for Regression

- Mean Squared Error (MSE) $= \frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$
- Root Mean Squared Error (RMSE) $= \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}^{(i)} - y^{(i)}\right)^2}$
- Mean Absolute Error (MAE) $= \frac{1}{n}\sum_{i=1}^{n}\left|\hat{y}^{(i)} - y^{(i)}\right|$
- R-Squared $R^2 = 1 - \frac{SS_{reg}}{SS_{avg}} = 1 - \frac{\sum_{i=1}^{n}\left(\hat{y}^{(i)} - y^{(i)}\right)^2}{\sum_{i=1}^{n}\left(\hat{y}^{(i)} - \bar{y}\right)^2}$
- Adjusted R-Squared $R^2_{adj} = 1 - \frac{\sum_{i=1}^{n}\left(\hat{y}^{(i)} - y^{(i)}\right)^2}{\sum_{i=1}^{n}\left(\hat{y}^{(i)} - \bar{y}\right)^2}\frac{(n-1)}{(n-k-1)}$

R-squared intuition

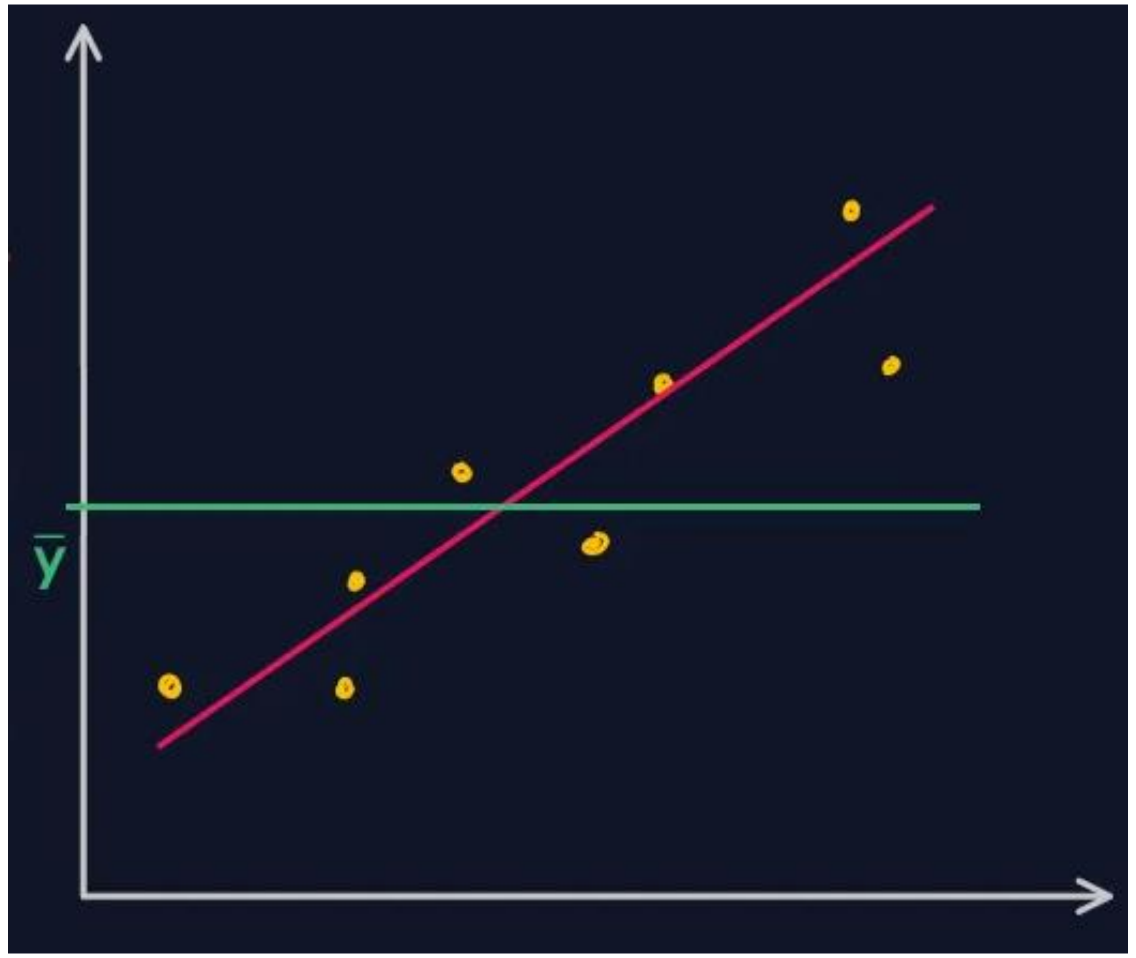


$$R^2 = 1 - \frac{SS_{reg}}{SS_{avg}} = 1 - \frac{\sum_{i=1}^{n} \left(\hat{y}^{(i)} - y^{(i)}\right)^2}{\sum_{i=1}^{n} \left(\hat{y}^{(i)} - \bar{y}\right)^2}$$

- Denominator is variance w.r.t. mean
- Numerator is variance w.r.t. regression line
- Lesser the variance wrt regression line the better

- How much variance is explained by linear regression?
  - More the merrier (Implies less error is left after regression)
- R-squared between 0 & 1. Higher the better

Adjusted R-squared intuition



$$R^2_{adj} = 1 - \frac{\sum_{i=1}^{n} \left(\hat{y}^{(i)} - y^{(i)}\right)^2}{\sum_{i=1}^{n} \left(\hat{y}^{(i)} - \bar{y}\right)^2} \frac{(n-1)}{(n-k-1)}$$

- If additional feature is added R squared increases
- But if the feature less useful in explaining variance, then adjusted R-squared decreases

- Penalized for using more features that do not add value

Recap

- Population and Sample Regression
- Simple Linear Regression Intuition
- Linear Regression Algorithm
- Gradient Descent
- Impact of Scaling in Gradient Descent
- Closed form analytical solution
- Types of Gradient Descent
- Regression Evaluation Metrics


QUESTIONS