AI VIET NAM
@aivietnam.edu.vn

Module 04 – Exercise Class

LINEAR REGRESSION
Vectorization

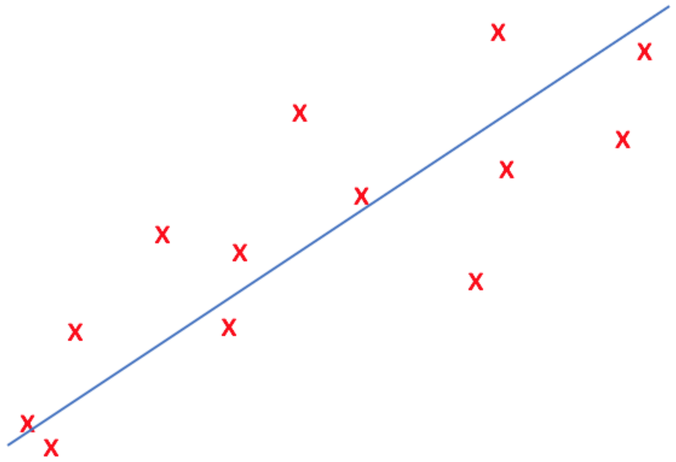
Nguyen Quoc Thai

AI VIET NAM
@aivietnam.edu.vn

Objectives

Linear Regression

- Introduction
- Stochastic Gradient Descent
- Mini Batch Gradient Descent
- Batch Gradient Descent



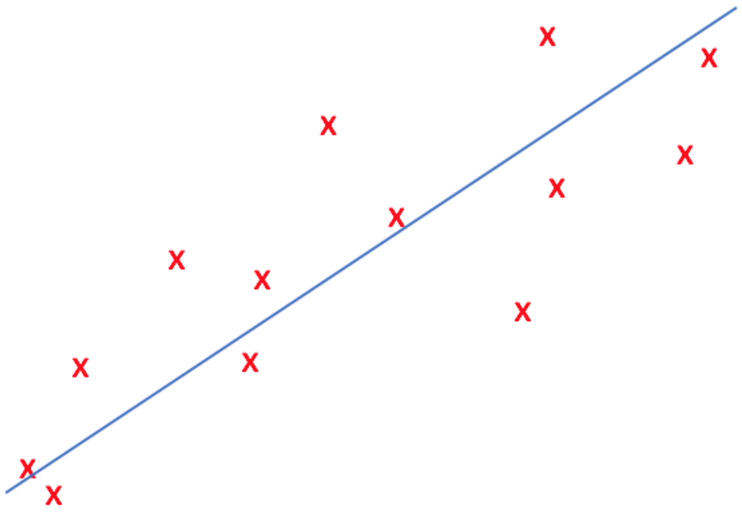
Linear Regression for Timeseries

- Bitcoin Dataset
- Feature Scaling
- Modeling
- Evaluation
- Inference

Outline

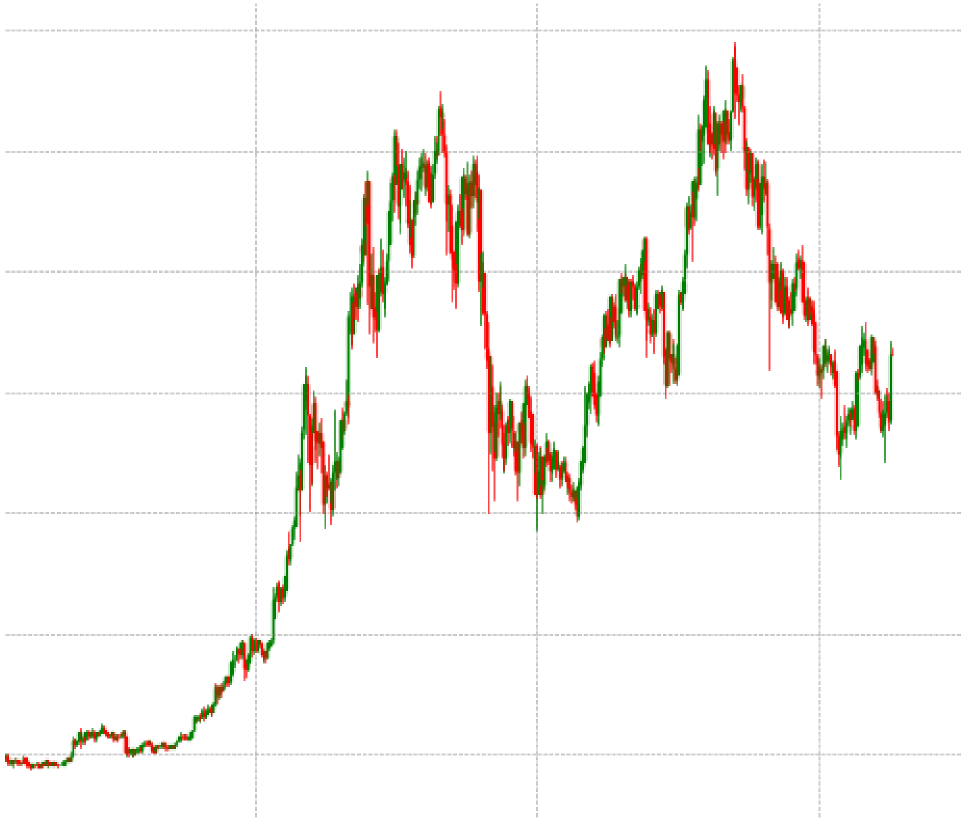
SECTION 1

Linear Regression



SECTION 2

Time Series Application



Linear Regression

Introduction

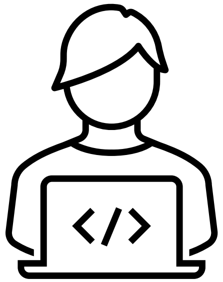
Data

Level	Salary
0	8
1	15
2	18
3	22
4	26
5	30
6	38
7	47

Level	Salary
3.5	???
10	???

Prediction

Learning



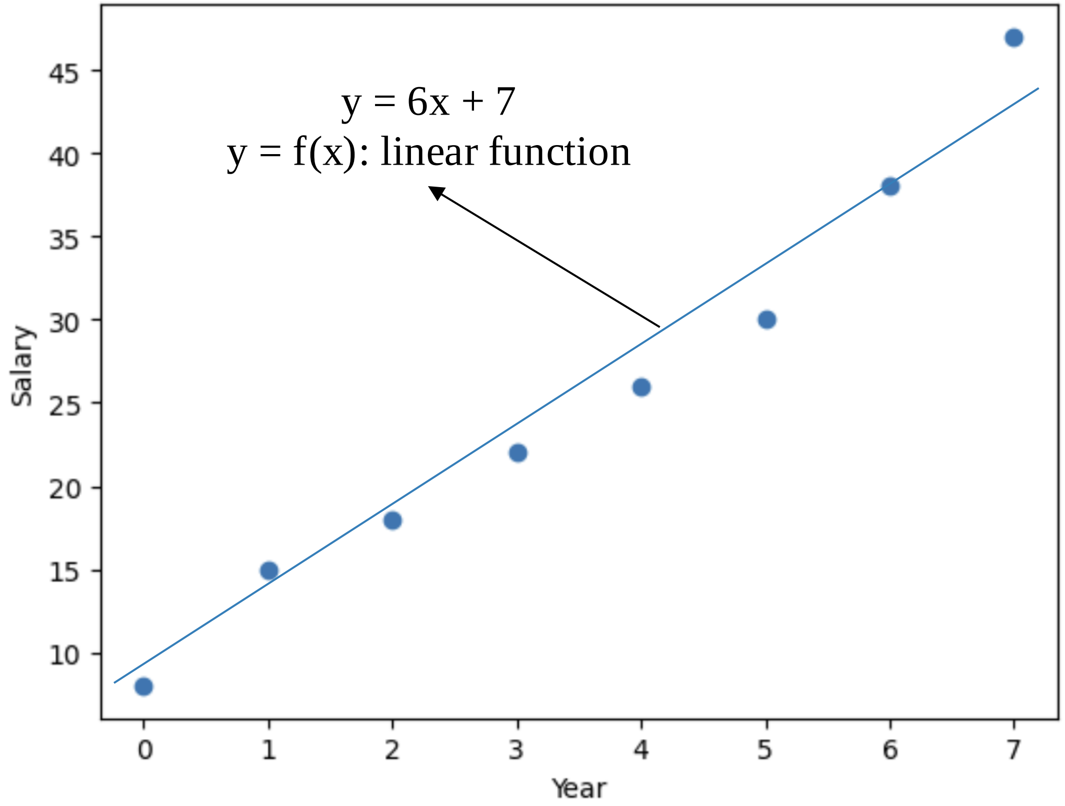
Linear Regression

Introduction

Data

Level	Salary
0	8
1	15
2	18
3	22
4	26
5	30
6	38
7	47

Visualization



Linear Regression

Introduction

Data

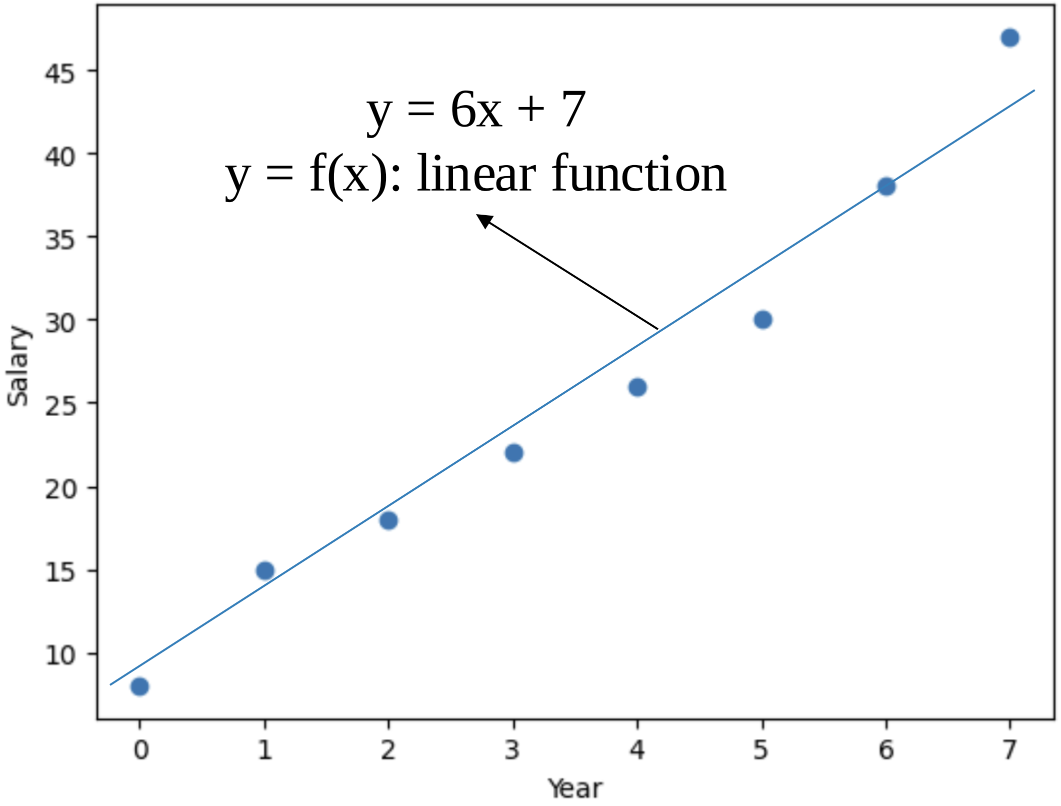
Level	Salary
0	8
1	15
2	18
3	22
4	26
5	30
6	38
7	47

Modeling

$y = wx + b$

Find w and b to fit the data

Visualization



Linear Regression

Stochastic Gradient Descent

1) Pick a sample (x, y) from training data

2) Compute the output $\hat{y}$

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

3) Compute loss

$$L = (\hat{y} - y)^2$$

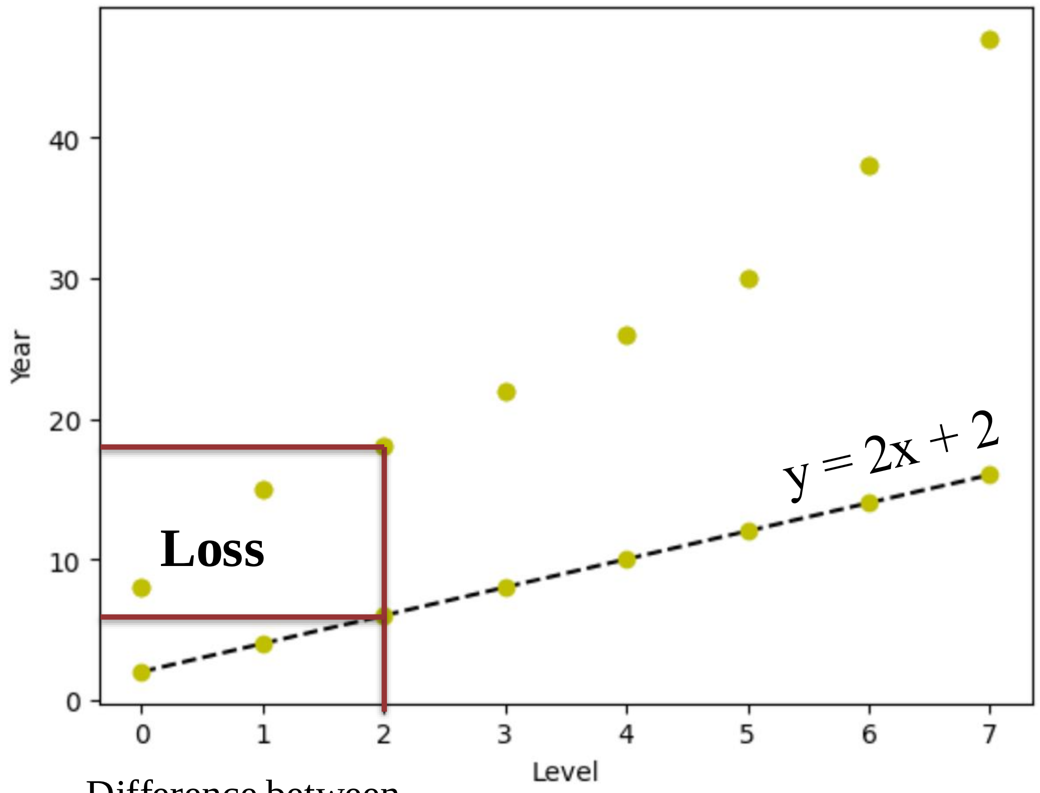
4) Compute derivative

$$\frac{\partial L}{\partial w} = 2x(\hat{y} - y) \qquad \frac{\partial L}{\partial b} = 2(\hat{y} - y)$$

5) Update parameters

$$w = w - \eta \frac{\partial L}{\partial w} \qquad b = b - \eta \frac{\partial L}{\partial b}$$

η is learning rate



Difference between predicted and actual value

Linear Regression

Stochastic Gradient Descent (Vectorization)

1) Pick a sample (x, y) from training data

2) Compute output $\hat{y}$

$$\hat{y} = \boldsymbol{\theta}^T \boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{\theta}$$

3) Compute loss

$$L = (\hat{y} - y)^2$$

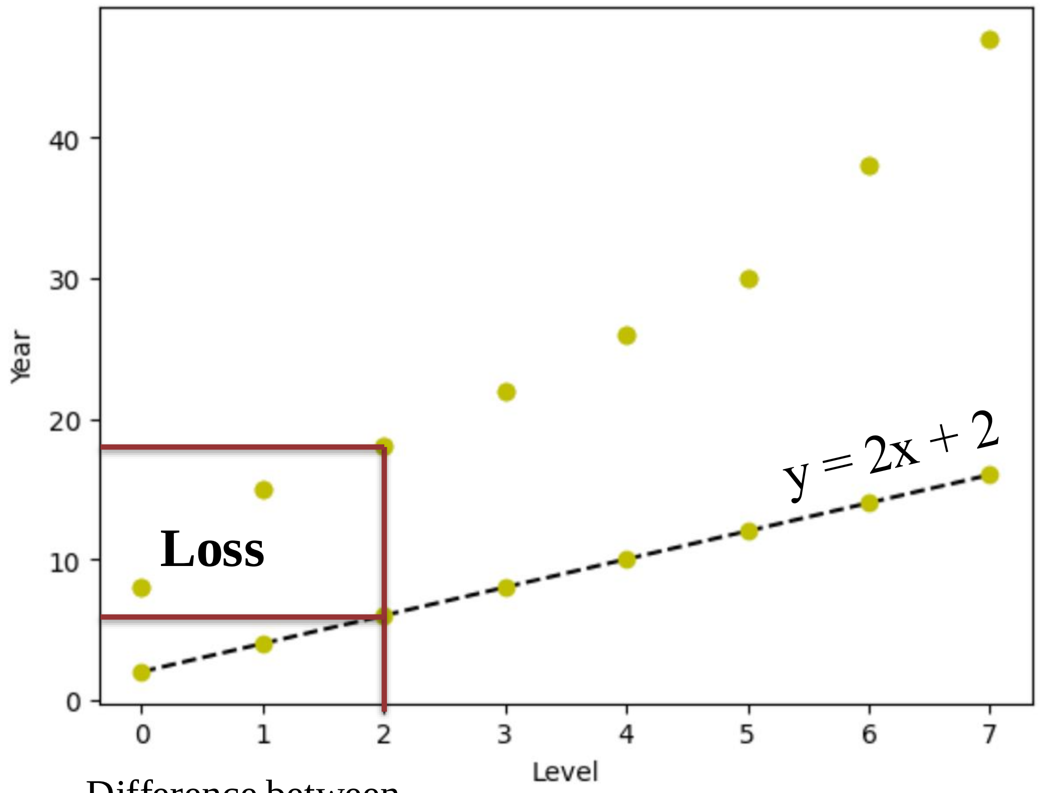
4) Compute derivative

$$\nabla_{\theta} \mathrm{L} = 2\boldsymbol{x}(\hat{y} - y)$$

5) Update parameters

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_{\theta} \mathrm{L}$$

η is learning rate



Difference between predicted and actual value

Linear Regression

Stochastic Gradient Descent (Vectorization)

1) Pick a sample (x, y) from training data

2) Compute output $\hat{y}$

$$\hat{y} = \boldsymbol{\theta}^T \boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{\theta}$$

3) Compute loss

$$L = (\hat{y} - y)^2$$

4) Compute derivative

$$\nabla_{\theta} \mathrm{L} = 2\boldsymbol{x}(\hat{y} - y)$$

5) Update parameters

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_{\theta} \mathrm{L}$$

η is learning rate

```
x = np.array([[2]])
y = np.array([18])
```

```
num_samples = x.shape[0]
num_samples
```

```
1
```

```
# append bias
x = np.hstack([x, np.ones((num_samples, 1))])
x
```

```
array([[2., 1.]])
```

```
# init weights
theta = np.array([2, 2]) # num_features: x.shape[1]
theta
```

```
array([2, 2])
```

Linear Regression

Stochastic Gradient Descent (Vectorization)

1) Pick a sample (x, y) from training data

2) Compute output $\hat{y}$

$$\hat{y} = \boldsymbol{\theta}^T \boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{\theta}$$

3) Compute loss

$$L = (\hat{y} - y)^2$$

4) Compute derivative

$$\nabla_{\theta} \mathrm{L} = 2\boldsymbol{x}(\hat{y} - y)$$

5) Update parameters

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_{\theta} \mathrm{L}$$

η is learning rate

```
# forward
def predict(x, theta):
    return x.dot(theta)

y_hat = predict(x, theta)
y_hat
```

```
array([6.])
```

Linear Regression

Stochastic Gradient Descent (Vectorization)

1) Pick a sample (x, y) from training data

2) Compute output $\hat{y}$

$$\hat{y} = \boldsymbol{\theta}^T \boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{\theta}$$

3) Compute loss

$$L = (\hat{y} - y)^2$$

4) Compute derivative

$$\nabla_{\theta} \mathrm{L} = 2\boldsymbol{x}(\hat{y} - y)$$

5) Update parameters

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_{\theta} \mathrm{L}$$

η is learning rate

```
# compute gradient
def compute_gradient(x, y, y_hat):
    d_theta = 2*x*(y_hat-y)
    return d_theta

d_theta = compute_gradient(x, y, y_hat)
d_theta
```

```
array([[-48., -24.]])
```

Linear Regression

Stochastic Gradient Descent (Vectorization)

1) Pick a sample (x, y) from training data

2) Compute output $\hat{y}$

$$\hat{y} = \boldsymbol{\theta}^T \boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{\theta}$$

3) Compute loss

$$L = (\hat{y} - y)^2$$

4) Compute derivative

$$\nabla_{\theta} \mathrm{L} = 2\boldsymbol{x}(\hat{y} - y)$$

5) Update parameters

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_{\theta} \mathrm{L}$$

η is learning rate

```
# update weights
lr = 0.1

def update_weights(theta, d_theta, lr):
    new_theta = theta - lr*d_theta
    return new_theta

new_theta = update_weights(theta, d_theta, lr)
new_theta
```

```
array([[6.8, 4.4]])
```

Linear Regression

Practice

1) Pick a sample (x, y) from training data

2) Compute output $\hat{y}$

$$\hat{y} = \boldsymbol{\theta}^T \boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{\theta}$$

3) Compute loss

$$L = (\hat{y} - y)^2$$

4) Compute derivative

$$\nabla_{\theta} \mathrm{L} = 2\boldsymbol{x}(\hat{y} - y)$$

5) Update parameters

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_{\theta} \mathrm{L}$$

η is learning rate

x = [1 3]
y = 22

Init θ
lr = 0.1

$\boxed{y = 2x + 2}$

Linear Regression

Mini Batch Gradient Descent (Vectorization)

1) Pick m samples (x, y) from training data

2) Compute output $\hat{y}$

$$\hat{y} = \boldsymbol{\theta}^T \boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{\theta}$$

3) Compute loss

$$L = (\hat{y} - y)^2$$

4) Compute derivative

$$\nabla_{\theta} \mathrm{L} = 2\boldsymbol{x}(\hat{y} - y)$$

5) Update parameters

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_{\theta} \mathrm{L}$$

η is learning rate

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

✓ 0.0s

```
(array([[2],
        [3],
        [1]]),
 array([3, 4, 2]))
```

```
x.shape, y.shape
```

✓ 0.0s

```
((3, 1), (3,))
```

```
num_samples = x.shape[0]
num_samples
```

✓ 0.0s

```
3
```

Linear Regression

Mini Batch Gradient Descent (Vectorization)

1) Pick m samples (x, y) from training data

2) Compute output $\hat{y}$

$$\hat{y} = \boldsymbol{\theta}^T \boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{\theta}$$

3) Compute loss

$$L = (\hat{y} - y)^2$$

4) Compute derivative

$$\nabla_\theta \mathrm{L} = 2\boldsymbol{x}(\hat{y} - y)$$

5) Update parameters

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_\theta \mathrm{L}$$

η is learning rate

```
# append bias
x = np.hstack([np.ones((num_samples, 1)), x])
x
```

✓ 0.0s

```
array([[1., 2.],
       [1., 3.],
       [1., 1.]])
```

```
# init weights
theta = np.array([2, 2]) # num_features: x.shape[1]
theta
```

✓ 0.0s

```
array([2, 2])
```

Linear Regression

Mini Batch Gradient Descent (Vectorization)

1) Pick m samples (x, y) from training data

2) Compute output $\hat{y}$

$$\hat{y} = \boldsymbol{\theta}^T \boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{\theta}$$

3) Compute loss

$$L = (\hat{y} - y)^2$$

4) Compute derivative

$$\nabla_{\theta} \mathrm{L} = 2\boldsymbol{x}(\hat{y} - y)$$

5) Update parameters

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_{\theta} \mathrm{L}$$

η is learning rate

```python
# forward
def predict(x, theta):
    return x.dot(theta)

y_hat = predict(x, theta)
y_hat
```

✓ 0.0s

```
array([6., 8., 4.])
```

Linear Regression

Mini Batch Gradient Descent (Vectorization)

1) Pick m samples (x, y) from training data

2) Compute output $\hat{y}$

$$\hat{y} = \boldsymbol{\theta}^T \boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{\theta}$$

3) Compute loss

$$L = (\hat{y} - y)^2$$

4) Compute derivative

$$\nabla_{\theta} \mathrm{L} = 2\boldsymbol{x}(\hat{y} - y)$$

5) Update parameters

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_{\theta} \mathrm{L}$$

η is learning rate

```
# compute loss
def compute_loss(y_hat, y):
    return (y_hat-y)*(y_hat-y)

loss = compute_loss(y_hat, y)
loss
```

✓ 0.0s

```
array([ 9., 16.,  4.])
```

Linear Regression

Mini Batch Gradient Descent (Vectorization)

1) Pick m samples (x, y) from training data

2) Compute output $\hat{y}$

$$\hat{y} = \boldsymbol{\theta}^T \boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{\theta}$$

3) Compute loss

$$L = (\hat{y} - y)^2$$

4) Compute derivative

$$\nabla_\theta \mathrm{L} = 2\boldsymbol{x}(\hat{y} - y)$$

5) Update parameters

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_\theta \mathrm{L}$$

η is learning rate

```python
# compute gradient
def compute_gradient(x, y, y_hat):
    d_theta = 2*x.T.dot(y_hat-y)
    return d_theta

d_theta = compute_gradient(x, y, y_hat)
d_theta
```

✓ 0.0s

```
array([18., 40.])
```

Linear Regression

Mini Batch Gradient Descent (Vectorization)

1) Pick m samples (x, y) from training data

2) Compute output $\hat{y}$

$$\hat{y} = \boldsymbol{\theta}^T \boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{\theta}$$

3) Compute loss

$$L = (\hat{y} - y)^2$$

4) Compute derivative

$$\nabla_{\theta} \mathrm{L} = 2\boldsymbol{x}(\hat{y} - y)$$

5) Update parameters

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_{\theta} \mathrm{L}$$

η is learning rate

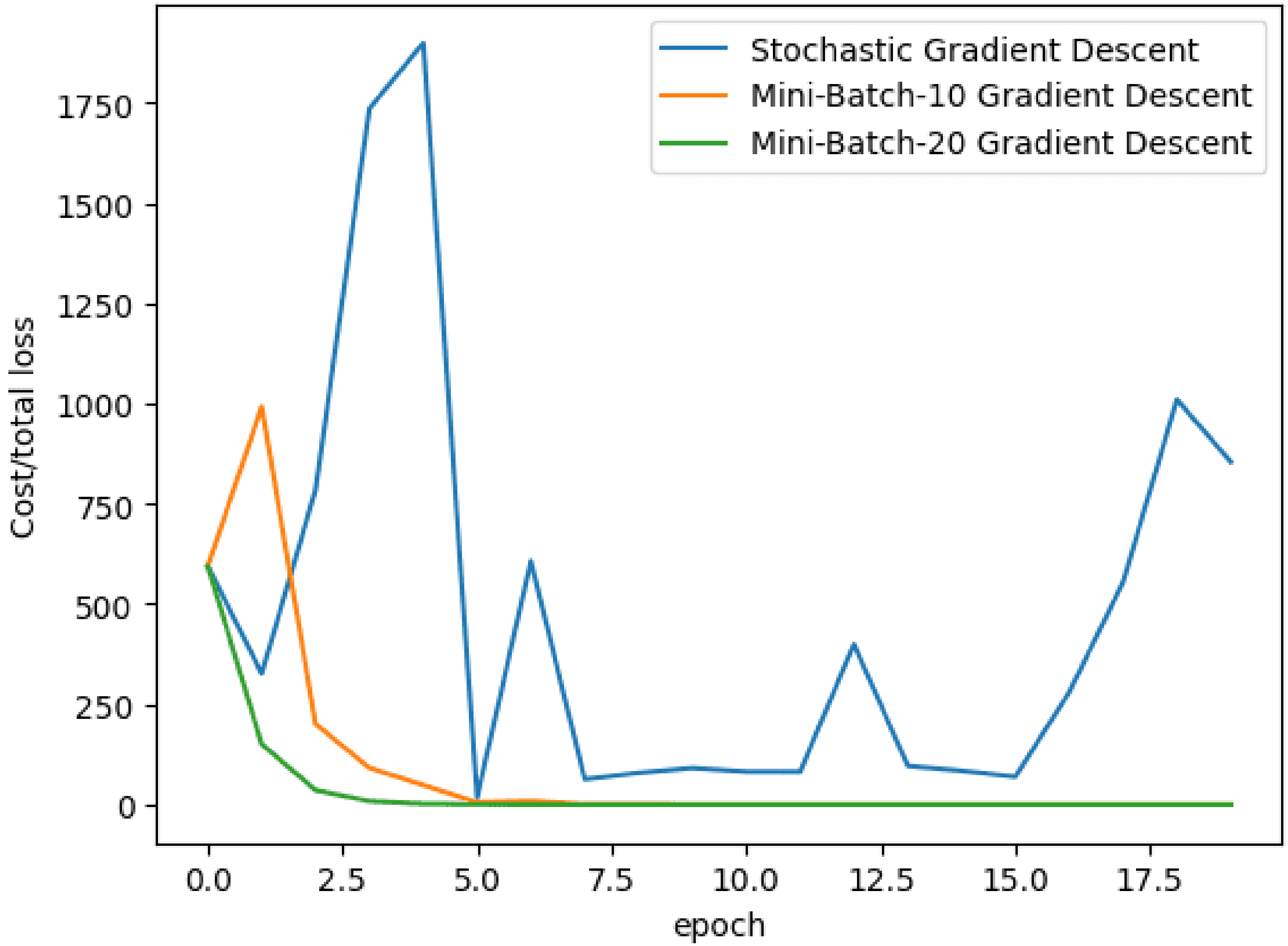
```python
# update weights
lr = 0.1

def update_weights(theta, d_theta, lr):
    new_theta = theta - lr*d_theta
    return new_theta

new_theta = update_weights(theta, d_theta, lr)
new_theta
```

✓ 0.0s

```
array([ 0.2, -2. ])
```

Linear Regression

Comparision

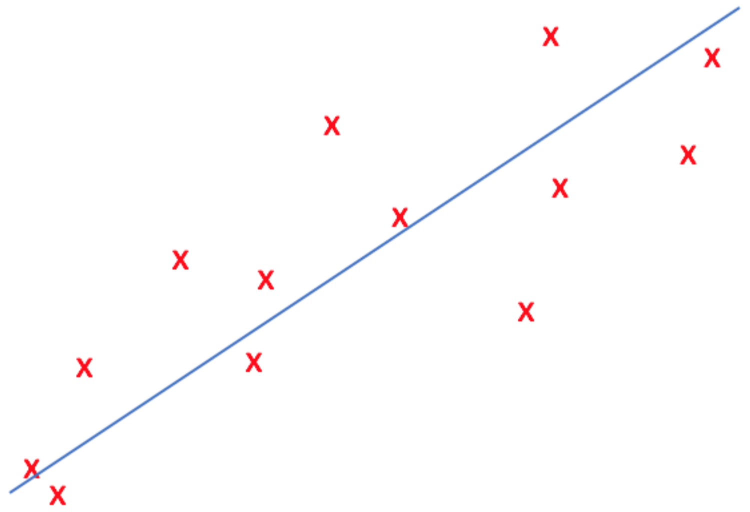


QUIZ TIME

Outline

SECTION 1

Linear Regression



SECTION 2

Time Series Prediction

Time Series Prediction

Bitcoin Dataset

```
df = pd.read_csv('/content/BTC-Daily.csv')
df = df.drop_duplicates()

```

```
df.head()
```

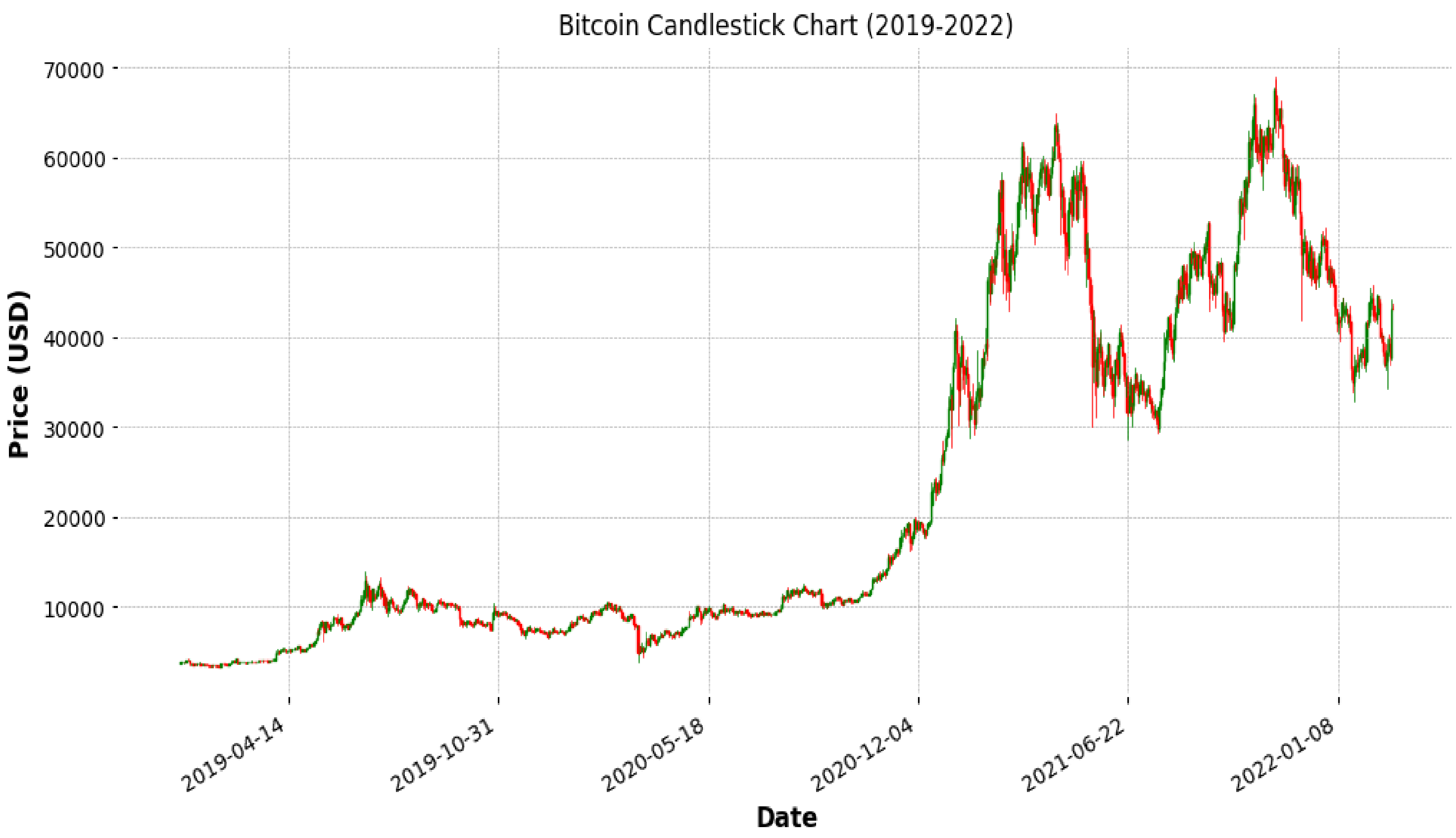
	unix	date	symbol	open	high	low	close	Volume BTC	Volume USD
0	1646092800	2022-03-01 00:00:00	BTC/USD	43221.71	43626.49	43185.48	43185.48	49.006289	2.116360e+06
1	1646006400	2022-02-28 00:00:00	BTC/USD	37717.10	44256.08	37468.99	43178.98	3160.618070	1.364723e+08
2	1645920000	2022-02-27 00:00:00	BTC/USD	39146.66	39886.92	37015.74	37712.68	1701.817043	6.418008e+07
3	1645833600	2022-02-26 00:00:00	BTC/USD	39242.64	40330.99	38600.00	39146.66	912.724087	3.573010e+07
4	1645747200	2022-02-25 00:00:00	BTC/USD	38360.93	39727.97	38027.61	39231.64	2202.851827	8.642149e+07

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2651 entries, 0 to 2650
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   unix        2651 non-null   int64
 1   date        2651 non-null   object
 2   symbol      2651 non-null   object
 3   open        2651 non-null   float64
 4   high        2651 non-null   float64
 5   low         2651 non-null   float64
 6   close       2651 non-null   float64
 7   Volume BTC  2651 non-null   float64
 8   Volume USD  2651 non-null   float64
dtypes: float64(6), int64(1), object(2)
memory usage: 186.5+ KB
```

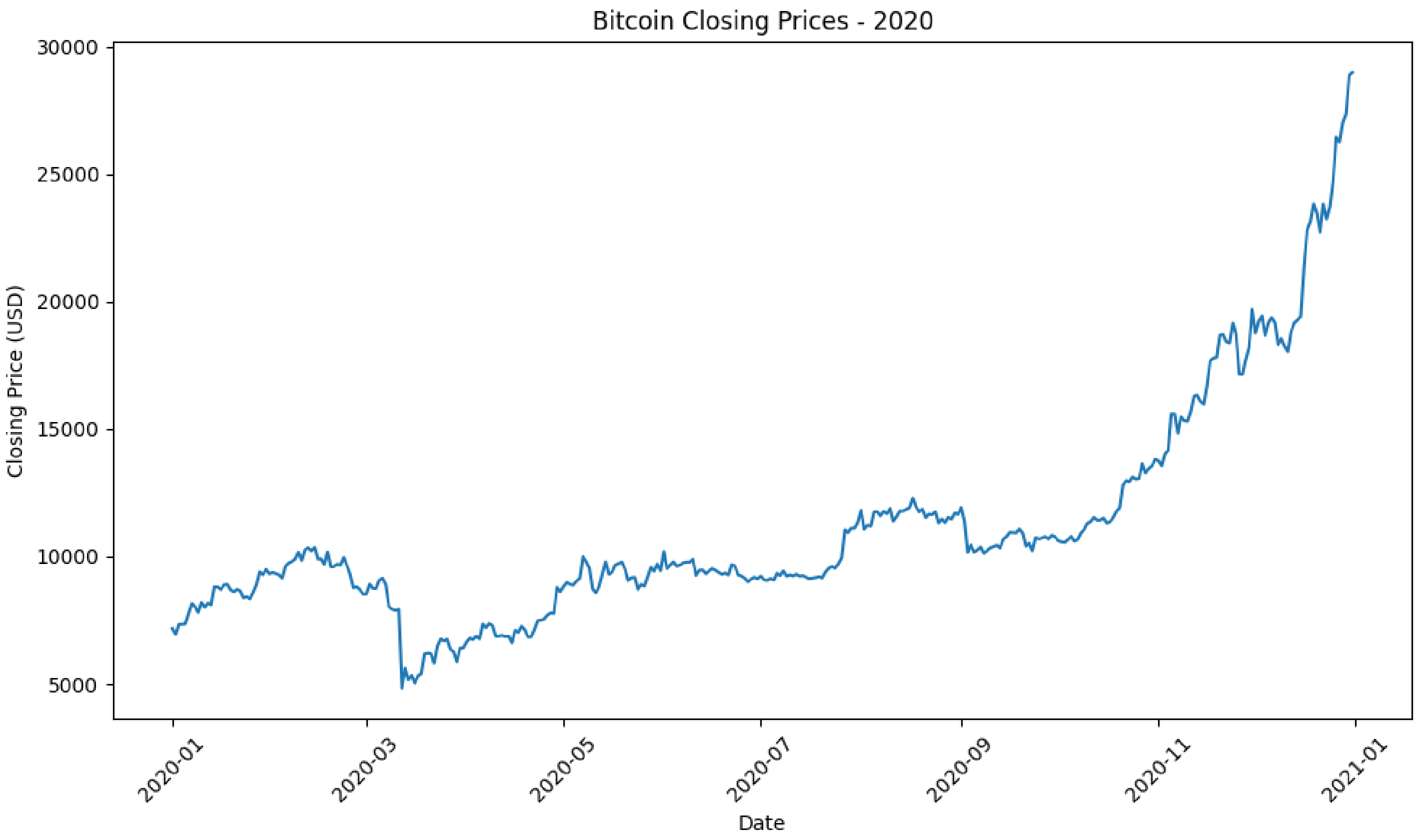
Time Series Prediction

Bitcoin Dataset



Time Series Prediction

Bitcoin Dataset



Time Series Prediction

Feature Scaling

MaxAbsScaler

$$x_{new} = \frac{x}{x_{max}}$$


```python
import numpy as np
from sklearn.preprocessing import MaxAbsScaler

X = np.array([[1], [2], [3], [4]])

scaler = MaxAbsScaler()
X_transform = scaler.fit_transform(X)
X_transform
```

✓ 0.0s

```
array([[0.25],
       [0.5 ],
       [0.75],
       [1.  ]])
```

Time Series Prediction

Feature Scaling

MinMaxScaler

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

1
2
3
4

0.0
0.33
0.67
1.0

```python
import numpy as np
from sklearn.preprocessing import MinMaxScaler

X = np.array([[1], [2], [3], [4]])

scaler = MinMaxScaler()
X_transform = scaler.fit_transform(X)
X_transform
```

✓ 0.0s

```
array([[0.        ],
       [0.33333333],
       [0.66666667],
       [1.        ]])
```

Time Series Prediction

Feature Scaling

StandardScaler

$$x_{new} = \frac{x - \mu}{\sigma}$$

$$\mu = \frac{1}{N}\sum x_i \qquad \sigma = \sqrt{\frac{1}{N}\sum_i (x_i - \mu)^2}$$

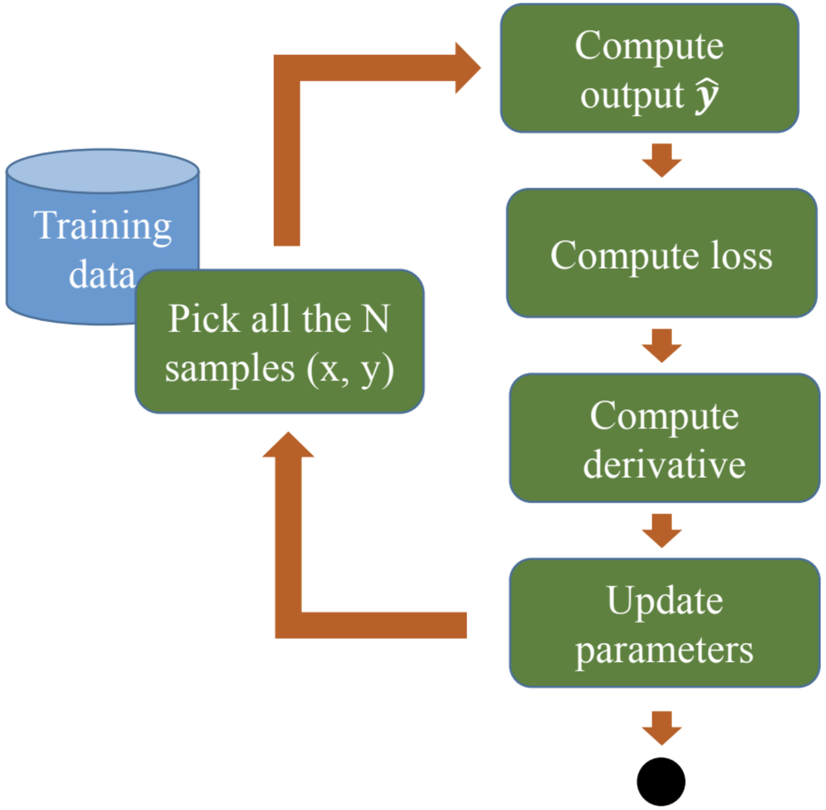
```python
import numpy as np
from sklearn.preprocessing import StandardScaler

X = np.array([[1], [2], [3], [4]])

scaler = StandardScaler()
X_transform = scaler.fit_transform(X)
X_transform
```

✓ 0.0s

```
array([[-1.34164079],
       [-0.4472136 ],
       [ 0.4472136 ],
       [ 1.34164079]])
```

Time Series Prediction

Modeling



1) Pick all the N samples from training data

2) Compute output $\hat{\mathbf{y}}$

$$\hat{\boldsymbol{y}} = \boldsymbol{x}\boldsymbol{\theta}$$

3) Compute loss

$$\boldsymbol{L}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = (\hat{\boldsymbol{y}} - \boldsymbol{y}) \odot (\hat{\boldsymbol{y}} - \boldsymbol{y})$$

4) Compute derivative

$$\boldsymbol{k} = 2(\hat{\boldsymbol{y}} - \boldsymbol{y})$$

$$L'_{\boldsymbol{\theta}} = \boldsymbol{x}^T \boldsymbol{k}$$

5) Update parameters

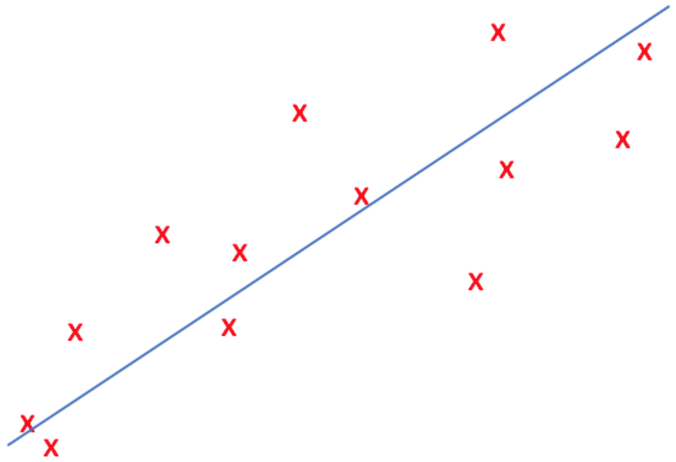
$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \frac{L'_{\boldsymbol{\theta}}}{N}$$

η is learning rate

Summary

Linear Regression

- Introduction
- Stochastic Gradient Descent
- Mini Batch Gradient Descent
- Batch Gradient Descent



Linear Regression for Timeseries

- Bitcoin Dataset
- Feature Scaling
- Modeling
- Evaluation
- Inference

Thanks!

Any questions?