ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA

TRÍ TUỆ NHÂN TẠO

Khoa Công Nghệ Thông Tin
TS. Nguyễn Văn Hiệu

Khoa Công Nghệ Thông Tin
TS. Nguyễn Văn Hiệu

TRÍ TUỆ NHÂN TẠO

Chương 5: Hồi quy tuyến tính

Nội dung

- Giới thiệu
- Hồi quy tuyến tính
  - Hồi quy tuyến tính đơn biến
  - Hồi quy tuyến tính đa biến

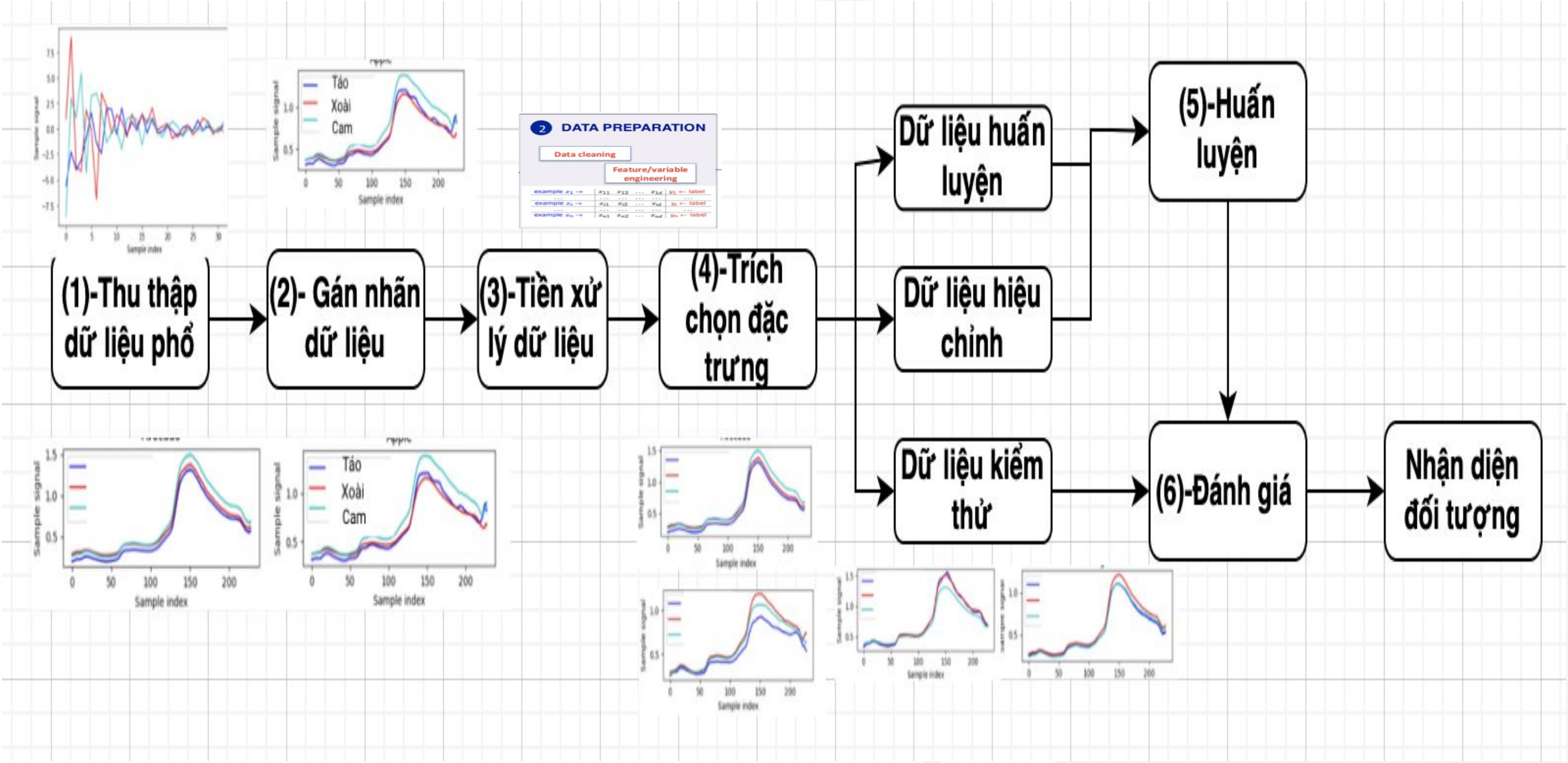
Giới thiệu

Giới thiệu

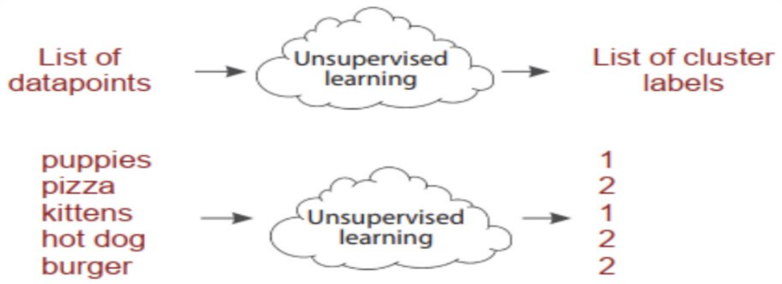
- Học vẹt: Nhớ và Làm lại
- Học hiểu: Quan sát và Khái quát
- Học làm gì:
  - Để nhận biết.
  - Để xây dựng.
  - Để thông minh hơn.
- Kiểu học:
  - Học có giám sát(có thầy)
  - Học không giám sát(không thầy)
  - Học tăng cường(có phần thưởng)
  - Học bán giám sát (một phần có thầy + một phần không có thầy)

Giới thiệu

- Quy trình xử lý công nghệ nhận diện đối tượng

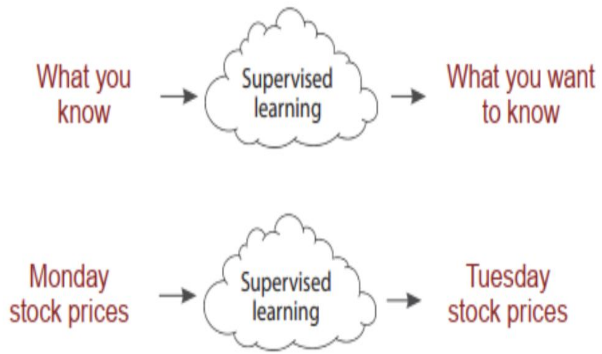


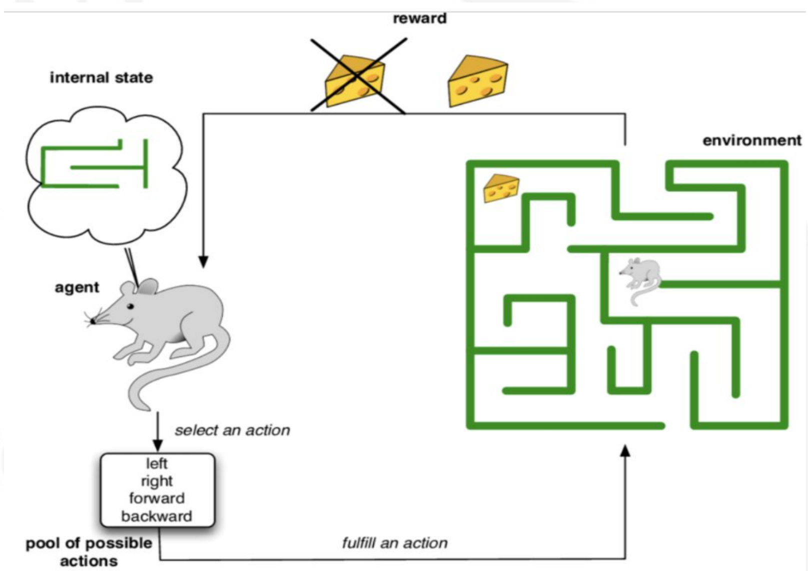
Giới thiệu



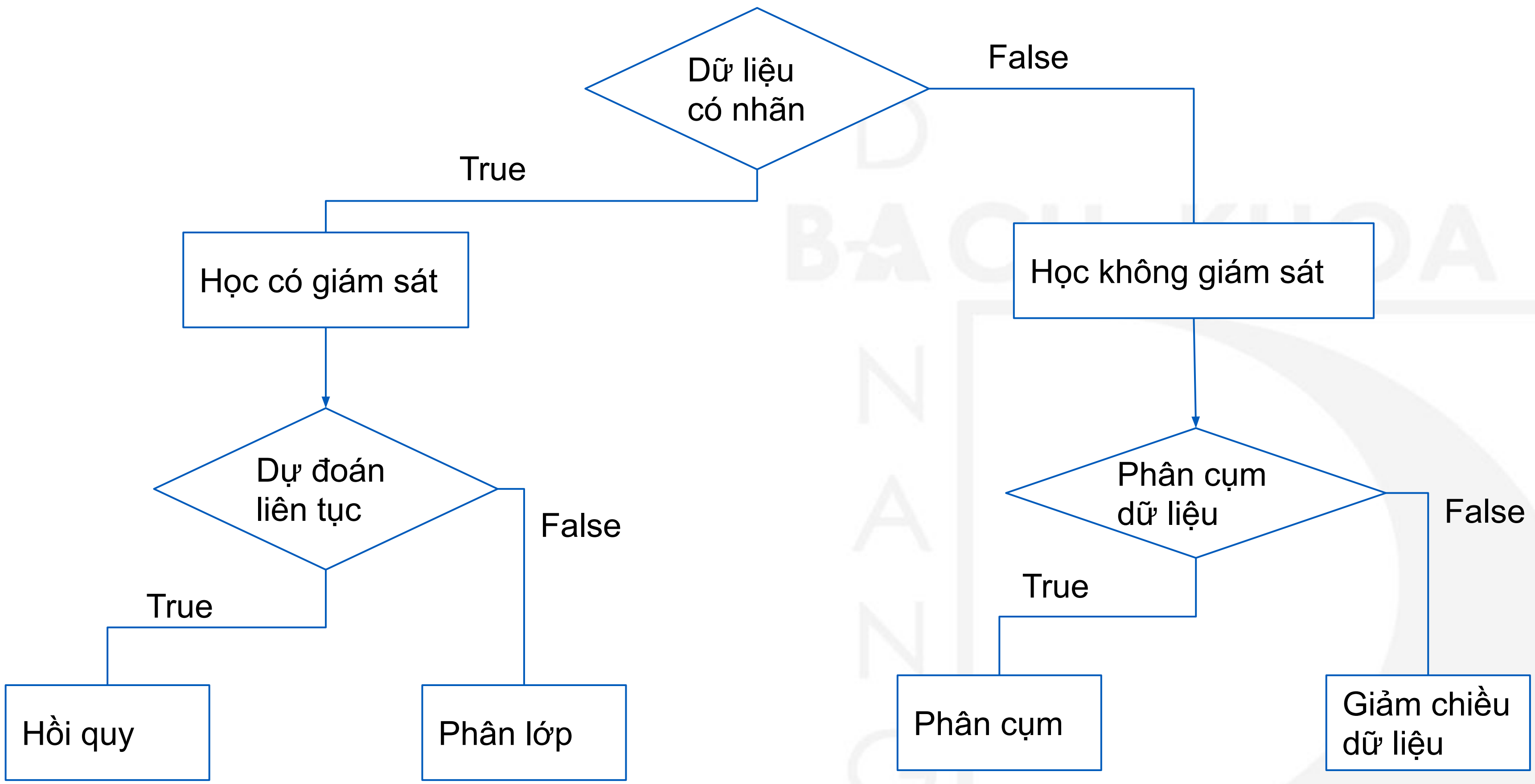
Học máy là gì ?

- Lĩnh vực tập trung vào việc phát triển các thuật toán/mô hình máy tính có khả năng tự học hỏi từ dữ liệu và cải thiện hiệu suất của chúng theo thời gian mà không cần lập trình cụ thể.
- Học có giám sát, không giám sát, học tăng cường





Giới thiệu


Dữ liệu có nhãn
True
False
Học có giám sát
Học không giám sát
Dự đoán liên tục
True
False
Phân cụm dữ liệu
True
False
Hồi quy
Phân lớp
Phân cụm
Giảm chiều dữ liệu

Giới thiệu


Machine Learning Algorithms
Deep Learning
Deep Boltzmann Machine (DBM)
Deep Belief Networks (DBN)
Convolutional Neural Network (CNN)
Stacked Auto-Encoders
Ensemble
Random Forest
Gradient Boosting Machines (GBM)
Boosting
Bootstrapped Aggregation (Bagging)
AdaBoost
Stacked Generalization (Blending)
Gradient Boosted Regression Trees (GBRT)
Neural Networks
Radial Basis Function Network (RBFN)
Perceptron
Back-Propagation
Hopfield Network
Regularization
Ridge Regression
Least Absolute Shrinkage and Selection Operator (LASSO)
Elastic Net
Least Angle Regression (LARS)
Rule System
Cubist
One Rule (OneR)
Zero Rule (ZeroR)
Repeated Incremental Pruning to Produce Error Reduction (RIPPER)
Regression
Linear Regression
Ordinary Least Squares Regression (OLSR)
Stepwise Regression
Multivariate Adaptive Regression Splines (MARS)
Locally Estimated Scatterplot Smoothing (LOESS)
Logistic Regression
Bayesian
Naive Bayes
Averaged One-Dependence Estimators (AODE)
Bayesian Belief Network (BBN)
Gaussian Naive Bayes
Multinomial Naive Bayes
Bayesian Network (BN)
Decision Tree
Classification and Regression Tree (CART)
Iterative Dichotomiser 3 (ID3)
C4.5
C5.0
Chi-squared Automatic Interaction Detection (CHAID)
Decision Stump
Conditional Decision Trees
M5
Dimensionality Reduction
Principal Component Analysis (PCA)
Partial Least Squares Regression (PLSR
Sammon Mapping
Multidimensional Scaling (MDS)
Projection Pursuit
Principal Component Regression (PCR)
Partial Least Squares Discriminant Analysis
Mixture Discriminant Analysis (MDA)
Quadratic Discriminant Analysis (QDA)
Regularized Discriminant Analysis (RDA)
Flexible Discriminant Analysis (FDA)
Linear Discriminant Analysis (LDA)
Instance Based
k-Nearest Neighbour (kNN)
Learning Vector Quantization (LVQ)
Self-Organizing Map (SOM)
Locally Weighted Learning (LWL)
Clustering
k-Means
k-Medians
Expectation Maximization
Hierarchical Clustering

Học có giám sát

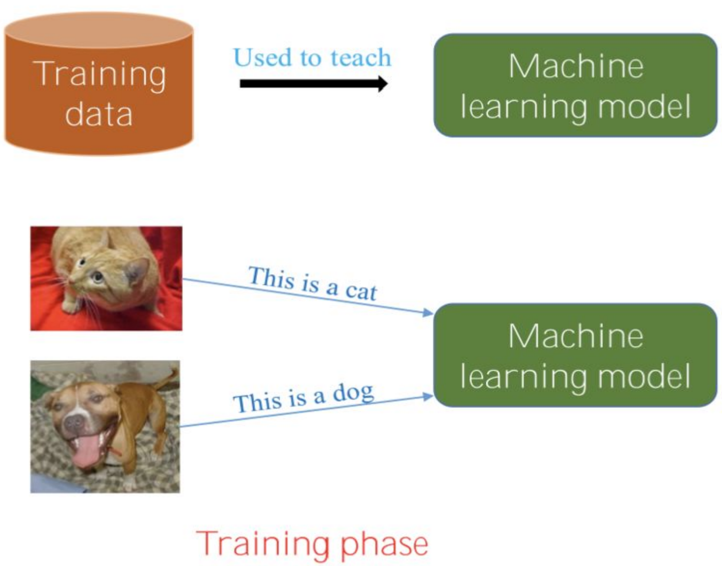
- Dữ liệu

  Dữ liệu vào và Nhãn đồng thời

- Dữ liệu huấn luyện
  - Cats
  - Dogs

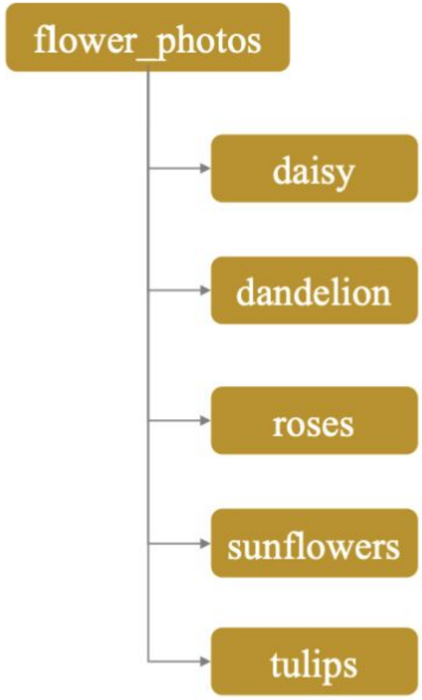
Học có giám sát

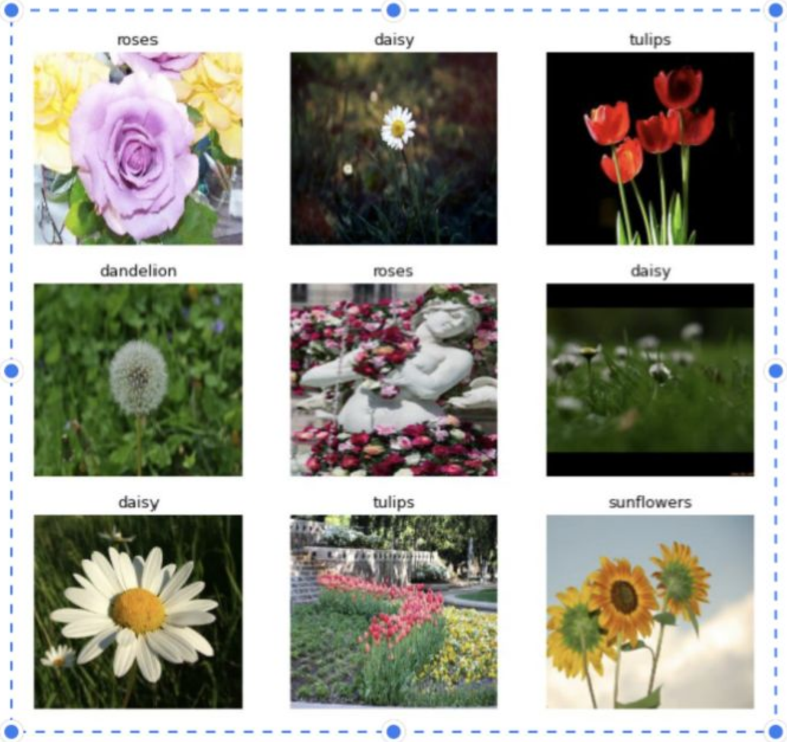
- Dữ liệu



Học có giám sát

- Dữ liệu



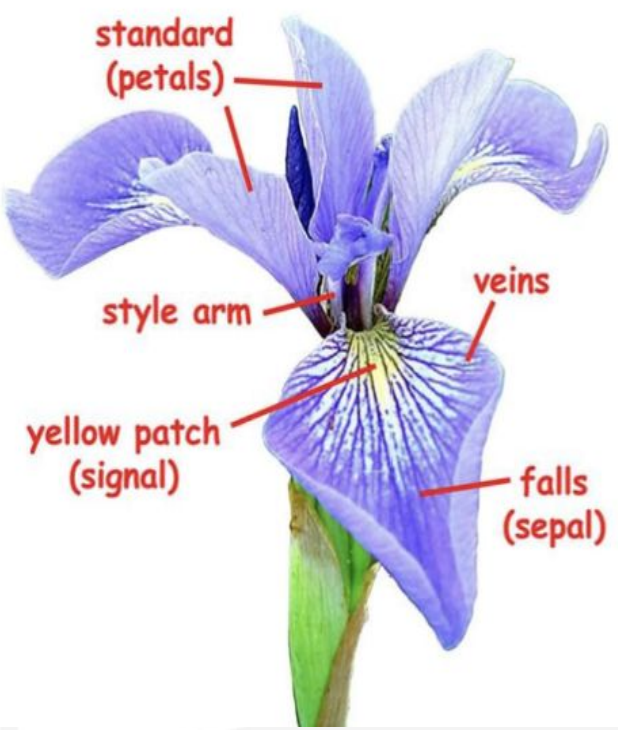


Học có giám sát

- Dữ liệu

Feature | Label

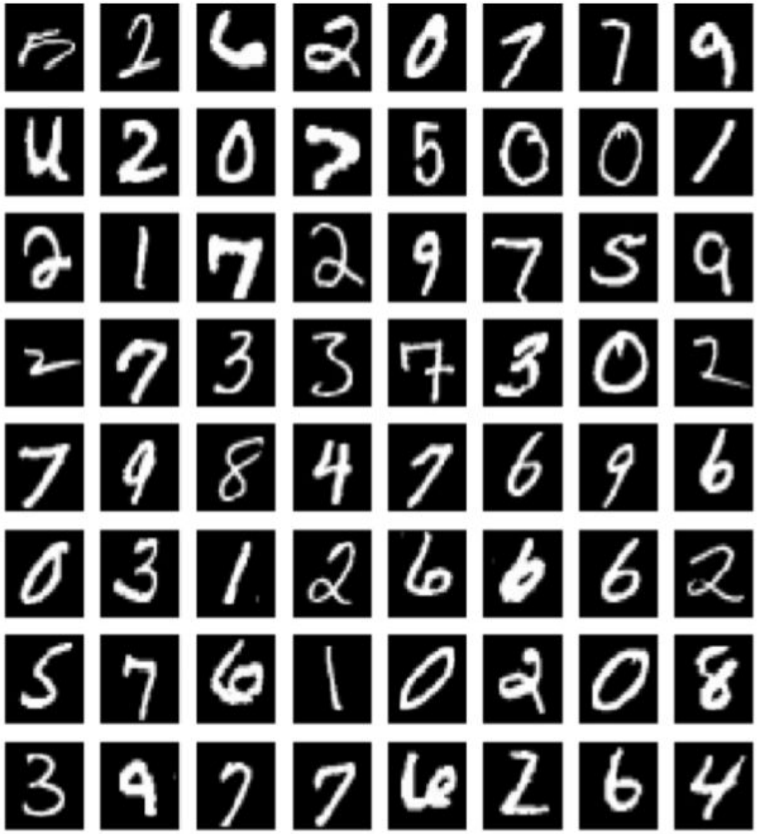
Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Label
5.1	3.5	1.4	0.2	0
4.9	3	1.4	0.2	0
4.7	3.2	1.3	0.2	0
6.4	3.2	4.5	1.5	1
6.9	3.1	4.9	1.5	1
5.5	2.3	4	1.3	1
4.9	2.5	4.5	1.7	2
7.3	2.9	6.3	1.8	2
6.7	2.5	5.8	1.8	2



Giới thiệu

MNIST dataset

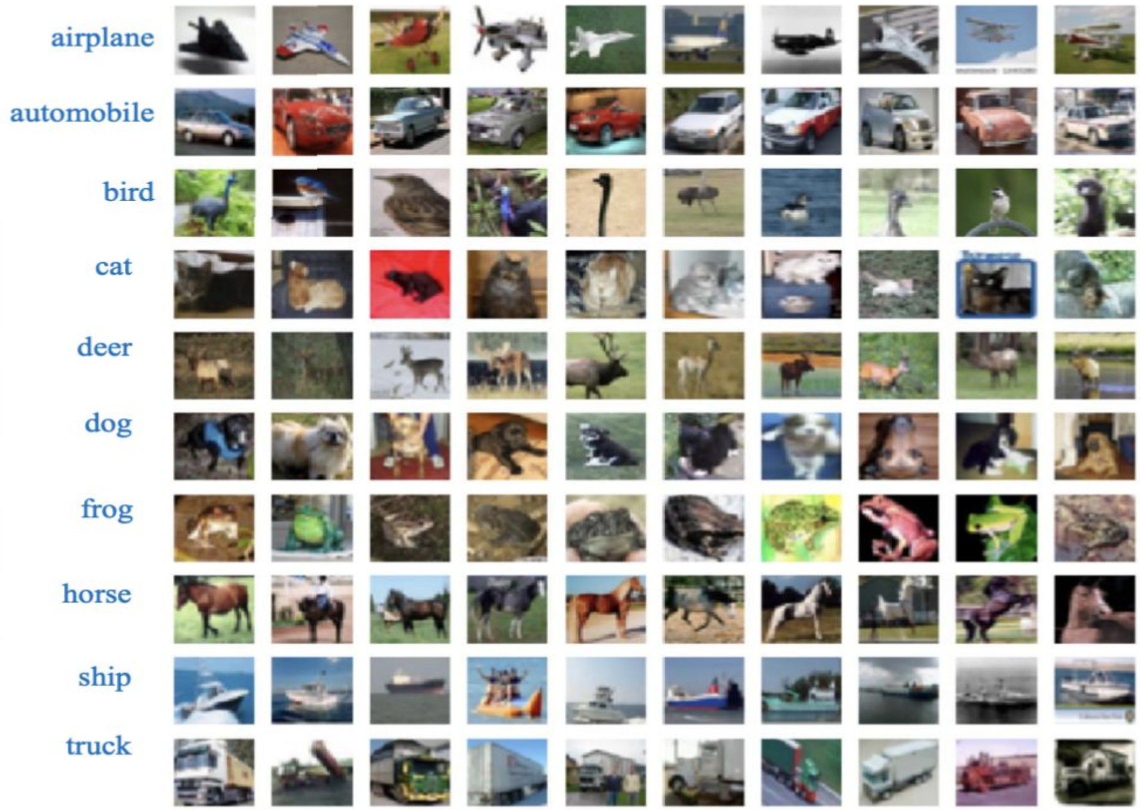
- Grayscale images
- Resolution=28x28
- Training set: 60000 samples
- Testing set: 10000 samples



Giới thiệu

Cifar-10 dataset

- Color images
- Resolution=32x32
- Training set: 50000 samples
- Testing set: 10000 samples



Hồi quy tuyến tính

Học có giám sát

- **Dữ liệu huấn luyện**: mẫu **x** với nhãn **y**

  $(x_1, y_1), \ldots, (x_n, y_n)$. Với x_i thuộc R^d

- **Hồi quy:** **y** là giá trị thực. **y** thuộc **R**

  f: $R^d \longrightarrow R$ **hàm f** gọi là **hàm hồi quy**

- **Phân lớp**: **y** là giá trị thực. ví dụ y thuộc {+, -}

  f: $R^d \longrightarrow \{-1, 1\}$ **f** gọi là **hàm phân lớp nhị phân**

Hồi quy tuyến tính(1)

- **Cho:** **Dữ liệu huấn luyện** mẫu **x** với nhãn **y**

  $(x_1,y_1),\ldots,(x_n,y_n)$. Với x_i thuộc R^d

$x_1 \rightarrow$	x_{11}	x_{12}	$\ldots$	x_{1d}	y_1
	$\ldots$	$\ldots$	$\ldots$	$\ldots$	
$x_i \rightarrow$	x_{i1}	x_{i2}	$\ldots$	x_{id}	y_i
	$\ldots$	$\ldots$	$\ldots$	$\ldots$	
$x_n \rightarrow$	x_{n1}	x_{n2}	$\ldots$	x_{nd}	y_n

- **Tác vụ:** huấn luyện (learning) hàm hồi quy f:

  $f: R^d \longrightarrow R$

  $f(x) = y$

- **Hồi quy tuyến tính** (linear regression): Mô hình hồi quy gọi là tuyến tính, nếu hàm hồi quy là tuyến tính.

Hồi quy tuyến tính(2)

- **Mô hình hồi quy tuyến tính**

$$f(x) = \beta_0 + \sum_{j=1}^{d} \beta_j x_j,\ \beta_j \in R,\ j = 1, \ldots, d$$

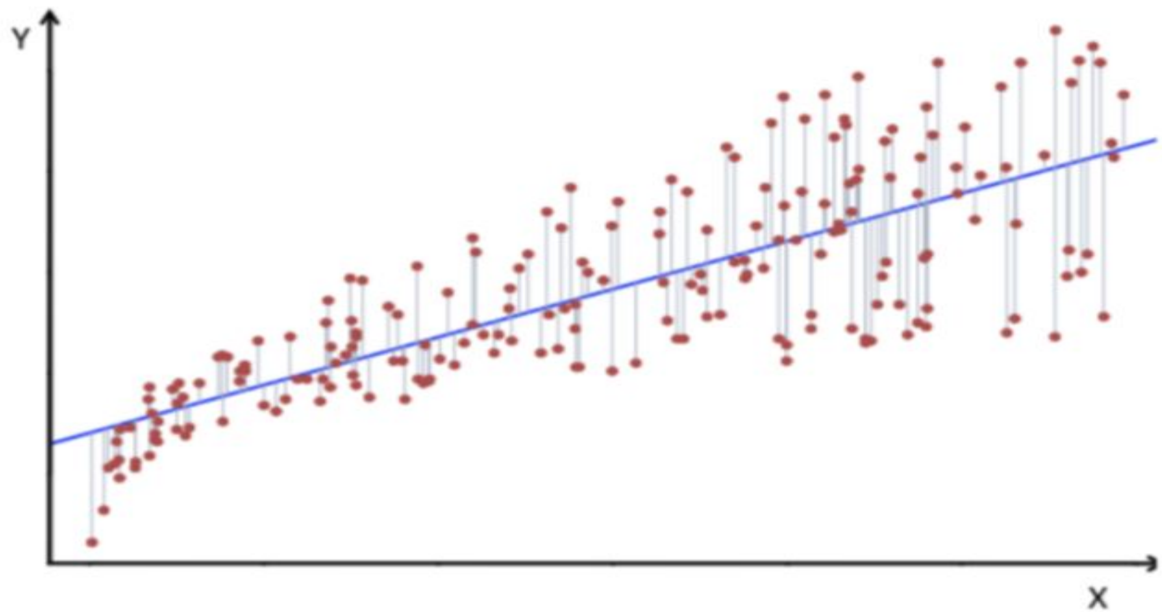
Huấn luyện mô hình tuyến tính —> Huấn luyện tham số

- **Ước lượng mất mát bằng kỹ thuật bình phương tối thiểu**
  - Mất mát tại một điểm dữ liệu: $loss(y_i,\ f(x_i)) = (y_i - f(x_i))^2$
  - Mất mát toàn cục là hàm số R, cần đạt giá trị bé nhất

$$R = \frac{1}{2n} \sum_{i=1}^{n} loss(y_i,\ f(x_i)) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

Hồi quy tuyến tính đơn biến(1)

- Trường hợp đặc biệt khi d = 1



$d = 1$, line in $\mathbb{R}^2$

Hồi quy tuyến tính đơn biến(1)

- Trường hợp đặc biệt khi d = 1
  - Dữ liệu

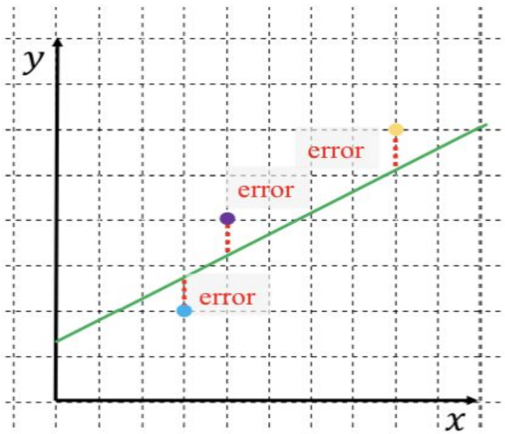
area	price
6.7	9.1
4.6	5.9
3.5	4.6
5.5	6.7

  - mô hình $f(x) = \beta_1 x + \beta_0$
  - $Price = \beta_1 \times area + \beta_0$

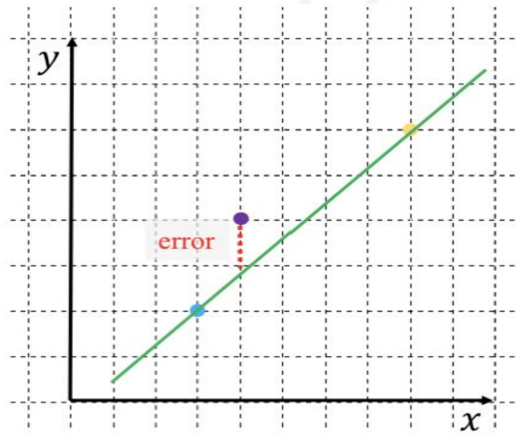
Hồi quy tuyến tính đơn biến(1)

area	price
6.7	9.1
4.6	5.9
3.5	4.6
5.5	6.7

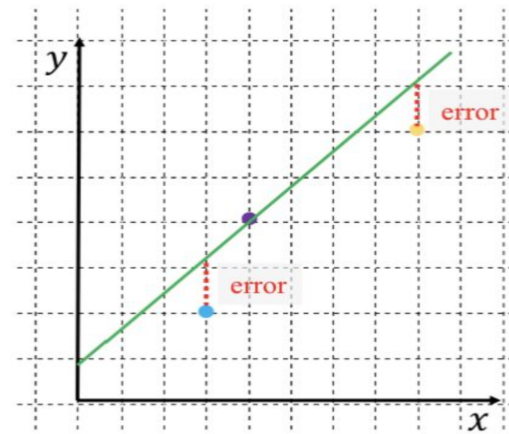
- Trường hợp đặc biệt khi d = 1



$$f(x) = \beta_1 x + \beta_0$$



$$f(x) = \beta_1 x + \beta_0$$



$$f(x) = \beta_1 x + \beta_0$$

- **Tìm siêu tham số sao cho lỗi nhỏ nhất ? Tìm như thế nào ?**

Hồi quy tuyến tính đơn biến(2)

- Trường hợp có 1 đặc trưng (d = 1)

$$f(x) = \beta_0 + \beta_1 x$$

- Cần tối ưu hàm mất mát toàn cục

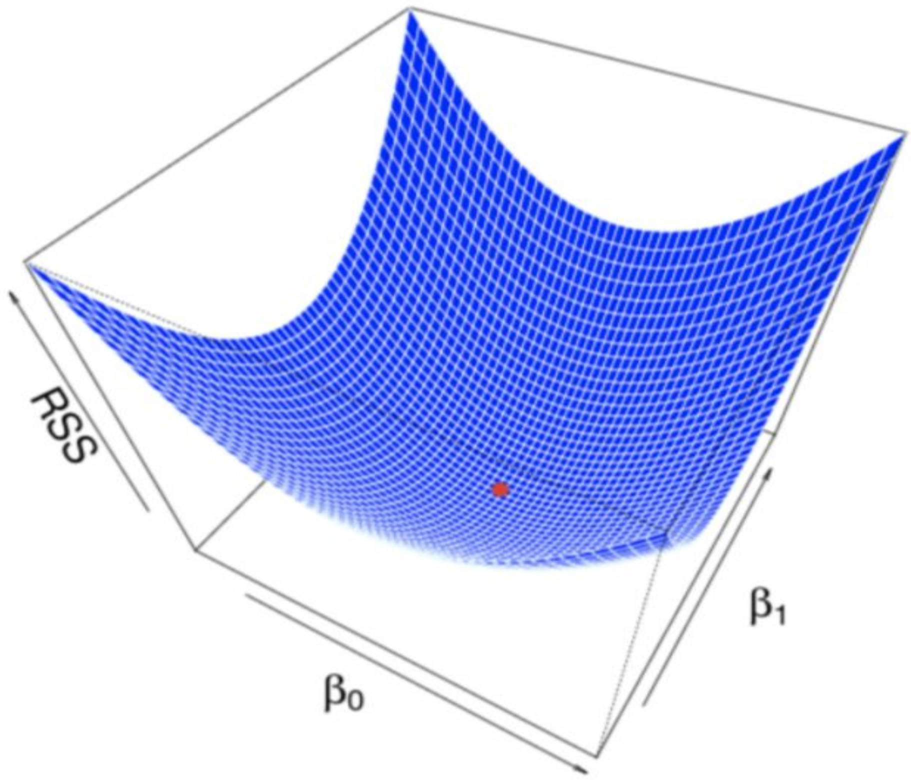
$$R = \frac{1}{2n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

$$R(\beta) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2$$

- Tìm kiếm tham số

$$R(\beta) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2$$

Hồi quy tuyến tính đơn biến (3)



Hồi quy tuyến tính đơn biến(4)

Find β_0 and β_1 so that:

$$argmin_\beta\left(\frac{1}{2n}\sum_{i=1}^{n}(y_i-\beta_0-\beta_1x_i)^2\right)$$

Minimize: $R(\beta_0, \beta_1)$, that is: $\frac{\partial R}{\partial \beta_0} = 0 \qquad \frac{\partial R}{\partial \beta_1} = 0$

$$\frac{\partial R}{\partial \beta_0} = 2 \times \frac{1}{2n}\sum_{i=1}^{n}(y_i-\beta_0-\beta_1x_i) \times \frac{\partial}{\partial \beta_0}(y_i-\beta_0-\beta_1x_i)$$

$$\frac{\partial R}{\partial \beta_0} = \frac{1}{n}\sum_{i=1}^{n}(y_i-\beta_0-\beta_1x_i) \times (-1) = 0$$

$$\beta_0 = \frac{1}{n}\sum_{i=1}^{n}y_i - \beta_1\frac{1}{n}\sum_{i=1}^{n}x_i$$

Hồi quy tuyến tính đơn biến(5)

$$\frac{\partial R}{\partial \beta_1} = 2 \times \frac{1}{2n} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i) \times \frac{\partial}{\partial \beta_1} (y_i - \beta_0 - \beta_1 x_i)$$

$$\frac{\partial R}{\partial \beta_1} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i) \times (-x_i) = 0$$

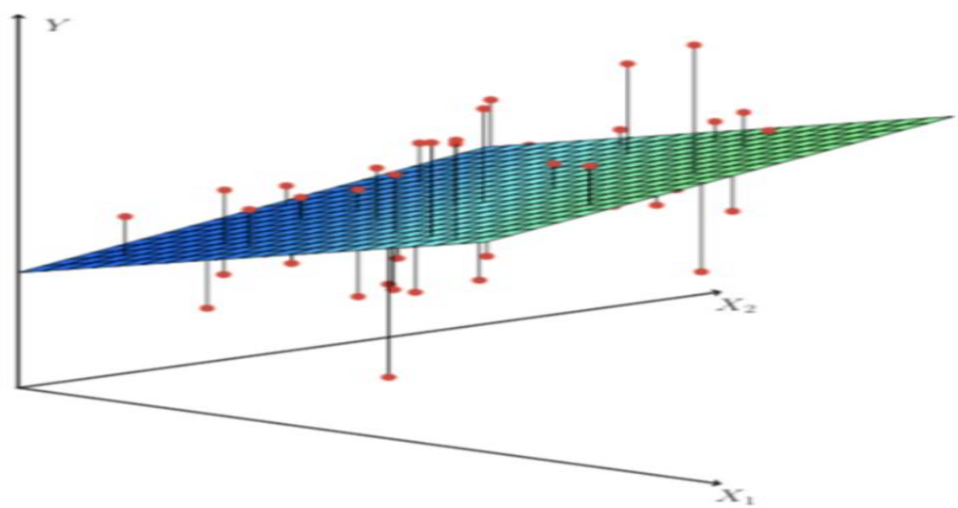
$$\beta_1 \sum_{i=1}^{n} {x_i}^2 = \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} \beta_0 x_i$$

Plugging β_0 in β_1:

$$\beta_1 = \frac{\sum_{i=1}^{n} y_i x_i - \frac{1}{n} \sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} x_i^2 - \frac{1}{n} \sum_{i=1}^{n} x_i \sum x_i}$$

Hồi quy tuyến tính đa biến (1)

$$f\left(x^{(i)}\right) = \beta_0 + \beta_1 x_1^{(i)} + \beta_2 x_2^{(i)} + \ldots + \beta_d x_d^{(i)}$$



$d = 2$, hyperplane is $\mathbb{R}^3$

Hồi quy tuyến tính đa biến(2)

- Số đặc trưng bằng d

$$f(x) = \beta_0 + \sum_{j=1}^{d} \beta_j x_j, \ \beta_j \in R, \ j = 1, \ldots, d$$

- Tìm tham số để hàm mất mát tối ưu

$$R = \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{d} \beta_j x_{ij} \right)^2$$

- Chuyển sang biểu diễn ma trận

Hồi quy tuyến tính đa biến(3)

- Biểu diễn bằng ma trận

$$X := \begin{pmatrix} 1 & x_{11} & \cdots & x_{1j} & \cdots & x_{1d} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{i1} & \cdots & x_{ij} & \cdots & x_{id} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & \cdots & x_{nj} & \cdots & x_{nd} \end{pmatrix}$$

$$y := \begin{pmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{pmatrix} \qquad \beta := \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_j \\ \vdots \\ \beta_d \end{pmatrix}$$

Hồi quy tuyến tính đa biến (4)

We want to find $(d+1)$ β's that minimize R. We write R:

$$R(\beta) = \frac{1}{2n}||(y - X\beta)||^2$$

$$R(\beta) = \frac{1}{2n}(y - X\beta)^T(y - X\beta)$$

$$\frac{\partial R}{\partial \beta} = -\frac{1}{n}X^T(y - X\beta)$$

We have that:

$$\frac{\partial^2 R}{\partial \beta} = -\frac{1}{n}X^T X$$

is positive definite which ensures that β is a minimum. We solve:

$$X^T(y - X\beta) = 0$$

The unique solution is: $\boxed{\beta = (X^T X)^{-1} X^T y}$

Đánh giá

- Chỉ hoạt động được khi $\mathbf{X^T X}$ có $\mathbf{det(X^T X)}$ khác 0
- Sẽ chậm nếu d lớn: $\mathbf{0(d^3)}$

Gradient Descent(1) (Chi tiết)

Gradient descent cho hàm một biến

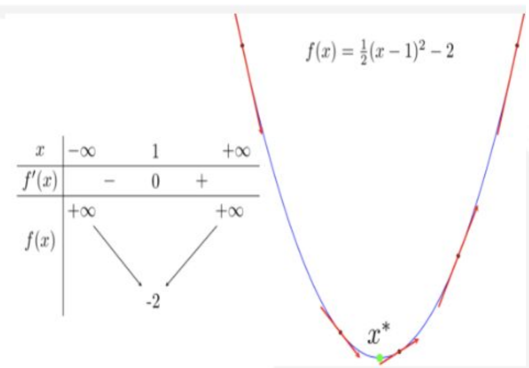
Xem ví dụ trước

- Nếu $f'(x_t) > 0$, thì x_t nằm bên phải x^*. Cần dịch chuyển sang trái

$$x_{t+1} = x_t + \Delta = x_t - \alpha.\, f'(x_t)$$

- Nếu $f'(x_t) < 0$, thì x_t nằm bên trái x^*. Cần dịch chuyển sang phải

$$x_{t+1} = x_t + \Delta = x_t - \alpha.\, f'(x_t)$$



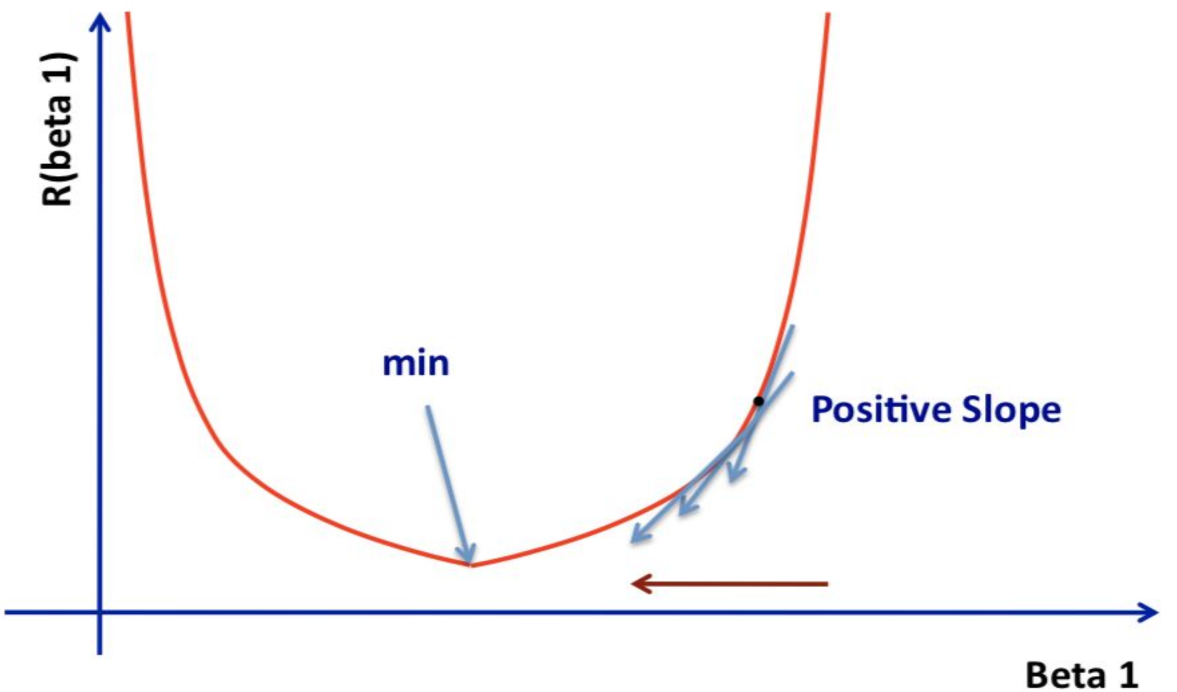
- Thuật toán Gradient descent:
  - Dự đoán một điểm khởi tạo $x_t = x_0$
  - Cập nhật x_t đến đạt đến kết quả chấp nhận

$$x_t := x_t - \alpha f'(x_t)$$

- Khái niệm "Gradient descent" chính dấu "-" : ngược hướng đàm hàm

Gradient Descent(2)

- Trực quan



Gradient Descent(3) (Chi tiết)

Gradient Descent cho hàm đa biến

- Bài toán:
  - Tối ưu cho hàm $f(\theta), \theta = (\theta_0, \theta_1, \ldots, \theta_n)^T$
- Đạo hàm tại điểm θ: $\Delta_\theta f(\theta)$
- **Thuật toán** (tương tự hàm một biến)
  - Dự đoán một điểm khởi tạo θ
  - Cập nhật θ đến khi nhận được kết quả chấp nhận được

$$\theta = \theta - \alpha \cdot \Delta_\theta f(\theta)$$

Gradient Descent(4)

Repeat until convergence:

Update **simultaneously** all β_j for ($j = 0$ and $j = 1$)

$$\beta_0 := \beta_0 - \alpha \frac{\partial}{\partial \beta_0} R(\beta_0, \beta_1)$$

$$\beta_1 := \beta_1 - \alpha \frac{\partial}{\partial \beta_1} R(\beta_0, \beta_1)$$

α is a learning rate.

Gradient Descent(5)

In the linear case:

$$\frac{\partial R}{\partial \beta_0} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i) \times (-1) = 0$$

$$\frac{\partial R}{\partial \beta_1} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i) \times (-x_i)$$

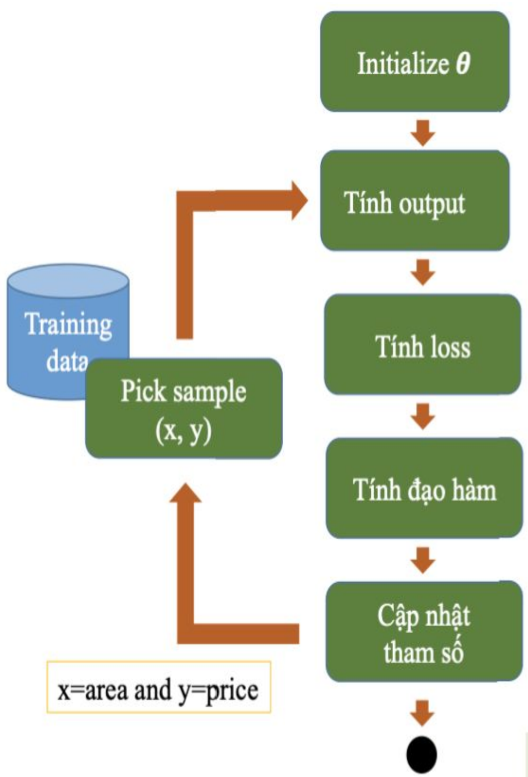
Repeat until convergence:

Update **simultaneously** all β_j for ($j = 0$ and $j = 1$)

$$\beta_0 := \beta_0 - \alpha\frac{1}{n}\sum_{i=1}^{n}(\beta_0 + \beta_1 x_i - y_i)$$

$$\beta_1 := \beta_1 - \alpha\frac{1}{n}\sum_{i=1}^{n}(\beta_0 + \beta_1 x_i - y_i)(x_i)$$

Hội quy đơn biến



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

2) Tính output $\hat{y}$

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

3) Tính loss

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

4) Tính đạo hàm

$$L'_w = 2x(\hat{y} - y)$$

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

5) Cập nhật tham số

$$w = w - \eta L'_w$$

$$b = b - \eta L'_b$$

η is learning rate

Hội quy đơn biến

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

2) Tính output $\hat{y}$

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

3) Tính loss

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

4) Tính đạo hàm

$$L'_w = 2x(\hat{y} - y)$$

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

5) Cập nhật tham số

$$w = w - \eta L'_w$$

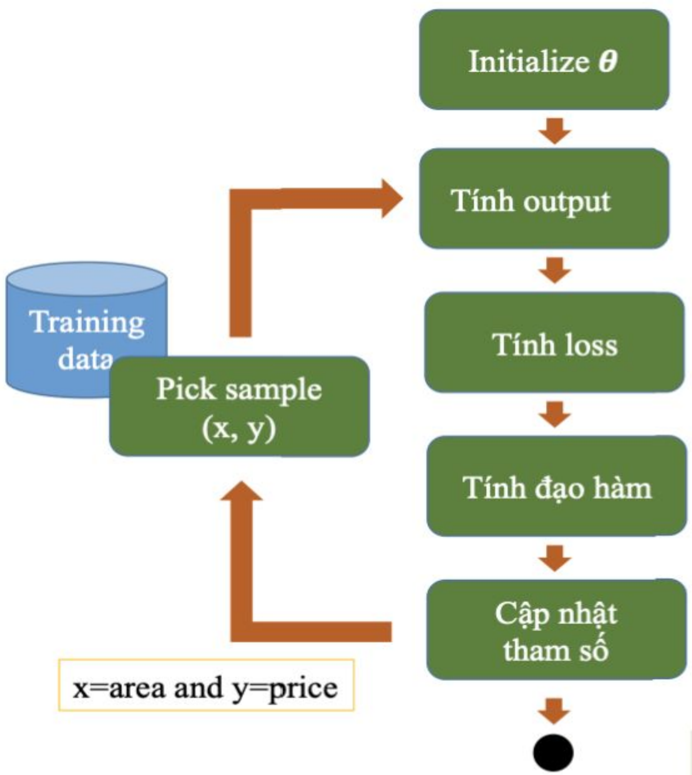
$$b = b - \eta L'_b$$

η is learning rate

```python
# forward
def predict(x, w, b):
    return x*w + b

# compute gradient
def gradient(y_hat, y, x):
    dw = 2*x*(y_hat-y)
    db = 2*(y_hat-y)

    return (dw, db)

# update weights
def update_weight(w, b, lr, dw, db):
    w_new = w - lr*dw
    b_new = b - lr*db

    return (w_new, b_new)
```

```python
# test sample
x = 6.7
y = 9.1

# init weights
b = 0.04
w = -0.34
lr = 0.01

# predict y_hat
y_hat = predict(x, w, b)
print('y_hat: ', y_hat)

# compute loss
loss = (y_hat-y)*(y_hat-y)
print('Loss: ', loss)

# compute gradient
(dw, db) = gradient(y_hat, y, x)
print('dw: ', dw)
print('db: ', db)

# update weights
(w, b) = update_weight(w, b, lr, dw, db)
print('w_new: ', w)
print('b_new: ', b)
```

Hội quy đa biến



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

2) Tính output $\hat{y}$

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

3) Tính loss

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

4) Tính đạo hàm

$$L'_{\theta} = 2\boldsymbol{x}(\hat{y} - y)$$

5) Cập nhật tham số

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

η is learning rate

Hội quy đa biến

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

2) Tính output $\hat{y}$

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

3) Tính loss

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

4) Tính đạo hàm

$$L'_\theta = 2x(\hat{y} - y)$$

5) Cập nhật tham số

$$\theta = \theta - \eta L'_\theta$$

η is learning rate

```python
import numpy as np

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

# compute gradient
def gradient(y_hat, y, x):
    dtheta = 2*x*(y_hat-y)

    return dtheta

# update weights
def update_weight(theta, lr, dtheta):
    dtheta_new = theta - lr*dtheta

    return dtheta_new
```

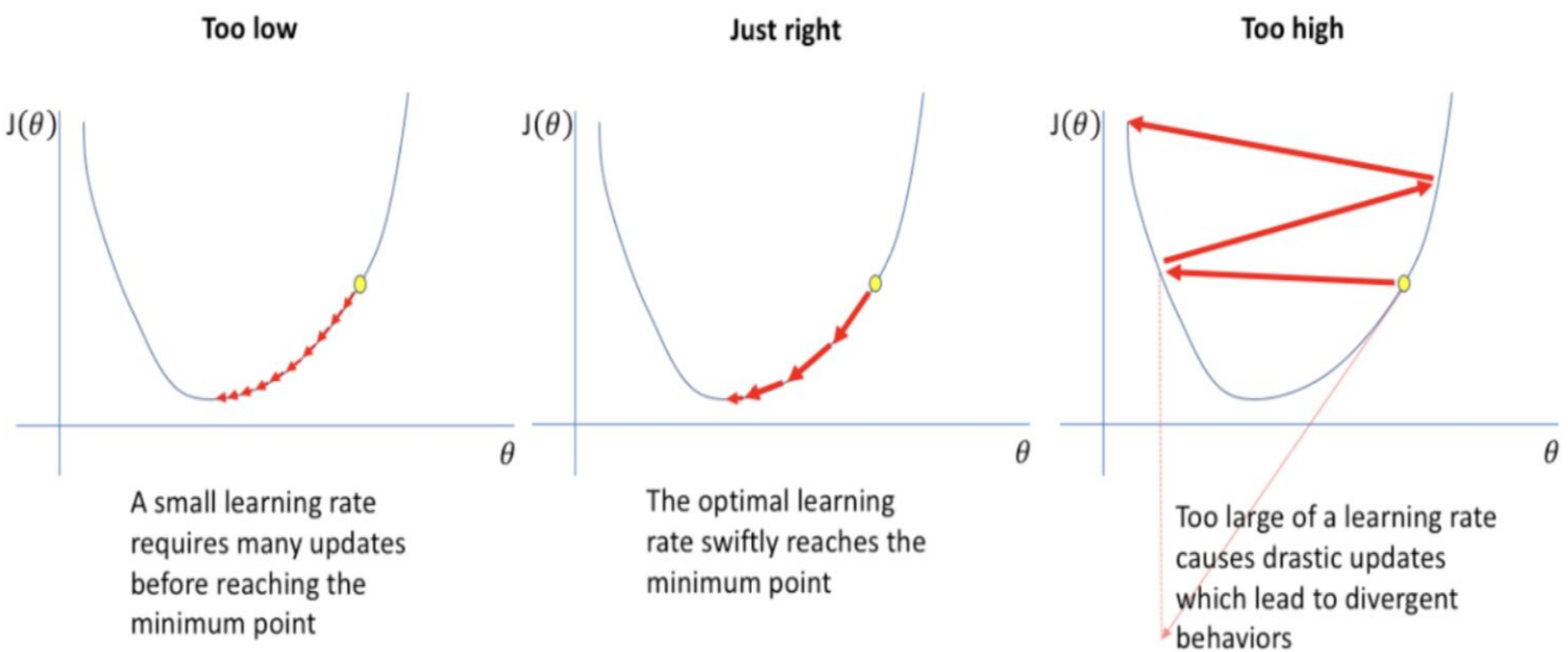
```python
# test sample
x = np.array([6.7, 1])
y = np.array([9.1])

# init weight
lr = 0.01
theta = np.array([-0.34, 0.04]) #[w, b]
print('theta', theta)

# predict y_hat
y_hat = predict(x, theta)
print('y_hat: ', y_hat)

# compute loss
loss = (y_hat-y)*(y_hat-y)
print('Loss: ', loss)

# compute gradient
dtheta = gradient(y_hat, y, x)
print('dtheta: ', dtheta)

# update weights
theta = update_weight(theta, lr, dtheta)
print('theta_new: ', theta)
```

Đánh giá

- Hiệu quả với số chiều lớn (d lớn)
- Phải chọn số bước lặp
- Phải chọn tham số học

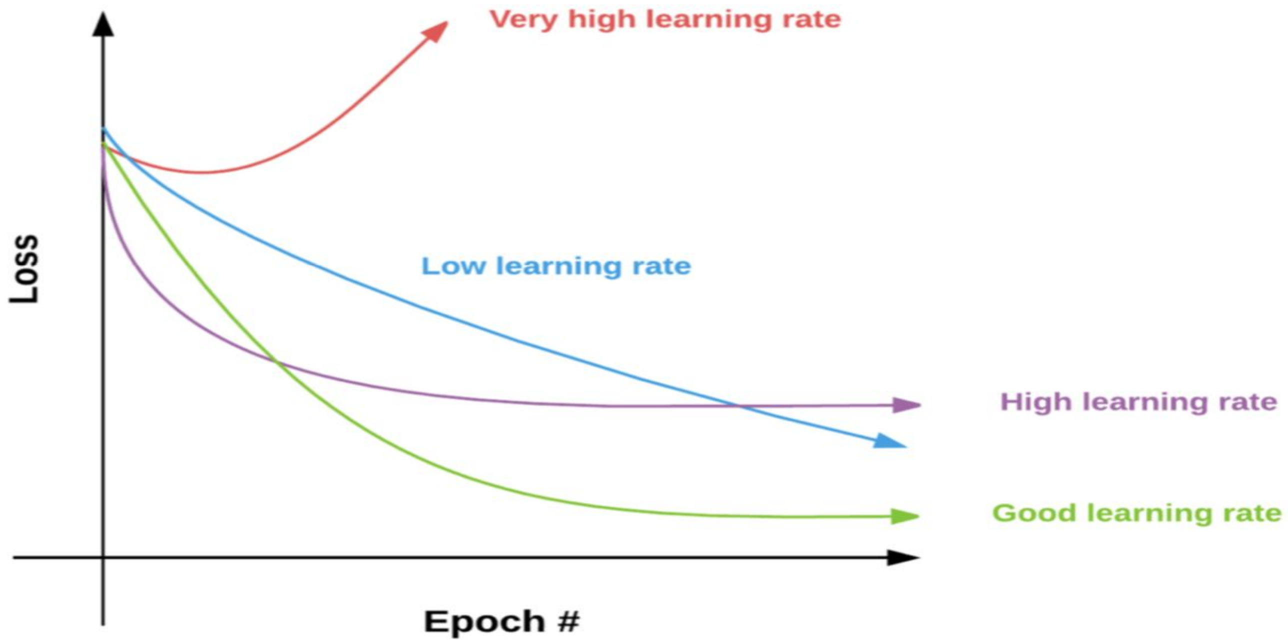
Đánh giá

- Xem xét tham số học



Đánh giá

- Xem xét tham số học



Hồi quy tuyến tính (N- mẫu)

- Sử dụng toàn tập dữ liệu

1) Pick all the N samples from training data

2) Tính output $\hat{y}^{(i)}$

$$\hat{y}^{(i)} = \boldsymbol{\theta}^T \boldsymbol{x}^{(i)} \qquad \text{for } 0 \leq i < N$$

3) Tính loss

$$L^{(i)} = (\hat{y}^{(i)} - y^{(i)})^2 \qquad \text{for } 0 \leq i < N$$

4) Tính đạo hàm

$$L_{\boldsymbol{\theta}}^{\prime(i)} = 2\boldsymbol{x}(\hat{y}^{(i)} - y^{(i)}) \qquad \text{for } 0 \leq i < N$$

5) Cập nhật tham số

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \frac{\sum_i L_{\boldsymbol{\theta}}^{\prime(i)}}{N} \qquad \eta \text{ is learning rate}$$

```python
# load data

import numpy as np
from numpy import genfromtxt
import matplotlib.pyplot as plt

data = genfromtxt('data.csv', delimiter=',')
areas  = data[:,0]
prices = data[:,1]
data_size = areas.size

print(type(areas))
print('areas: ', areas)
print('prices: ', prices)
print('data_size: ', data_size)

plt.scatter(areas, prices)
plt.xlabel('areas')
plt.ylabel('prices')
plt.xlim(3,7)
plt.ylim(4,10)
plt.show()
```

Hồi quy tuyến tính (N- mẫu)

- Sử dụng toàn tập dữ liệu

1) Pick all the N samples from training data

2) Tính output $\hat{y}^{(i)}$

$$\hat{y}^{(i)} = \boldsymbol{\theta}^T \boldsymbol{x}^{(i)} \qquad \text{for } 0 \le i < N$$

3) Tính loss

$$L^{(i)} = (\hat{y}^{(i)} - y^{(i)})^2 \qquad \text{for } 0 \le i < N$$

4) Tính đạo hàm

$$L_{\boldsymbol{\theta}}^{\prime(i)} = 2\boldsymbol{x}(\hat{y}^{(i)} - y^{(i)}) \qquad \text{for } 0 \le i < N$$

5) Cập nhật tham số

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \frac{\sum_i L_{\boldsymbol{\theta}}^{\prime(i)}}{N} \qquad \eta \text{ is learning rate}$$

```python
import numpy as np
from numpy import genfromtxt

data = genfromtxt('data.csv', delimiter=',')
areas  = data[:,0]
prices = data[:,1]
data_size = areas.size

# vector [x, b]
data = np.c_[areas, np.ones((data_size, 1))]

n_epochs = 10
lr = 0.01

theta = np.array([[-0.34],[0.04]])
```

Hồi quy tuyến tính (N- mẫ

- Sử dụng toàn tập dữ liệu

1) Pick all the N samples from training data

2) Tính output $\hat{y}^{(i)}$

$$\hat{y}^{(i)} = \boldsymbol{\theta}^T \boldsymbol{x}^{(i)} \qquad \text{for } 0 \le i < N$$

3) Tính loss

$$L^{(i)} = (\hat{y}^{(i)} - y^{(i)})^2 \qquad \text{for } 0 \le i < N$$

4) Tính đạo hàm

$$L_{\boldsymbol{\theta}}^{\prime(i)} = 2\boldsymbol{x}(\hat{y}^{(i)} - y^{(i)}) \qquad \text{for } 0 \le i < N$$

5) Cập nhật tham số

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \frac{\sum_i L_{\boldsymbol{\theta}}^{\prime(i)}}{N} \qquad \eta \text{ is learning rate}$$

```python
losses = [] # for debug
for epoch in range(n_epochs):
    sum_of_losses = 0
    gradients = np.zeros((2,1))

    for index in range(data_size):
        # get data
        x_i = data[index:index+1]
        y_i = prices[index:index+1]

        # compute output y_hat_i
        y_hat_i = x_i.dot(theta)

        # compute loss
        l_i = (y_hat_i - y_i)*(y_hat_i - y_i)

        # compute gradient
        g_l_i = 2*(y_hat_i - y_i)
        gradient = x_i.T.dot(g_l_i)

        # accumulate gradient
        gradients = gradients + gradient
        sum_of_losses = sum_of_losses + l_i

    # normalize
    sum_of_losses = sum_of_losses/data_size
    gradients     = gradients/data_size

    # for debug
    losses.append(sum_of_losses[0][0])

    # update
    theta = theta - lr*gradients
```

Hồi quy tuyến tính (m- mẫu)

- sử dụng m mẫu; m - mini batch size

1) Pick m samples $(x^{(i)}, y^{(i)})$ from training data

1.1) Tính output $\hat{y}^{(i)}$

$$\hat{y}^{(i)} = \boldsymbol{\theta}^T \boldsymbol{x}^{(i)} \quad \text{for } 0 \le i < m$$

1.2) Tính loss

$$L^{(i)} = (\hat{y}^{(i)} - y^{(i)})^2 \quad \text{for } 0 \le i < m$$

1.3) Tính đạo hàm

$$L_{\boldsymbol{\theta}}'^{(i)} = 2\boldsymbol{x}(\hat{y}^{(i)} - y^{(i)}) \quad \text{for } 0 \le i < m$$

2) Cập nhật tham số

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \frac{\sum_i L_{\boldsymbol{\theta}}'^{(i)}}{m}$$

η is learning rate

```python
# Load data

import numpy as np
from numpy import genfromtxt
import matplotlib.pyplot as plt

data = genfromtxt('data.csv', delimiter=',')
areas  = data[:,0]
prices = data[:,1]
data_size = areas.size

print(type(areas))
print('areas: ', areas)
print('prices: ', prices)
print('data_size: ', data_size)

plt.scatter(areas, prices)
plt.xlabel('areas')
plt.ylabel('prices')
plt.xlim(3,7)
plt.ylim(4,10)
plt.show()
```

Hồi quy tuyến tính (m- mẫu)

- m - mini batch size

1) Pick m samples $(x^{(i)}, y^{(i)})$ from training data

1.1) Tính output $\hat{y}^{(i)}$

$$\hat{y}^{(i)} = \boldsymbol{\theta}^T \boldsymbol{x}^{(i)} \qquad \text{for } 0 \leq i < m$$

1.2) Tính loss

$$L^{(i)} = (\hat{y}^{(i)} - y^{(i)})^2 \qquad \text{for } 0 \leq i < m$$

1.3) Tính đạo hàm

$$L_{\boldsymbol{\theta}}^{\prime(i)} = 2\boldsymbol{x}(\hat{y}^{(i)} - y^{(i)}) \qquad \text{for } 0 \leq i < m$$

2) Cập nhật tham số

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \frac{\sum_i L_{\boldsymbol{\theta}}^{\prime(i)}}{m}$$

η is learning rate

```python
# Load data

import numpy as np
from numpy import genfromtxt
import matplotlib.pyplot as plt

data = genfromtxt('data.csv', delimiter=',')
areas  = data[:,0]
prices = data[:,1]
data_size = areas.size

print(type(areas))
print('areas: ', areas)
print('prices: ', prices)
print('data_size: ', data_size)

plt.scatter(areas, prices)
plt.xlabel('areas')
plt.ylabel('prices')
plt.xlim(3,7)
plt.ylim(4,10)
plt.show()
```

Hồi quy tuyến tính (m- mẫu)

- m - mini batch size

1) Pick m samples $(x^{(i)}, y^{(i)})$ from training data

1.1) Tính output $\hat{y}^{(i)}$

$$\hat{y}^{(i)} = \boldsymbol{\theta}^T \boldsymbol{x}^{(i)} \qquad \text{for } 0 \le i < m$$

1.2) Tính loss

$$L^{(i)} = (\hat{y}^{(i)} - y^{(i)})^2 \qquad \text{for } 0 \le i < m$$

1.3) Tính đạo hàm

$$L'^{(i)}_{\boldsymbol{\theta}} = 2\boldsymbol{x}(\hat{y}^{(i)} - y^{(i)}) \qquad \text{for } 0 \le i < m$$

2) Cập nhật tham số

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \frac{\sum_i L'^{(i)}_{\boldsymbol{\theta}}}{m} \qquad \eta \text{ is learning rate}$$

```python
# vector [x, b]
data = np.c_[areas, np.ones((data_size, 1))]

# init weight
lr = 0.01
theta = np.array([-0.34, 0.04]) #[w, b]

# number of epochs
epoch_max = 10

# mini-batch size
m = 2
```

Hồi quy tuyến tính (m- mẫu)

- m - mini batch size

1) Pick m samples $(x^{(i)}, y^{(i)})$ from training data

1.1) Tính output $\hat{y}^{(i)}$

$$\hat{y}^{(i)} = \boldsymbol{\theta}^T \boldsymbol{x}^{(i)} \qquad \text{for } 0 \le i < m$$

1.2) Tính loss

$$L^{(i)} = (\hat{y}^{(i)} - y^{(i)})^2 \qquad \text{for } 0 \le i < m$$

1.3) Tính đạo hàm

$$L_{\boldsymbol{\theta}}^{\prime(i)} = 2\boldsymbol{x}(\hat{y}^{(i)} - y^{(i)}) \qquad \text{for } 0 \le i < m$$

2) Cập nhật tham số

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \frac{\sum_i L_{\boldsymbol{\theta}}^{\prime(i)}}{m} \qquad \eta \text{ is learning rate}$$

```python
for epoch in range(epoch_max):
    for j in range(0, data_size, m):

        # some variables
        sum_of_losses = 0
        gradients = np.zeros((2,))
        for index in range(j, j+m):
            # get mini-batch
            x_i = data[index]
            y_i = prices[index]

            # predict y_hat_i
            y_hat_i = x_i.dot(theta)

            # compute loss
            l_i = (y_hat_i - y_i)*(y_hat_i - y_i)

            # compute gradient
            gradient_i = x_i*2*(y_hat_i - y_i)

            # accumulate gradients
            gradients = gradients + gradient_i
            sum_of_losses = sum_of_losses + l_i

        # normalize
```

Hồi quy tuyến tính (1- mẫu)

- Sử dụng 1 mẫu

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

2) Tính output $\hat{y}$

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

3) Tính loss

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

4) Tính đạo hàm

$$L'_{\boldsymbol{\theta}} = 2\boldsymbol{x}(\hat{y} - y)$$

5) Cập nhật tham số

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

η is learning rate

```python
import numpy as np

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

# compute gradient
def gradient(y_hat, y, x):
    dtheta = 2*x*(y_hat-y)

    return dtheta

# update weights
def update_weight(theta, lr, dtheta):
    dtheta_new = theta - lr*dtheta

    return dtheta_new
```

Hồi quy tuyến tính (1- mẫu)

- Sử dụng 1 mẫu

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

2) Tính output $\hat{y}$

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

3) Tính loss

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

4) Tính đạo hàm

$$L'_{\boldsymbol{\theta}} = 2\boldsymbol{x}(\hat{y} - y)$$

5) Cập nhật tham số

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

η is learning rate

```python
# vector [x, b]
data = np.c_[areas, np.ones((data_size, 1))]
print(data)

# init weight
n = 0.01
theta = np.array([-0.34, 0.04]) #[w, b]
print('theta', theta)
```

Hồi quy tuyến tính (1

- Sử dụng 1 mẫu

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

2) Tính output $\hat{y}$

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

3) Tính loss

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

4) Tính đạo hàm

$$L'_{\boldsymbol{\theta}} = 2\boldsymbol{x}(\hat{y} - y)$$

5) Cập nhật tham số

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

η is learning rate

```
# number of epochs
epoch_max = 10

for epoch in range(epoch_max):
    for i in range(data_size):
        # get a sample
        x = data[i]
        y = prices[i:i+1]

        # predict y_hat
        y_hat = predict(x, theta)

        # compute loss
        loss = (y_hat-y)*(y_hat-y)

        # compute gradient
        dtheta = gradient(y_hat, y, x)

        # update weights
```

Demo

Diện tích	Giá bán
30	448.524
32.4138	509.248
34.8276,	535.104
...	...

Giá nhà cho 91m^2 là : [1375.19238926]

Demo