

Học máy với Python

Session 1

NGÔ VĂN LINH

Tổng quan

❑ Session 1:

- Linear Regression
- Làm quen với Python
- Triển khai thuật toán Linear Regression
- Biểu diễn dữ liệu
 - Tiền xử lý văn bản: Bag of words, TF-IDF, Word2vec
 - Tiền xử lý ảnh:.....

Tổng quan

- Session 2:
 - K-Means và Support Vector Machines (SVMs)
 - Triển khai thuật toán K-Means
 - Làm quen với thư viện Scikit-Learn: K-Means, SVMs

Tổng quan

- Session 3:

- Neural Networks: Multi-layer Perceptron (MLP)
- Làm quen với tensorflow (1): Multi-layer Perceptron

Tổng quan

❑ Session 4:

- Làm quen với tensorflow (2): Recurrent Neural Networks
- Tổng kết

Session 1

Linear Regression

❑ Nội dung chính:

1. Nhắc lại kiến thức
2. Hỏi đáp

1. Bài toán hồi quy

- Cho tập dữ liệu $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$, trong đó mỗi điểm dữ liệu (x_i, y_i) bao gồm 2 thành phần:
 - $x_i = [x_{i1}, x_{i2}, \dots, x_{iK}]^T$ là một vector K chiều
 - $y_i \in \mathbb{R}$ là một số thực
- Giả thiết rằng tồn tại hàm f tuyến tính sao cho $y_i \cong f(x_i)$:

$$f(x_i) = w_0 + w_1 x_{i1} + \dots + w_K x_{iK} = \mathbf{w} \mathbf{x}_i$$

1. Bài toán hồi quy

- ❑ Lỗi trên tập dữ liệu D:

$$\text{RSS}(f) / N = \frac{1}{N} \sum_{i=1}^N (y_i - f(x_i))^2$$

- ❑ Nghiệm w^* tối thiểu hóa L:

$$w^* = (X^T X)^{-1} X Y \quad \text{với } X = \begin{pmatrix} 1 & x_{11} & x_{12} \dots x_{1K} \\ 1 & x_{21} & x_{22} \dots x_{2K} \\ \dots & & \\ 1 & x_{N1} & x_{N2} \dots x_{NK} \end{pmatrix}, Y = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{pmatrix}$$

1. Bài toán hồi quy

- ❑ Ridge Regression: thêm vào đại lượng phạt $\lambda\|w\|_2^2$ vào RSS(f)

$$L = \sum_{i=1}^N (y_i - f(x_i))^2 + \lambda \sum_{j=0}^K w_j^2$$

- ❑ Minimize L ta có w^* lúc này là^[1]:

$$w^* = (X^T X + \lambda I_{K+1})^{-1} X Y \text{ với } I_{K+1} \text{ là ma trận đơn vị}$$

[1] <https://stats.stackexchange.com/questions/69205/how-to-derive-the-ridge-regression-solution>

1. Bài toán hồi quy

Matrix algebra

Operation	Input	Complexity
Matrix multiplication	Two $n \times n$ matrices	$O(n^{2.373})$
Matrix multiplication	One $n \times m$ matrix & one $m \times p$ matrix	$O(nmp)$
Matrix inversion	One $n \times n$ matrix	$O(n^3)$ $O(n^{2.807})$

- ❑ Khi kích thước dữ liệu quá lớn việc tính nghịch đảo ma trận quá tốn kém. Các giải quyết là tối ưu sử dụng stochastic gradient.

1. Bài toán hồi quy

- ❑ Các lược đồ tối ưu dựa vào gradient:

$$w = w - \text{learning_rate} * \nabla_w L$$

- ❑ $\nabla_w L = E_q[B(w)]$, lấy mẫu ngẫu nhiên data từ phân phối q và gọi $b(w)$ là gradient trên tập mẫu:

$$w = w - \text{learning_rate} * b(w) . \quad b \text{ là lấy mẫu độc lập từ } B.$$

Việc tối ưu theo stochastic gradient đảm bảo tính hội tụ và về mặt thực nghiệm cho kết quả tốt hơn trên các hàm non-convex so với gradient thông thường.

- ❑ Hiểu đơn giản là ta chia dữ liệu thành các minibatch rồi tối ưu parameter theo gradient của minibatch đó. Lặp lại trên dữ liệu nhiều epoch.

1. Bài toán hồi quy

- ❑ Có thể sử dụng các phương pháp tối ưu dựa trên gradient để minimize hàm lỗi. Áp dụng khi kích thước dữ liệu quá lớn việc tính nghịch đảo ma trận quá tốn kém
- ❑ Lược đồ tối ưu:

Linear Regression: $w = w - \text{learning_rate} * x^T(xw - y)$

Ridge Regression: $w = w - \text{learning_rate} * [x^T(xw - y) + \lambda w]$

2. Hỏi đáp



Làm quen với Python

- ❑ Tên gọi: /'paɪθən/ /'paɪ.θə:n/
- ❑ Nội dung chính:
 - 1. Giống và khác
 - 2. Kiểu dữ liệu
 - 3. Lệnh if
 - 4. Phép lặp
 - 5. Đọc, ghi file
 - 6. Xử lý ngoại lệ
 - 7. Lệnh assert
 - 8. Hàm
 - 9. Lập trình hướng đối tượng
 - 10. Numpy
- ❑ Python IDE/Editor: **Pycharm**, vim, gedit, ...

1. Giống và khác

- ❑ Không có khai báo biến, biến được tạo ra ngay khi khởi tạo:

`x = 1.0`

`x = 'Python'`

- ❑ Không có ký tự kết thúc lệnh (`;`)
- ❑ Các khối lệnh phân biệt nhau bởi khoảng cách với lề
- ❑ Các phép toán so sánh: `> < == !=`
- ❑ Các phép toán logic: `and or not`
- ❑ Hàm mũ: `34 -> 3 ** 4`
- ❑ Hàm print: `print "Python"`

2. Kiểu dữ liệu

- a) Kiểu number
- b) Kiểu string
- c) Kiểu list
- d) Kiểu tuple
- e) Kiểu set
- f) Kiểu dictionary

2. Kiểu dữ liệu

a) Kiểu number:

➤ int, float

➤ Tự động chuyển đổi kiểu:

a = 1

b = 2.0

c = a + b

>> output: c = 3.0

2. Kiểu dữ liệu

b) Kiểu string: Ép kiểu

➤ string -> number

```
float_number = float('2018')  
int_number = int('2018')
```

➤ number -> string

```
year = str(2018)
```

2. Kiểu dữ liệu

b) Kiểu string: Thao tác với string

➤ Cộng

```
name = 'Summer ' + str(2018)
>> name: 'Summer 2018'
```

➤ Nhân

```
name = '1' * 5
>> name: '*****'
```

➤ Format

```
name = 'Data {} {}'.format('Science', 2018)
>> name: 'Data Science 2018'
```

2. Kiểu dữ liệu

b) Kiểu string: Thao tác với string

➤ Split

```
list_1 = 'Data Science 2018'.split()  
list_2 = 'Data--Science--2018'.split('--')  
>> list_1 = list_2: ['Data', 'Science', '2018']
```

➤ Join

```
list_1 = ['Data', 'Science', '2018']  
name = '--'.join(list_1)  
>> name: 'Data--Science--2018'
```

➤ Replace

```
'Data Science'.replace('e', '-')  
>> 'Data Sci-nc-'
```

2. Kiểu dữ liệu

b) Kiểu string: Thao tác với string

- **Isspace** a = ' '.isspace()
 b = '\t\t'.isspace()
 c = '-'.isspace()
 >> a: True, b: True, c: False
- **Isalpha** a = 'abcde'.isalpha()
 b = 'abcde123'.isalpha()
 >> a: True, b: False
- **Isdigit** a = '1234'.isdigit()
 b = 'abc123'.isdigit()
 >> a: True, b: False

2. Kiểu dữ liệu

c) Kiểu list:

➤ Khởi tạo

```
list_a = [1,2,3,4,5]
list_b = range(5) # [0,1,2,3,4]
list_c = range(1,10,2) # [1,3,5,7,9]
list_d = range(5, 1, -1) # [5,4,3,2]
list_e = [x for x in range(10) if x % 2 == 0]
            # [0, 2, 4, 6, 8]
list_f = []
list_f.append('Data')
list_f.append('Science')
list_f.append(2018)
# ['Data', 'Science', 2018]
```

2. Kiểu dữ liệu

c) Kiểu list: Thao tác với list

➤ Trích xuất ra các phần tử

```
src = range(10) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
a = src[:7] # [0, 1, 2, 3, 4, 5, 6]
b = src[3:] # [3, 4, 5, 6, 7, 8, 9]
c = src[3:7] # [3, 4, 5, 6]
d = src[:-1] # [0, 1, 2, 3, 4, 5, 6, 7, 8]
e = src[:-2] # [0, 1, 2, 3, 4, 5, 6, 7]
```

➤ Lấy số phần tử

```
src = range(10) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
length = len(src) # length = 10
```

2. Kiểu dữ liệu

c) Kiểu list: Thao tác với list

➤ Enumerate

```
src = ['Data', 'Science', 2018]
for index, element in enumerate(src):
    print index, element
```

```
>> Output:
0 Data
1 Science
2 2018
```

2. Kiểu dữ liệu

c) Kiểu list: Thao tác với list

- Kiểm tra một phần tử có/không thuộc list hay không

```
src = ['Data', 'Science', 2018]
exp_1 = 'Data' in src # True
exp_2 = 2018 not in src # False
```

- Sắp xếp một list

```
src = [(3, 'A'), (2, 'C'), (1, 'D'), (4, 'B')]
src.sort()
print src # [(1, 'D'), (2, 'C'), (3, 'A'), (4, 'B')]
```

```
src.sort(key=lambda (amount, name): name)
print src # [(3, 'A'), (4, 'B'), (2, 'C'), (1, 'D')]
```

2. Kiểu dữ liệu

c) Kiểu list: Thao tác với list

➤ Xóa một phần tử khỏi list

```
src = [1,2,3,4,5]
del src[3]
print src # [1, 2, 3, 5]
```

➤ Ghép nhiều list lại thành một

```
a = [1,2,3,4,5]
b = ['A', 'B', 'C', 'D']
c = ['x', 'y', 'z', 't']
d = zip(a, b, c)
print d
# [(1, 'A', 'x'), (2, 'B', 'y'), (3, 'C', 'z'), (4, 'D', 't')]
```

2. Kiểu dữ liệu

d) Kiểu tuple:

➤ Fixed list

```
src = (3,2,1)  
src[2] = 5 # can't do that
```

➤ Thường dùng để đóng gói dữ liệu

```
src = [(20, 7, 'Session 1'), (21, 7, 'Session 2')]
```

2. Kiểu dữ liệu

e) Kiểu set:

- Tập hợp các phần tử phân biệt
- Thu gọn một list có phần tử trùng lặp

```
src = [1,2,2,3,3,3,4,4,4,4]  
reduced_list = list(set(src)) # [1, 2, 3, 4]
```

2. Kiểu dữ liệu

f) Kiểu dictionary:

➤ Khởi tạo

```
dict = {}
dict['A'] = 1
dict['B'] = 2
print dict # {'A': 1, 'B': 2}
```

➤ Kiểm tra xem 1 giá trị có là key của dictionary hay không

```
print 'A' in dict # True
print 'C' in dict # False
```

➤ Lấy danh sách keys và values

```
print dict.keys() # ['A', 'B']
print dict.values() # [1, 2]
```

3. Lệnh if

❑ elif là else if

❑ Ví dụ:

```
a = False
b = 1
if a and b != 1 :
    print 'One'
elif b == 1:
    print 'Two'
else:
    print 'Three'
>> Output:
'Two'
```

4. Phép lặp

❑ Lệnh for

```
src = range(10)
for i in src:
    print i
```

❑ Lệnh while

```
i = 0
while i < len(src):
    print src[i]
    i += 1
```

❑ Thoát khỏi vòng lặp: break

❑ Bỏ qua bước lặp: continue

5. Đọc ghi file

❑ Đọc nội dung từ file

```
with open('file.txt') as f:  
    content = f.read()
```

❑ Ghi nội dung ra file

```
with open('file.txt', 'w') as f:  
    content = f.write('Data Science Lab')
```

6. Xử lý ngoại lệ

❑ Lấy tên ngoại lệ

```
division = 10/0
>> Output:

Traceback (most recent call last):
  File "jdoodle.py", line 2, in <module>
    division = 10/0
ZeroDivisionError: integer division or modulo by zero
Command exited with non-zero status 1
```

❑ Xử lý ngoại lệ:

```
try:
    division = 10/0
except ZeroDivisionError:
    print "Division by zero occurs"
```

7. Lệnh assert

- ❑ **assert**: đặt điều kiện trước khi đoạn code được thực hiện

```
a = 5  
b = 0  
assert b != 0  
c = a / b
```

- ❑ Đặt thông báo cho lệnh assert

```
a = 5  
b = 0  
assert b != 0, "Some Notification"  
c = a/b
```

8. Hàm

❑ Định nghĩa hàm

```
def do_sum(a,b):  
    return a + b  
print do_sum(3,5) # 8
```

❑ Tham số có giá trị mặc định:

```
def do_sum(a,b=5):  
    return a + b  
print do_sum(3) # 8
```

❑ Hàm trong hàm

```
def do_sum(a,b):  
    def decrease(value):  
        return value - 1  
  
    return decrease(a+b)  
  
print do_sum(3,5) # 7
```

9. Lập trình hướng đối tượng

□ Cấu trúc của một lớp:

```
class Person:  
    def __init__(self, name, age, phone_number):  
        self._name = name  
        self._age = age  
        self._phone_number = phone_number  
        self._selected_items = []  
  
    def select(self, item):  
        self._selected_items.append(item)  
  
    def get_info(self):  
        return 'Name: {}\nAge: {}\nPhone number: {}'.format(  
            self._name, self._age, self._phone_number)
```

Hàm khởi tạo

Hàm chức năng

9. Lập trình hướng đối tượng

□ Khởi tạo và sử dụng đối tượng:

```
nam = Person('Nam', 18, 123456789)
nam.select('Book')
nam.select('Pen')
print nam._selected_items
print nam.get_info()
>> Output:
['Book', 'Pen']
Name: Nam
Age: 18
Phone number: 123456789
```

10. Numpy

- ❑ Hỗ trợ các phép toán trên mảng, ma trận, ...
- ❑ Kiểu dữ liệu chuẩn: **numpy array**
- ❑ Import thư viện numpy: `import numpy as np`
- ❑ Khởi tạo đối tượng numpy array:

```
np_array_1 = np.array([1,2,3,4,5])
np_array_2 = np.array(
    [[1,2,3],
     [4,5,6],
     [7,8,9]])
```

10. Numpy

- ❑ Cộng: `np_array_1 + np_array_2`
- ❑ Trừ: `np_array_1 - np_array_2`
- ❑ Nhân: `np_array_1.dot(np_array_2)`
- ❑ Chia: `np_array_1 / np_array_2`
- ❑ Chuyển vị: `np_array.transpose()`
- ❑ Nghịch đảo: `np.linalg.inv(np_array)`

=> Xem thêm: <https://docs.scipy.org/doc/numpy/reference/routines.array-manipulation.html>

Triển khai thuật toán Linear Regression

❑ Nội dung chính:

1. Dữ liệu sử dụng
2. Cross-validation
3. Triển khai

1. Dữ liệu sử dụng

- ❑ Tập dữ liệu Death Rate:

<https://people.sc.fsu.edu/~jburkardt/datasets/regression/x28.txt>

- ❑ Có tất cả 60 điểm dữ liệu, mỗi điểm dữ liệu có 15 thuộc tính và 1 giá trị death rate tương ứng.

index	A1	A2		A13	A14	A15	Death Rate
1	36	27	...	15	59	59	921.870
2	35	23	...	10	39	57	997.875
3	44	29	...	6	33	54	962.354
4	47	45	...	8	24	56	982.291
...	

2. Cross-validation

- ❑ Một tập dữ liệu D thường có 2 phần: D_{train} và D_{test}
- ❑ D_{train} dùng để huấn luyện mô hình
- ❑ D_{test} để đánh giá hiệu quả của mô hình
- ❑ Cross-validation (k-fold cross-validation) dùng để lựa chọn tham số cho mô hình (với ridge regression, đó là giá trị LAMBDA λ).

2. Cross-validation

- Áp dụng 5-fold cross-validation vào việc lựa chọn LAMBDA
- 5-fold cross-validation được tiến hành như sau:
 - Chia D_{train} thành 5 phần (xấp xỉ) bằng nhau: D_1, D_2, D_3, D_4, D_5
 - Với mỗi D_i ($i = \overline{1, 5}$), ta thực hiện:
 - * Huấn luyện mô hình trên $D_{train} \setminus D_i$
 - * Tính lỗi trên D_i
 - Tính lỗi trung bình qua 5 lần
 - Lựa chọn LAMBDA đem lại lỗi trung bình nhỏ nhất.
 - Huấn luyện mô hình trên toàn bộ D_{train} với LAMBDA tìm được và đánh giá hiệu quả mô hình trên D_{test}

2. Cross-validation

- ❑ Xem thêm các kỹ thuật khác cho lựa chọn tham số tại bài giảng số 8, môn Học Máy của thầy Thân Quang Khoát.

<http://is.hust.edu.vn/~khoattq/lectures/ML-1-2018/L8-Model-assessment.pdf>

3. Triển khai

- ❑ Triển khai thuật toán Ridge Regression (trường hợp tổng quát của Linear Regression)
 - > Đọc dữ liệu
 - > Chuẩn hóa dữ liệu
 - > Xây dựng mô hình
- ❑ Lựa chọn **LAMBDA** theo phương pháp cross-validation

3. Triển khai

Đọc dữ liệu:

- > Đọc file
- > Chia nội dung thành từng dòng
- > Chia mỗi dòng thành các features
- > X: features từ A1 → A15
- > Y: feature cuối cùng, B

I	A1	A2	A3	...	A13	A14	A15	B
1	36	27	71	...	15	59	59	921.870
2	35	23	72	...	10	39	57	997.875
3	44	29	74	...	6	33	54	962.354
4	47	45	79	...	8	24	56	982.291
5	43	35	77	...	38	206	55	1071.289
6	53	45	80	...	32	72	54	1030.380
7	43	30	74	...	32	62	56	934.700
8	45	30	73	...	4	4	56	899.529

3. Triển khai

□ Chuẩn hóa dữ liệu:

- > Các features có miền giá trị lệch nhau
- > Chuẩn hóa để đưa về 1 miền chung
- > Có nhiều phương pháp^[1], ta chọn

“Feature Scaling”:

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

với $X' \in [0,1]^{N \times 15}$, N là số điểm dữ liệu

I	A1	A2	A3	...	A13	A14	A15	B
1	36	27	71	...	15	59	59	921.870
2	35	23	72	...	10	39	57	997.875
3	44	29	74	...	6	33	54	962.354
4	47	45	79	...	8	24	56	982.291
5	43	35	77	...	38	206	55	1071.289
6	53	45	80	...	32	72	54	1030.380
7	43	30	74	...	32	62	56	934.700
8	45	30	73	...	4	4	56	899.529

[1] [https://en.wikipedia.org/wiki/Normalization_\(statistics\)](https://en.wikipedia.org/wiki/Normalization_(statistics))

3. Triển khai

☐ Chuẩn hóa dữ liệu:

> Nhắc lại công thức

=> Ta cần thêm feature $x_{i0} = 1$

vào mỗi điểm dữ liệu

$$w^* = (X^T X + \lambda I_{K+1})^{-1} X Y$$

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} \dots x_{1K} \\ 1 & x_{21} & x_{22} \dots x_{2K} \\ \dots & \dots & \dots \\ 1 & x_{N1} & x_{N2} \dots x_{NK} \end{pmatrix}$$

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{pmatrix}$$

3. Triển khai

☐ Chuẩn hóa dữ liệu:

```
25 def normalize_and_add_ones(X):
26     X = np.array(X)
27     X_max = np.array([[np.amax(X[:, column_id])
28                         for column_id in range(X.shape[1])]
29                         for _ in range(X.shape[0])])
30     X_min = np.array([[np.amin(X[:, column_id])
31                         for column_id in range(X.shape[1])]
32                         for _ in range(X.shape[0])])
33
34     X_normalized = (X - X_min) / (X_max - X_min)
35
36     ones = np.array([[1] for _ in range(X_normalized.shape[0])])
37     return np.column_stack((ones, X_normalized))
```

I	A1	A2	A3	...	A13	A14	A15	B
1	36	27	71	...	15	59	59	921.870
2	35	23	72	...	10	39	57	997.875
3	44	29	74	...	6	33	54	962.354
4	47	45	79	...	8	24	56	982.291
5	43	35	77	...	38	206	55	1071.289
6	53	45	80	...	32	72	54	1030.380
7	43	30	74	...	32	62	56	934.700
8	45	30	73	...	4	4	56	899.529

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

3. Triển khai

□ Triển khai mô hình:

> Xây dựng lớp **RidgeRegression**

```
40 class RidgeRegression:  
41     def __init__(self):  
42         return  
43  
44     def fit(self, X_train, Y_train, LAMBDA):...  
53  
54     def predict(self, W, X_new):...  
58  
59     def compute_RSS(self, Y_new, Y_predicted):...  
63  
64     def get_the_best_LAMBDA(self, X_train, Y_train):...
```

$$w^* = (X^T X + \lambda I_{K+1})^{-1} X Y$$

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} \dots x_{1K} \\ 1 & x_{21} & x_{22} \dots x_{2K} \\ \dots & \dots & \dots \\ 1 & x_{N1} & x_{N2} \dots x_{NK} \end{pmatrix}$$

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{pmatrix}$$

3. Triển khai

□ Triển khai mô hình:

> Hàm **fit**

```
44 def fit(self, X_train, Y_train, LAMBDA):
45     assert len(X_train.shape) == 2 and \
46         X_train.shape[0] == Y_train.shape[0]
47
48     W = np.linalg.inv(
49         X_train.transpose().dot(X_train) +
50         LAMBDA * np.identity(X_train.shape[1]))
51     ).dot(X_train.transpose()).dot(Y_train)
52     return W
```

$$w^* = (X^T X + \lambda I_{K+1})^{-1} X Y$$

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} \dots x_{1K} \\ 1 & x_{21} & x_{22} \dots x_{2K} \\ \dots & \dots & \dots \\ 1 & x_{N1} & x_{N2} \dots x_{NK} \end{pmatrix}$$

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{pmatrix}, W = \begin{pmatrix} w_0 \\ w_1 \\ \dots \\ y_K \end{pmatrix}$$

3. Triển khai

□ Triển khai mô hình: > Hàm **fit_gradient**

```
13     def fit_gradient_descent(self,X_train,Y_train,LAMBDA,learning_rate,max_num_epoch=100,batch_size=128):
14         W = np.random.randn(X_train.shape[1])
15         last_loss=10e+8
16         for ep in range(max_num_epoch):
17             arr = np.array(range(X_train.shape[0]))
18             np.random.shuffle(arr)
19             X_train=X_train[arr]
20             Y_train=Y_train[arr]
21             total_minibatch = int(np.ceil(X_train.shape[0]/batch_size))
22             for i in range(total_minibatch):
23                 index = i *batch_size
24                 X_train_sub = X_train[index:index+batch_size]
25                 Y_train_sub = Y_train[index:index+batch_size]
26                 grad = X_train_sub.T.dot(X_train_sub.dot(W) -Y_train_sub) + LAMBDA * W
27                 W = W - learning_rate*grad
28                 new_loss = self.compute_RSS(self.predict(W,X_train),Y_train)
29                 if(np.abs(new_loss - last_loss) <= 1e-5):
30                     break
31                 last_loss=new_loss
32         return W
```

3. Triển khai

□ Triển khai mô hình:

> Hàm **predict**

```
54 def predict(self, W, X_new):  
55     X_new = np.array(X_new)  
56     Y_new = X_new.dot(W)  
57     return Y_new
```

$$Y_{\text{new}} = X_{\text{new}}W$$

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} \dots x_{1K} \\ 1 & x_{21} & x_{22} \dots x_{2K} \\ \dots & \dots & \dots \\ 1 & x_{N1} & x_{N2} \dots x_{NK} \end{pmatrix}$$

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{pmatrix}, W = \begin{pmatrix} w_0 \\ w_1 \\ \dots \\ y_K \end{pmatrix}$$

3. Triển khai

- Triển khai mô hình:

- > Hàm **compute_RSS**

```
59 def compute_RSS(self, Y_new, Y_predicted):  
60     loss = 1. / Y_new.shape[0] * \  
61         np.sum((Y_new - Y_predicted) ** 2)  
62     return loss
```

$$\text{RSS}(f) / N = \frac{1}{N} \sum_{i=1}^N (y_i - f(x_i))^2$$

3. Triển khai

Triển khai mô hình:

> Xác định giá trị LAMBDA tốt nhất:

- * B1: Xác định miền giá trị tìm kiếm
- * B2: Thực hiện cross-validation với từng giá trị LAMBDA có thể
- * B3: Xác định giá trị LAMBDA tốt nhất trong miền
- * B4: Quay trở lại bước 1

3. Triển khai

☐ Triển khai mô hình: Hàm `get_the_best_LAMBDA`

```
64 def get_the_best_LAMBDA(self, X_train, Y_train):
65     def cross_validation(num_folds, LAMBDA):...
66
67
68
69     def range_scan(best_LAMBDA, minimum_RSS, LAMBDA_values):...
70
71
72
73     best_LAMBDA, minimum_RSS = range_scan(best_LAMBDA=0, minimum_RSS=10000 ** 2,
74                                         LAMBDA_values=range(50)) # [0, 1, 2, ..., 49]
75
76
77     LAMBDA_values = [k * 1. / 1000 for k in range(
78         max(0, (best_LAMBDA - 1) * 1000), (best_LAMBDA + 1) * 1000, 1)
79         ] # step size = 0.001
80
81
82     best_LAMBDA, minimum_RSS = range_scan(best_LAMBDA=best_LAMBDA, minimum_RSS=minimum_RSS,
83                                         LAMBDA_values=LAMBDA_values)
84
85     return best_LAMBDA
```

3. Triển khai

❑ Triển khai mô hình: Hàm `range_scan`

```
79 def range_scan(best_LAMBDA, minimum_RSS, LAMBDA_values):
80     for current_LAMBDA in LAMBDA_values:
81         aver_RSS = cross_validation(num_folds=5, LAMBDA=current_LAMBDA)
82         if aver_RSS < minimum_RSS:
83             best_LAMBDA = current_LAMBDA
84             minimum_RSS = aver_RSS
85     return best_LAMBDA, minimum_RSS
```

3. Triển khai

❑ Triển khai mô hình: Hàm **cross_validation**

```
65 def cross_validation(num_folds, LAMBDA):
66     row_ids = np.array(range(X_train.shape[0]))
67     # np.split() requires equal divisions
68     valid_ids = np.split(row_ids[:len(row_ids) - len(row_ids) % num_folds], num_folds)
69     valid_ids[-1] = np.append(valid_ids[-1], row_ids[len(row_ids) - len(row_ids) % num_folds:])
70     train_ids = [[k for k in row_ids if k not in valid_ids[i]] for i in range(num_folds)]
71     aver_RSS = 0
72     for i in range(num_folds):
73         valid_part = {'X': X_train[valid_ids[i]], 'Y': Y_train[valid_ids[i]]}
74         train_part = {'X': X_train[train_ids[i]], 'Y': Y_train[train_ids[i]]}
75         W = self.fit(train_part['X'], train_part['Y'], LAMBDA)
76         Y_predicted = self.predict(W, valid_part['X'])
77         aver_RSS += self.compute_RSS(valid_part['Y'], Y_predicted)
78     return aver_RSS / num_folds
```

3. Triển khai

☐ Triển khai mô hình: Chạy thử

```
100 > if __name__ == '__main__':
101     X, Y = get_data(path='../../datasets/death-rates-data.txt')
102     # normalization
103     X = normalize_and_add_ones(X)
104     X_train, Y_train = X[:50], Y[:50]
105     X_test, Y_test = X[50:], Y[50:]
106
107     ridge_regression = RidgeRegression()
108     best_LAMBDA = ridge_regression.get_the_best_LAMBDA(X_train, Y_train)
109     print 'Best LAMBDA:', best_LAMBDA
110     W_learned = ridge_regression.fit(
111         X_train=X_train, Y_train=Y_train, LAMBDA=best_LAMBDA
112     )
113     Y_predicted = ridge_regression.predict(W=W_learned, X_new=X_test)
114
115     print ridge_regression.compute_RSS(Y_new=Y_test, Y_predicted=Y_predicted)
```

4. Tiền xử lý dữ liệu

1. Tiền xử lý và biểu diễn văn bản (bag of words, TF-IDF)
2. Biểu diễn vector từ (Word2vec)

Biểu diễn Bag of words và TF-IDF cho doc

- ❑ TF-IDF = term frequency–inverse document frequency
- ❑ Được sử dụng cho dữ liệu dạng văn bản (text)
- ❑ Biểu diễn TF-IDF đối với 1 văn bản d trong một tập văn bản (corpus) D:

$$r_d = [\text{tf-idf}(w_1, d, D), \text{tf-idf}(w_2, d, D), \dots, \text{tf-idf}(w_{|V|}, d, D)]$$

với $r_d \in \mathbb{R}^{|V|}$ là 1 vector $|V|$ chiều

$V = \{w_i\}$ là từ điển (tập hợp các từ xuất hiện trong D) đối với D

Biểu diễn bag of words, TF-IDF

- Trong đó, mỗi giá trị $\text{tf-idf}(w_i, d, D)$ được tính như sau:

$$\text{tf-idf}(w_i, d, D) = \text{tf}(w_i, d) \times \text{idf}(w_i, D)$$

$$\text{với } \text{tf}(w_i, d) = \frac{f(w_i, d)}{\max\{f(w_j, d) : w_j \in V\}}$$

$$\text{idf}(w_i, D) = \log_{10} \frac{|D|}{|\{d' \in D : w_i \in d'\}|}$$

- Trong đó, $f(w_i, d)$ là số lần xuất hiện của từ w_i trong văn bản d .

Biểu diễn TF-IDF

□ Xác định từ điển V:

➤ Với mỗi văn bản d trong D:

* B1: Tách d thành các từ theo punctuations^[1] ta thu được W_d :

‘Data-Science Lab;2018’ -> [‘Data’, ‘Science’, ‘Lab’, ‘2018’]

* B2: Loại bỏ từ dừng (stop words^[2]) khỏi W_d :

$$W_d = W_d \setminus \{\text{stop_words}\}$$

a, an, the, have, for,

* B3: Đưa các từ về dạng gốc (stemming^[3]) :

$$W_d = \{\text{stem}(w) : w \in W_d\}$$

trong đó $\text{stem}(w)$ là dạng gốc của w

[1] <https://en.wikipedia.org/wiki/Punctuation>

[2] <https://www.ranks.nl/stopwords>

Biểu diễn TF-IDF

- Một cách xác định từ điển V:
 - Với mỗi văn bản d trong D: thu được W_d
 - Cuối cùng, ta có:

$$V = \bigcup_{d \in D} W_d$$

Biểu diễn TF-IDF

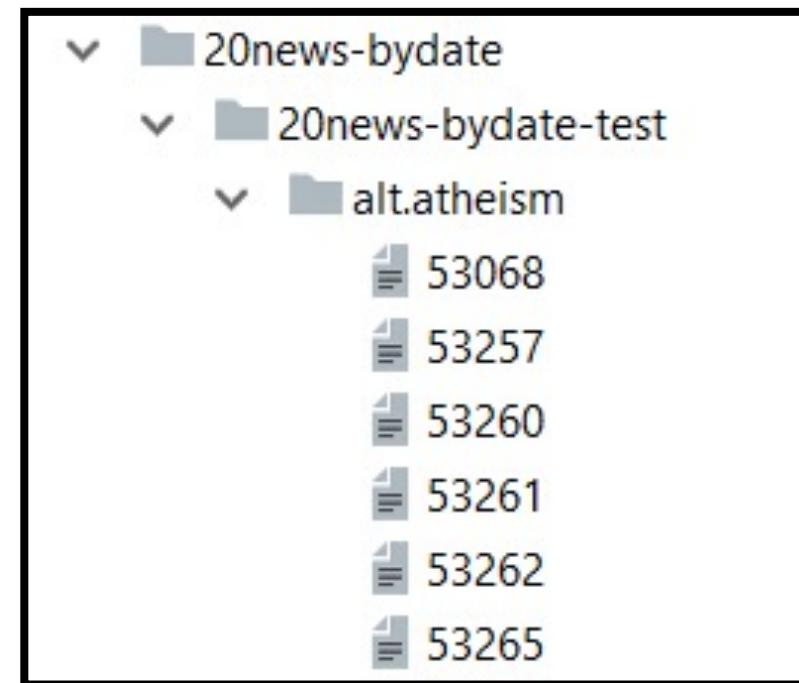
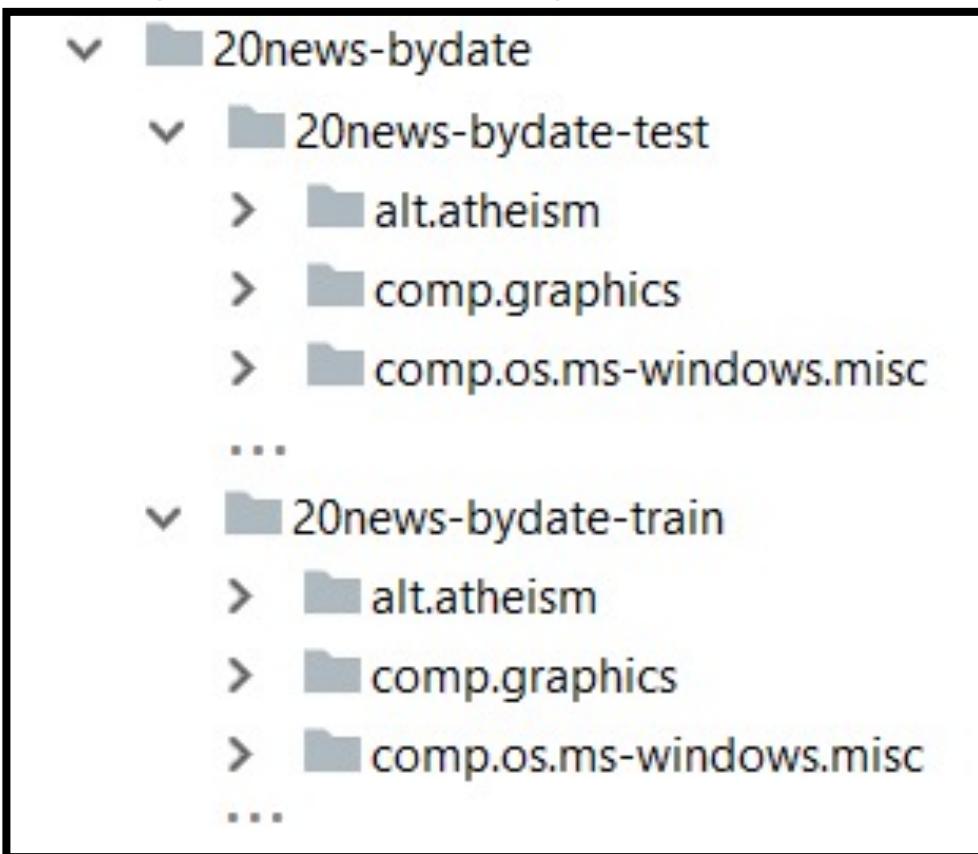
- ❑ Tập dữ liệu thực hành: 20newsgroups

<http://qwone.com/~jason/20Newsgroups/>

- ❑ Tải [20news-bydate.tar.gz](#)
- ❑ Bao gồm xấp xỉ 20,000 bài báo, thuộc 20 nhóm tin tức khác nhau.
- ❑ Tập dữ liệu này sẽ được sử dụng để thực hành với K-Means, SVMs và Neural Networks
- ❑ Tiền xử lý: tính biểu diễn tf-idf cho tất cả các văn bản có trong tập dữ liệu.

Biểu diễn TF-IDF

❑ Tiền xử lý: cấu trúc cây thư mục

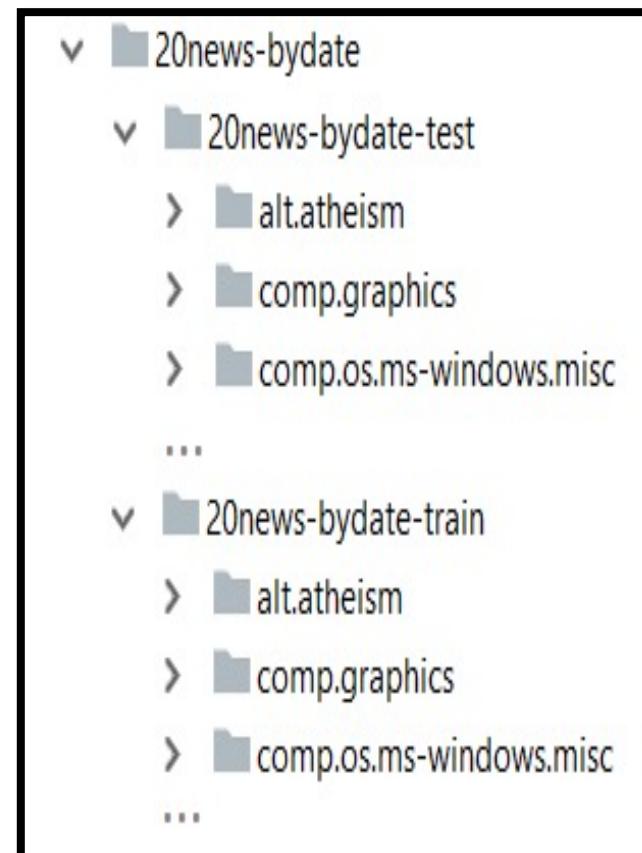


Biểu diễn TF-IDF

- ❑ Tiền xử lý: đọc dữ liệu và tập hợp dữ liệu

> Lấy danh sách các thư mục và newsgroups

```
87 def gather_20newsgroups_data():
88     path = '../datasets/20news-bydate/'
89     dirs = [path + dir_name + '/'
90             for dir_name in.listdir(path)
91             if not isfile(path + dir_name)]
92     train_dir, test_dir = (dirs[0], dirs[1]) if 'train' in dirs[0] \
93     else (dirs[1], dirs[0])
94     list_newsgroups = [newsgroup
95                         for newsgroup in.listdir(train_dir)]
96     list_newsgroups.sort()
```



Biểu diễn TF-IDF

- ❑ Tiền xử lý: đọc dữ liệu và tập hợp dữ liệu

> Thu thập dữ liệu

```
98  with open('../datasets/20news-bydate/stop_words.txt') as f:  
99      stop_words = f.read().splitlines()  
100     from nltk.stem.porter import PorterStemmer  
101     stemmer = PorterStemmer()  
102  
103     def collect_data_from(parent_dir, newsgroup_list):...  
128  
129     train_data = collect_data_from(  
130         parent_dir=train_dir,  
131         newsgroup_list=list_newsgroups  
132     )  
133     test_data = collect_data_from(  
134         parent_dir=test_dir,  
135         newsgroup_list=list_newsgroups  
136     )
```

Biểu diễn TF-IDF

- ❑ Tiền xử lý: đọc dữ liệu và tập hợp dữ liệu

> Thu thập dữ liệu: hàm **collect_data_from**

```
103 def collect_data_from(parent_dir, newsgroup_list):
104     data = []
105     for group_id, newsgroup in enumerate(newsgroup_list):
106         label = group_id
107         dir_path = parent_dir + '/' + newsgroup + '/'
108         files = [(filename, dir_path + filename)
109                   for filename in listdir(dir_path)
110                   if isfile(dir_path + filename)]
111     files.sort()
```

Biểu diễn TF-IDF

- ❑ Tiền xử lý: đọc dữ liệu và tập hợp dữ liệu

> Thu thập dữ liệu: hàm `collect_data_from`

```
112     for filename, filepath in files:  
113         with open(filepath) as f:  
114             text = f.read().lower()  
115             # remove stop words then stem remaining words  
116             words = [stemmer.stem(word)  
117                     for word in re.split('\W+', text)  
118                     if word not in stop_words]  
119             # combine remaining words  
120             content = ' '.join(words)  
121             assert len(content.splitlines()) == 1  
122             data.append(str(label) + '<fff>' +  
123                             filename + '<fff>' + content)  
124     return data
```

Biểu diễn TF-IDF

- ❑ Tiền xử lý: đọc dữ liệu và tập hợp dữ liệu

> Ghi ra file:

```
135     full_data = train_data + test_data
136     with open('../datasets/20news-bydate/20news-train-processed.txt', 'w') as f:
137         f.write('\n'.join(train_data))
138
139     with open('../datasets/20news-bydate/20news-test-processed.txt', 'w') as f:
140         f.write('\n'.join(test_data))
141
142     with open('../datasets/20news-bydate/20news-full-processed.txt', 'w') as f:
143         f.write('\n'.join(full_data))
```

Biểu diễn TF-IDF

- ❑ Tiền xử lý: đọc dữ liệu và tập hợp dữ liệu

> Output: **20news-train-processed.txt**

```
1 0<fff>49960<fff>mathew mathew manti co uk subject alt atheism faq at
2 0<fff>51060<fff>mathew mathew manti co uk subject alt atheism faq in
3 0<fff>51119<fff>i3150101 dbstu1 rz tu bs de benedikt rosenau subject
4 0<fff>51120<fff>mathew mathew manti co uk subject re univers violat
5 0<fff>51121<fff>strom watson ibm com rob strom subject re soc motss
6 0<fff>51122<fff>i3150101 dbstu1 rz tu bs de benedikt rosenau subject
7 0<fff>51123<fff>keith cco caltech edu keith allan schneider subject
8 0<fff>51124<fff>i3150101 dbstu1 rz tu bs de benedikt rosenau subject
9 0<fff>51125<fff>keith cco caltech edu keith allan schneider subject
10 0<fff>51126<fff>keith cco caltech edu keith allan schneider subject
```

Biểu diễn TF-IDF

- ❑ Tiền xử lý: tạo từ điển và tính trước giá trị idf

```
61     def generate_vocabulary(data_path):
62         def compute_idf(df, corpus_size):...
63
64
65
66         with open(data_path) as f:
67             lines = f.read().splitlines()
68             doc_count = defaultdict(int)
69             corpus_size = len(lines)
70
71         for line in lines:
72             features = line.split('<fff>')
73             text = features[-1]
74             words = list(set(text.split()))
75             for word in words:
76                 doc_count[word] += 1
```

Biểu diễn TF-IDF

- Tiền xử lý: tạo từ điển và tính trước giá trị idf

```
78 words_idfs = [(word, compute_idf(document_freq, corpus_size))
79             for word, document_freq in
80             zip(doc_count.keys(), doc_count.values())
81             if document_freq > 10 and not word.isdigit()]
82 words_idfs.sort(key=lambda (word, idf): -idf)
83 print 'Vocabulary size: {}'.format(len(words_idfs))
84 with open('../datasets/20news-bydate/words_idfs.txt', 'w') as f:
85     f.write('\n'.join([word + '<ffff>' + str(idf) for word, idf in words_idfs]))
```

Biểu diễn TF-IDF

- ❑ Tiền xử lý: tạo từ điển và tính trước giá trị **idf**

```
62     def compute_idf(df, corpus_size):  
63         assert df > 0  
64         return np.log10(corpus_size * 1. / df)
```

$$\text{idf}(w_i, D) = \log_{10} \frac{|D|}{|\{d' \in D : w_i \in d'\}|}$$

Biểu diễn TF-IDF

- Tiền xử lý: tạo từ điển và tính trước giá trị **idf**

1	aargh<fff>3.23382650162
2	ahmet<fff>3.23382650162
3	xvt<fff>3.23382650162
4	unbvm1<fff>3.23382650162
5	deskwr1t<fff>3.23382650162
6	oversight<fff>3.23382650162
7	coliseum<fff>3.23382650162
8	amorc<fff>3.23382650162
9	spacelab<fff>3.23382650162
10	brotherhood<fff>3.23382650162
11	blond<fff>3.23382650162
12	laden<fff>3.23382650162
13	dickinson<fff>3.23382650162
14	comedi<fff>3.23382650162
15	mje<fff>3.23382650162
16	durban<fff>3.23382650162

Biểu diễn TF-IDF

☐ Tiết xử lý: tính tf-idf

```
13     def get_tf_idf(data_path):
14         # get pre-computed idf values
15         with open('../datasets/20news-bydate/words_idfs.txt') as f:
16             words_idfs = [(line.split('<fff>')[0], float(line.split('<fff>')[1]))
17                           for line in f.read().splitlines()]
18
19             word_IDs = dict([(word, index)
20                               for index, (word, idf) in enumerate(words_idfs)])
21             idfs = dict(words_idfs)
22
23             with open(data_path) as f:
24                 documents = [
25                     (int(line.split('<fff>')[0]),
26                      int(line.split('<fff>')[1]),
27                      line.split('<fff>')[2])
28                     for line in f.read().splitlines()]
```

Biểu diễn TF-IDF

❑ Tiền xử lý: tính tf-idf

```
30     data_tf_idf = []
31     for document in documents:
32         label, doc_id, text = document
33         words = [word for word in text.split() if word in idfs]
34         word_set = list(set(words))
35         max_term_freq = max([words.count(word)
36                             for word in word_set])
```

Biểu diễn TF-IDF

❑ Tiền xử lý: tính tf-idf

```
38         words_tfidfs = []
39         sum_squares = 0.0
40         for word in word_set:
41             term_freq = words.count(word)
42             tf_idf_value = term_freq * 1. / max_term_freq * idfs[word]
43             words_tfidfs.append((word_IDs[word], tf_idf_value))
44             sum_squares += tf_idf_value ** 2
45
46         words_tfidfs_normalized = [str(index) + ':'
47                                     + str(tf_idf_value / np.sqrt(sum_squares))
48                                     for index, tf_idf_value in words_tfidfs]
49
50         sparse_rep = ' '.join(words_tfidfs_normalized)
51         data_tf_idf.append((label, doc_id, sparse_rep))
```

Biểu diễn TF-IDF

- Tiền xử lý: Ghi **data_tf_idf** ra file

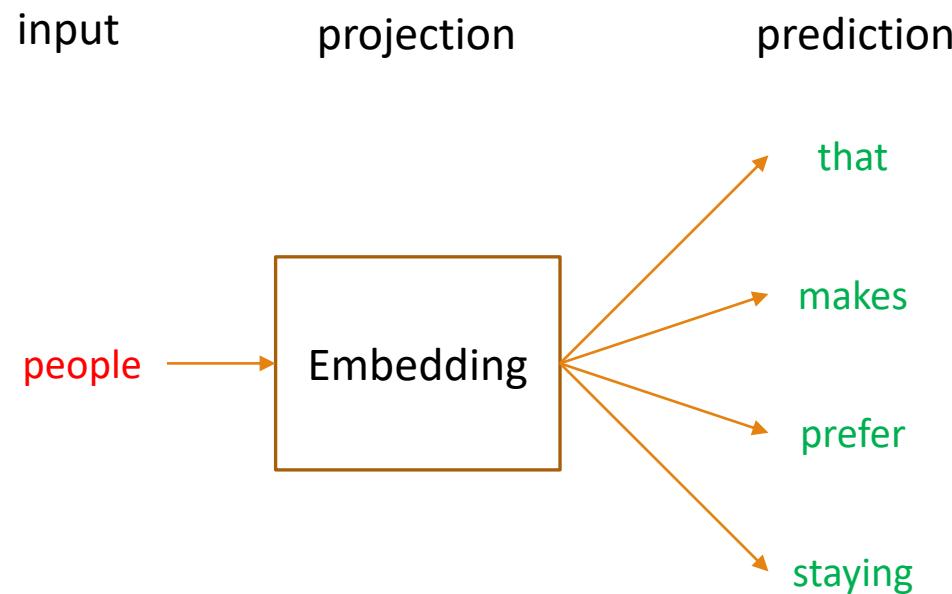
```
1 0<fff>49960<fff>9370:0.0967868358288 8553:0.229793603446 7347:0.132
2 0<fff>51060<fff>6637:0.0264613146274 9930:0.0291134757707 2501:0.03
3 0<fff>51119<fff>8648:0.16054261467 5695:0.211258519544 9671:0.62138
4 0<fff>51120<fff>10288:0.127635484843 9671:0.372830827488 10305:0.04
5 0<fff>51121<fff>7763:0.538997270778 10255:0.195617964114 9473:0.400
6 0<fff>51122<fff>10030:0.0719050870906 10174:0.0610246365229 3930:0.
7 0<fff>51123<fff>10164:0.7012549744 10288:0.170180646457 9410:0.5434
8 0<fff>51124<fff>8717:0.172847984547 5979:0.225946768213 8475:0.7192
9 0<fff>51125<fff>9370:0.137114684091 9004:0.150470242337 10022:0.205
10 0<fff>51126<fff>10243:0.205727965198 9511:0.395948539607 8294:0.506
11 0<fff>51127<fff>9385:0.546772136773 10235:0.285813329788 10261:0.74
12 0<fff>51128<fff>9868:0.341744976285 10054:0.297276273338 9333:0.417
13 0<fff>51130<fff>10160:0.211947851892 9241:0.341062262272 10255:0.31
14 0<fff>51131<fff>10288:0.145869125535 10058:0.169030324161 9977:0.18
15 0<fff>51132<fff>9370:0.274229368182 10288:0.0850903232285 10135:0.1
```

Word2vec: Biểu diễn vector cho từ

- Để thu được biểu diễn word2vec của từ, có 2 mô hình:
 1. Skip-Gram
 2. CBOW (Continuous Bag-of-Word Model)

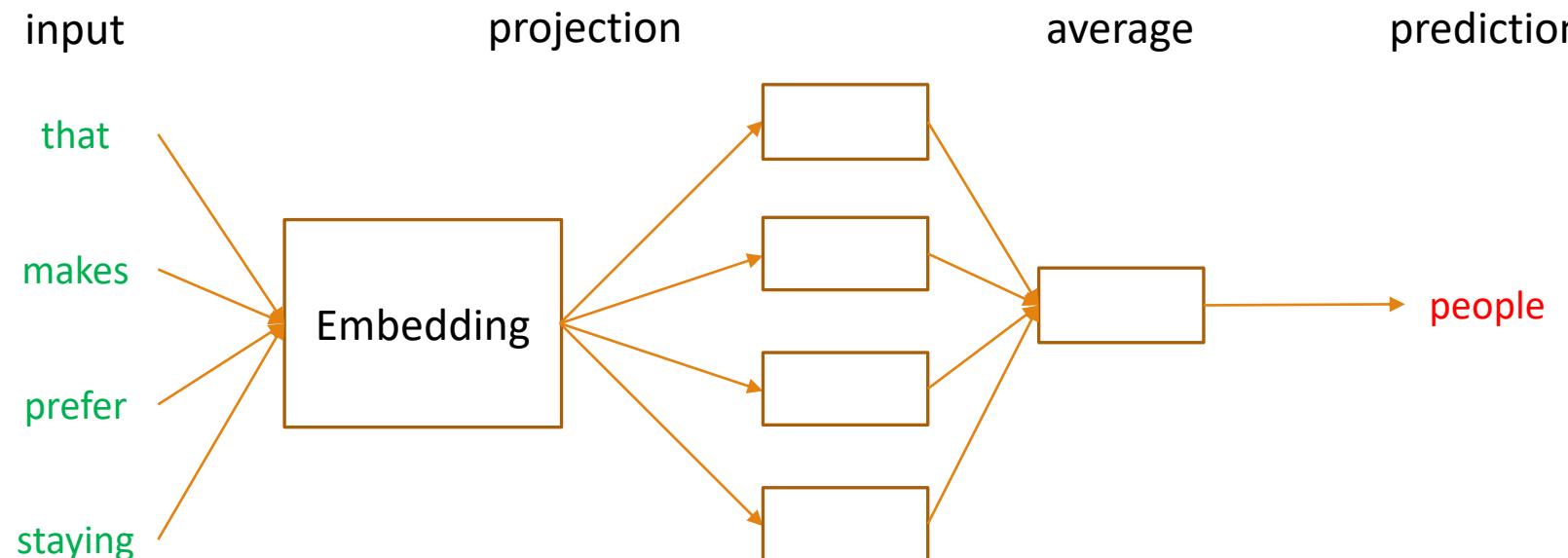
1. Skip-Gram

- ❑ Sử dụng **center word** làm **input** và **context words** làm **target**



2. CBOW

- Sử dụng context words làm **input** và center word làm **target**



Word2vec

- ❑ Sau khi huấn luyện, thu được ma trận **Word Embedding**, mỗi Hàng là một vector biểu diễn cho một từ.
- ❑ Lưu ý: ma trận Word Embedding cũng thay đổi khi training
- ❑ **Word Embedding** thường là tầng đầu tiên trong rất nhiều mô hình Deeplearning hiện nay.
- ❑ Xem chi tiết tại:

[1] <https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf>

[2] <https://arxiv.org/pdf/1411.2738.pdf>

Tổng kết

- ❑ Ôn lại Linear Regression: RSS, công thức nghiệm
- ❑ Làm quen với Python: cú pháp cơ bản, numpy
- ❑ Triển khai thuật toán Linear Regression: đọc và chuẩn hóa dữ liệu, triển khai theo phong cách hướng đối tượng, sử dụng cross-validation để tìm giá trị LAMBDA
- ❑ Biểu diễn TF-IDF: công thức TF-IDF, cách triển khai trong thực tế

Chuẩn bị cho Session 2

- Triển khai Kmeans
- Sử dụng thư viện scikit-learn cho Kmeans, SVMs

Thank you