# 情報工学実験4：データマイニング班
# (week 4) 線形回帰モデル(OLS, GD)の実装演習

1. （復習）
    1. 機械学習概観、検討演習
    2. モデルとは？（問題設定、アルゴリズム、モデル）
    3. 線形回帰モデル
    4. 仮説、損失関数、目的関数
    5. 最小二乗法
2. 最小二乗法の解法例
3. 実装演習
    1. クラスデザイン
    2. Numpy, Python チュートリアル
    3. データセット読み込み用モジュール
    4. 線形回帰モデル用モジュール
4. Linear Regression with GD
5. グラフ描画の例
6. 参考サイト

実験ページ: http://ie.u-ryukyu.ac.jp/~tnal/2019/info4/dm/

# Example: *Iris* flower data set

review

http://en.wikipedia.org/wiki/Iris_flower_data_set

(1) What is experience E?
(2) What is task T?
(3) How to measure the performance P?

- Classification
  - In Classification, the samples belong to two or more classes and we want to learn from already labeled data how to predict the class of unlabeled data.
  - E.g., distinguishes the species from each other.
  - Dataset = samples vs. features and classes

- Teach data
- supervisory signal
- output data, Y
- target
- 1 class in 3 classes

- Input data, X
- 4 features or attributes

**Fisher's *Iris* Data**

| Sepal length | Sepal width | Petal length | Petal width | Species |
|---|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 | *I. setosa* |
| 4.9 | 3.0 | 1.4 | 0.2 | *I. setosa* |
| 4.7 | 3.2 | 1.3 | 0.2 | *I. setosa* |
| 4.6 | 3.1 | 1.5 | 0.2 | *I. setosa* |
| 5.0 | 3.6 | 1.4 | 0.2 | *I. setosa* |

1 sample

# Example: boston house prices dataset

http://archive.ics.uci.edu/ml/datasets/Housir

review

(1) What is experience E?
(2) What is task T?
(3) How to measure the performance P?

- Regression
  - If the desired output consists of one or more continuous variables, then the task is called *regression*.
  - E.g., concerns housing values in suburbs of Boston.
  - Dataset = samples vs. features and continuous variables

13 features

Continuous variable

| CRIM | ZN | INDUS | （中略） | LSTAT | MEDV |
|---|---|---|---|---|---|
| 6.32E−03 | 1.80E+01 | 2.31E+00 | | 4.98E+00 | 24.00 |
| 2.73E−02 | 0.00E+00 | 7.07E+00 | | 9.14E+00 | 21.60 |
| 2.73E−02 | 0.00E+00 | 7.07E+00 | | 4.03E+00 | 34.70 |

1 sample

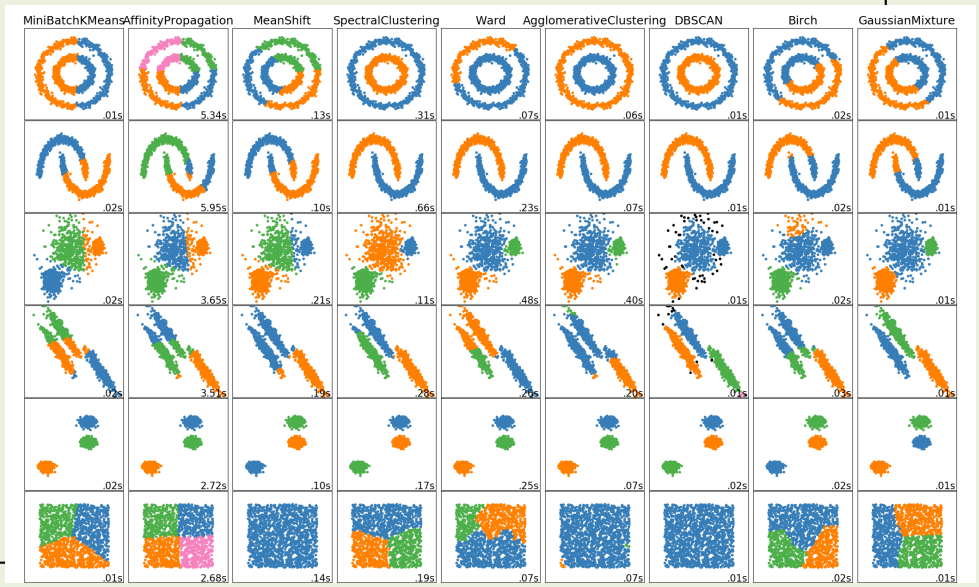
# Example: *Overview of clustering methods*

https://scikit-learn.org/stable/modules/clustering.html

review

(1) What is experience E?
(2) What is task T?
(3) How to measure the performance P?

- Clustering
  - Clustering is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense or another) to each other than to those in other groups (clusters).
  - Training data consists of a set of input vectors x **without any corresponding target values**.
  - Dataset = samples vs. features



# Terminology

review

- ML types
  - supervised, unsupervised, semi-supervised
  - (reinforcement learning, genetic algorithm,,,)
- Task types
  - classification, regression, clustering
- sample
- features, attributes
  - numerical value
  - categorical value
  - true or false
- supervisory signal, teacher, class, label, target variable

- input, output
- Input types
  - training data / training set
  - test (for evaluation)
  - validation (for hyper params)
- model
- parameters
  - hyper parameters
  - weights, parameters
- learn, fit
- predict, estimate
- evaluation
  - open or close test
  - cross validation

# Exercises for clustering

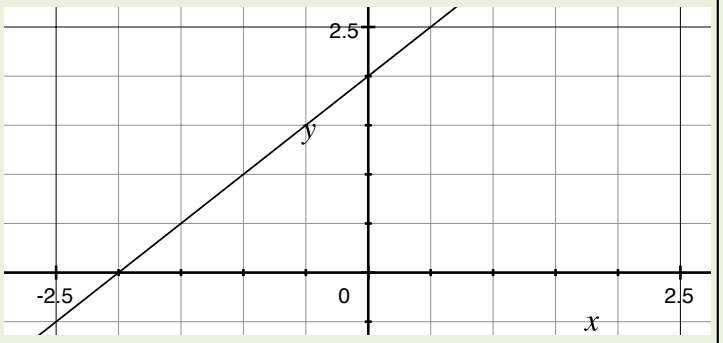
- Make a group of 2~4 students.
  - Choose one kind of problem settings on machine learning.
  - Try to design an example under the problem setting.
    - Input? Features? Output?
    - What is experience E?
    - What is task T?
    - How to measure the performance P?

# 情報工学実験4：データマイニング班
# (week 4) 線形回帰モデル(OLS, GD)の実装演習

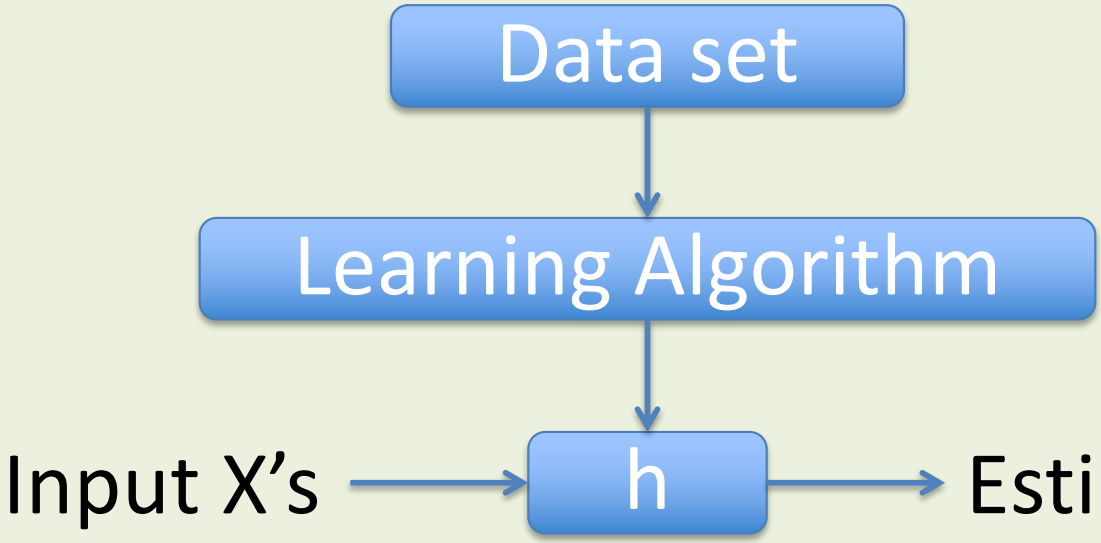
1. （復習）
    1. 機械学習概観、検討演習
    2. モデルとは？（問題設定、アルゴリズム、モデル）
    3. 線形回帰モデル
    4. 仮説、損失関数、目的関数
    5. 最小二乗法
2. 最小二乗法の解法例
3. 実装演習
    1. クラスデザイン
    2. Numpy, Python チュートリアル
    3. データセット読み込み用モジュール
    4. 線形回帰モデル用モジュール
4. Linear Regression with GD
5. グラフ描画の例
6. 参考サイト

# Models

- Represent by any formulas with (sometimes one) parameters for the relationship between input X's and output Y's.
  - In machine learning, the formulas called as "hypothesis".
  - E.g., h = a*x + b
    - a, b: parameters
  - Parameterized model.
  - Predictive model. (e.g., a=1, b=2)



# Problem <-> Algorithm + Model




Linear Regression Model

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 = \sum \theta_i x_i = \sum \theta_i \Phi_i(x)$$

$$h_\theta(x) = \theta_0 + \theta_1 x$$

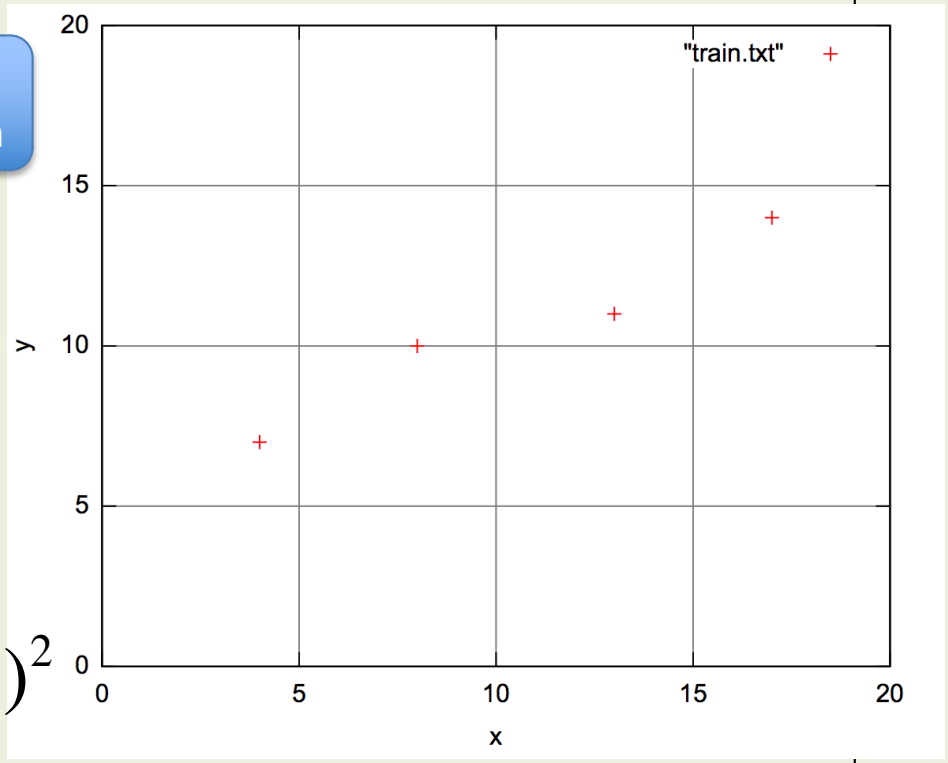
- How do we prepare a model?
- How do we evaluate the goodness?
- How do we choose the appropriate parameters?

# Linear Regression Model

- Training datasets
  - (x,y) = (4,7), (8,10), (13,11), (17,14)
- Hypothesis

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Assumption 1
Linear function




- Parameters
  - θ0, θ1
- Cost function

Assumption 2
Squared error

$$J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

- Objective function (measurement of the goodness)

$$\min_\theta J(\theta_0,\theta_1)$$

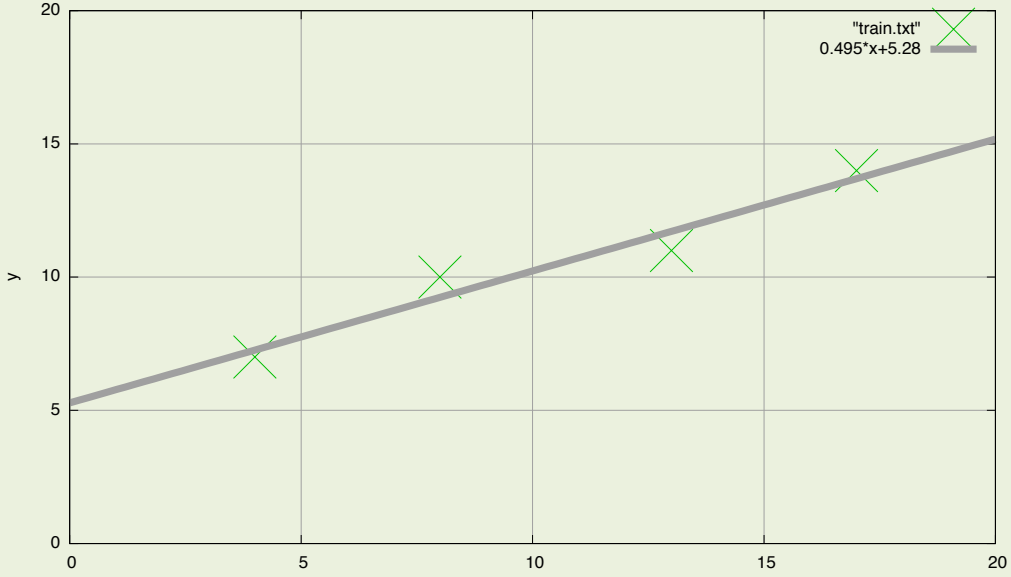
# 情報工学実験4：データマイニング班
# (week 4) 線形回帰モデル(OLS, GD)の実装演習

1. （復習）
    1. 機械学習概観、検討演習
    2. モデルとは？（問題設定、アルゴリズム、モデル）
    3. 線形回帰モデル
    4. 仮説、損失関数、目的関数
    5. 最小二乗法
2. 最小二乗法の解法例
3. 実装演習
    1. クラスデザイン
    2. Numpy, Python チュートリアル
    3. データセット読み込み用モジュール
    4. 線形回帰モデル用モジュール
4. Linear Regression with GD
5. グラフ描画の例
6. 参考サイト

# Ordinary Least Squares

review

$$h(x) = \theta_0 + \theta_1 x$$ (x,y)=(4,7), (8,10), (13,11), (17,14)




$$7 = \theta_0 + 4\theta_1$$

$$0 = \theta_0 + 4\theta_1 - 7$$

$$e_1 := \theta_0 + 4\theta_1 - 7$$

$$e_1^2 = (\theta_0 + 4\theta_1 - 7)^2$$

$$E = \sum e_i^2 \geq 0$$

$$= (\theta_0 + 4\theta_1 - 7)^2 + (\theta_0 + 8\theta_1 - 10)^2 + (\theta_0 + 13\theta_1 - 11)^2 + (\theta_0 + 17\theta_1 - 14)^2$$

$$= 538\theta_1^2 + 84\theta_0\theta_1 + 4\theta_0^2 - 978\theta_1 - 84\theta_0 + 466$$

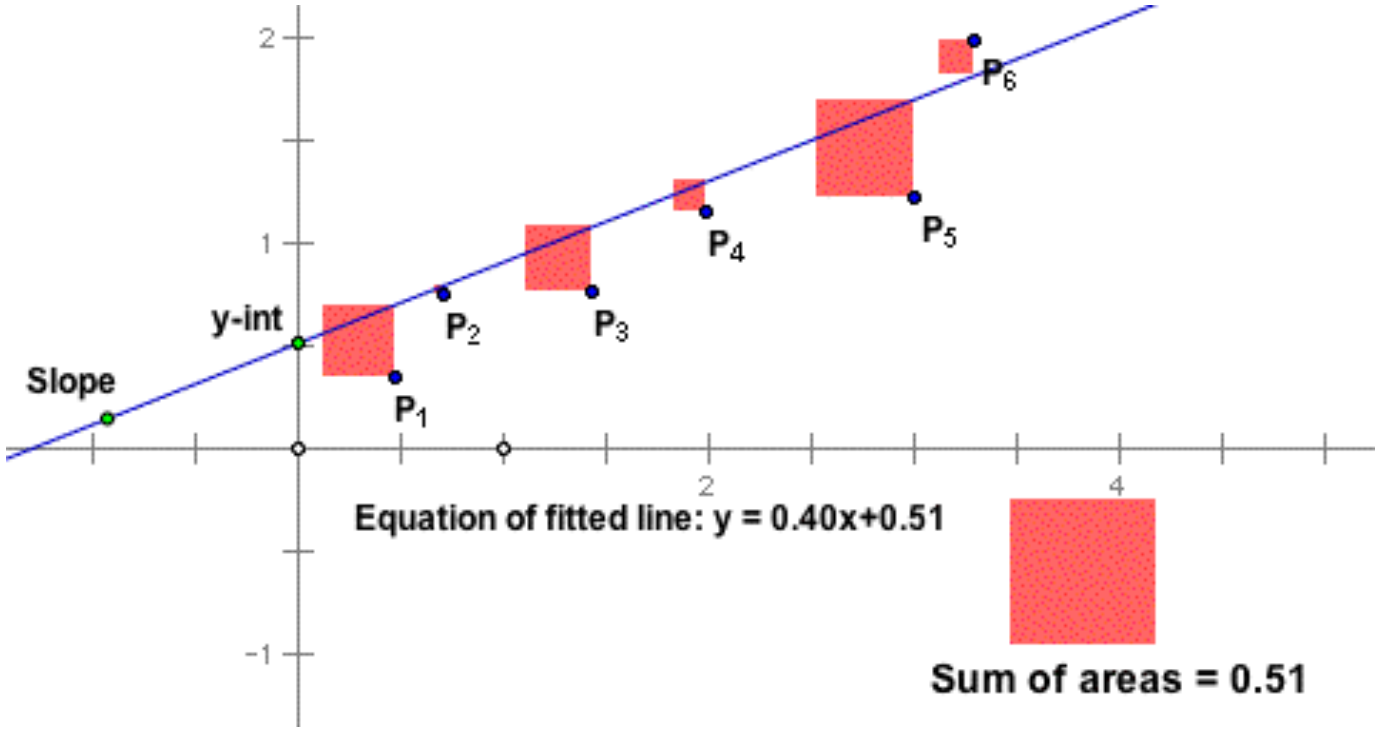
$$= (2\theta_1 + 21\theta_0 - 21)^2 + 97(\theta_0 - 48/97)^2 + 121/97$$

θ0=1029/194≒5.28, θ1=48/97≒0.495

h(x) = 5.28 + 0.495x

Ref., http://gihyo.jp/dev/serial/01/machine-learning/0008

# Ordinary Least Squares (OLS)




https://inst.eecs.berkeley.edu/~ee127a/book/login/l_ols_main.html

# 情報工学実験4：データマイニング班
# (week 4) 線形回帰モデル(OLS, GD)の実装演習

1. （復習）
    1. 機械学習概観、検討演習
    2. モデルとは？（問題設定、アルゴリズム、モデル）
    3. 線形回帰モデル
    4. 仮説、損失関数、目的関数
    5. 最小二乗法
2. 最小二乗法の解法例
3. 実装演習
    1. クラスデザイン
    2. Numpy, Python チュートリアル
    3. データセット読み込み用モジュール
    4. 線形回帰モデル用モジュール
4. Linear Regression with GD
5. グラフ描画の例
6. 参考サイト

# OLS resolver (1/2)

residual sum of squares

$$RSS(\theta) = \sum_{i}^{N} (y_i - h_\theta(x_i))^2$$

$$= \sum_{i}^{N} (y - \theta_0 - x_i\theta_1)^2$$

$$= (Y - X\theta)^T (Y - X\theta)$$

$$\frac{\partial}{\partial \theta} RSS(\theta) = 0$$

$$X^T X\theta = X^T Y$$

$$\theta = (X^T X)^{-1} X^T Y$$

cond., ($X^TX$) is nonsigular (regular matrix).

$$X = \begin{bmatrix} x^{00} & x^{01} & \dots & x^{0M} \\ x^{10} & x^{11} & \dots & x^{1M} \\ \dots & \dots & \dots & \dots \\ x^{N0} & x^{N1} & \dots & x^{NM} \end{bmatrix}$$

$$\theta = \begin{bmatrix} \theta^0 \\ \theta^1 \\ \dots \\ \theta^M \end{bmatrix}$$

$$Y = \begin{bmatrix} y^0 \\ y^1 \\ \dots \\ y^N \end{bmatrix}$$

# OLS resolver (2/2)

$$RSS(\theta) = \sum_i^N (y_i - h_\theta(x_i))^2 = \sum_i^N (y - \theta_0 - x_i\theta_1)^2 = (Y - X\theta)^T (Y - X\theta)$$

$$= (Y^T - \theta^T X^T)(Y - X\theta) = Y^T Y - \theta^T X^T Y - Y^T X\theta + \theta^T X^T X\theta$$

$$\frac{\partial}{\partial\theta} RSS(\theta) = 0, \frac{\partial x^T a}{\partial x} = a, \frac{\partial a^T x}{\partial x} = a$$

$$\frac{\partial \theta^T X^T Y}{\partial\theta} = X^T Y$$

$$\frac{\partial Y^T X\theta}{\partial\theta} = (Y^T X)^T = X^T Y$$

$$\frac{\partial \theta^T X^T X\theta}{\partial\theta} = \frac{\partial \theta^T (X^T X\theta)}{\partial\theta} + \frac{\partial (\theta^T X^T X)\theta}{\partial\theta} = X^T X\theta + (\theta^T X^T X)^T = 2X^T X\theta$$

$$\frac{\partial}{\partial\theta} RSS(\theta) = -2X^T Y + 2X^T X\theta = 0$$

$$X^T X\theta = X^T Y$$

$$\theta = (X^T X)^{-1} X^T Y$$

# 情報工学実験4：データマイニング班
# (week 4) 線形回帰モデル(OLS, GD)の実装演習

1. （復習）
    1. 機械学習概観、検討演習
    2. モデルとは？（問題設定、アルゴリズム、モデル）
    3. 線形回帰モデル
    4. 仮説、損失関数、目的関数
    5. 最小二乗法
2. 最小二乗法の解法例
3. 実装演習
    1. クラスデザイン
    2. Numpy, Python チュートリアル
    3. データセット読み込み用モジュール
    4. 線形回帰モデル用モジュール
4. Linear Regression with GD
5. グラフ描画の例
6. 参考サイト

# Implementation of OLS resolver

# Class design / How to use

```
# from numpy as np
# X = np.array([[1,4],[1,8],[1,13],[1,17]])
# Y = np.array([7, 10, 11, 14])
>>> import datasets
>>> X, Y = datasets.load_linear_example1()
>>> import regression
>>> model = regression.LinearRegression()
>>> model.fit(X, Y)
>>> model.theta
array([ 5.30412371,  0.49484536])
>>> model.predict(X)
array([  7.28350515,   9.2628866 ,  11.7371134 ,  13.71649485])
>>> model.score(X, Y) # RSS
1.2474226804123705
```

X = [x0, x1]

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 = \sum \theta_i x_i$$

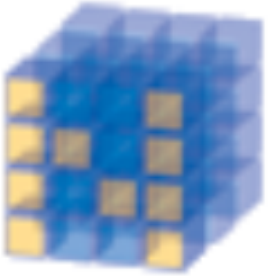
$$h_\theta(x) = \theta_0 + \theta_1 x$$

https://github.com/naltoma/regression-test.git

# Linear Algebra with NumPy

http://www.numpy.org

- import numpy as np
- help(np)
  - Provides
    - 1. An array object of arbitrary homogeneous items
    - 2. Fast mathematical operations over arrays
    - 3. Linear Algebra, Fourier Transforms, Random Number Generation



NumPy

Base N-dimensional array package

# Linear Algebra Practice 1

https://github.com/naltoma/intro_jupyter_sklearn

```
>>> import numpy as np
# create an array, similar to matrix.
# if you wan to use concrete matrix object, check `np.mat()'.
>>> a = np.array([[1,2,3],[4,5,6]])
>>> a
array([[1, 2, 3],
       [4, 5, 6]])
>>> type(a)
<class 'numpy.ndarray'>
>>> a.shape
(2, 3)
```

# Linear Algebra Practice 2

```
>>> a
array([[1, 2, 3],
       [4, 5, 6]])
>>> a + 1
array([[2, 3, 4],
       [5, 6, 7]])
>>> a * 2
array([[ 2,  4,  6],
       [ 8, 10, 12]])
```

```
>>> a.T
array([[1, 4],
       [2, 5],
       [3, 6]])
>>> a*a       # elementwise product
array([[ 1,  4,  9],
       [16, 25, 36]])
>>> np.dot(a,a.T)   # dot product of two arrays
array([[14, 32],
       [32, 77]])
>>> np.linalg.inv(np.dot(a,a.T))
array([[ 1.42592593, -0.59259259],
       [-0.59259259,  0.25925926]])
```

# Numpy Tools 1

```
# return evenly spaced values
within a given interval.
>>> np.arange(0,1,0.3)
array([ 0. ,  0.3,  0.6,  0.9])
>>> np.arange(8)
array([0, 1, 2, 3, 4, 5, 6, 7])
# gives a new shape
>>> np.reshape(np.arange(6),(2,3))
array([[0, 1, 2],
       [3, 4, 5]])
>>> np.reshape(np.arange(6),(3,2))
array([[0, 1],
       [2, 3],
       [4, 5]])
```

```
# return evenly spaced
numbers over a specified
interval.
>>> np.linspace(0,2,3)
array([ 0.,  1.,  2.])
>>> np.linspace(0,2,4)
array([ 0.        , 0.66666667,
1.33333333,  2.        ])


>>> np.zeros((2,3))
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])
>>> np.ones((2,3))
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.]])
```

# Numpy Tools 2

```
>>> np.random.seed(0)
>>> np.random.random((2,3))
array([[ 0.5488135 ,  0.71518937,
0.60276338],
       [ 0.54488318,  0.4236548 ,
0.64589411]])
>>> np.random.seed(0)
>>> np.random.random((2,3))
array([[ 0.5488135 ,  0.71518937,
0.60276338],
       [ 0.54488318,  0.4236548 ,
0.64589411]])
```

```
>>>
a=np.reshape(np.arange(12),(3,4))
>>> a
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
>>> a[:,0]
array([0, 4, 8])
>>> a[:,0:2]
array([[0, 1],
       [4, 5],
       [8, 9]])
>>> a[:,0:3]
array([[ 0,  1,  2],
       [ 4,  5,  6],
       [ 8,  9, 10]])
```

# Python Tips

- reloading module
  - import importlib
  - importlib.reload(module)
  - http://docs.python.jp/3/library/importlib.html#module-importlib

# [reprint] Class design / How to use

```
# from numpy as np
# X = np.array([[1,4],[1,8],[1,13],[1,17]])
# Y = np.array([7, 10, 11, 14])
>>> import datasets
>>> X, Y = datasets.load_linear_example1()
>>> import regression
>>> model = regression.LinearRegression()
>>> model.fit(X, Y)
>>> model.theta
array([ 5.30412371,  0.49484536])
>>> model.predict(X)
array([  7.28350515,   9.2628866 ,  11.7371134 ,  13.71649485])
>>> model.score(X, Y) # RSS
1.2474226804123705
```

https://github.com/naltoma/regression-test.git

# prepare a repository

you can use any repositories including GitHub.

local> cd ~/GIT

local> git init --bare regression-test

local> cd ~/temp

local> git clone ~/GIT/regression-test

local> cd regression-test

# datasets.py

```
# testing
>>> import datasets
>>> X, Y = datasets.load_linear_example1()
>>> X
array([[ 1,  4],
       [ 1,  8],
       [ 1, 13],
       [ 1, 17]])
>>> X[0]
array([1, 4])
>>> Y
array([ 7, 10, 11, 14])
```

```
import numpy as np


def load_linear_example1():
    X = np.array([[1,4],[1,8],[1,13],[1,17]])
    Y = np.array([7, 10, 11, 14])
    return X, Y
```

if correct, then add & commit!

# regression.py (ver.1)

```
# testing (cont.)
>>> import regression
>>> model = regression.LinearRegression()
(this class returns an instance only)
>>> model.x
>>> #nothing
```

if correct, then add & commit!

```
import numpy as np

class LinearRegression:
    x = None
    theta = None
    y = None

    def fit(self, x, y):
        pass

    def predict(self, x):
        pass

    def score(self, x, y):
        pass
```

# regression (ver.2: fit())

```
# testing (cont.)
>>> import importlib
>>> importlib.reload(regression)
>>> model = regression.LinearRegression()
>>> model.fit(X, Y)
>>> model.theta
array([ 5.30412371,  0.49484536])
```

if correct, then add & commit!

```
def fit(self, x, y):
    temp = np.linalg.inv(np.dot(x.T,x))
    self.theta = np.dot(np.dot(temp,x.T),y)
```

$$\theta = (X^T X)^{-1} X^T Y$$

# regression (ver.3: predict())

```
# testing (cont.)
>>> importlib.reload(regression)
>>> model = regression.LinearRegression()
>>> model.fit(X, Y)
>>> model.predict(X)
array([  7.28350515,   9.2628866 ,  11.7371134 ,  13.71649485])
```

```
def predict(self, x):
    return np.dot(x, self.theta)
```

if correct, then add & commit!

# regression (ver.4: score())

```
# testing (cont.)
>>> importlib.reload(regression)
>>> model = regression.LinearRegression()
>>> model.fit(X, Y)
>>> model.score(X, Y)
1.2474226804123705
```

RSS: residual sum of squares

if correct, then add & commit!

```
def score(self, x, y):
    error = self.predict(x) - y
    return (error**2).sum()
```

# [reprint] Class design / How to use

```
# from numpy as np
# X = np.array([[1,4],[1,8],[1,13],[1,17]])
# Y = np.array([7, 10, 11, 14])
>>> import datasets
>>> X, Y = datasets.load_linear_example1()
>>> import regression
>>> model = regression.LinearRegression()
>>> model.fit(X, Y)
>>> model.theta
array([ 5.30412371,  0.49484536])
>>> model.predict(X)
array([  7.28350515,   9.2628866 ,  11.7371134 ,  13.71649485])
>>> model.score(X, Y) # RSS
1.2474226804123705
```

https://github.com/naltoma/regression-test.git

# 情報工学実験4：データマイニング班
# (week 4) 線形回帰モデル(OLS, GD)の実装演習

1. （復習）
    1. 機械学習概観、検討演習
    2. モデルとは？（問題設定、アルゴリズム、モデル）
    3. 線形回帰モデル
    4. 仮説、損失関数、目的関数
    5. 最小二乗法
2. 最小二乗法の解法例
3. 実装演習
    1. クラスデザイン
    2. Numpy, Python チュートリアル
    3. データセット読み込み用モジュール
    4. 線形回帰モデル用モジュール
4. Linear Regression with GD
5. グラフ描画の例
6. 参考サイト

# Linear Regression with GD

# Gradient descent algorithm

review

Repeat until convergence {

$$\theta_i := \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta_0, \theta_1)$$

}

(1) Start with any parameters.
(2) Update the parameters simultaneously, until convergence.

## Simple example

$$J(\theta) = \theta^2, \alpha = 0.1$$

$$new\_\theta = \theta - \alpha \frac{d}{d\theta} J(\theta)$$

$$= \theta - 0.1 * 2\theta = \theta - 0.2\theta = 0.8\theta$$

- e.g., α=0. 1, θ=3, J(θ)=9
- 1st update
  - New_θ = 0.8*3 = 2.4
  - J(θ) = 2.4**2 = 5.76
- 2nd update
  - New_θ = 0.8*2.4 = 1.92
  - J(θ) = 1.92**2 = 3.6864
- 3rd update
  - New_θ = 1.536
  - J(θ) = 2.359296
- 4th update
  - New_θ = 1.2288000000000001
  - J(θ) = 1.5099494400000002

# Update the parameters simultaneously

$$\theta_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

**Good!**

```
temp0 = calculate new theta_0
temp1 = calculate new theta_1
theta_0 = temp0
theta_1 = temp1
```

**BAD!**

```
temp0 = calculate new theta_0
theta_0 = temp0
temp1 = calculate new theta_1
theta_1 = temp1
```

# Partial differentiation with theta 0 (intercept)

$$\frac{\partial}{\partial\theta_0}J(\theta_0,\theta_1) = \frac{\partial}{\partial\theta_0}\frac{1}{2m}\sum_{i=1}^{m}(h_\Theta(x^i)-y^i)^2$$

$$= \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x^i)-y^i)\frac{\partial}{\partial\theta_0}(h_\Theta(x^i)-y^i)$$

$$= \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x^i)-y^i)\frac{\partial}{\partial\theta_0}(\theta_0x_0+\theta_1x_1-y^i)$$

$$= \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x^i)-y^i)x_0$$

$$= \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x^i)-y^i)$$

# Partial differentiation with theta 1 (slope)

$$\frac{\partial}{\partial\theta_1}J(\theta_0,\theta_1) = \frac{\partial}{\partial\theta_1}\frac{1}{2m}\sum_{i=1}^{m}(h_\Theta(x^i)-y^i)^2$$

$$= \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x^i)-y^i)\frac{\partial}{\partial\theta_1}(h_\Theta(x^i)-y^i)$$

$$= \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x^i)-y^i)\frac{\partial}{\partial\theta_1}(\theta_0 x_0+\theta_1 x_1-y^i)$$

$$= \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x^i)-y^i)x_1$$

# Updating formula on GD

$$\frac{\partial}{\partial \theta_i} J(\Theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\Theta(x^i) - y^i) x_i$$

$$Therefore,$$

$$\theta_i := \theta_i - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\Theta(x^i) - y^i) x^i$$

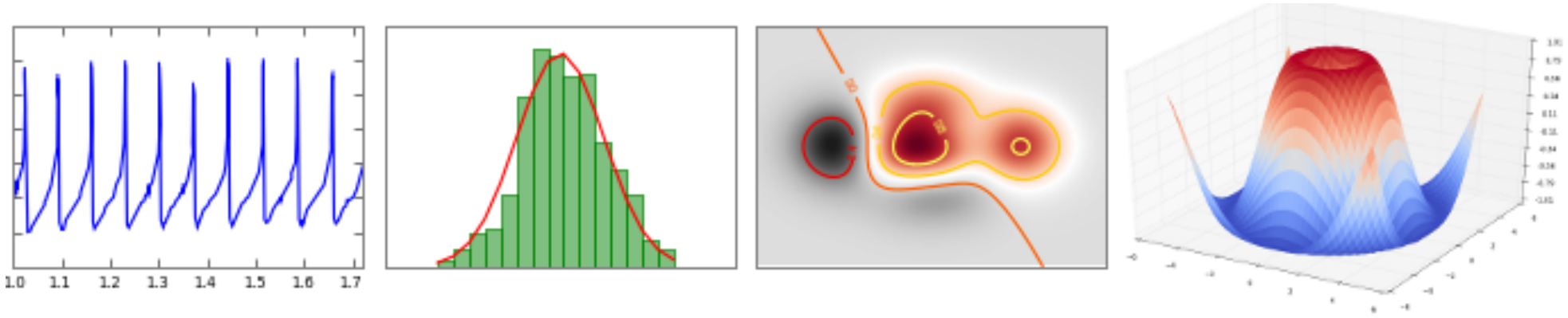
# 情報工学実験4：データマイニング班
## (week 4) 線形回帰モデル(最小二乗法)の実装演習

1. （復習）機械学習概観、検討演習
2. （復習）モデルとは？（問題設定、アルゴリズム、モデル）
3. （復習）線形回帰モデル
4. （復習）仮説、損失関数、目的関数
5. （復習）最小二乗法
6. 最小二乗法の解法例
7. 実装演習
    1. クラスデザイン
    2. Numpy, Python チュートリアル
    3. データセット読み込み用モジュール
    4. 線形回帰モデル用モジュール
8. グラフ描画の例
9. 参考サイト

# 情報工学実験4：データマイニング班
# (week 4) 線形回帰モデル(OLS, GD)の実装演習

1. （復習）
    1. 機械学習概観、検討演習
    2. モデルとは？（問題設定、アルゴリズム、モデル）
    3. 線形回帰モデル
    4. 仮説、損失関数、目的関数
    5. 最小二乗法
2. 最小二乗法の解法例
3. 実装演習
    1. クラスデザイン
    2. Numpy, Python チュートリアル
    3. データセット読み込み用モジュール
    4. 線形回帰モデル用モジュール
4. Linear Regression with GD
5. グラフ描画の例
6. 参考サイト

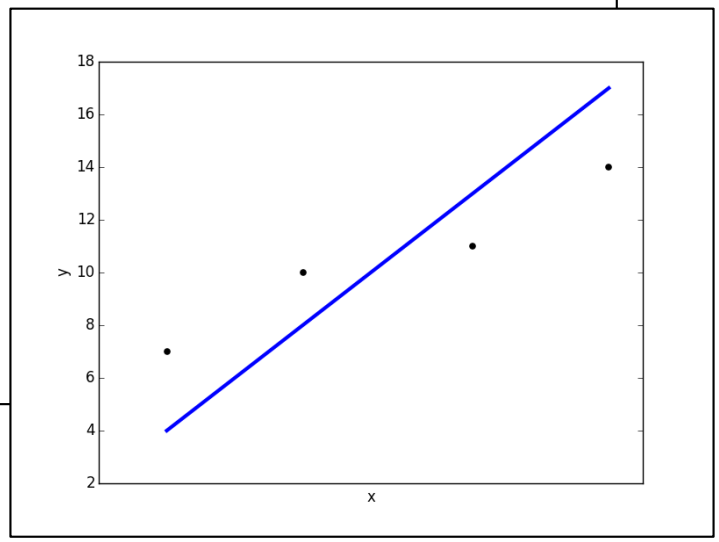
# matplotlib




http://matplotlib.org

# 2D Graph with Matplotlib

```
import numpy as np
x = np.array([4, 8, 13, 17])
y = np.array([7, 10, 11, 14])
estimated = 5.30412371 + 0.49484536*x
import matplotlib.pyplot as plt
plt.scatter(x, y, color="black")
plt.plot(x, estimated, color='blue', linewidth=3)
plt.xlabel('x')
plt.ylabel('y')
plt.xticks(())
plt.show()
```

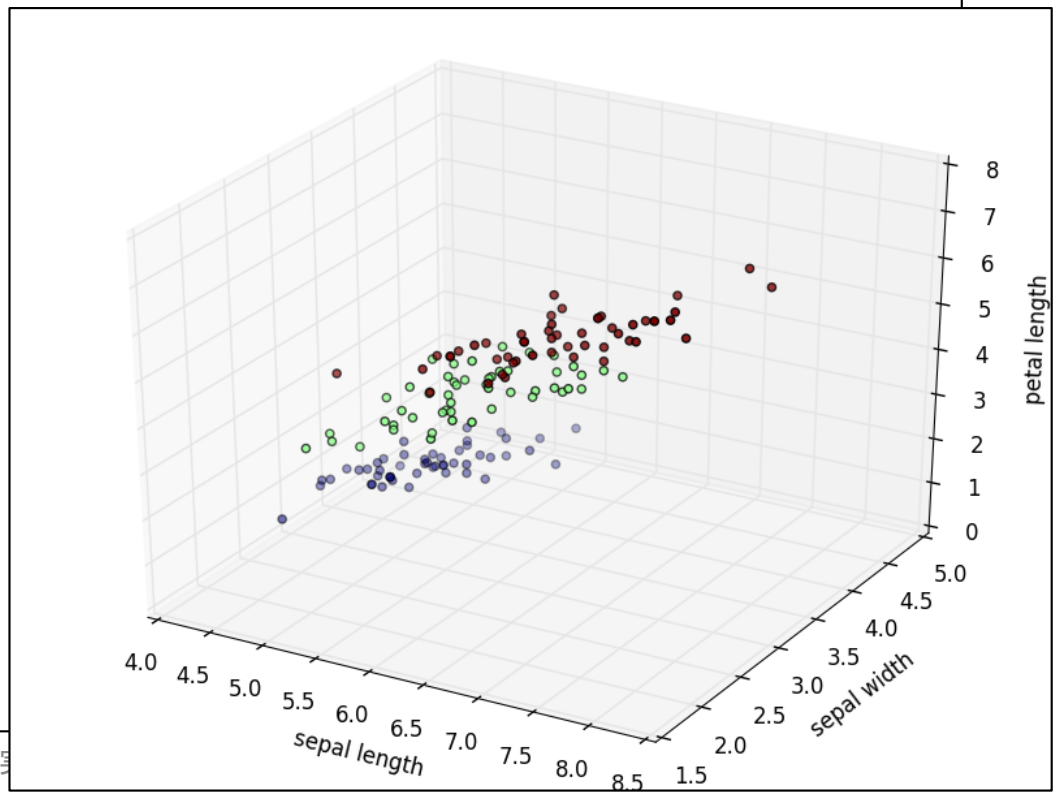



hg clone ssh://info3dm@shark//home/info3dm/HG/2014/tnal/regression
cd regression
python matplotlib-2d.py

# 3D Graph with Matplotlib

```
from sklearn import datasets
iris = datasets.load_iris()
data = iris.data
target = iris.target

import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(1)
ax = Axes3D(fig)
X = data[:,0]
Y = data[:,1]
Z = data[:,2]
labels = target
ax.scatter(X, Y, Z, c=labels)
ax.set_xlabel("sepal length")
ax.set_ylabel("sepal width")
ax.set_zlabel("petal length")
plt.show()
```



# References


- Machine Learning | Coursera, https://class.coursera.org/ml-007
- The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Second Edition, February 2009, http://statweb.stanford.edu/~tibs/ElemStatLearn/
- Machine Learning in Action, http://www.manning.com/pharrington/
- 機械学習 はじめよう 第11回 線形回帰を実装してみよう, http://gihyo.jp/dev/serial/01/machine-learning/0011
- 回帰分析 (regression analysis) – 機械学習の「朱鷺の杜Wiki」, http://ibisforest.org/index.php?回帰分析
- わかりやすいパターン認識, http://www.amazon.co.jp/わかりやすいパターン認識-石井-健一郎/dp/4274131491
- 正規方程式の導出, http://mathtrain.jp/seikiequ
- Linear Regression using Gradient Descent, https://towardsdatascience.com/linear-regression-using-gradient-descent-97a6c8700931