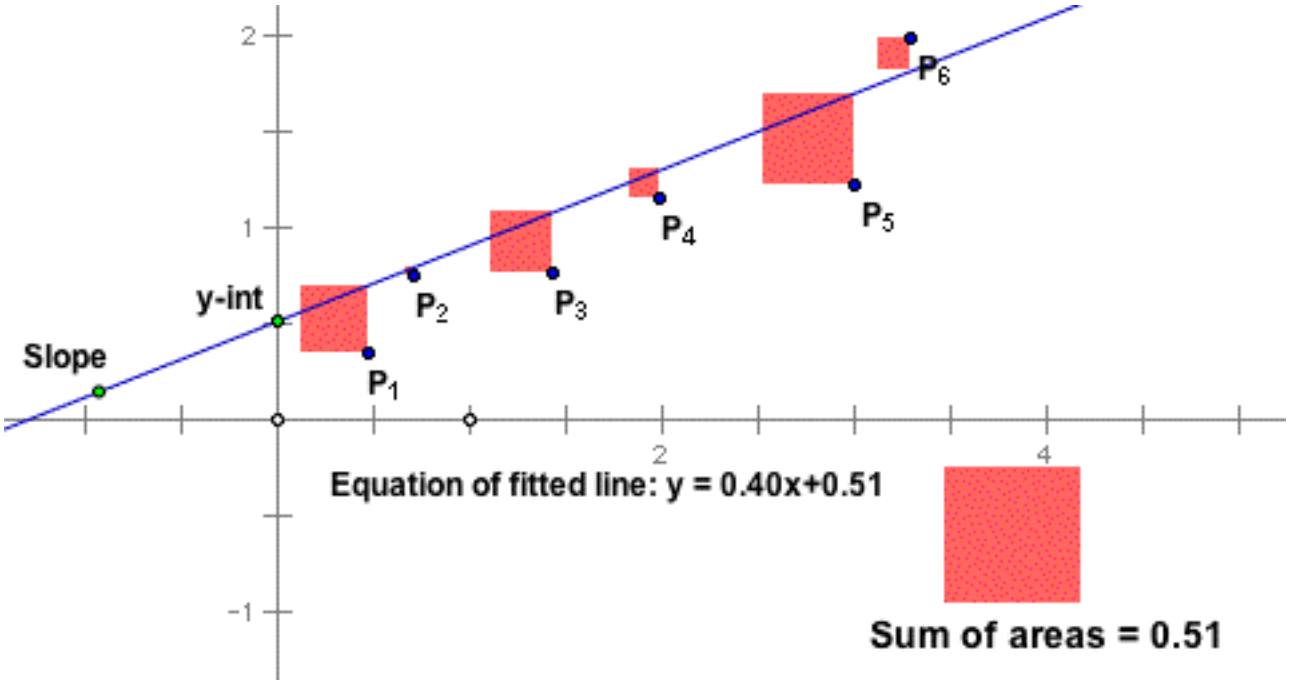
情報工学実験4：データマイニング班
(week 4) 線形回帰モデル(最小二乗法)の実装演習

1. （復習）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
    5. グラフ描画の例
4. 参考サイト

実験ページ: http://ie.u-ryukyu.ac.jp/~tnal/2014/info4/dm/

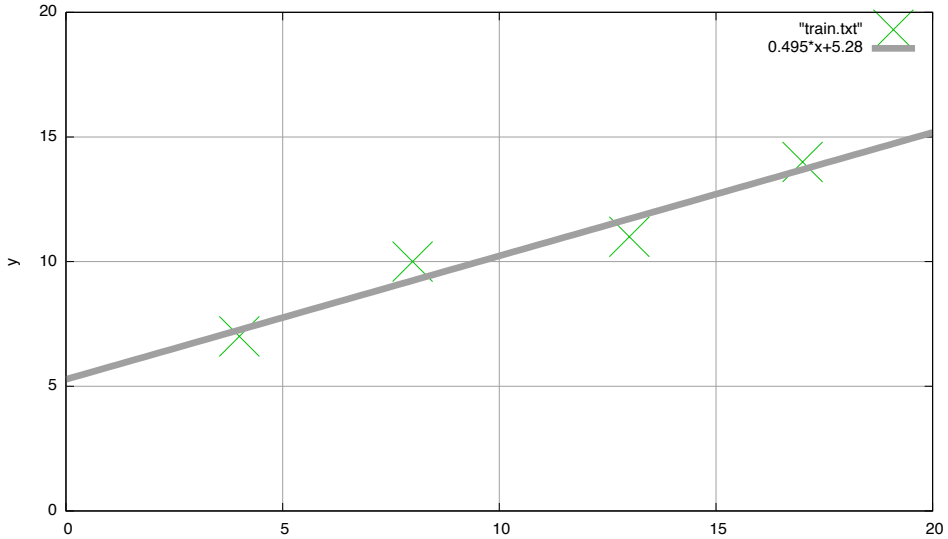
Ordinary Least Squares (OLS)



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

Ordinary Least Squares

$$h(x) = \theta_0 + \theta_1 x$$ (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

OLS resolver

$$RSS(\theta) = \sum_i^N (y_i - h_\theta(x_i))^2$$

$$= \sum_i^N (y - \theta_0 - x_i\theta_1)^2$$

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

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

$$X^T (Y - X\theta) = 0$$

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

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

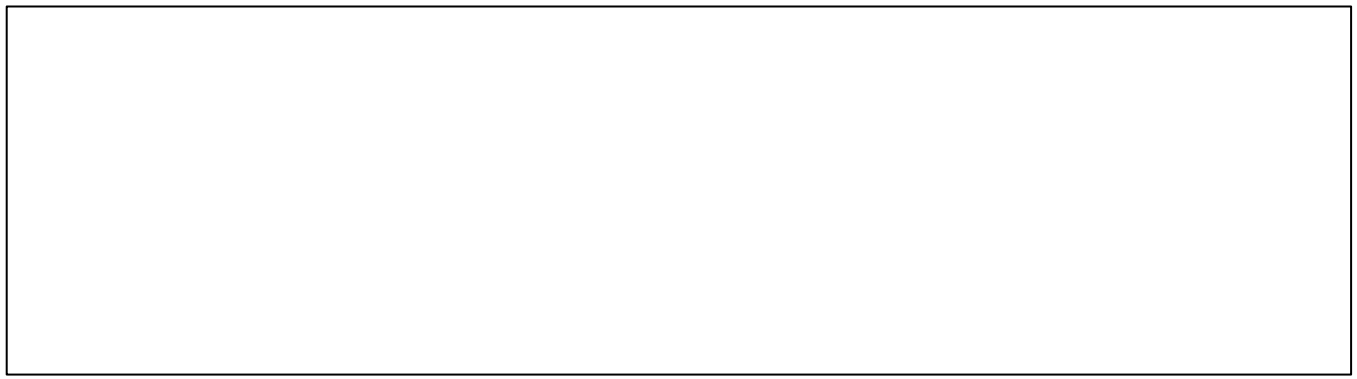
cond., (X^TX) is nonsigular (regular matrix).

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

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

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

Implementation of OLS resolver



Class design / How to use

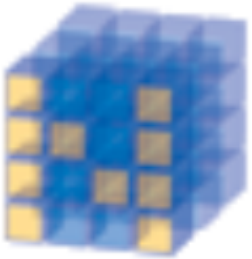
```
# from numpy as np
# X = np.array([[1,4],[1,8],[1,13],[1,17]])
# Y = np.array([7, 10, 11, 14])
>>> import datasets
>>> X, Y = datasets.load_linear_example1()
>>> import regression
>>> model = regression.LinearRegression()
>>> model.fit(X, Y)
>>> model.theta
array([ 5.30412371,  0.49484536])
>>> model.predict(X)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

hg clone ssh://info3dm@shark//home/info3dm/HG/2014/tnal/regression

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

```
>>> 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,  0.2925926]])
```

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
```

hg clone ssh://info3dm@shark//home/info3dm/HG/2014/tnal/regression

prepare a repository

```
local> ssh info3dm@shark
shark> cd ~/HG/2014/e1257xx
shark> hg init regression
shark> exit
local> hg clone ssh://info3dm@shark//home/info3dm/HG/2014/ex1257xx/regression
local> cd regression
```

datasets.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

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
```

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

if correct, then commit & push!!

regression.py (ver.1)

```
# testing
>>> import regression
>>> model = regression.LinearRegression()
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import numpy as np

class LinearRegression:

    input = None
    theta = None
    output = None
```

```
    def fit(self, input, output):
        pass

    def predict(self, input):
        pass

    def score(self, input, output):
        pass
```

regression (ver.2: fit())

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

```
def fit(self, input, output):
    self.theta =
np.dot(np.dot(np.linalg.inv(np.dot(input.T,input)
),input.T),output)
```

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

regression (ver.3: predict())

```
# testing
>>> 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, input):
    return np.dot(input, self.theta)
```

regression (ver.4: score())

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

RSS: residual sum of squares

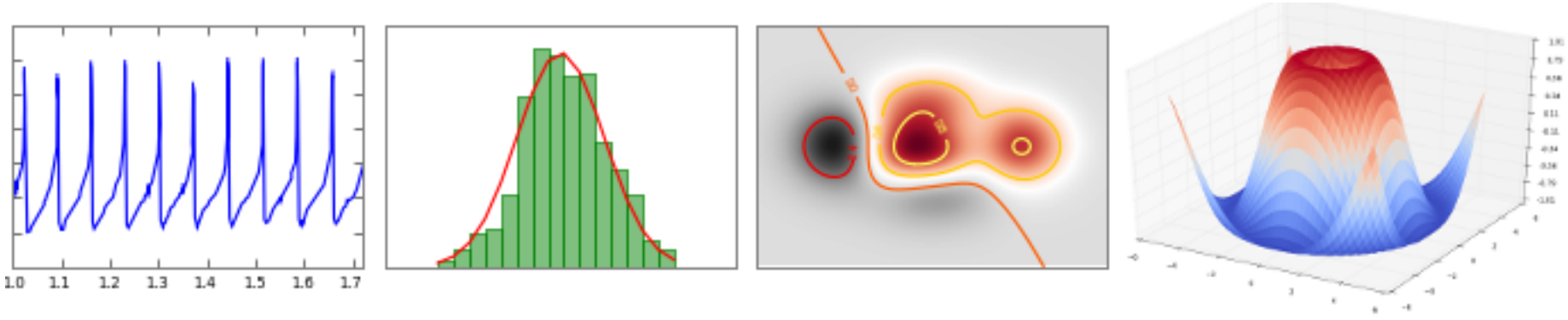
```
def score(self, input, output):
    error = self.predict(input) - output
    return (error**2).sum()
```

[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
```

hg clone ssh://info3dm@shark//home/info3dm/HG/2014/tnal/regression

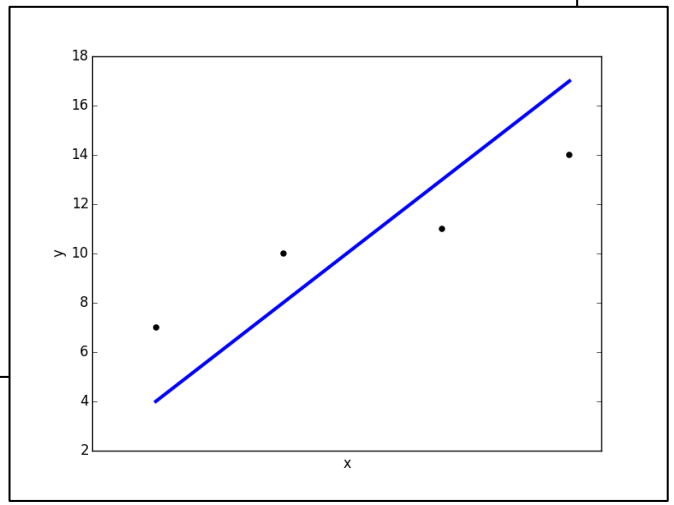
matplotlib



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()
```

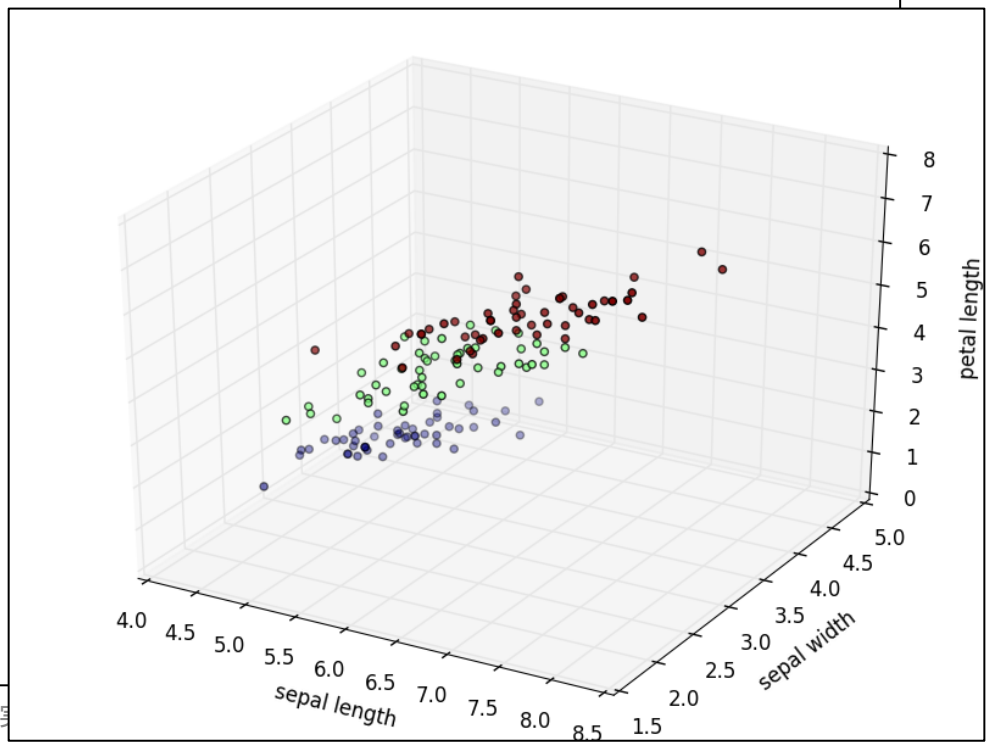


hg clone ssh://info3dm@shark//home/info3dm/HG/2014/tnal/regression
cd regression
python matplotlib-2d.py

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

$$RSS = \sum (y - h_\theta(x))^2$$

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

$$(y^0 - \theta^0 x^{00} - \theta^1 x^{01})^2 + (y^1 - \theta^0 x^{10} - \theta^1 x^{11})^2 + \ldots + (y^N - \theta^0 x^{N0} - \theta^1 x^{N1})^2 = 0$$

$$\begin{bmatrix} x^{00} & x^{01} & \ldots & x^{0M} \\ x^{10} & x^{11} & \ldots & x^{1M} \\ \ldots & \ldots & \ldots & \ldots \\ x^{N0} & x^{N1} & \ldots & x^{NM} \end{bmatrix} \begin{bmatrix} \theta^0 \\ \theta^1 \\ \ldots \\ \theta^M \end{bmatrix} = \begin{bmatrix} y^0 \\ y^1 \\ \ldots \\ y^N \end{bmatrix}$$

$$X\theta = Y$$

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

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

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

$$J(\theta) = \frac{1}{2}\sum (x^i\theta_j - y^i)^2 = \frac{1}{2}(X\theta - Y)^T (X\theta - Y)$$

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

$$X^T X\theta - X^T Y = 0$$

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