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/2020/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

- Input data, X
- 4 features or attributes

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

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/Housin

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?

回帰タスク（regression）の例を考えてみよう。

情報工学実験4：データマイニング班

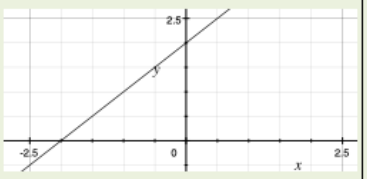
(week 4) 線形回帰モデル(OLS, GD)の実装演習

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

review

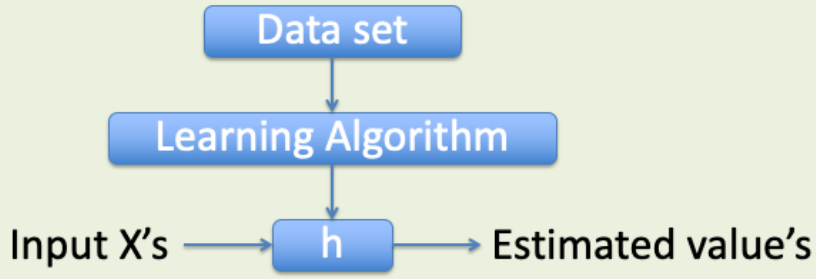
Models

- 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

review



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

review

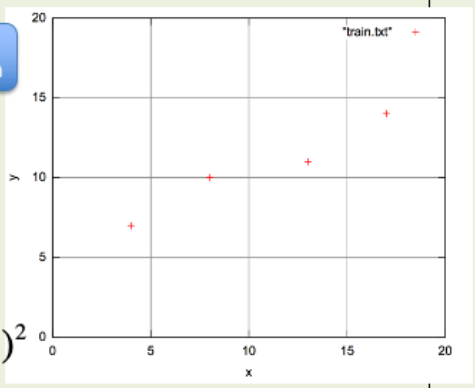
- Training datasets
  - (x,y) = (4,7), (8,10), (13,11), (17,14)
- Hypothesis

  Assumption 1
  Linear function

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

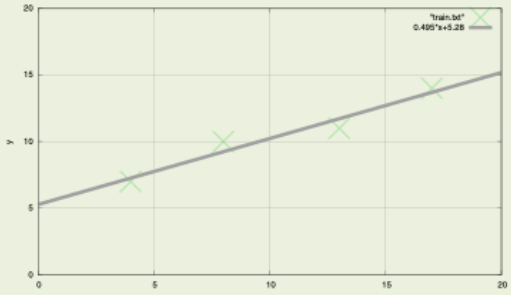
(week 4) 線形回帰モデル(OLS, GD)の実装演習

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)



$7 = \theta_0 + 4\theta_1$

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

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

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

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

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

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

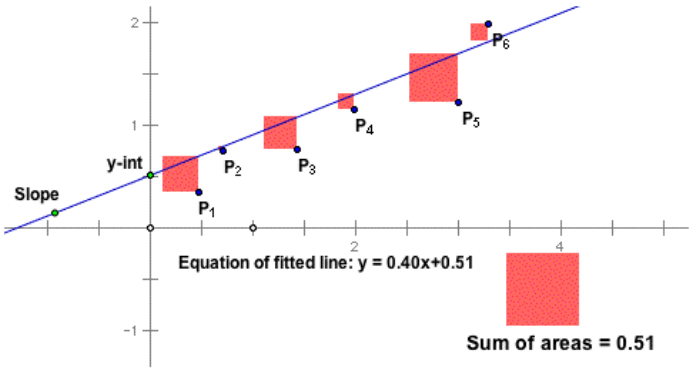
$= (2\theta_1 + 21\theta_0 - 21)^2 + 97(\theta_0 - 48/97)^2 + 121/97$

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)



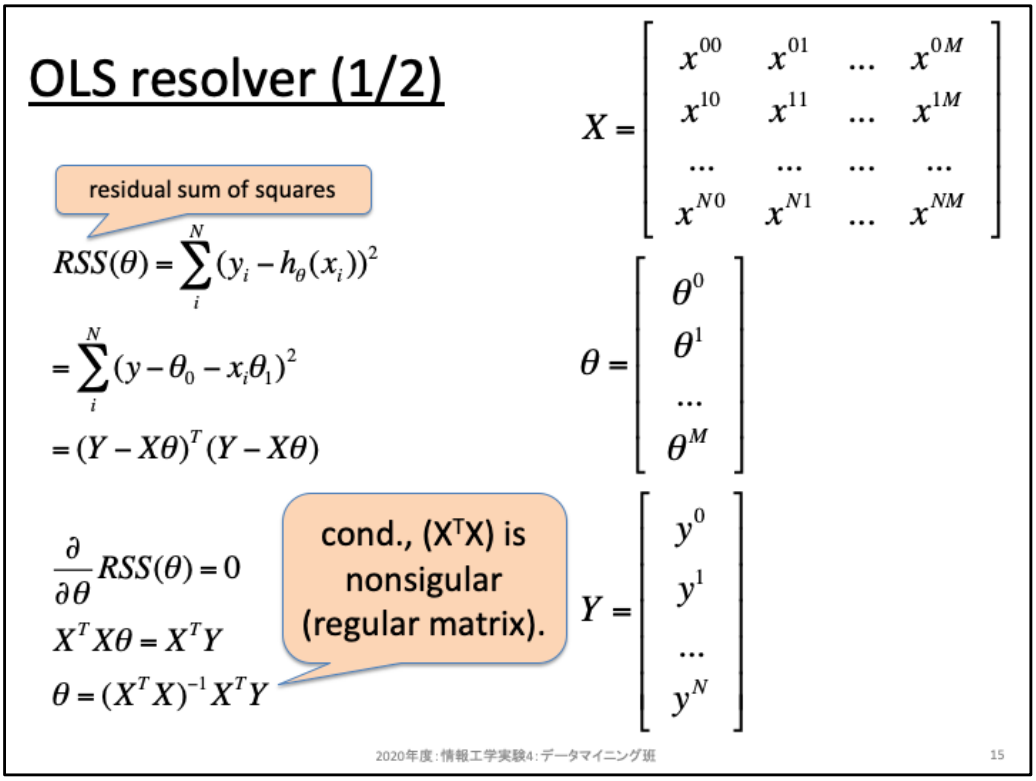
https://inst.eecs.berkeley.edu/~ee127a/book/login/l_ols_main.html

最小二乗法とは、このように実測値（サンプル）と予測値（モデル）との2乗誤差を小さくすることを目的関数とするため、説明変数が1次元ならばこのように「誤差の2乗＝面積」となり、面積の総和が小さくなるようにモデルを構築することを目指していることと同等である。

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



一般化するため、各サンプルxをM次元の特徴ベクトルとする。スライドで0～M個の特徴が並んでおりM+1個に見えるが、0番目のxはバイアス項として利用するための定数（全てのサンプルで1）とする。そのため、x00～x0Mがサンプル0番目の特徴ベクトルである。サンプルがN+1個用意されているものとし、これを行列Xとする。なお、サンプル数を特定していれば、サンプル数はN個（0～N-1なり、1～Nなり）で良い。ここでは単に特徴ベクトルの0～Mという表記に合わせただけである。

バイアス項θ0を含め、各特徴量に対するパラメータθをM+1個の縦ベクトルとして用意する。
サンプル数分の教師データyについても縦ベクトルYとして用意する。

行列X、ベクトルθ、ベクトルYを用いて残差平方和を書き直すとスライド左上のようになる。これを偏微分により0となる方程式に書き直し、θについて展開すると左下のようになる。従って、OLSではこの行列計算を求めることで最適なパラメータを求めることが可能だ。しかし前提として、(X^T・X)の逆行列が存在することが条件である点に注意を要する。

式の展開過程をより詳細に確認してみよう。

OLS resolver (2/2)

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

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

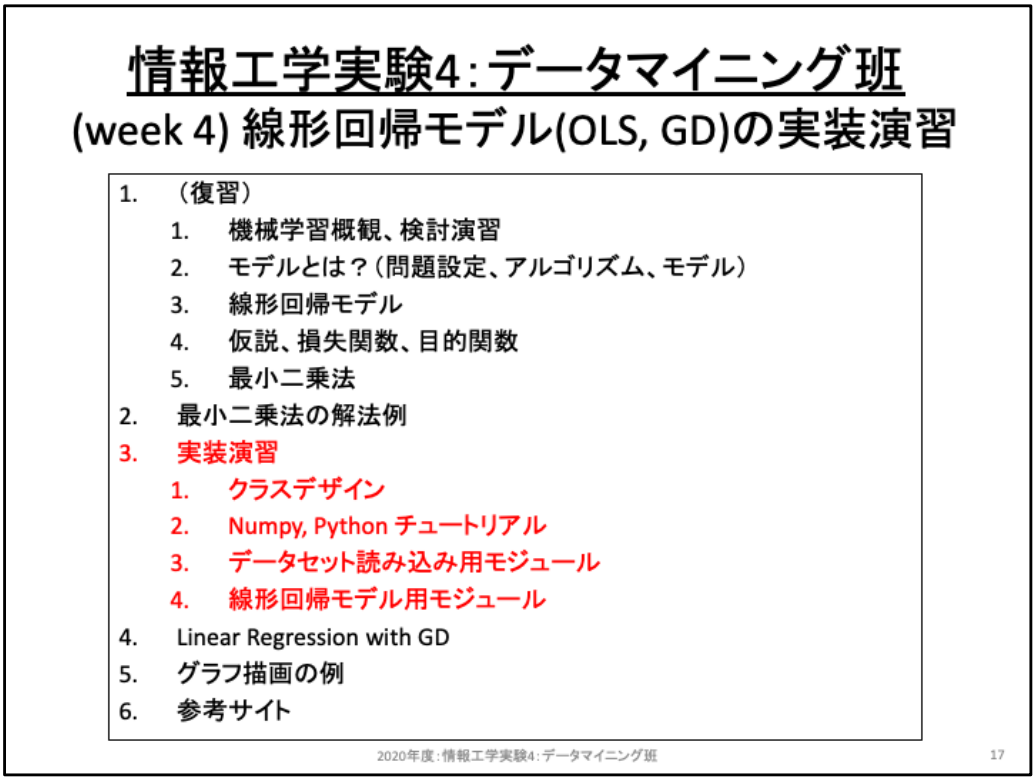
$$\frac{\partial\theta^TX^TY}{\partial\theta}=X^TY$$
$$\frac{\partial Y^TX\theta}{\partial\theta}=(Y^TX)^T=X^TY$$
$$\frac{\partial\theta^TX^TX\theta}{\partial\theta}=\frac{\partial\theta^T(X^TX\theta)}{\partial\theta}+\frac{\partial(\theta^TX^TX)\theta}{\partial\theta}=X^TX\theta+(\theta^TX^TX)^T=2X^TX\theta$$
$$\frac{\partial}{\partial\theta}RSS(\theta)=-2X^TY+2X^TX\theta=0$$
$$X^TX\theta=X^TY$$
$$\theta=(X^TX)^{-1}X^TY$$

2020年度：情報工学実験4：データマイニング班 16

スライド上部が残差平方和を行列・ベクトル演算で書き直した内容である。特に右から2つ目の式が右端のようになることについて、数次元・数サンプルの行列・ベクトルを実際に書いて確認してみよう。置き直すことができれば、2行目の展開はそのまま行列演算を書き下しただけである。この4つの項からなる残差平方和について偏微分が0となるθを求めていくことになる。なお、行列演算では乗算順序に意味がある点に注意を要する。ここではオレンジ背景に示した公式を用いて求めていくこととする。

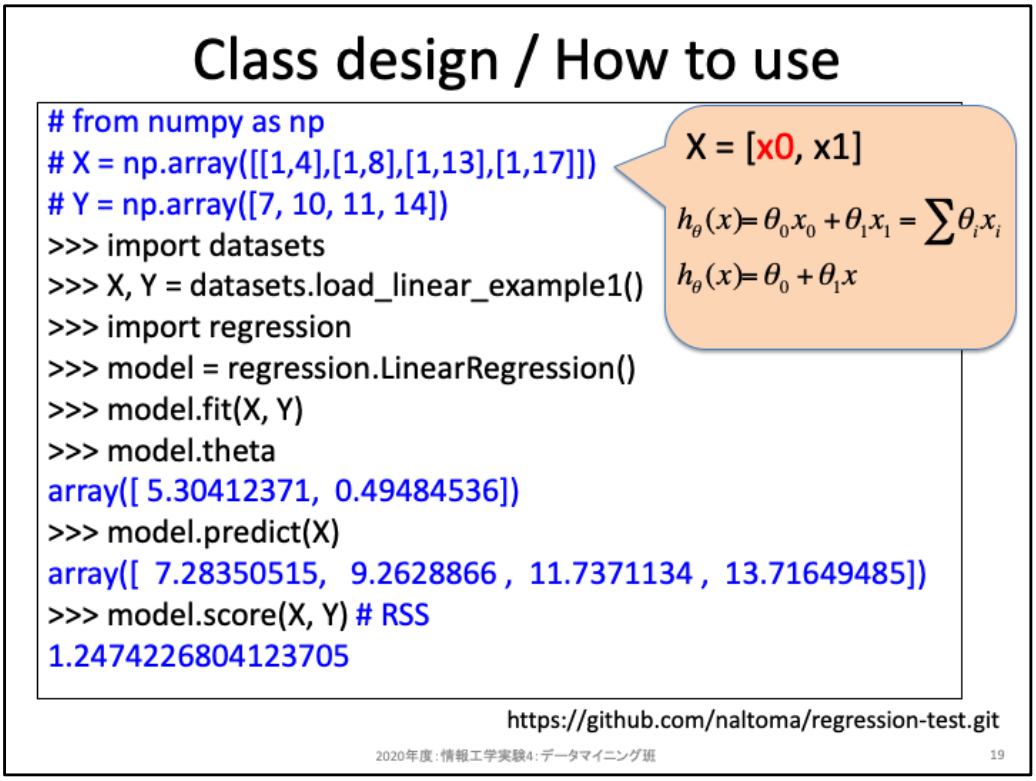
項目1番目について。これはθがそもそも乗算されていない定数項であるため、偏微分により削除されてしまう。
項目2番目について。これは微分するシータの転地が左から掛けられている。このような場合にはオレンジ背景の中央より、定数項がそのまま残る。
項目3番目について。これは微分するシータが右から掛けられている。このような場合にはオレンジ背景の右端を利用すると、残った項の転置行列が残る。
項目4番目について。これはθが両側から掛けられており、部分積分に分解しよう。分解後はこれまでに利用した公式を当てはめるだけとなり、求めることができる。

最後に、項目1～項目4の総和が0となるよう方程式をたて、θについて展開し直したのが最後の3行である。



OLSでは行列演算によりθを求めることができることを確認した。これを実装してみよう。

Implementation of OLS resolver



実装する際に、入力X、出力Yを np.array 形式で用意するものとする。これは scikit-learn を模倣した設計だ。実際のデータセットは datasets モジュールで用意するものとし、上記コード2行によりX, Yを受け取る。x0がバイアス用の定数(=1)なのは前述のとおりだ。そのため、例えばサンプル1番目の [1, 4] は、実際には1次元の特徴4だけが本来の入力であり、1はバイアス用の入力だ。この点は scikit-learn とは異なっている。Scikit-learnに限らず、このようなバイアス用の定数項はモデル側で準備してあるため、本来は特徴ベクトルに含める必要がない。今回は確認しやすくするためだけに、このような設計をしている。

その後、線形回帰モデルを regression モジュールで用意しておき、モデルを用意したあとで model.fit により適応(学習)させる。このfit関数の中身が先程求めた行列演算になる。また、学習で得られたパラメータは model.theta に保存されるものとしよう。学習後は model.predict により予測する個が可能であり、model.score で残差平方和によるスコアを返すようにしよう。

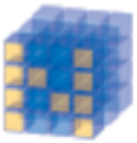
このように、システムをゼロから作る際にはどのように利用するかという視点から設計図を考えてみると良い。今回はこの設計図を元に、実装していこう。

その前に、便利なモジュールである numpy について軽く紹介する。

Linear Algebra with NumPy

http://www.numpy.org

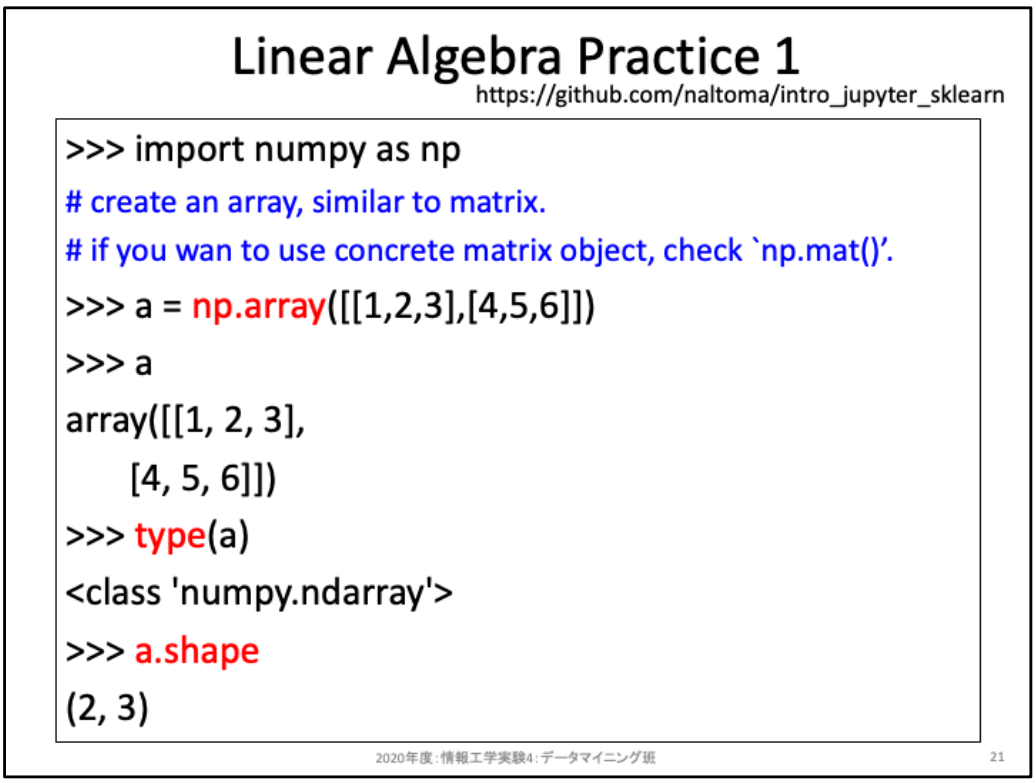
- import numpy as np
- help(np)
  - Provides
    - 1. An array object of arbitrary homogeneous items
    - 2. Fast mathematical operations over arrays
    - 3. Linear Algebra, Fourier Transforms, Random Number Generation



NumPy

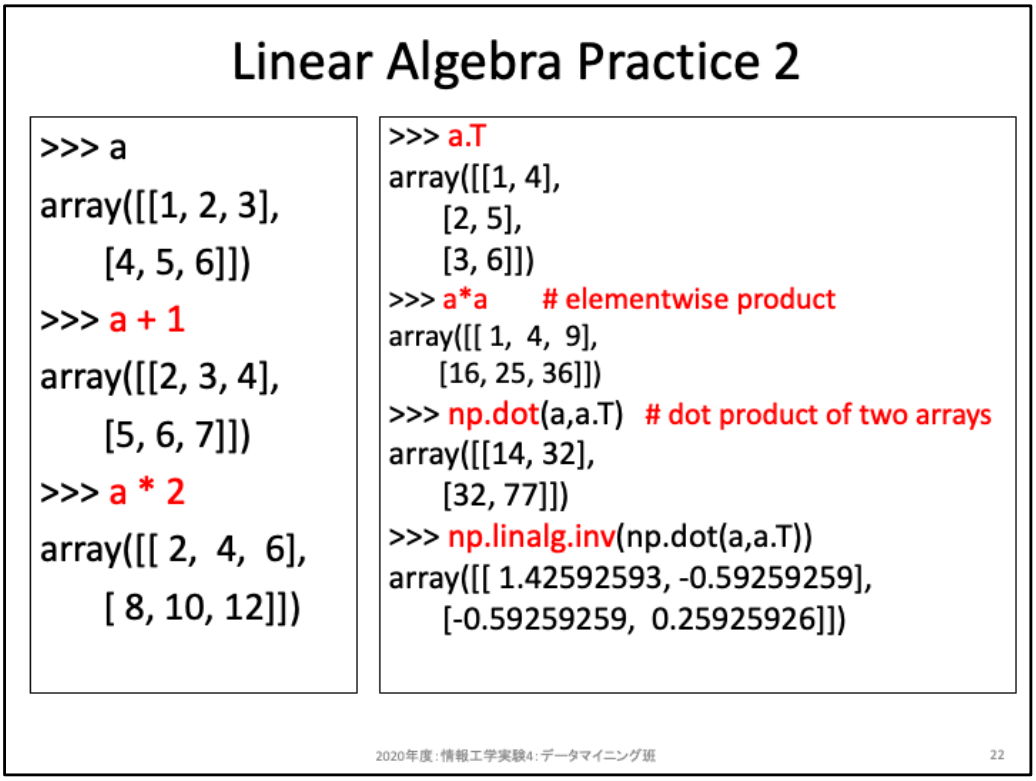
Base N-dimensional array package

行列演算にはnumpyモジュールを使おう。各種ライブラリの多くはC言語で書かれており、numpyはそれらをwrapperとして利用している。極めて高速に処理可能だ。慣例的に np というエイリアス（別称）を付けて利用する。

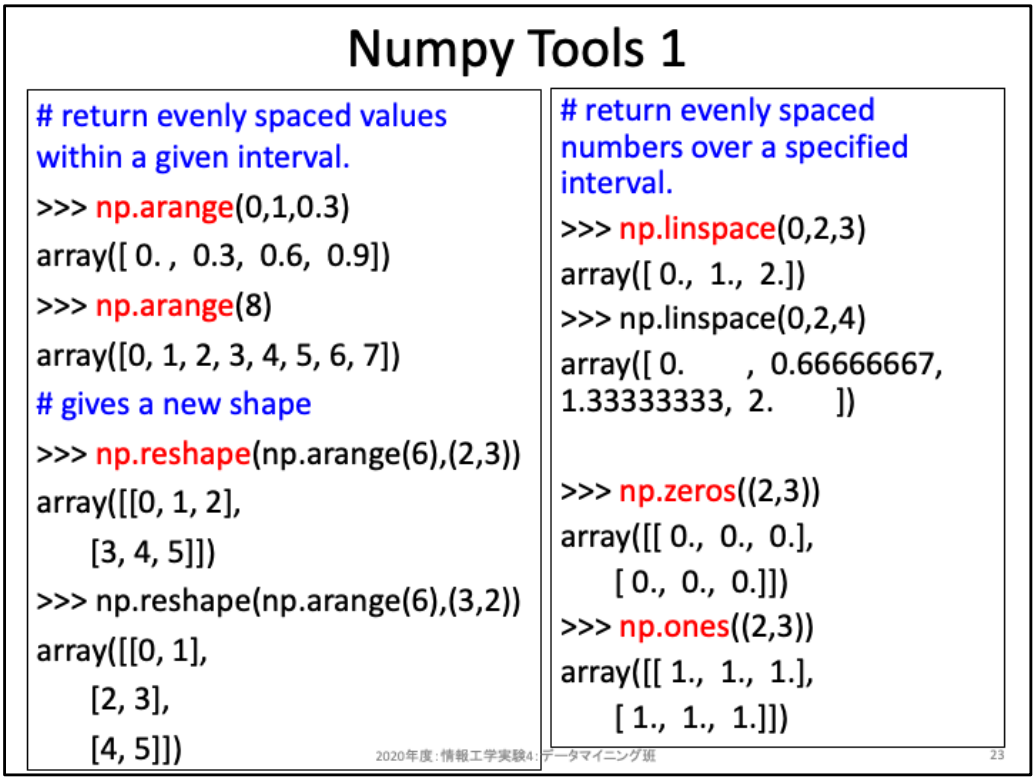


np.arrayは厳密には行列ではない。厳密に行列を扱いたいのであれば np.matrix があるが、機械学習モジュールではより柔軟に扱える np.array を利用することが多い。

ここでは [1, 2, 3] という1行が1つのサンプルを表しており、3次元ベクトルとして記載している。これを2行分（2サンプル分）の行列として定義した例である。データを読み込んだ際に行と列が逆になっていないか、確認できるようにしよう。



行列に直接スカラー演算を行うと、要素内演算が行われる。
行列演算をしたいなら、スライド右のように専用の演算子もしくは関数を使う必要がある。
np.linalg.inv は逆行列を求める関数である。

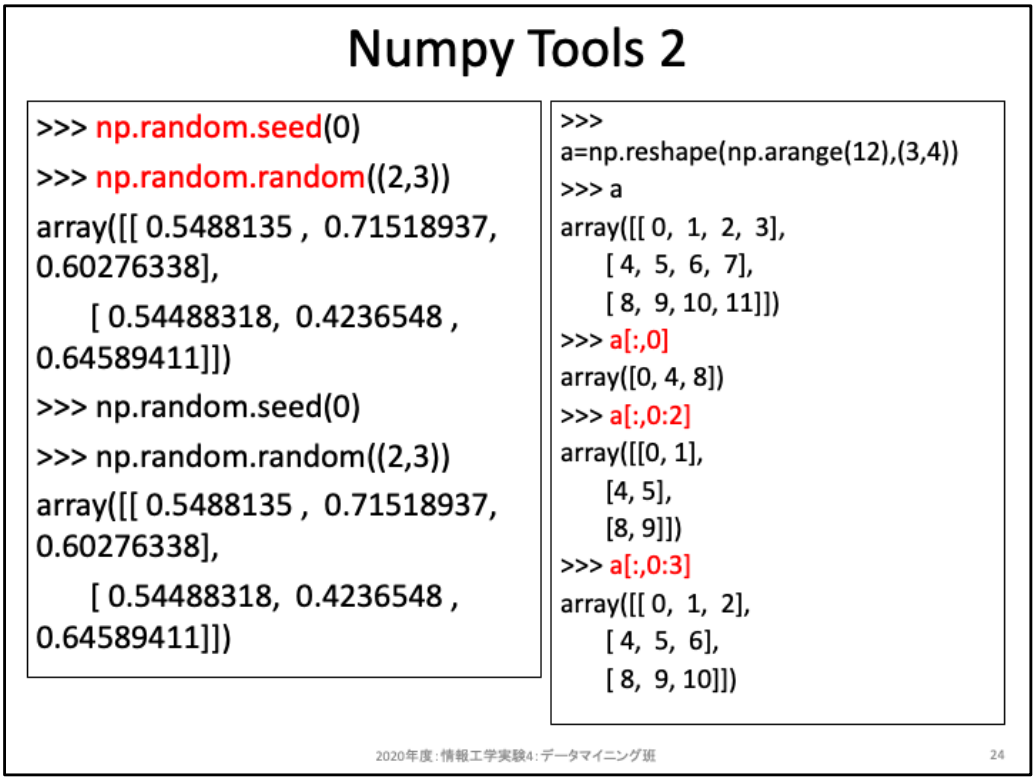


Np.arangeはstartからstopまでをstep毎に区切ったサンプルを用意する。startを省略すると0となり、stepを省略すると1となるため、rangeと同等である。

Np.reshapeは行列の要素数を買えずに、行列の形だけを変更する関数である。

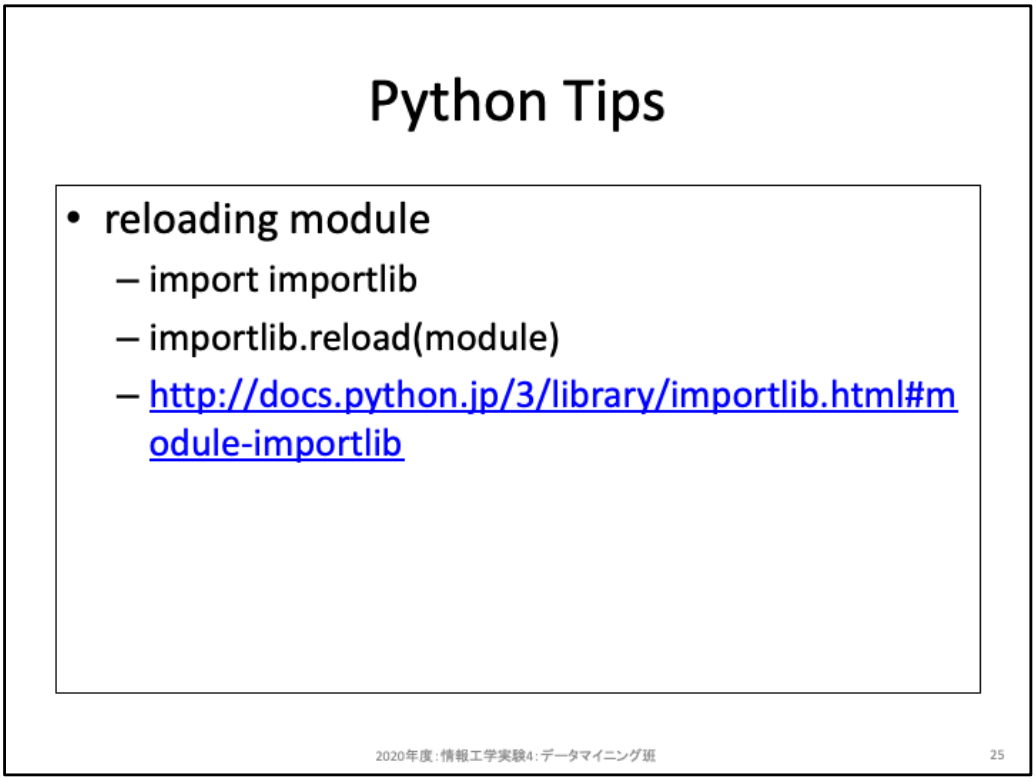
Np.linspaceはstartからstopまでをnum個に等分割したサンプルを用意する関数である。

Np.zerosはゼロ行列、onesは1行列を生成する際に用いる。



Np.randomは乱数を扱う際に利用する。擬似乱数列を固定したい場合にはseedを固定しよう。

スライド右は行列の列を指定して要素を取り出す例である。
a[:,0] は0列を取り出している。
a[:,0:2] は0列～1列を取り出している。



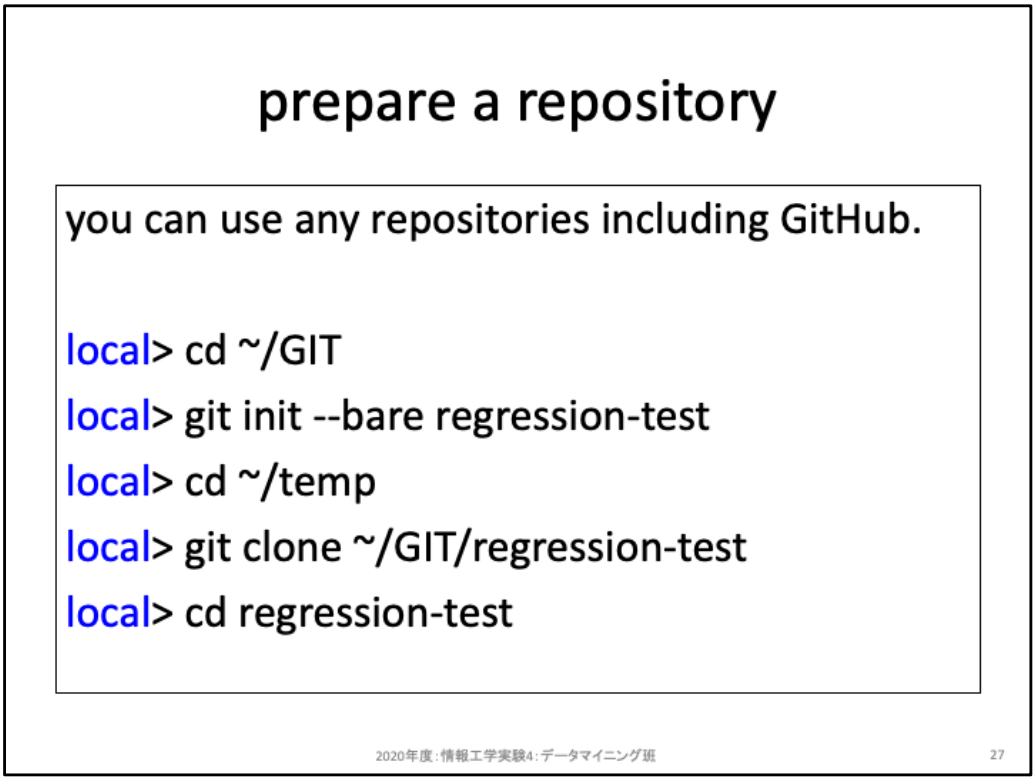
IDEで実行するなら問題ないが、Pythonインタプリタ上で実行しながらコード編集をしていると、「編集し直したはずのモジュールが読み込まれない」状況に陥る。これはPython 3 が過去に読み込んだ同じ名前のモジュールを再読み込みするのは無駄だという判断を自動で行うため、import module_name と改めて記述しても実際にはスキップされて読み込まれない。これを強制的に再読込みさせたいのであれば、上記の importlib.reload を使おう。

[reprint] Class design / How to use

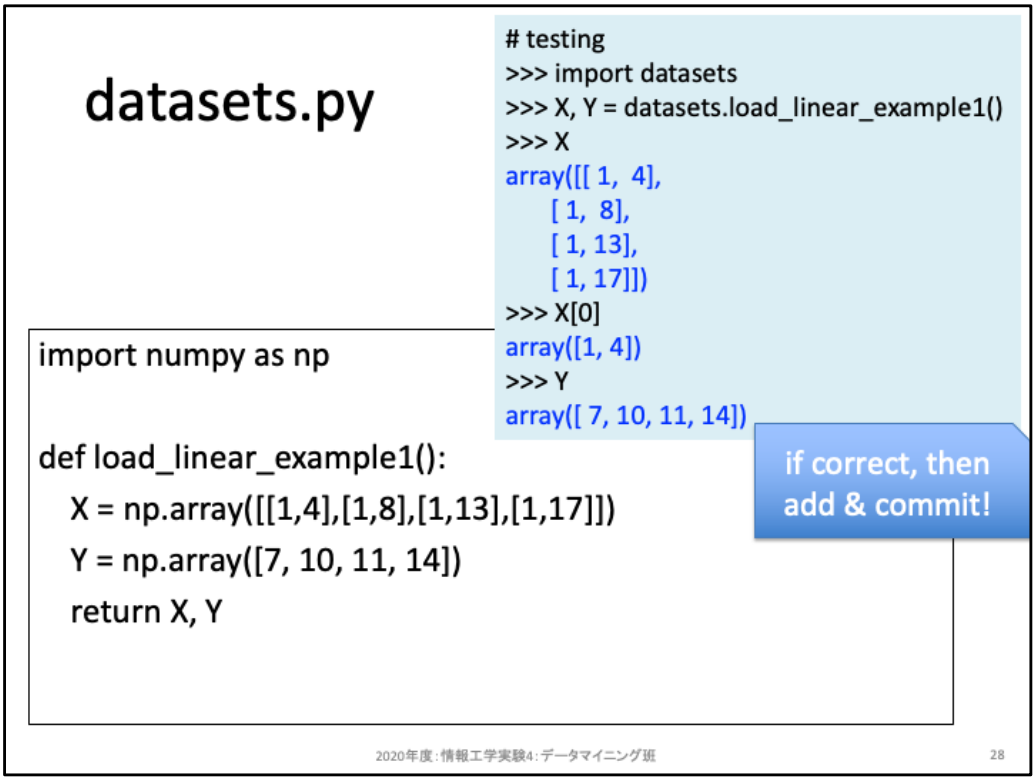
```
# from numpy as np
# X = np.array([[1,4],[1,8],[1,13],[1,17]])
# Y = np.array([7, 10, 11, 14])
>>> import datasets
>>> X, Y = datasets.load_linear_example1()
>>> import regression
>>> model = regression.LinearRegression()
>>> model.fit(X, Y)
>>> model.theta
array([ 5.30412371,  0.49484536])
>>> model.predict(X)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

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

改めて設計図を示す。これを元に実装していこう。



バージョン管理の練習を兼ねてやろう。スライド上では ~/GIT/regression-test をベアリポジトリとする例である。GitHubで用意したい場合にはそれでも構わない。



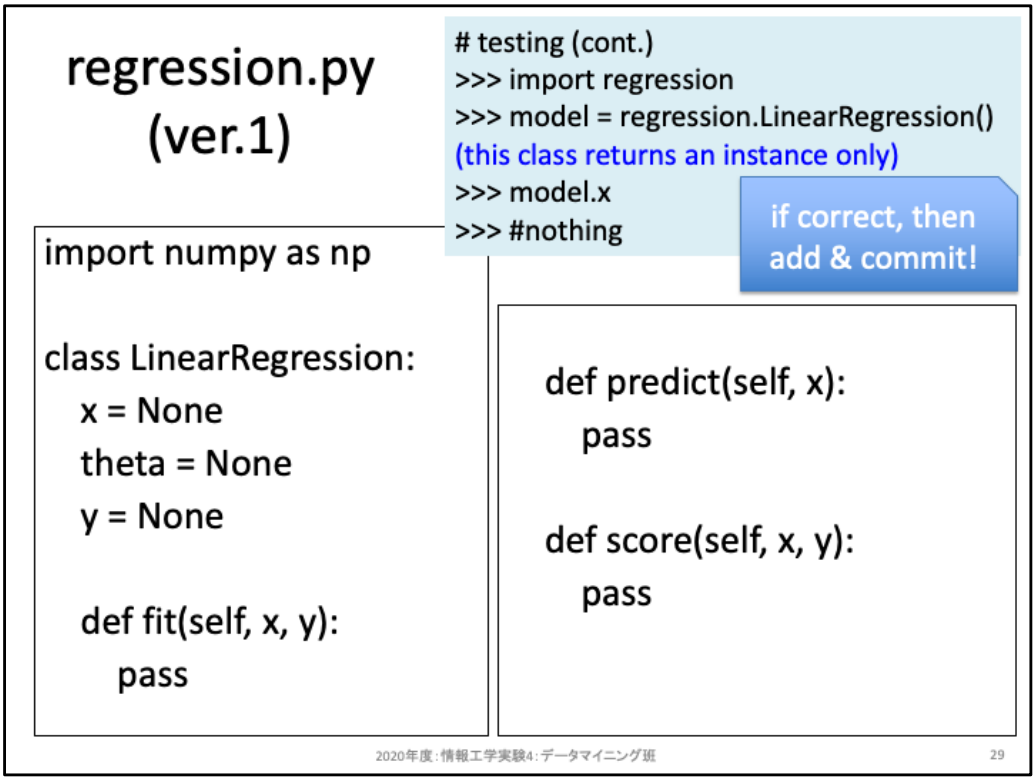
まず datasets.py を実装しよう。関数本体は3行だけで記述できる。この際、テストをdoctest, pytest 等の単体テストで記載すること。テスト駆動開発にするか、関数実装後にテストを記述して動作確認するかはどちらでも構わない。テスト駆動開発で取り組むなら、SyntaxErrorを避けるため次のように pass 文を使うとよいだろう。

```
# doctestの記述例
def load_linear_example1():
    """
    >>> X[0]
    array([1, 4])
    """
    pass
```

passは、何も実行せずに次の行に移動するだけの命令文だが、関数定義やifブロック等のようなインデントが必要となる状況下において何も命令文を記述しないと、それ自体が文法エラーとなってしまいテストを実行できない。これを避けるため、passを利用している。他にも「明示的に if文 の False ブロックは処理が存在しない」ことを示すために使うこともある。

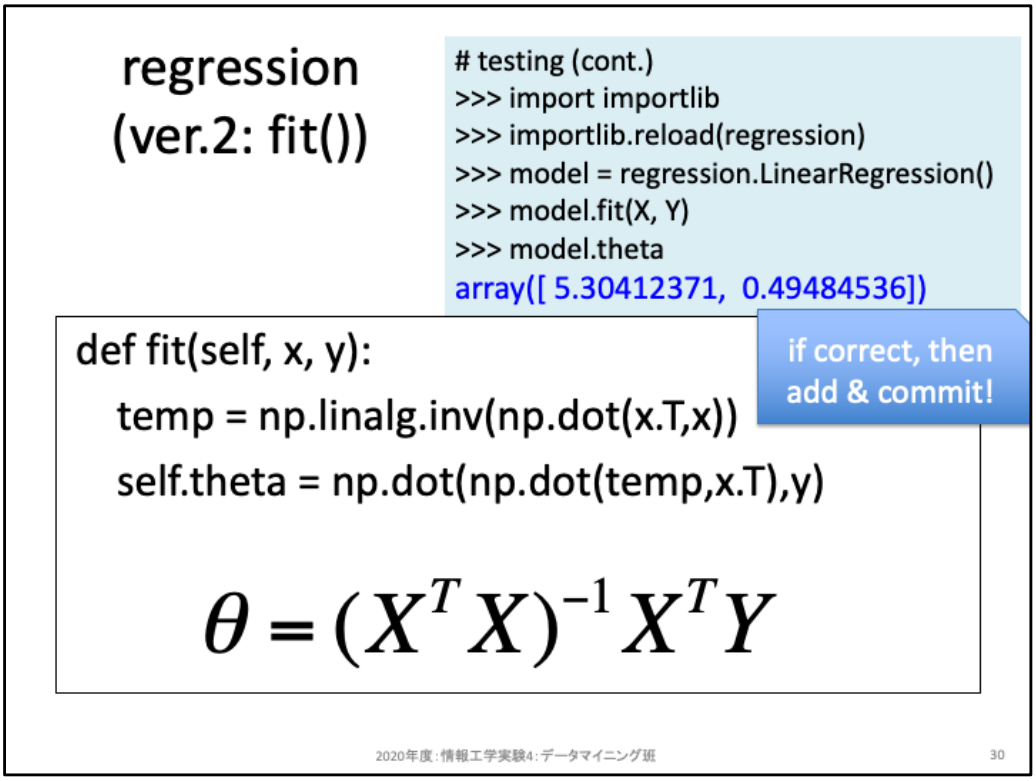
テストを記述したら、そのテストが通るようにコードを編集していこう。テストの動作方法は各ライブラリのドキュメントを参照しよう。IDEならテスト用の実行設定が必要だし、ターミナル上から実行するならオプション指定が必要になることが多い。

テストにより動作確認を終えたら、git add, git commit, git pushしよう。このように動作確認できる都度pushしていくことにしよう。



regression.pyを実装していこう。最初はfit, predict, score関数は全て pass としておき、モデルに与えられる入力・出力と、モデルで用意すべきthetaを変数として宣言（初期値None）としているだけだ。このように、まずは外側だけを用意して文法上は動作するようにしよう。

ここではテストは書かず、右上にあるように import して変数が用意されている（中身がない）ことを確認できれば、pushしよう。



fit関数の実装である。行列演算で求められることが分かったので、npモジュールを利用して演算しよう。なお、np.linalg.invによる逆行列が求められない際のエラー処理はここでは省略している。また、1行にまとめて書いてもよいが、長くなり、見通しが悪くなるため2行に分けて記載している。

水色背景のテストが通ることを確認せよ。確認できたらpushまでやろう。

regression (ver.3: predict())

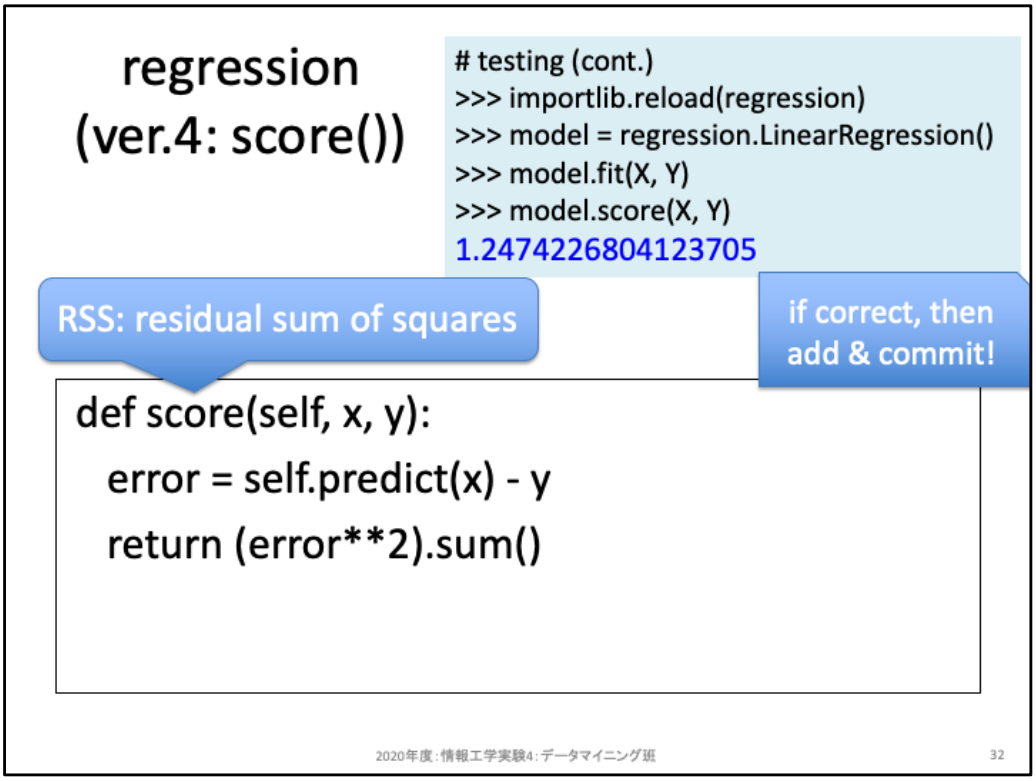
```
# testing (cont.)
>>> importlib.reload(regression)
>>> model = regression.LinearRegression()
>>> model.fit(X, Y)
>>> model.predict(X)
array([  7.28350515,   9.2628866 ,  11.7371134 ,  13.71649485])
```

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

if correct, then add & commit!

predict関数は、fit関数により求めたパラメータself.thetaを用いて予測する（行列演算する）だけであり、直接return文の中で処理している。

水色背景のテストが通ることを確認せよ。確認できたらpushまでやろう。



score関数は残差平方和により求める。サンプル数をfor分でループさせ、1個ずつ予測させて残渣を求め、、、という書き方でも実現できるが、npであれば上記のようにサンプル数分まとめて予測させ、そのままベクトル演算として残渣を求めることができる。実行速度は圧倒的にこちらが早い（ループ文は遅い）ので、npによる行列・ベクトル処理の仕方を学ぼう。

水色背景のテストが通ることを確認せよ。確認できたらpushまでやろう。

[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

以上の実装により、設計図に基づいた動作確認をし終えたはずだ。

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

線形回帰モデルをGD（最急降下法）で解く際の流れを再確認してみよう。

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
```

2020年度：情報工学実験4：データマイニング班 37

このスライドはθをアップデートする際の注意点を示している。パラメータは2つのみであり、θ0はバイアス項、θ1は入力xに対する重みパラメータである。

左下のようにパラメータの更新はまとめてやること。右下のように「θ0を更新後、そのθ0を用いてθ1について更新処理する」というやり方では、θ1更新前にθ0が移動しており、その影響が混ざるため不適切である。（実際にどのように影響を受けるのかは可視化して確認してみよう）

Partial differentiation with theta 0 (intercept)

$$\begin{aligned}
\frac{\partial}{\partial\theta_0}J(\theta_0,\theta_1) &= \frac{\partial}{\partial\theta_0}\frac{1}{2m}\sum_{i=1}^{m}(h_\Theta(x^i)-y^i)^2 \\
&= \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x^i)-y^i)\frac{\partial}{\partial\theta_0}(h_\Theta(x^i)-y^i) \\
&= \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x^i)-y^i)\frac{\partial}{\partial\theta_0}(\theta_0x_0+\theta_1x_1-y^i) \\
&= \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x^i)-y^i)x_0 \\
&= \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x^i)-y^i)
\end{aligned}$$

バイアス項θ0は、1次方程式 y = ax + b における切片b（intercept）に相当する。θ0について偏微分をする際にはθ1を定数として扱う。このため展開途中でθ0が掛かっていない項は削除され、右辺はx0だけが残る。ここでx0は定数項1であったため、最終的には残差だけが残る。

Partial differentiation with theta 1 (slope)

$$\begin{aligned}\frac{\partial}{\partial\theta_1}J(\theta_0,\theta_1) &= \frac{\partial}{\partial\theta_1}\frac{1}{2m}\sum_{i=1}^{m}(h_\Theta(x^i)-y^i)^2 \\ &= \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x^i)-y^i)\frac{\partial}{\partial\theta_1}(h_\Theta(x^i)-y^i) \\ &= \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x^i)-y^i)\frac{\partial}{\partial\theta_1}(\theta_0x_0+\theta_1x_1-y^i) \\ &= \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x^i)-y^i)x_1\end{aligned}$$

θ1は入力xに対する重みパラメータであり、1次方程式 y = ax + b における傾きaに相当する。θ1について偏微分をする際にはθ0を定数として扱う。このためθ1が掛かっていない項は削除され、最終的には残差に入力を掛けるだけとなる。

Updating formula on GD

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

$$Therefore,$$

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

θ0,θ1の偏微分値について導出した。θ0については残差のみ、θ1については残差に入力xを掛ける形になっていた。Θ0についてもx0を1としておけば、θ1と同様に残差に入力xを掛ける形となり、同じ形になっている。これをm次元で一般化すると上記スライドのようになり、偏微分値は残差に入力を掛け、パラメータの更新式は偏微分値に移動幅を調整するハイパーパラメータαを掛けて逆方向に調整することになる。

これでGDにおける更新式も導出できた。これをどのように実装するかは各自取り組んでみよう。参考までに実装例を以下に示す。

https://github.com/naltoma/regression-test/blob/master/regression_gd.py

Linear Regression using Gradient Descent
https://towardsdatascience.com/linear-regression-using-gradient-descent-97a6c8700931

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

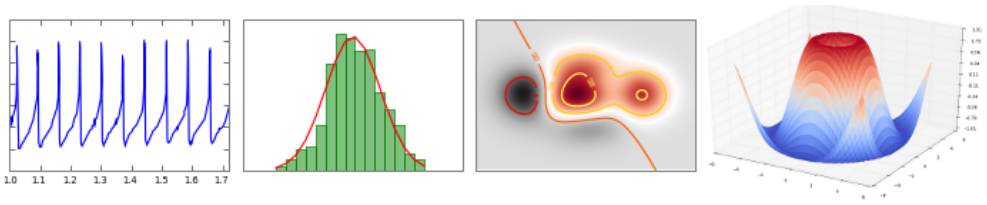
1. （復習）機械学習概観、検討演習
2. （復習）モデルとは？（問題設定、アルゴリズム、モデル）
3. （復習）線形回帰モデル
4. （復習）仮説、損失関数、目的関数
5. （復習）最小二乗法
6. 最小二乗法の解法例
7. 実装演習
    1. クラスデザイン
    2. Numpy, Python チュートリアル
    3. データセット読み込み用モジュール
    4. 線形回帰モデル用モジュール
8. グラフ描画の例
9. 参考サイト

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

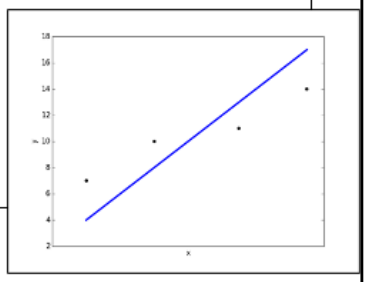
matplotlib



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()
```

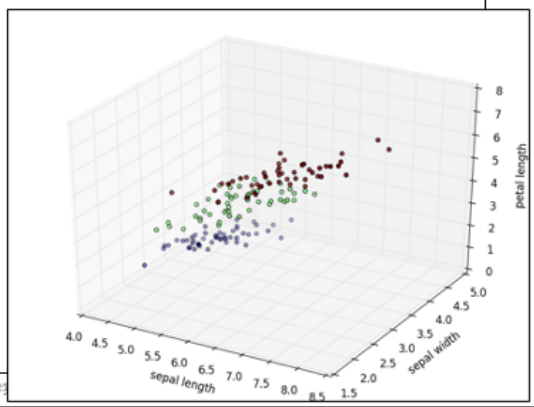


Matplotlibによる基本的な描画コードを例示している。直線や曲線を描画するためには多数のサンプル点を用意し、各サンプル点における具体的な値を求め、それらの点と点の間を線グラフとして結ぶ形で描画される。このため曲線の場合にはnp.linspace あたりでサンプルを用意すると楽に描画できるだろう。

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()
```



これはiris flower datasetの特徴量1次元目～3次元目までをXYZ軸として用意し、花の種別を色で指定したコード例である。

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