BPStudy#133〜Pythonによる機械学習・データ分析　9/25, 2018

機械学習と数学

辻真吾（@tsjshg）

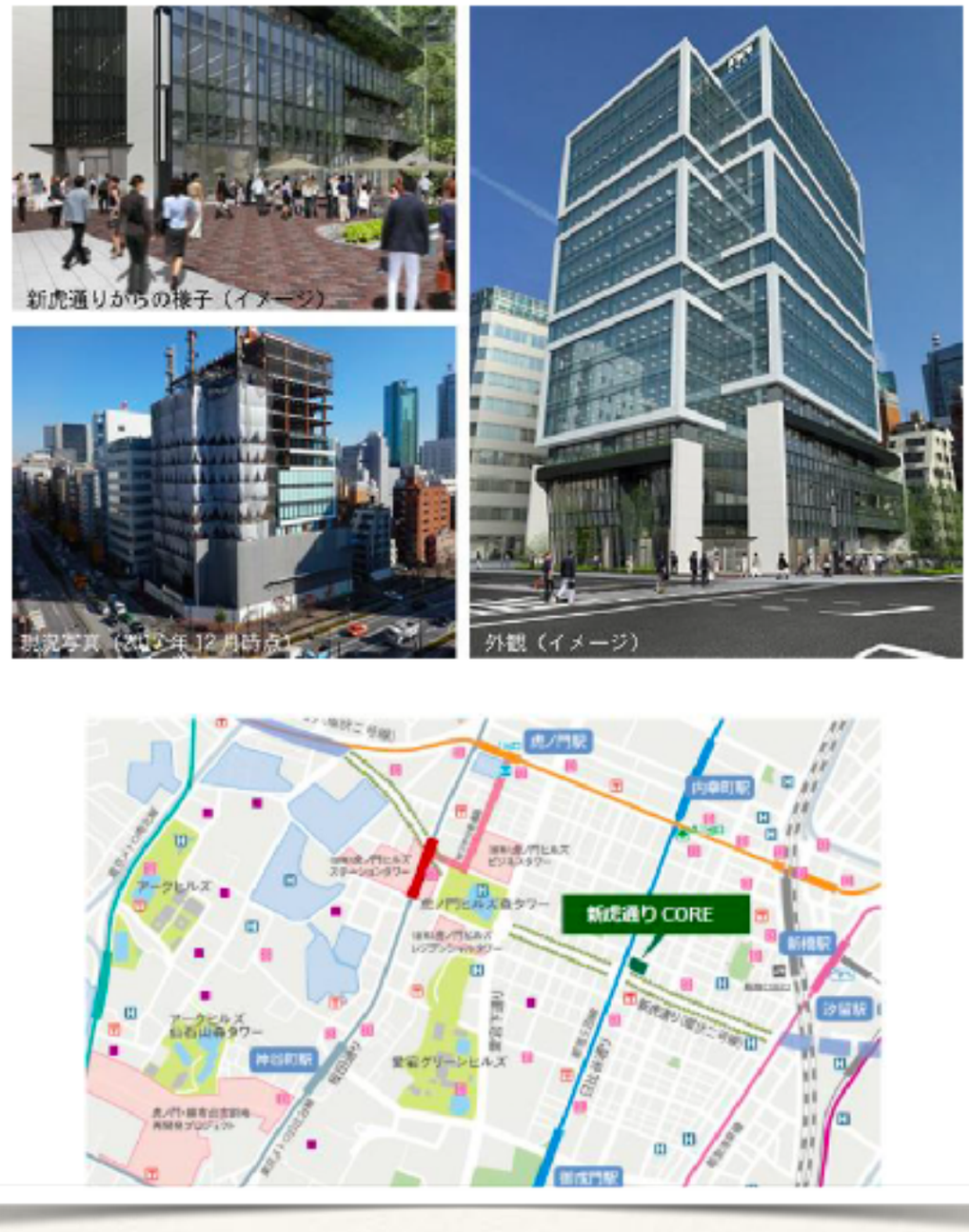
shingo.tsuji@gmail.com

自己紹介

- 辻真吾（TSUJI, Shingo）1975年東京都足立区出身
- 東京大学先端科学技術研究センターゲノムサイエンス分野に所属
  - 修士を出た年に創業した「いい生活」で3年ほど働いてました
    （SolarisにOracleとWebLogic入れて、ServletとJSP書いてました）
- プログラミング言語 LOGO, BASIC, C/C++, Java, そしてPython
- Start Python Clubというコミュニティをやっています

Start Python Club

- 9/14【Stapy x MUFG共催】
  PythonGlobalMeetupでは、KY氏はじめBeProudのみなさまにお世話になりました
- 「みんなのPython勉強会」次回は11/14（水）
- 近々、PyLadies Tokyoとのコラボも企画中



もくじ

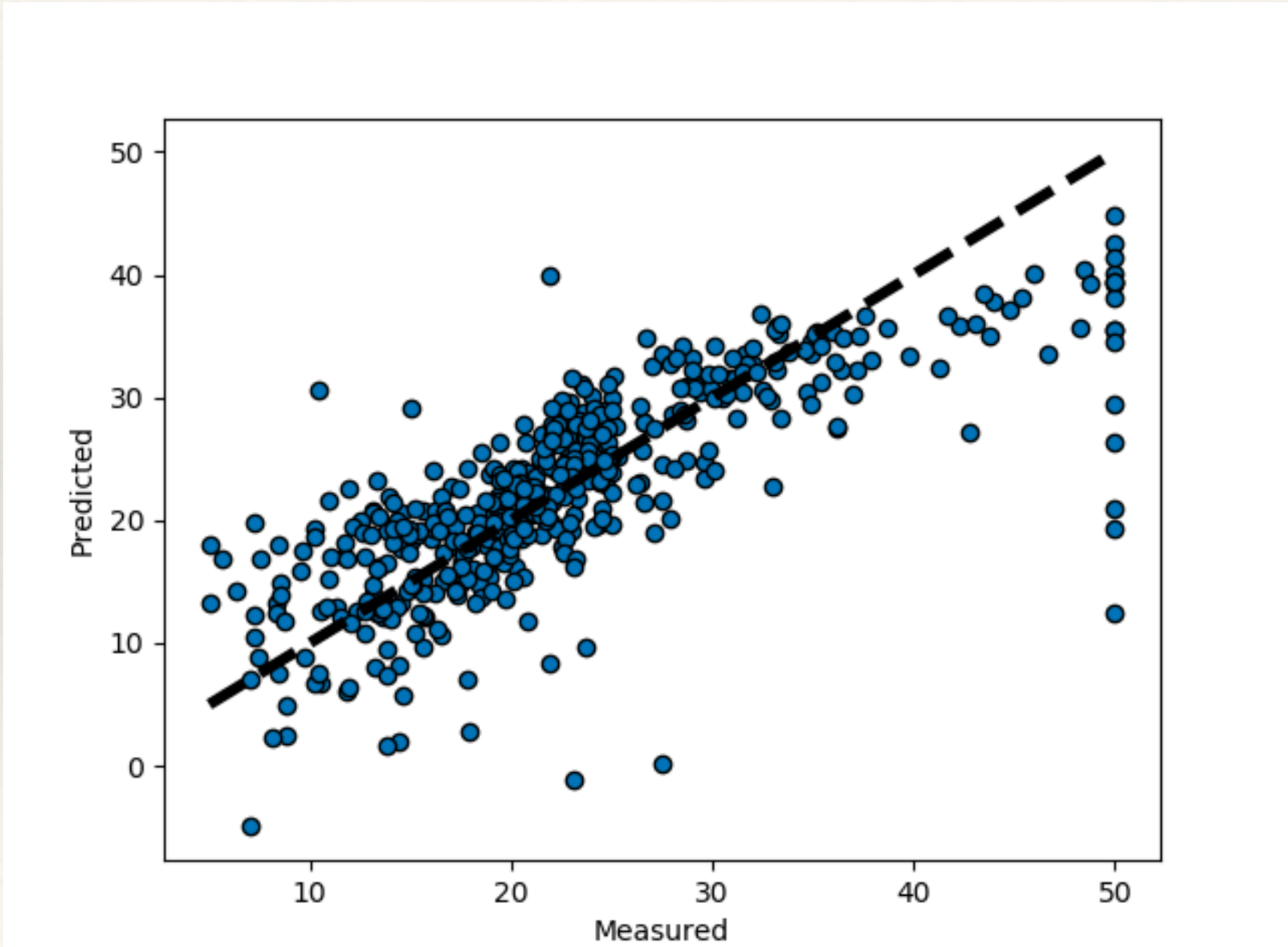
- 数学は好きですが・・・
- 機械学習と数学
  - どこまで必要か？
  - ロジステック回帰を題材に具体例を通じて考える
- どうやって学んでいけばよいか？

どこまで数学が必要か？

- 機械学習アルゴリズムを作る人は絶対必要
- アルゴリズムを実装する人も必要（数値計算の知識も）
  - 今夜はこの2カテゴリーは対象外
- 使う人を想定します
  - 数学の何をどこまで理解するべきか？
  - そもそも理解する必要があるのか？

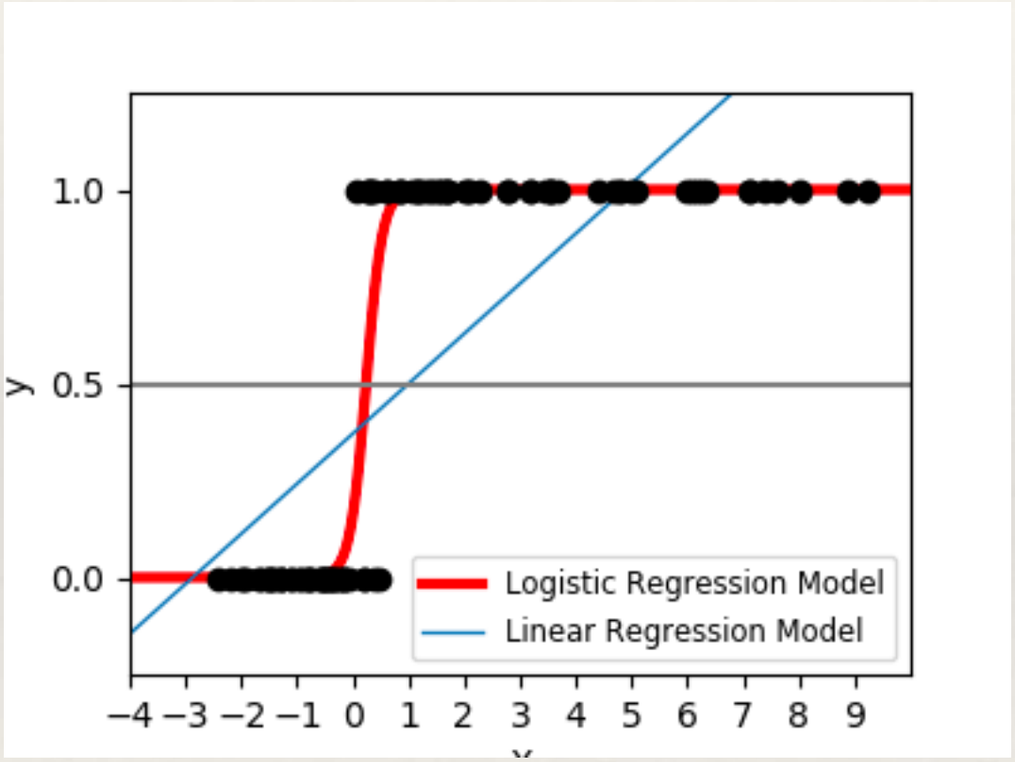
単回帰（regression）

$$y = ax + b$$



http://scikit-learn.org/stable/auto_examples/plot_cv_predict.html#sphx-glr-auto-examples-plot-cv-predict-py

ロジステック回帰



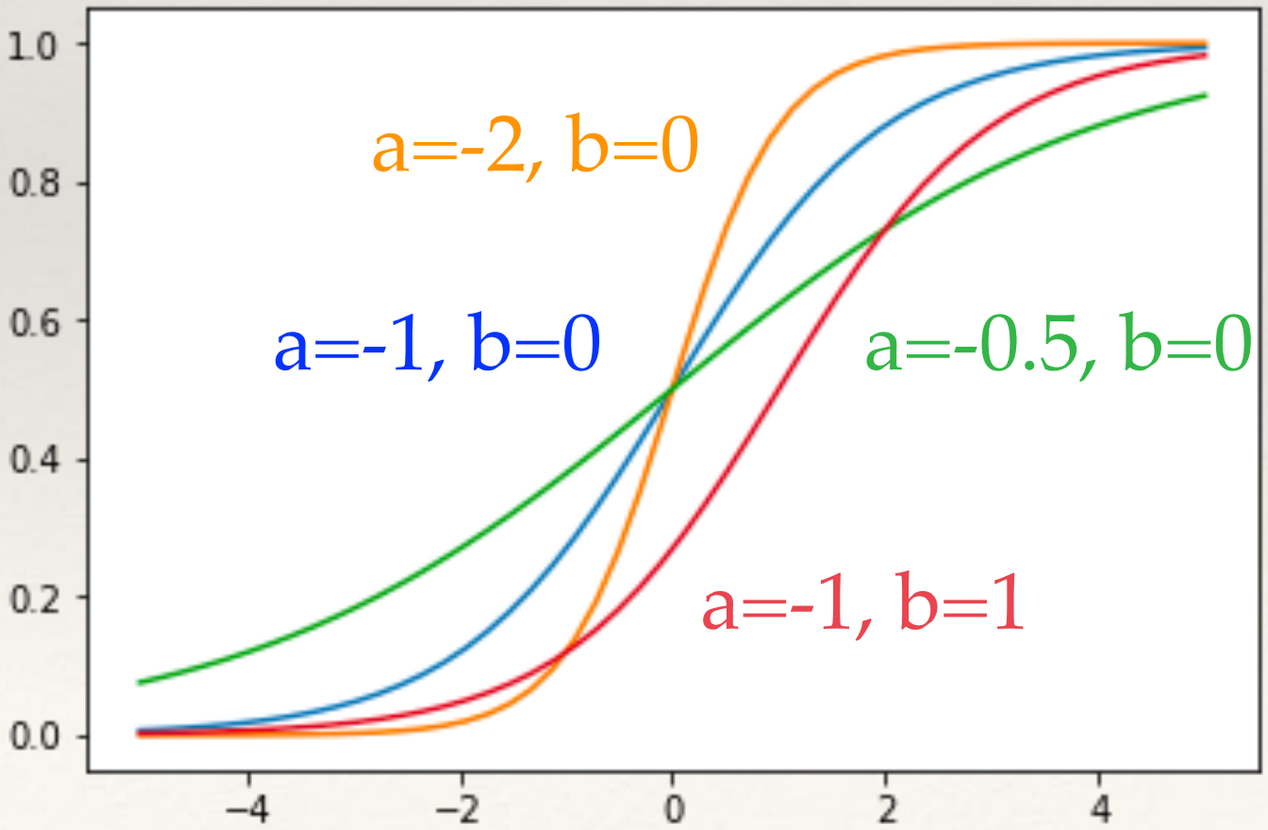
$$y = \frac{1}{1 + e^{ax+b}}$$

シグモイド関数

描いてみればいい

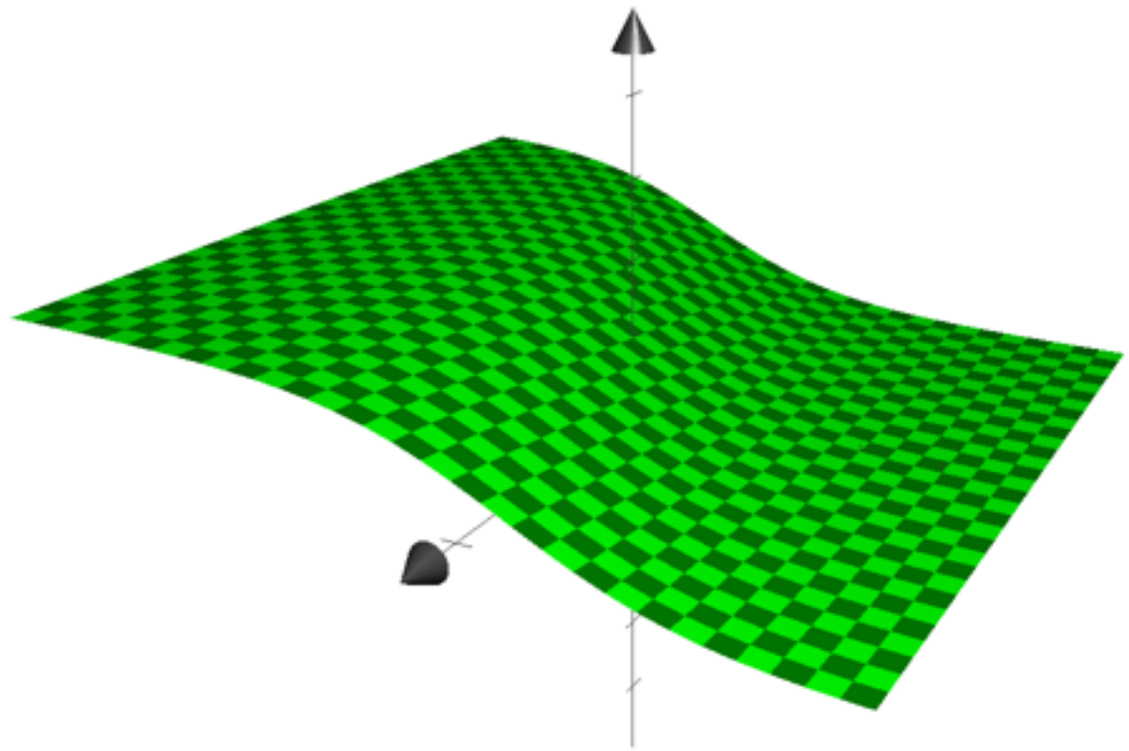
```
1  def sigmoid_f(x, a=-1, b=0):
2      return 1.0 / (1.0 + np.exp(a*x+b))
3
4  x = np.linspace(-5,5)
5  plt.plot(x, sigmoid_f(x))
6  plt.plot(x, sigmoid_f(x, a=-2))
7  plt.plot(x, sigmoid_f(x, a=-0.5))
8  plt.plot(x, sigmoid_f(x, b=1))
```

$$y = \frac{1}{1 + e^{ax+b}}$$



aの絶対値が大きくなると、
変化が急になる。

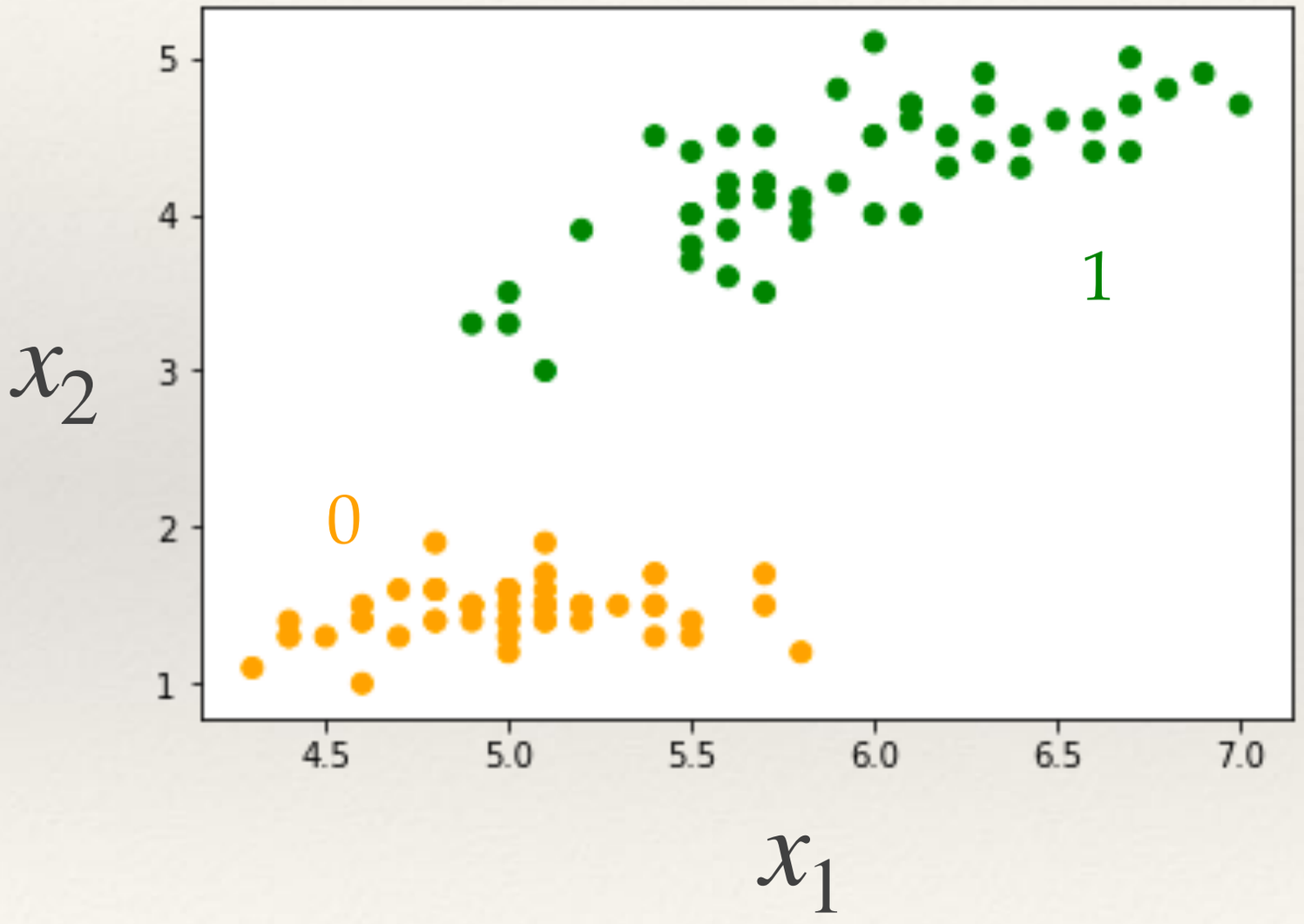
説明変数が2つ



$$y = \frac{1}{1 + e^{-2x_1 - 0.5x_2}}$$

irisのデータ

変数が2つのシグモイド関数をこのデータにあてはめる



#pydatatext 264ページ

scikit-learn

$$y = \frac{1}{1 + e^{-w_1 x_1 - w_2 x_2}}$$

1.1.11. Logistic regression

Logistic regression, despite its name, is a linear model for classification rather than regression. Logistic regression is also known in the literature as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.

The implementation of logistic regression in scikit-learn can be accessed from class LogisticRegression . This implementation can fit binary, One-vs- Rest, or multinomial logistic regression with optional L2 or L1 regularization.

As an optimization problem, binary class L2 penalized logistic regression minimizes the following cost function:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1).$$

Cを大きく設定すると、重みベクトルwのノルムが小さくなる

http://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

ハイパーパラメータCの調整

```
1  from sklearn.linear_model import LogisticRegression
2  clf = LogisticRegression()
3  clf.fit(X,y)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

```
1  clf.coef_
```

```
array([[-1.40943548,  2.98712545]])
```

```
1  from sklearn.linear_model import LogisticRegressionCV
2  clf_cv = LogisticRegressionCV()
3  clf_cv.fit(X,y)
```

```
LogisticRegressionCV(Cs=10, class_weight=None, cv=None, dual=False,
          fit_intercept=True, intercept_scaling=1.0, max_iter=100,
          multi_class='ovr', n_jobs=1, penalty='l2', random_state=None,
          refit=True, scoring=None, solver='lbfgs', tol=0.0001, verbose=0)
```

```
1  clf_cv.coef_
```

```
array([[0.27853061, 1.04682995]])
```

ほとんど同じコードで、全部やってくれる！

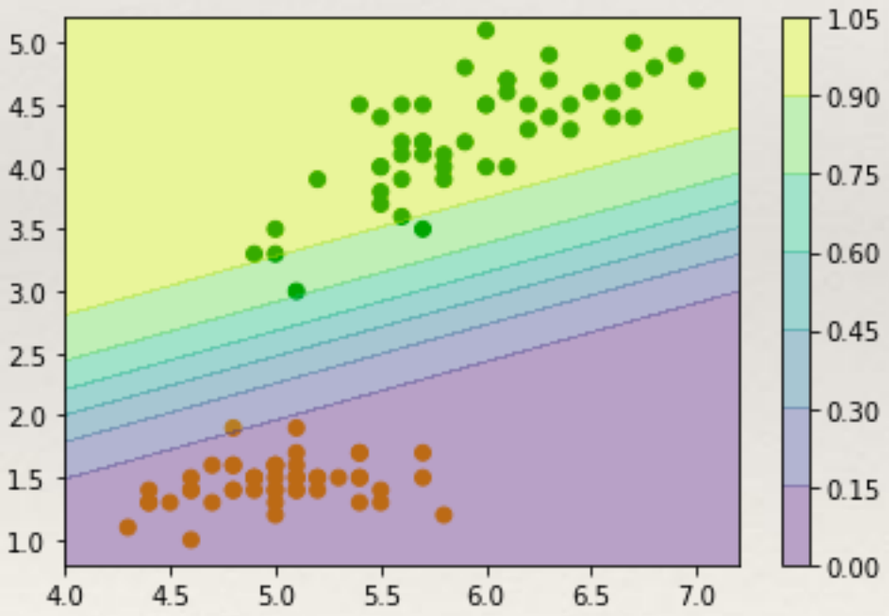
モデルの可視化

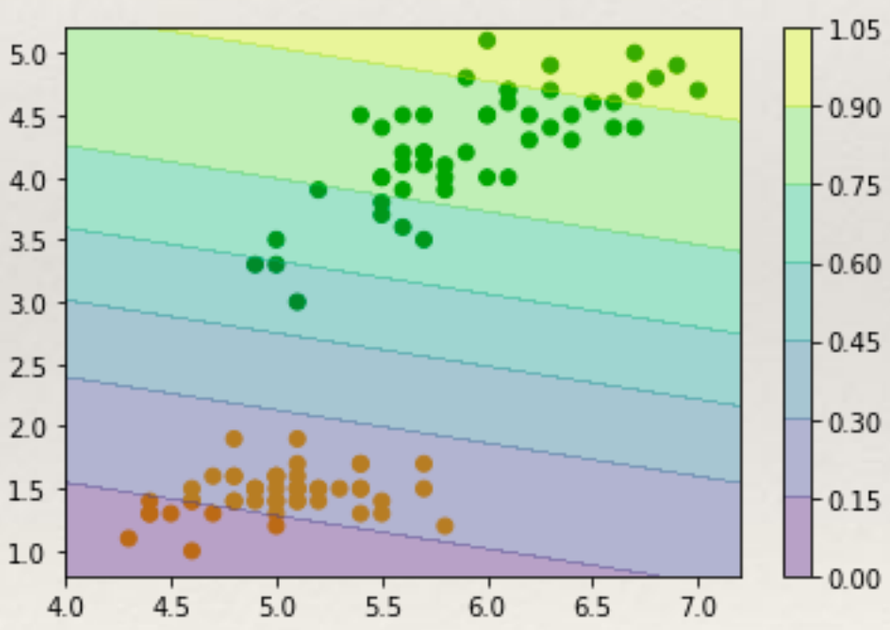
C=1

$$y = \frac{1}{1 + e^{1.41x_1 - 2.99x_2}}$$

C=0.046

$$y = \frac{1}{1 + e^{-0.28x_1 - 1.04x_2}}$$






```
1  plt.scatter(X[:,0], X[:,1], c = ['g' if x else 'orange' for x in y])
2  plt.contourf(xx, yy ,z, alpha=0.4)
3  plt.colorbar()
```

xxやyyの生成にはmeshgridを利用

#pydatatext 117ページ

Cを調節することの意味

- 重みwのノルムが変わる
- 大きくなると急な変化、小さくなると緩やかな変化
- 急な変化は過学習（over fitting）の可能性が高い
- 汎化性能の向上

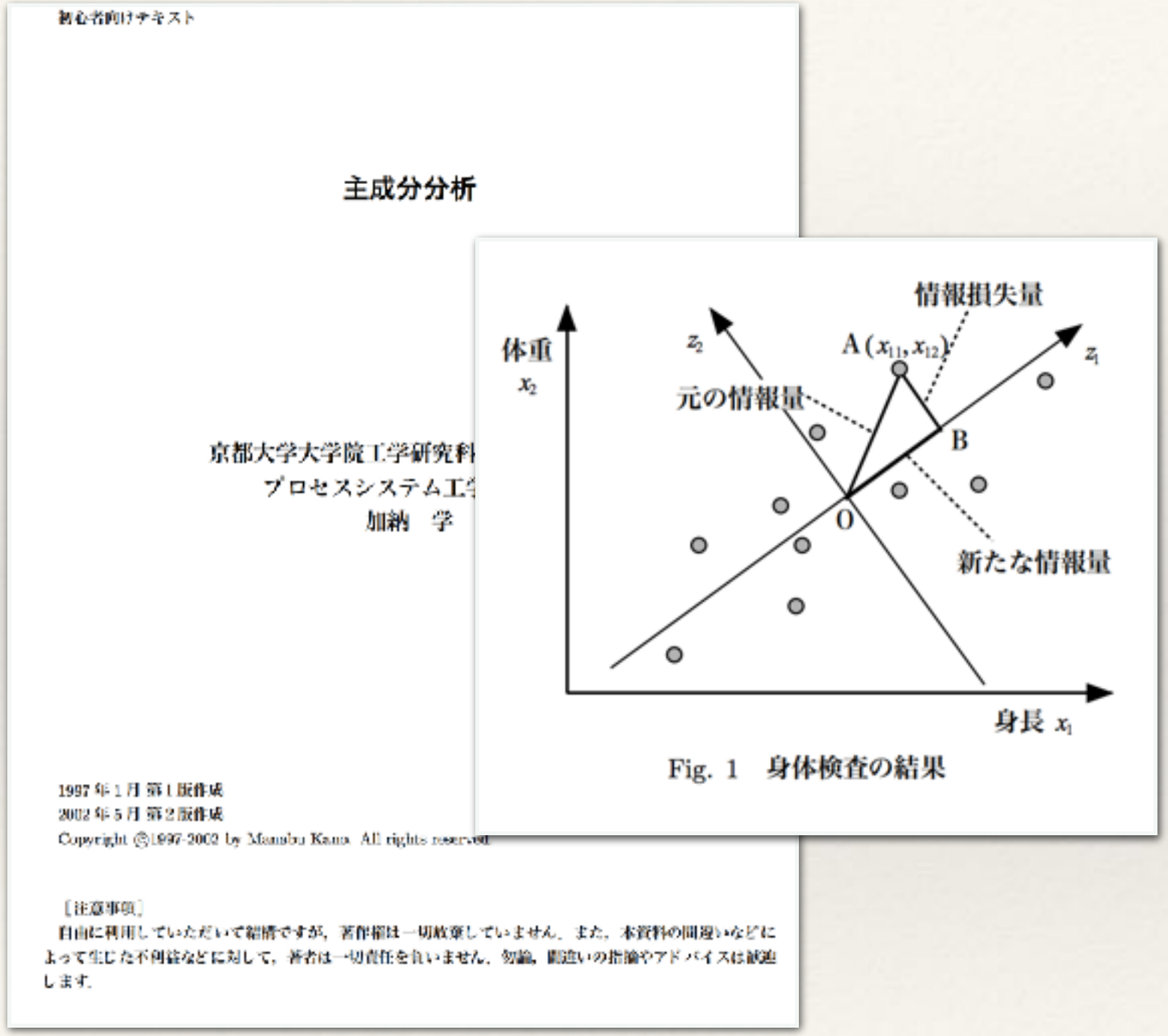
ここまでのまとめ

- 数式が読める（意味を掴める）ことは重要
- わからなければ、コンピュータに手伝ってもらう
  - グラフを描いて実感する
    - SymPyなど数式をそのまま処理してくれるライブラリもある
- scikit-learnが賢いので実は何も知らなくても、そこそこ良いモデルが作れる
  - みんな忙しいので、お客さんが満足していれそれでいい？

どのように学べばよいか？

- まず全体象を理解する（細かい事は気にしない）
- 目的駆動型学習
  - （例）Deep Learningの仕組みを理解したい！
  - （例）主成分分析（PCA）を手計算できるようになりたい！
- わからないところを調べながら進む

たとえば主成分分析



初心者向けテキスト

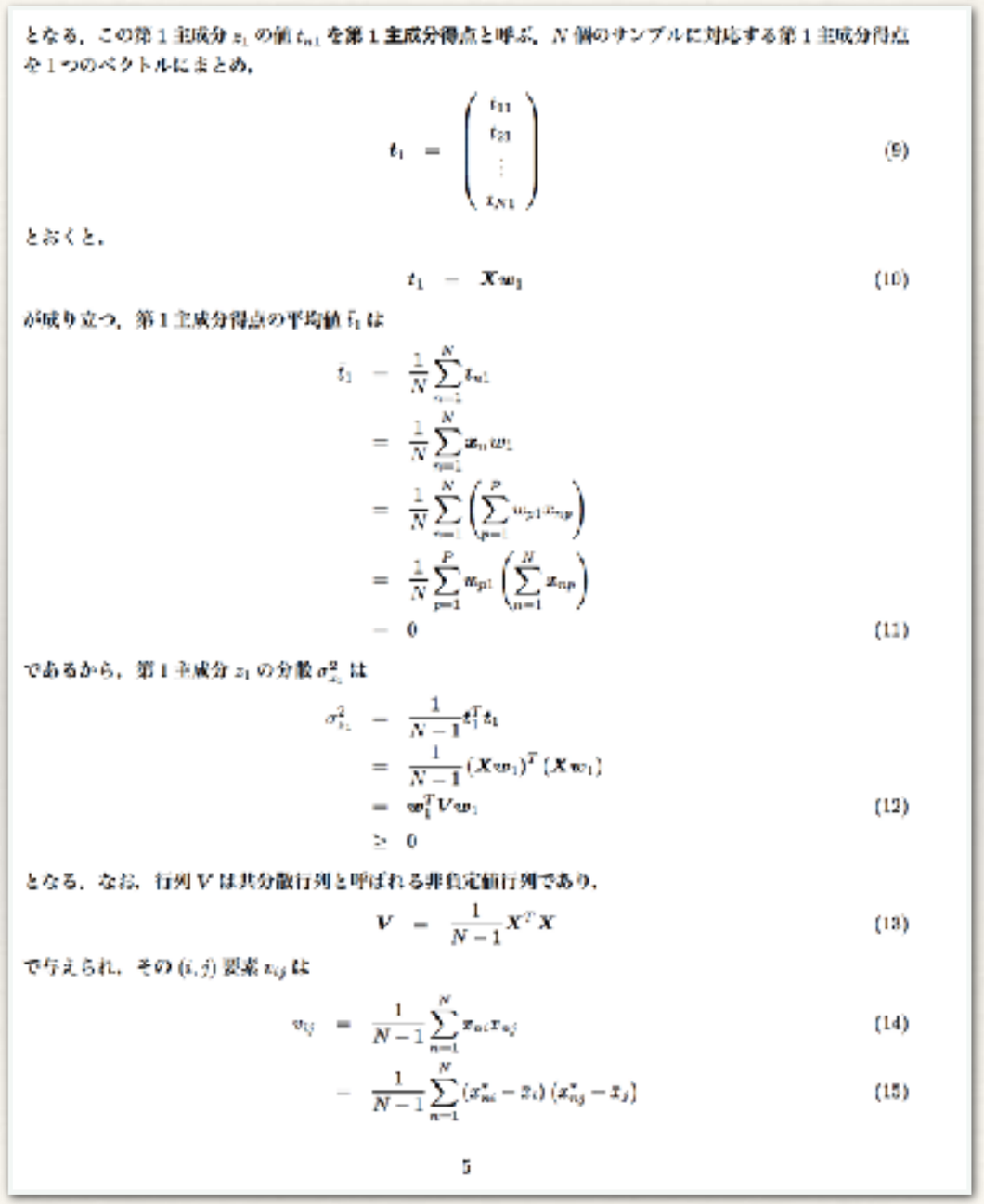
主成分分析

京都大学大学院工学研究科
プロセスシステム工学
加納　学

1997年1月 第1版作成
2002年5月 第2版作成

［注意事項］
自由に利用していただいて結構ですが，著作権は一切放棄していません．また，本資料の間違いなどによって生じた不利益などに対して，著者は一切責任を負いません．勿論，間違いの指摘やアドバイスは感謝します．

Fig. 1　身体検査の結果



となる．この第1主成分 z_1 の値 t_{n1} を**第1主成分得点**と呼ぶ．N 個のサンプルに対応する第1主成分得点を1つのベクトルにまとめ，

$$\boldsymbol{t}_1 = \begin{pmatrix} t_{11} \\ t_{21} \\ \vdots \\ t_{N1} \end{pmatrix} \tag{9}$$

とおくと，

$$\boldsymbol{t}_1 = \boldsymbol{X}\boldsymbol{w}_1 \tag{10}$$

が成り立つ．第1主成分得点の平均値 $\bar{t}_1$ は

$$\begin{aligned} \bar{t}_1 &= \frac{1}{N}\sum_{n=1}^{N} x_{n1} \\ &= \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{w}_1 \\ &= \frac{1}{N}\sum_{n=1}^{N}\left(\sum_{p=1}^{P} w_{p1} x_{np}\right) \\ &= \frac{1}{N}\sum_{p=1}^{P} w_{p1}\left(\sum_{n=1}^{N} x_{np}\right) \\ &= 0 \end{aligned} \tag{11}$$

であるから，第1主成分 z_1 の分散 $\sigma_{z_1}^2$ は

$$\begin{aligned} \sigma_{z_1}^2 &= \frac{1}{N-1}\boldsymbol{t}_1^T \boldsymbol{t}_1 \\ &= \frac{1}{N-1}(\boldsymbol{X}\boldsymbol{w}_1)^T(\boldsymbol{X}\boldsymbol{w}_1) \\ &= \boldsymbol{w}_1^T \boldsymbol{V} \boldsymbol{w}_1 \\ &\geq 0 \end{aligned} \tag{12}$$

となる．なお，行列 $\boldsymbol{V}$ は共分散行列と呼ばれる非負定値行列であり，

$$\boldsymbol{V} = \frac{1}{N-1}\boldsymbol{X}^T\boldsymbol{X} \tag{13}$$

で与えられ，その (i,j) 要素 v_{ij} は

$$v_{ij} = \frac{1}{N-1}\sum_{n=1}^{N} x_{ni} x_{nj} \tag{14}$$

$$= \frac{1}{N-1}\sum_{n=1}^{N}(x_{ni}^* - \bar{x}_i)(x_{nj}^* - \bar{x}_j) \tag{15}$$

5

http://manabukano.brilliant-future.net/document/text-PCA.pdf

手元に教科書

- 一通りの内容が書いてある教科書はわからないところを解決するたのに便利
- 教科書がわからないと意味がないので、コンピュータの助けを借りられる内容がよいかも

本腰を入れてみたい人へ

- 機械学習のエッセンス（加藤公一）
- かなり本格的な内容
- #pydatatextと同じ3章が数学
  - 線形代数と微分積分について親玉みたいな内容

まとめ

- 数式を読めるようになろう
- 数式の意味がわからなかったら、コンピュータの助けを借りる
- scikit-learnは優秀
- 目的を決めて、教科書を片手に、コンピュータの助けを借りながら長い旅路を楽しむ