

# AI理論と実践講座

# 本日のAgenda

- AIを学ぶために必要な基礎知識
- 線形回帰
- Pythonによる線形回帰

※PCAについては次回に回させていただきます

AIを学ぶために必要な基礎知識

# English and Googling Skill

英語の情報がオリジナルで品質が高い

情報が日本語化されるのを待っていては遅すぎる (SPEED)

オリジナルに近い情報が正確 (QUALITY)

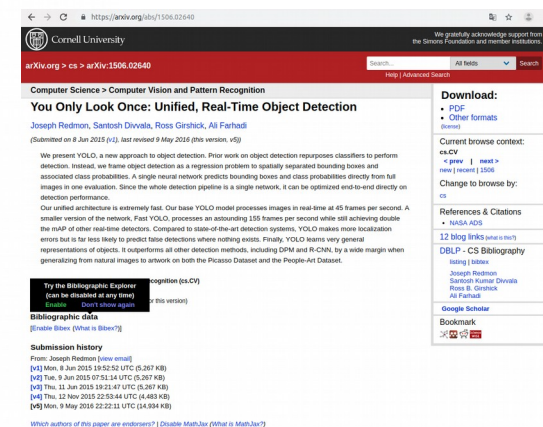
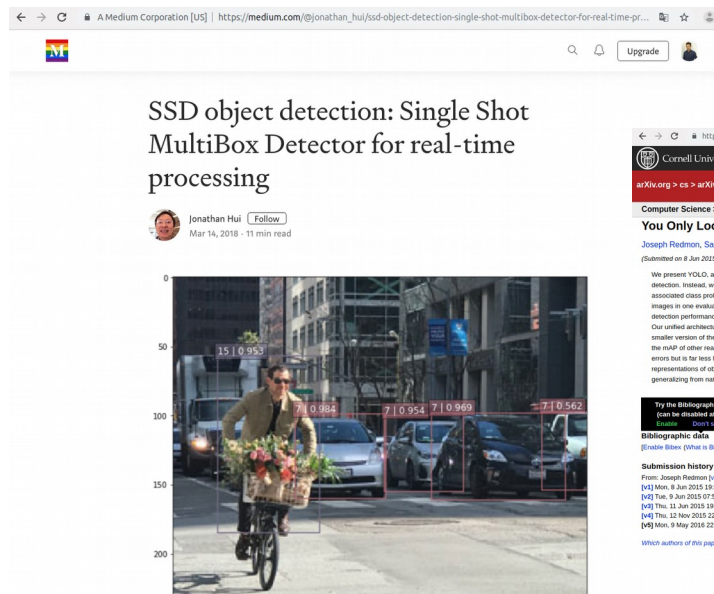
英語の情報が圧倒的に多い (QUANTITY)

# オススメ

コンテンツプラットフォーム Mediumがオススメ  
良質の解説記事が多い (専門知識がなくても読める)

原論文にあたりたいときにはarxivにほとんどある  
最近は読みやすい論文が多い

全部読み切る必要はなく、途中までなんとなく読むだけでも十分役に立つ



# オススメ 2

英語で受講できるe-learningには良質なものがあります  
Coursera, Udacity, edX etc.

The image displays three overlapping screenshots of e-learning platforms:

- Udacity:** The top screenshot shows the Udacity homepage with a navigation bar including 'Programs', 'Career', 'For Enterprise', 'Sign In', and a 'GET STARTED' button. The main banner features the text 'Be in demand' and 'Learn the latest tech skills to pro'.
- Coursera:** The middle screenshot shows the Coursera homepage with a search bar, a '検索' (Search) button, and a '何を学習しますか?' (What do you want to learn?) prompt. It also includes links for '企業用' (Enterprise), 'ログイン' (Login), and '参加は無料' (Participation is free).
- edX:** The bottom screenshot shows the edX homepage with a navigation bar including 'Courses', 'Programs & Degrees', 'Schools & Partners', and 'edX for Business'. The main banner features the text 'Access 2500+ Online Courses from 140 Top Institutions. Start Today!' and a 'Find courses' button. Below the banner is a search bar with the text 'What do you want to learn?'. At the bottom, there is a row of logos for partner institutions: MIT, Massachusetts Institute of Technology, Harvard University, Berkeley University of California, The University of Texas System, Boston University, The Hong Kong Polytechnic University, Duke University, Google, M (Mitsubishi), IBM, Imperial College London, Stanford, and Penn (University of Pennsylvania). A 'NEW! MicroBachelors™ Programs for' banner is also visible at the bottom left.

# プログラミング

プログラムを作る敷居は非常に低くなった

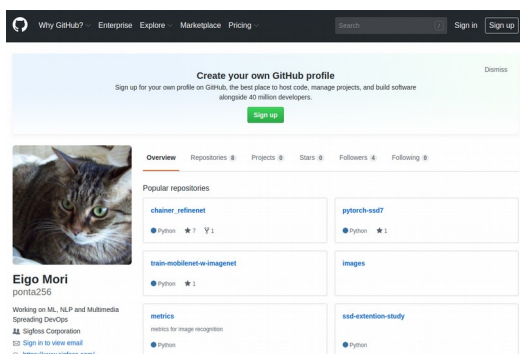
PythonならExcel学ぶのと同じ感覚で学べる

最初の入門部分は易しい入門書などを一通りした方が早い  
(わからないところはどんどん飛ばす)

あとは人のコードをコピーしたり、GitHubから落としてきたり



## プログラムが作れるようになると世界が変わります

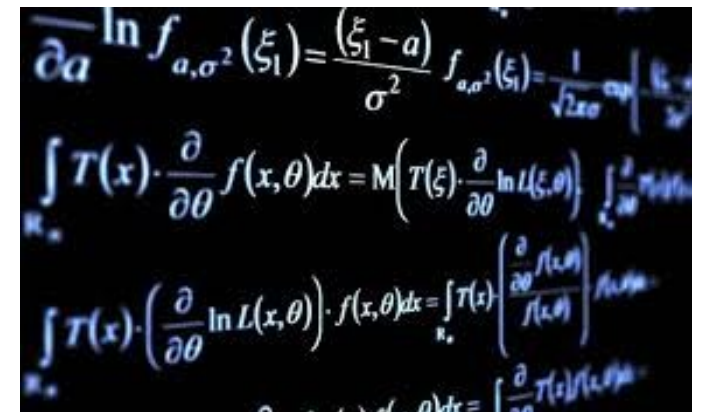


# 数学

**確率と統計に関してはぜひとも学びたい  
入門書を繰り返し読むことがオススメ**

線形代数と微積分は理論的な背景を理解したり、論文を読むときには必要  
実際にプログラムを開発する現場ではPythonのライブラリなどにすでに組み込み済みで**マストの知識ではない**

高校レベルの線形代数と微積分でも十分以上に役に立つ  
これも入門書を繰り返し読むことがオススメ


$$\frac{\partial}{\partial a} \ln f_{a, \sigma^2}(\xi_1) = \frac{(\xi_1 - a)}{\sigma^2} f_{a, \sigma^2}(\xi_1) = \frac{1}{\sqrt{2\pi\sigma^2}} \left( \frac{\xi_1 - a}{\sigma^2} \right) e^{-\frac{(\xi_1 - a)^2}{2\sigma^2}}$$
$$\int_{\mathcal{X}} T(x) \cdot \frac{\partial}{\partial \theta} f(x, \theta) dx = M \left( T(\xi) \cdot \frac{\partial}{\partial \theta} \ln L(\xi, \theta) \right) = \int_{\mathcal{X}} T(x) \cdot \left( \frac{\partial}{\partial \theta} \ln L(x, \theta) \right) \cdot f(x, \theta) dx = \int_{\mathcal{X}} T(x) \cdot \left( \frac{\partial}{\partial \theta} \ln L(x, \theta) \right) \cdot f(x, \theta) dx = \int_{\mathcal{X}} T(x) \cdot \left( \frac{\partial}{\partial \theta} \ln L(x, \theta) \right) \cdot f(x, \theta) dx$$



# 線形回帰

# 機械学習

一般にAIは**入力データ**に対し、何らかの基準で**判断を行い、推論を出力する**

エキスパートシステムのような手法では、その基準を人間が与えるのに対し、**機械学習ではその基準をコンピュータが自ら学ぶ**

コンピュータにどう学ばせたらいいか？

# 単回帰分析

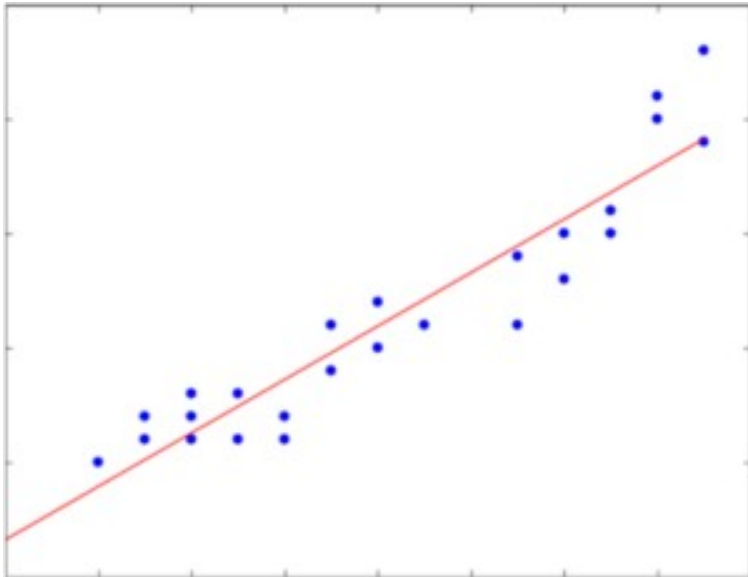
単回帰分析はある変数( $X$ )から、もう一方の変数( $Y$ )を予測する

(相関分析は変数間にどのくらいの関係性があるのかを調べる)

$Y$ を目的変数、 $X$ を説明変数と呼ぶ

# 目的変数と説明変数の選択

予測したい目的変数に相関がありそうな説明変数を選ぶ  
グラフ化することであらかじめ相関性が予想できる



年齢(X)と年収(Y)

部屋数(X)と家賃(Y)

体重(X)と身長(Y)

体重(X)とアルコール耐性(Y)

# 単回帰分析のモデル

$$Y = aX + b$$

学習データからaとbを求める

未知のXに対してもYが求まる  
(年齢入れば年収が求まる)

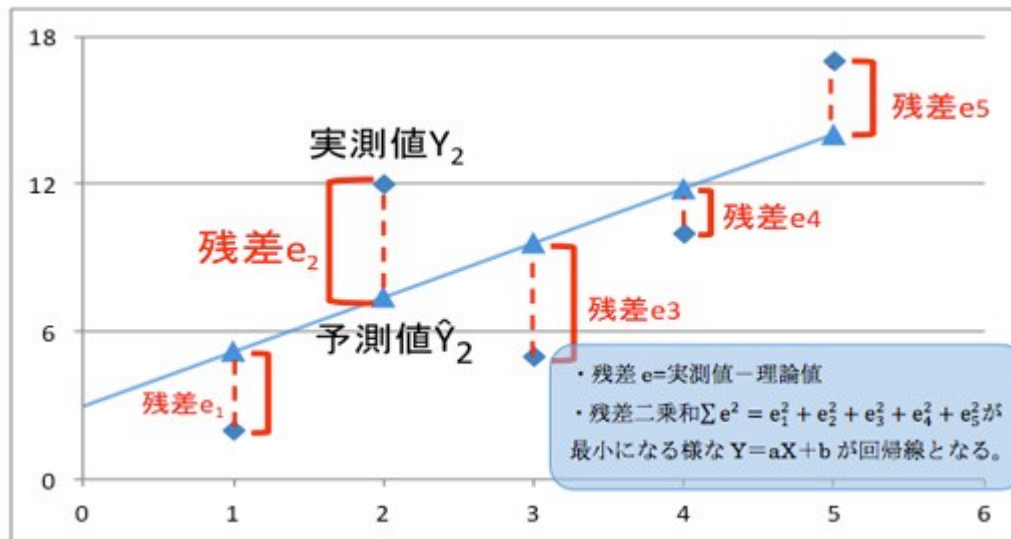
誤差がゼロのaとbが求まるときには単純に方程式を解くので  
機械学習は不要 (現実世界にはほぼない)

相関はありそうだけど誤差はあるものに、最もそれっぽいaと  
bを見つけるのが単回帰分析の機械学習

# 誤差を決める

機械学習は定めた誤差(LOSS)が最小になるようにパラメータを決定する

誤差の定義を決めることは極めて重要



単回帰分析の誤差定義

# 単回帰分析のアルゴリズム

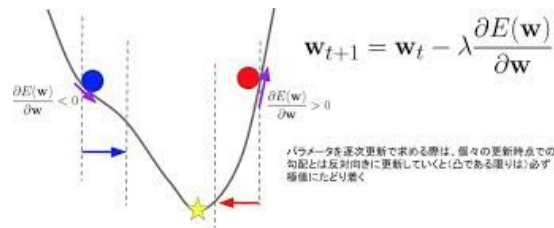
## 最急降下法

最急降下法は以下の手順で処理を進めます。

1. 任意のa, bを設定する
2. Loss(a, b)が最小となるようにa, bを動かす (偏微分)
3. a, bを更新して2.のプロセスへ戻る

**aとbを少しずつ動かしながら最適な値を見つける**

**この方法はDeep Learningなども同じ**



# 最急降下法の実装

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_boston

boston = load_boston()

x = boston['data'] # 物件の情報
x = x[:, 5]
y = boston['target'] # 家賃

alpha = 0.04
itera = 10000
cost = np.zeros(itera)
b = 1 # bの初期値
a = 1 # aの初期値
m = len(y)

for i in range(itera):
    cost[i] = (1/(2*m))*np.sum(np.square(b+a*x-y))
    b = b - alpha*(1/(m))*np.sum(b+a*x-y)
    a = a - alpha*(1/(m))*np.sum((b+a*x-y)*x)

print(b, a)
```



# 重回帰分析

単回帰分析  $Y = aX + b$

重回帰分析  $Y = b_1X_1 + b_2X_2 + b_3X_3 + b_4X_4 + \dots + b_0$

単回帰が説明変数1つを利用するのに対し、重回帰は複数の説明変数を利用する



Pythonによる線形回帰とPCA

# scikit-learnによる回帰分析

```
from sklearn.datasets import load_boston
from sklearn.linear_model import LinearRegression

boston = load_boston()

x = boston['data'] # 物件の情報
y = boston['target'] # 家賃

x = x[:, [5]] # 平均部屋数

model = LinearRegression()
model.fit(x_train, y_train)

a = model.coef_
b = model.intercept_

y_out = model.predict([[7.0]])
```

# 演習

問1.

部屋数の代わりにNOx濃度を説明変数として単回帰分析を行ってください

問2.

散布図を作って回帰曲線を描画してください

問3.

すべての特徴量を説明変数として重回帰分析を行ってください

問4.

自動車データセット <http://archive.ics.uci.edu/ml/datasets/Automobile> を使って、engine-sizeを用いた単回帰分析およびlength, width, heightを用いた重回帰分析を行ってください

Q&A