

# 電気情報工学セミナーⅡ

実験のためのPython & Git入門（機械学習による関数の近似実験）

～ 第3回 多項式による関数の近似 ～

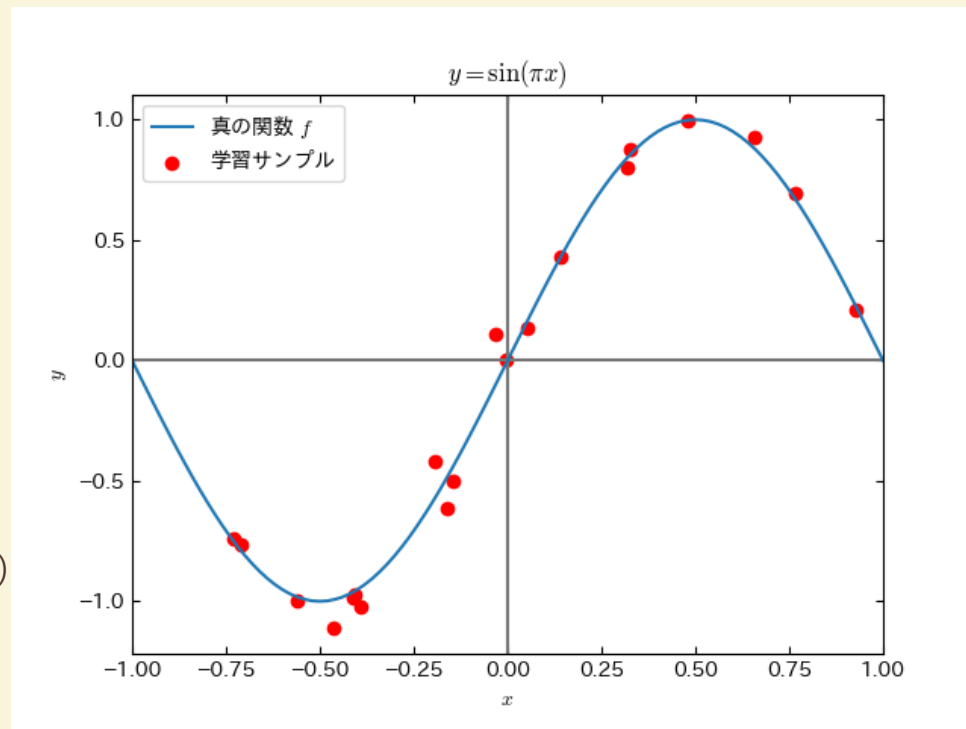
池原研究室

# 回帰分析実験の詳細

# 回帰分析の実験条件

データ点の集合  $\{(x_i, y_i)\}_{i=1}^N$  から  $y_i$  を生成した関数  $y = f(x)$  を予測

- データ点の数 :  $N = 20$
- 関数の定義域 :  $[\alpha, \beta]$  ;  $\alpha = -1, \beta = 1$
- データ点の  $x$  座標 :  $x_i \sim \mathcal{U}(\alpha, \beta)$  (一様分布)
- データ点の  $y$  座標 :  $y_i = f(x_i) + \epsilon_i$
- ノイズ :  $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$  (正規分布)
- ノイズの強さ :  $\sigma = r \left( \max_{x \in [\alpha, \beta]} f(x) - \min_{x \in [\alpha, \beta]} f(x) \right)$  ;  $r = 0.05$
- 真の関数 :  $f(x) = \sin(\pi x)$



# 評価指標（コンセプト）

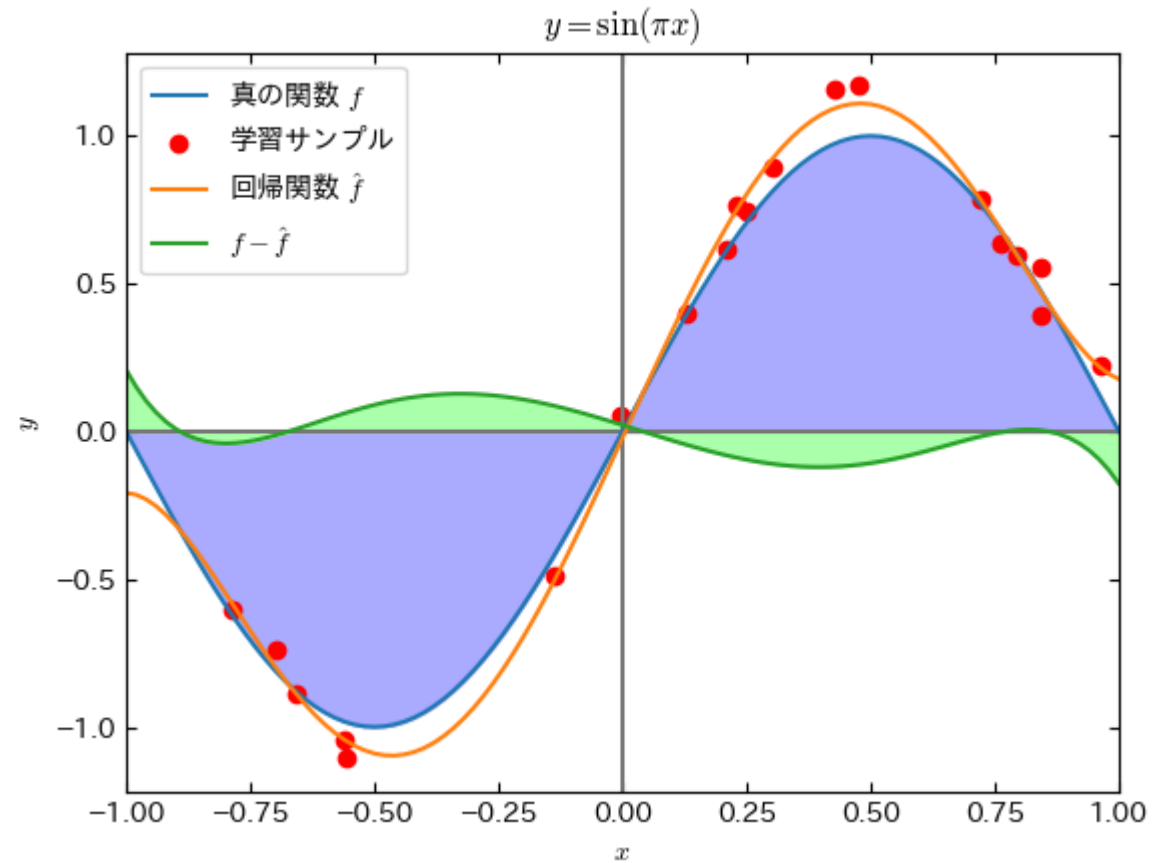
$f$ と $\hat{f}$ が近い（差が小さい）ほどよい

$$\int_{\alpha}^{\beta} dx |f(x) - \hat{f}(x)|$$

$$\int_{\alpha}^{\beta} dx |f(x)| + \varepsilon$$

0 除算回避のため  
小さい正数を足す

後ろに続く関数を $x$ で積分



# 評価指標をコンピュータで計算するために

$f$ と $\hat{f}$ が近い（差が小さい）ほどよい

定義域を同じ間隔でサンプリング

$$x_m = \frac{m}{M} \alpha + \left(1 - \frac{m}{M}\right) \beta$$

$$\frac{\int_{\alpha}^{\beta} dx |f(x) - \hat{f}(x)|}{\int_{\alpha}^{\beta} dx |f(x)| + \varepsilon} \approx \frac{\sum_{m=0}^M |f(x_m) - \hat{f}(x_m)|}{\sum_{m=0}^M |f(x_m)| + \varepsilon}$$

0 除算回避のため  
小さい正数を足す

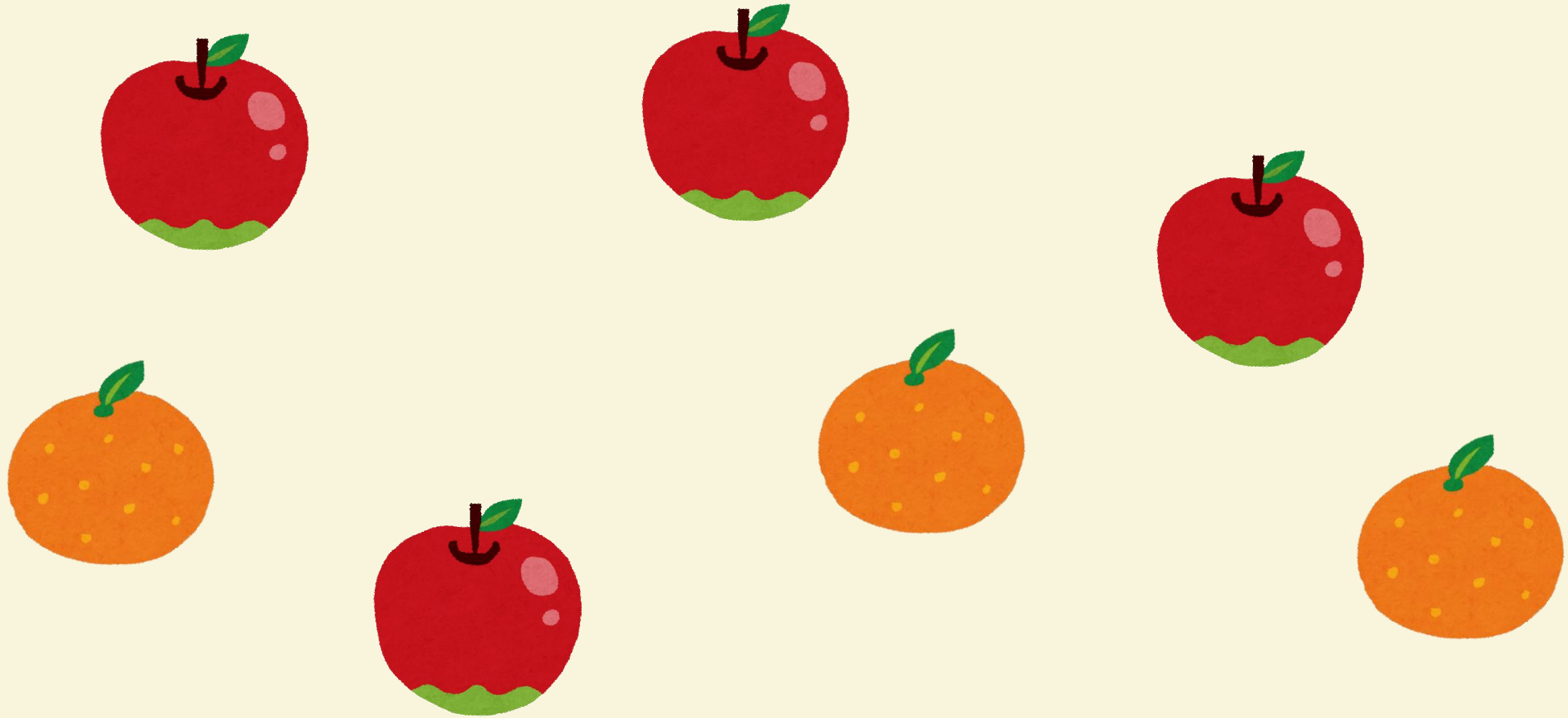
後ろに続く関数を $x$ で積分

※分母分子ともに生じた $\frac{1}{M+1}$ で約分している

評価用サンプル点の数： $M = 100$   
0除算回避のための正数： $\varepsilon = 10^{-8}$

# 線形代数と 多項式フィッティング

# 状況を説明してみよう



# 計算してみよう

$f(x) = 24x^2 + 46x$  とする。

①  $\frac{d}{dx} f(x)$

②  $\int_0^1 dx f(x)$



## 計算してみよう（回答例）

$f(x) = 24x^2 + 46x$  とする。

$$\textcircled{1} \quad \frac{d}{dx} f(x) = \lim_{a \rightarrow 0} \frac{f(x+a) - f(x)}{a} = \lim_{a \rightarrow 0} (48x + 46 + 24a) = 48x + 46$$

$$\textcircled{2} \quad \int_0^1 dx f(x) = \lim_{n \rightarrow \infty} \sum_{k=1}^n \frac{1}{n} f\left(\frac{k}{n}\right) = \lim_{n \rightarrow \infty} \left( 31 + \frac{35}{n} + \frac{4}{n^2} \right) = 31$$

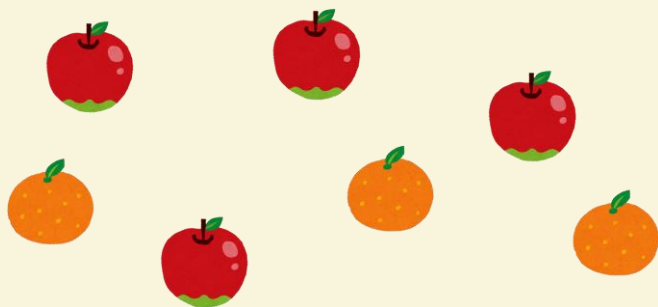
## 計算してみよう（写像の線形性を活用）

$f(x) = 24x^2 + 46x$  とする。

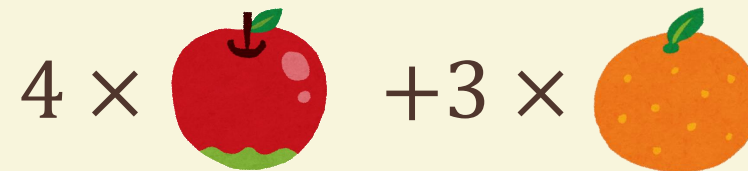
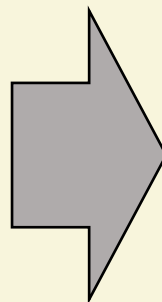
$$\textcircled{1} \quad \frac{d}{dx} f(x) = 24 \cdot \frac{d}{dx} x^2 + 46 \cdot \frac{d}{dx} x = 24 \cdot 2x + 46 \cdot 1 = 48x + 46$$

$$\textcircled{2} \quad \int_0^1 dx f(x) = 24 \cdot \int_0^1 dx x^2 + 46 \cdot \int_0^1 dx x = 24 \cdot \frac{1}{3} + 46 \cdot \frac{1}{2} = 31$$

# 線形代数のココロ

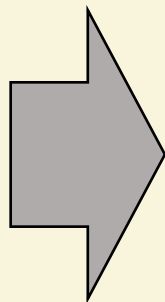


よくわからないもの

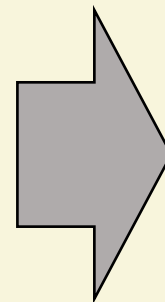


知っているものの線形結合

$T(v)$



$$T\left(\sum_{i=1}^n a_i v_i\right)$$



$$\sum_{i=1}^n a_i T(v_i)$$

$v$ がよく性質がわかっているものの  
線形結合に分解できたら...

$T$ が線形写像なら

# 多項式フィッティング

真の関数 $f$ を $d$ 次多項式で近似

$$\hat{f}_n(x) = a_0 + a_1x + a_2x^2 + \cdots + a_dx^d$$

(よく性質がわかっている部品として $x^p$ を採用)

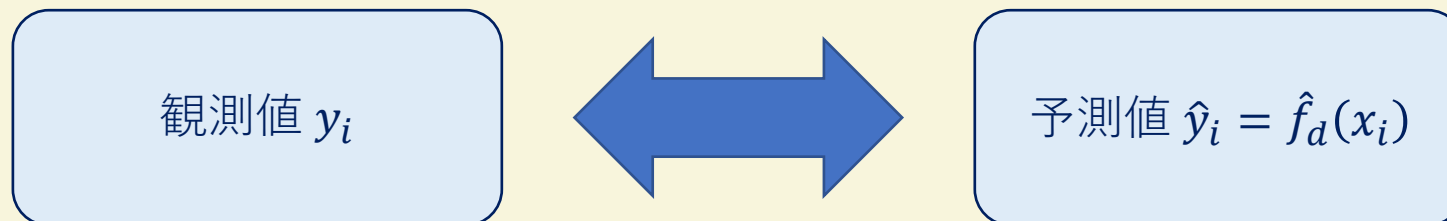
## 補足 1

$n$ 回微分可能な関数は  
 $n$ 次多項式でよく近似できる  
(テイラーの定理)

## 補足 2

よく性質がわかっている部品として  
 $\sin(kx)$ と $\cos(kx)$ を採用する  
フーリエ級数展開も有名

# パラメータの求め方



平均的に誤差が小さくなるように

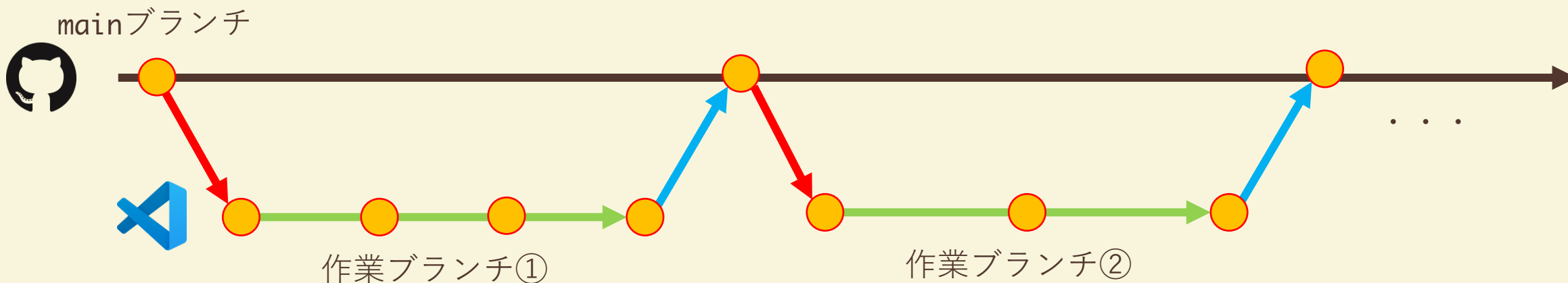
誤差関数  $E = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$  を最小化  $\Rightarrow \frac{\partial E}{\partial a_p} = 0$  を各  $p$  でみたす  $\{a_p\}_{p=0}^d$  を求める

$$\begin{pmatrix} x_1^0 & \cdots & x_N^0 \\ \vdots & \ddots & \vdots \\ x_1^d & \cdots & x_N^d \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} x_1^0 & \cdots & x_N^0 \\ \vdots & \ddots & \vdots \\ x_1^d & \cdots & x_N^d \end{pmatrix} \begin{pmatrix} x_1^0 & \cdots & x_1^d \\ \vdots & \ddots & \vdots \\ x_N^0 & \cdots & x_N^d \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_d \end{pmatrix} \quad \text{つまり} \quad X^\top \mathbf{y} = X^\top X \mathbf{a} \quad \text{を満たせばよい}$$

$$\mathbf{a} = (X^\top X)^{-1} X^\top \mathbf{y}$$

# GitHubを使った個人開発

# 大まかな流れ



## 計画

- GitHubにIssueを作成
  - やりたいことなどをまとめる
- GitHub上でブランチを作成

## 実装

- プログラムを書いてコミットの繰り返し
- 失敗したらブランチごと削除できる

## 記録

- GitHubでPull Requestを発行
  - 作業の成果をまとめる
- 作業内容をmainブランチに統合 (merge)

# GitHub①：Issueを作成する

解決すべき課題

(個人で使う際には) やりたいことや方針をまとめるのにGitHubでIssueを作成するとよい

評価指標の算出 #22

Open fkzk opened this issue 2 hours ago · 1 comment

fkzk commented 2 hours ago

Owner

$\|t - y\|^2 / \|t\|^2$  を評価基準にする

fkzk commented 32 seconds ago

Owner Author

$\sin(\pi x)$  と  $2\sin(\pi x)$  とで結果が変わらないようにL1ノルムで  $\|t - y\| / \|t\|$  を評価基準にした方がよさげ

取り組んでみてわかったこともメモできる



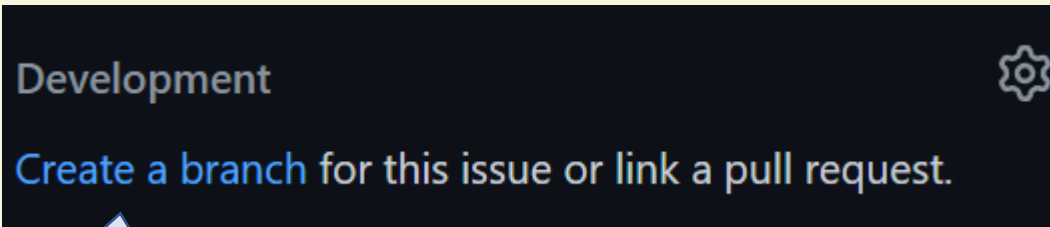
# GitHub②：Issueのためのブランチを作成

Issueの解決は  
難しい場合もある

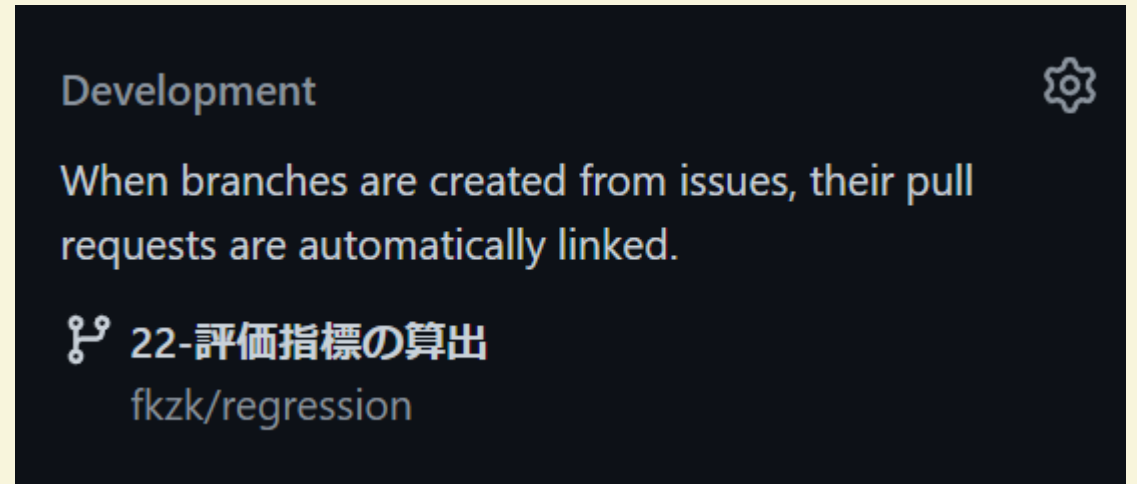


mainブランチ（成功だけからなる履歴）とは別に  
作業用のブランチ（失敗したら消せる履歴）があると便利

Issue画面の右側にある



作成開始は  
ここから



- ☺ 自動でIssueと対応したブランチの名前を提案してくれる
- ☺ クリックだけでブランチを作成可能
- ☺ 手元でブランチを切り替えるコマンドも教えてくれる

# Git①：GitHubで作ったブランチを手元で利用

①GitHubのリポジトリ（origin）の現状を取得（fetch）

```
$ git fetch origin
```

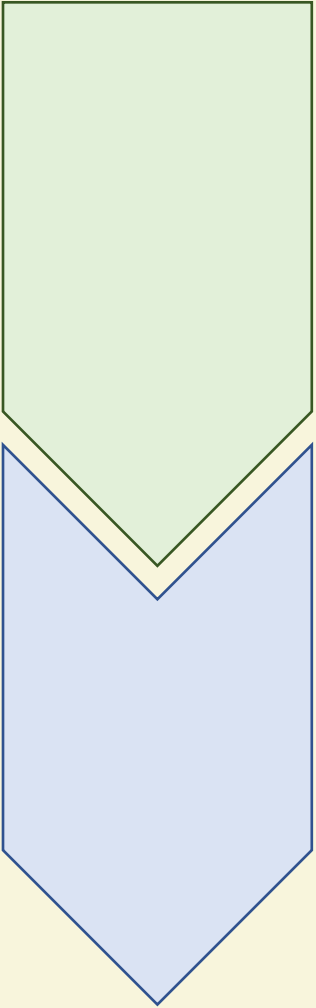
- ✓ originはcloneした時点でgitが勝手につけたクローン元の名前
- ✓ 別の名前のリモートリポジトリの現状を取得する場合はoriginの代わりにリモートリポジトリ名を書く

②ブランチを切り替える

```
$ git switch ブランチ名
```

- ✓ ローカル（手元のPC）に存在しないブランチの場合はリモートに同じ名前のブランチがないか（fetchした中から）探して切り替える
- ✓ 古いバージョンのgitではswitchの代わりにcheckoutを使う必要があることも

## Git②：Gitでコードをコミット（復習）

- 
- ①「今回変更した」「今からセーブしたいファイル」を宣言（**ステージング**）

```
$ git add ファイル名
```

- ✓ VSCodeならソース管理の画面で+を押す

- ②ステージングしたファイルをセーブ（**コミット**）

```
$ git commit -m 'コミットメッセージ'
```

- ✓ VSCodeではコミットメッセージをテキストボックスに書いてから「コミット」ボタンを押す

## Git③ : GitHubに作業状況を送る

① リモート（GitHub）の変更状態を取り込む（1人なら複数PCでの開発時）

```
$ git fetch origin  
$ git merge origin/ブランチ名
```

追跡対象のファイルを  
(commitしたことがある)  
すべてcommit (or stash)  
してから始めること

✓ 自分が編集した部分と同じ所に別の変更を加えられていると  
手作業の修正が必要に（下線の状態をconflictが生じているという）

② リモート（GitHub）に変更状態を送る

```
$ git push origin
```

VSCodeでは  
①②がまとめてできる

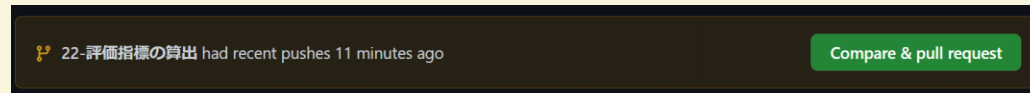
✓ ソース管理

メッセージ ('22-評...

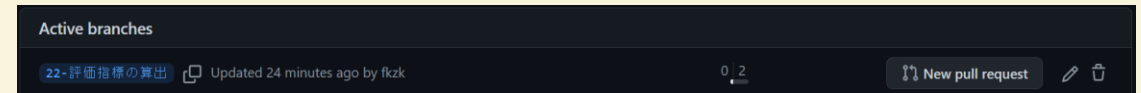
🔄 変更の同期 2↑

# GitHub③ : Pull Requestを送る

送信直後はトップにリンクが出てくる



なくなっていたらブランチ一覧から

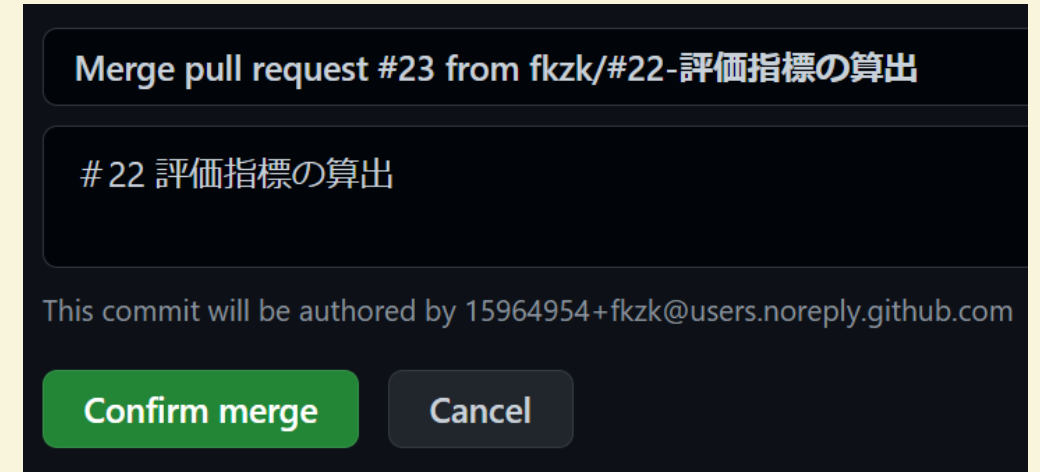
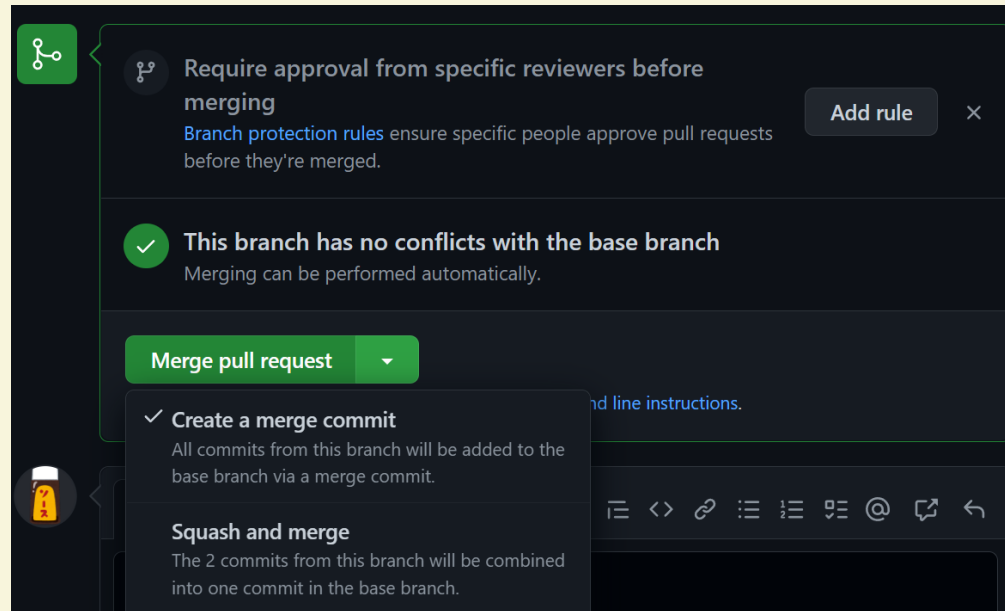


本来は...  
「こういう変更したけど  
取り込んでみませんか？」  
という提案をする機能

個人で使うなら  
まとめノート感覚でOK

コミット一覧のページから  
マウスオーバーで  
見えるので詳しく書くと◎

# GitHub④ : Pull Requestを取り入れる

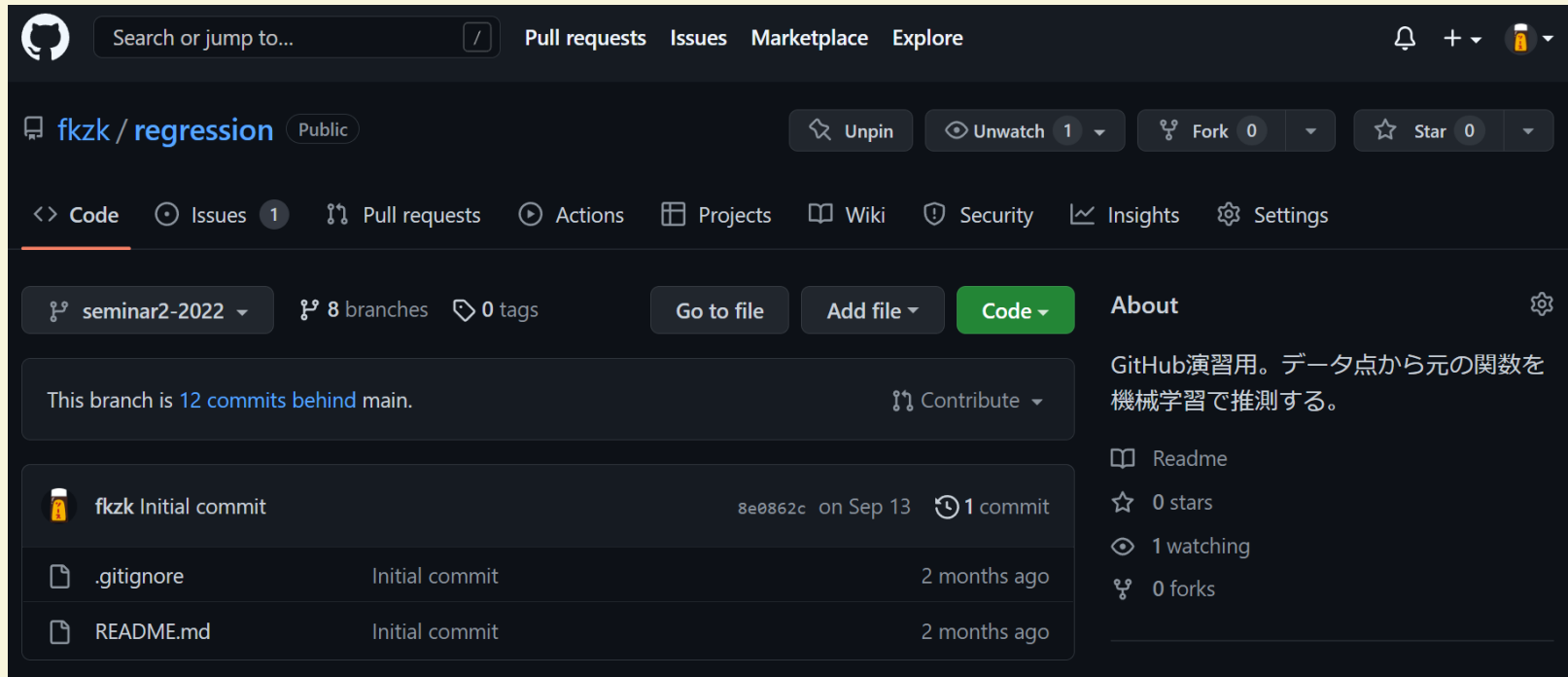


(本来は他人が書いたコードの検証などが必要になるが)  
個人で使う分には何も考えずに2クリックでOK

# 実践編

# 今回以降のコード

<https://github.com/fkzk/regression/tree/seminar2-2022>



※授業ではTAはseminar2-2022というブランチを中心に開発を進めていく  
(参考にするときだけ気にすればいい；学生はmainブランチを中心に進める)

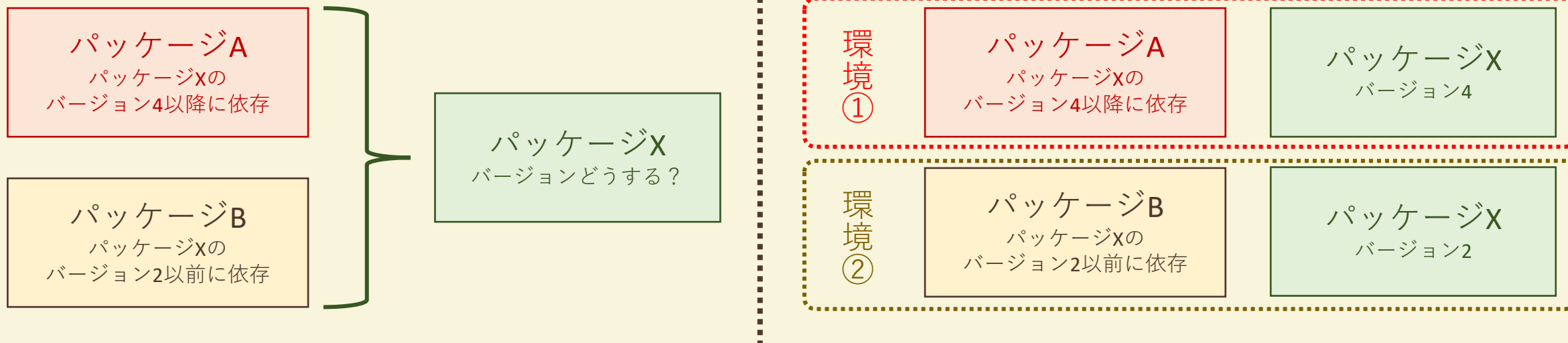


# Pythonの仮想環境を作成

\$HOME/.bash\_profile内で  
定義した変数を参照

仮想環境名

```
$ python -m venv $VENV/seminar2
```



# Pythonの仮想環境をactivateする

Windows

```
$ . $VENV/seminar2/Scripts/activate
```

Mac / Linux

```
$ . $VENV/seminar2/bin/activate
```

. コマンドはシェルスクリプトを自分自身で実行するコマンド  
今回は設定変更してくれるスクリプト (**activate**) を読み込んだ

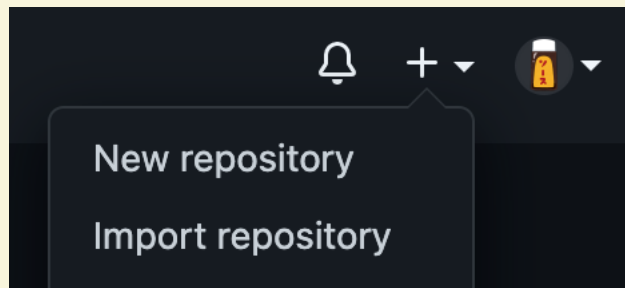
# 利用する外部パッケージを仮想環境にインストール

```
$ python -m pip install --upgrade pip wheel
$ python -m pip install japanize-matplotlib
```

- ✓ PyPI (The Python Package Index) と呼ばれるパッケージ公開サービス上から同じ名前で登録されているパッケージを探してインストールしてくれる
- ✓ 依存しているパッケージがある場合はそれもまとめてインストールしてくれる



# GitHub リポジトリを作成



リポジトリ：開発履歴の保管庫

非公開にもできるが  
無料ユーザは機能が制限される

トップページに表示される  
説明用Markdownファイルの生成

Gitで管理しないファイルの  
設定を言語に合わせ自動生成

A screenshot of the 'Create a new repository' form on GitHub. The form is titled 'Create a new repository' and includes a subtitle: 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. The form has several sections: 'Owner' (set to 'fkzk'), 'Repository name' (set to 'regression' with a red warning icon and a message 'The repository regression already exists on this account. [View about symmetrical-dollop?](#)'), 'Description (optional)' (empty text area), 'Public/Private' (Public is selected), 'Initialize this repository with:' (checkbox for 'Add a README file' is checked), 'Add .gitignore' (dropdown menu set to 'Python'), 'Choose a license' (dropdown menu set to 'None'), and a footer note: 'This will set `main` as the default branch. Change the default name in your [settings](#).'

省略可能だが設定すると  
1. `git init`  
2. 最初のcommit  
をGitHubがやってくれる

# GitHubリポジトリをcloneする（実は復習）

`$HOME/.bash_profile`内で  
定義した変数を参照

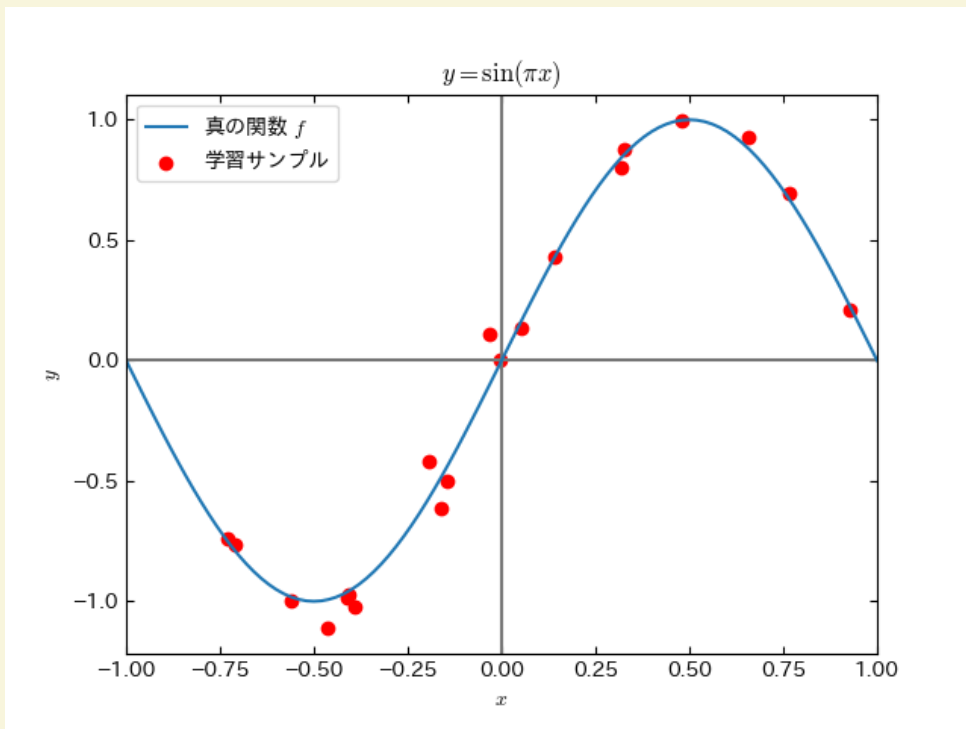
```
$ cd $SEMINAR2  
$ git clone github:/ユーザ名/regression
```

`$HOME/.bash_profile`内で  
Host `github`から始まる設定を書いたため  
本来は`git@github.com`と書く

VSCoideでディレクトリを開く

```
$ code -r regression
```

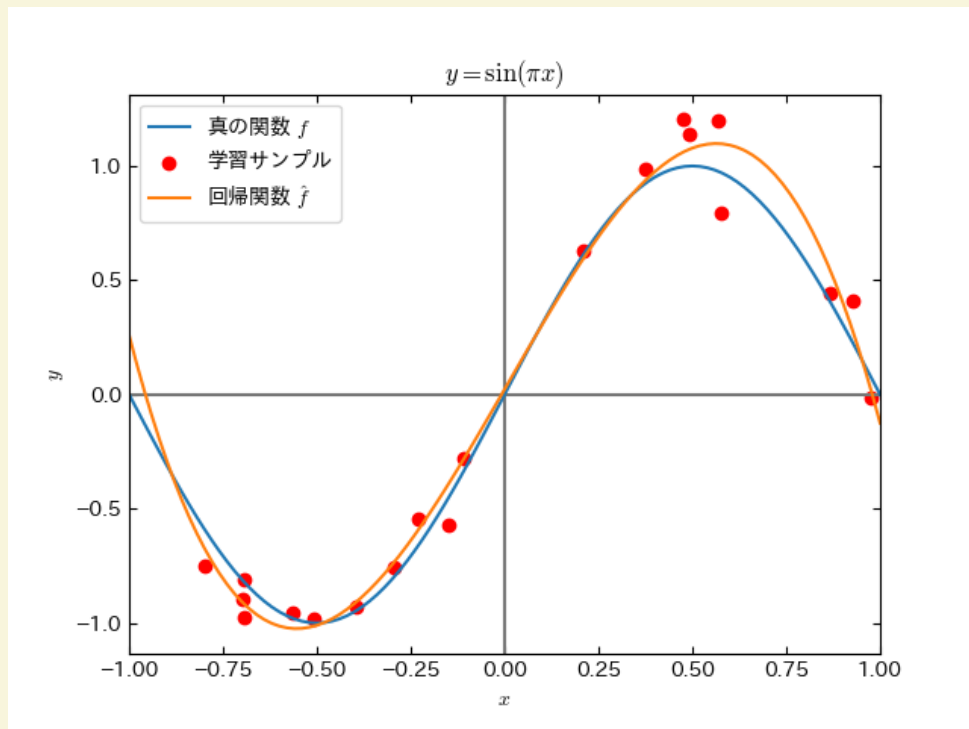
# グラフを作成



- ◆  $\sin(\pi x)$ のグラフを作成した
- ◆ グラフの見た目を整えた
- ◆ 学習サンプルをランダムに生成した
- ◆ グラフに判例を追加した

- ✓ 外部パッケージの利用にはimport文を使う
- ✓ グラフの作成にはmatplotlibというパッケージを利用
- ✓ matplotlibはnumpyという数値計算ライブラリのデータをグラフにすることが可能
- ✓ matplotlibはFigure (図) の中にAxes (グラフ) を入れるのが基本
  - 1つのFigureに複数のAxesを入れることもできる
- ✓ 滑らかに見える曲線も短い線分の連続でできている
- ✓ グラフの見た目はmatplotlibrcというファイルで調整可能
  - プログラムによる調整も可能だがmatplotlibrcの利用を推奨
- ✓ numpyでは配列の中身に対して一気に同じ処理を実行できる
- ✓ numpyは乱数の生成も可能
- ✓ 日本語を使いたい場合はjapanese-matplotlibを利用
- ✓ Gitで管理したくないファイルは.gitignoreに記述

# 多項式フィッティングを実装



- ◆ 学習サンプルの $x_i^p$ からなる行列`sample_X`を計算した
- ◆ 多項式の係数 $a$ を推定した
- ◆ 予測値を計算し、プロットした

学習サンプル $\{(x_i, y_i)\}_{i=1}^N$ を近似する  
 $d$ 次多項式の係数 $\{a_p\}_{p=0}^d$ が満たす関係

$d = 3$ で実装

$$\begin{pmatrix} x_1^0 & \cdots & x_N^0 \\ \vdots & \ddots & \vdots \\ x_1^d & \cdots & x_N^d \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} x_1^0 & \cdots & x_N^0 \\ \vdots & \ddots & \vdots \\ x_1^d & \cdots & x_N^d \end{pmatrix} \begin{pmatrix} x_1^0 & \cdots & x_1^d \\ \vdots & \ddots & \vdots \\ x_N^0 & \cdots & x_N^d \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_d \end{pmatrix}$$
$$X^T \quad \mathbf{y} = X^T \quad X \quad \mathbf{a}$$

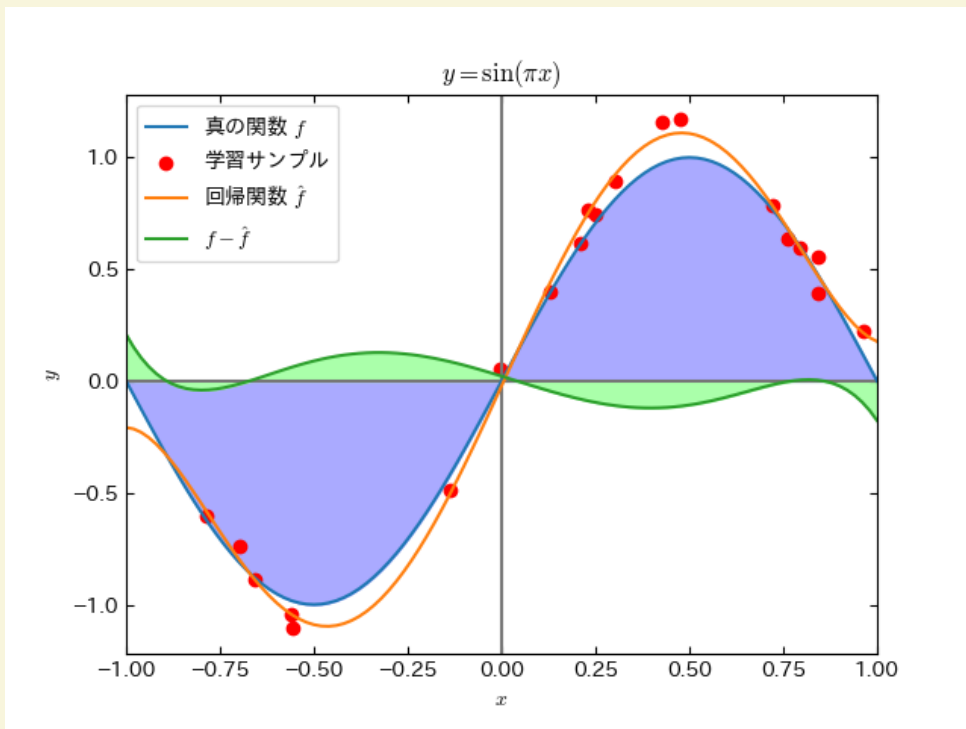
パラメータ推定

$$\mathbf{a} = (X^T X)^{-1} X^T \mathbf{y}$$

推論  
( $y$ の予測)

$$\hat{y} = \sum_{p=0}^d a_p x^p = (x^0 \quad \cdots \quad x^d) \mathbf{a}$$

# 評価指標の算出



参考画像：作成しない予定

◆ 評価指標を算出した

定義域を同じ間隔でサンプリング

$$x_m = \frac{m}{M}\alpha + \left(1 - \frac{m}{M}\right)\beta$$

$$\frac{\sum_{m=0}^M |f(x_m) - \hat{f}(x_m)|}{\sum_{m=0}^M |f(x_m)| + \varepsilon}$$

評価用サンプル点の数： $M = 100$   
0除算回避のための正数： $\varepsilon = 10^{-8}$