全体説明

- 次ページからの Q1〜Q3 のうち，どれか 1 つを自由に選んで，python でコードを書いて提出してください
- 問題に応じて以下のディレクトリに python ライブラリを作成し，アプリケーションコード（動作させるコード）は `main.py` に記述してください．

  Q1 `van_del_pol/`

  Q2 `linear_regression/`

  Q3 `graph_mst/`
- 問題ごとに，何をライブラリに記述し，何を `main.py` に記述するかは指定しています．
- **Jupyter notebook の形での提出は受け付けません．**指定されたコードを指定された場所に記述してください．
- ライブラリに記述するコードは，再利用可能かつユニットテストが可能であるように記述してください．
- もし既存のライブラリを利用する場合，弊社側でコードを動かせるように，使用したライブラリとそのバージョンを `requirements.txt` や `poetry.lock` などの形で掲載してください．
- コードは zip ファイルの形にまとめて提出してください．github などでコードを公開してしまわないように注意してください．

Q1. van del Pol 方程式の数値計算

van del Pol 方程式

van del Pol 方程式は，以下で表される 2 階の常微分方程式である．

$$x''(t) - a(1 - x(t)^2)x'(t) + x = 0$$

この方程式はオランダの電気工学者 B. van del Pol が，真空管電気回路内の安定な振動のモデルとして提案したものである．この方程式の解を数値計算で求め，その答えを適切に図示することが課題である．$y(t) = x'(t)$ とおけば，

$$\begin{aligned} x'(t) &= y(t) \\ y'(t) &= a(1 - x(t)^2)y(t) - x(t) \end{aligned}$$

というベクトル値の 1 階の常微分方程式になるので，以下この形で考える．

使う解法

一般のベクトル値の微分方程式 $x'(t) = F(x(t))$ に対して，その数値計算法は色々とあるが，ここでは以下の 2 つの手法を考える．ただし，τ は時間刻み幅とする．

陽的 Euler 法

$$x_{n+1} = x_n + \tau F(x_n)$$

Heun 法

$$\begin{aligned} k_n^{(1)} &= F(x_n) \\ k_n^{(2)} &= F(x_n + \tau k_n^{(1)}) \\ x_{n+1} &= x_n + \frac{\tau}{2}(k_n^{(1)} + k_n^{(2)}) \end{aligned}$$

課題

数値計算

van del Pol 方程式の解を，陽的 Euler 法と Heun 法で計算するコードを，`van_del_pol/`ディレクトリにライブラリとして作成せよ．

どのような入出力の関数を定義するかは受験者の自由であるが，その入出力の根拠は技術面接での説明対象となる．余力があれば他の手法を自身で調査して実装してもよい．（ボーナス）

図示

matplotlib パッケージを用いて，以下の数値計算の結果を図示し，その結果をファイルに保存するコードを `main.py` に書け．

- 時間に対する $x(t)$, $y(t)$ それぞれの推移のプロット
- $(x(t), y(t))$ の 2 次元空間上での軌跡

数値実験

次のケースにおいて，Euler 法と Heun 法での数値計算結果を図示せよ．

- $a = 2$ で $\tau = 0.1, 0.01, 0.001$ と変えたとき
- $a = 10$ で $\tau = 0.1, 0.01, 0.001$ と変えたとき

考察（ボーナス）

定数 a の値と刻み幅 τ を変えると，計算が破綻する場合がある．どのような場合にそうなるかを見つけ，その理由について調査せよ．

test/docstring（ボーナス）

自身の書いたコードに対して，適切な docstring と test を書け．

Q2. 線形回帰モデル

`data/regression_train.csv` と `data/regression_test.csv` は，人間の寸法データを測定したデータである．このデータを用いて，身長 (`height`)・体重 (`weight`)・年齢 (`age`) の 3 つを説明変数，他の寸法サイズ (`BicepC`) を目的変数とする線形回帰モデルを作り，その評価までを行うのが課題である．

線形回帰モデル

m 次元のベクトルからなる説明変数 $x(1 \times m$ 行列) と，1 次元の目的変数 y のペアで与えられる N 個のデータ

$$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$$

があるとする．

このデータを最もよく説明する線形回帰モデル $y = x\beta + \beta_0$ の係数 $\beta(m \times 1$ 行列) とバイアス β_0 を求めたい．その手法としては，以下で与えられる二乗誤差を最小化する最小二乗法が有名である．

$$E(\beta, \beta_0) = \sum_{j=1}^{N} [y_j - (x_j\beta + \beta_0)]^2 .$$

課題

A. 線形回帰（最小二乗法）

以下の 2 つの機能を持つ python ライブラリのコードを `linear_regression/`に記述せよ．

- N 個のデータが与えられたとき，上の $E(\beta, \beta_0)$ を最小化するような係数 β とバイアス β_0 を求める．
- 係数 β とバイアス β_0，そして，説明変数 x が与えられたとき，線形回帰モデルによる y の推定値を計算する．

ただし，scikit-learn など、線形回帰計算自体が行える既存のライブラリは使わず，numpy を用いて書くこと．また，入力データと出力の与え方は自由に決めて良い．

B. 線形回帰以外の回帰

線形回帰以外の回帰モデルを 1 つ実装せよ．

- **ただし，scikit-learn など、線形回帰計算自体が行える既存のライブラリは使わず**，numpy を用いて書くこと．
- python ライブラリのコードは，`linear_regression` 以外の適切な名前をつけたディレクトリに作成せよ．

データへの適用

以下を行うコードを `main.py` に記述せよ．

- 学習データ (data/regression_train.csv) について，身長 (height)・体重 (weight)・年齢 (age) の 3 つを説明変数，他の寸法サイズ (BicepC) を目的変数とし，上で実装した 2 つの回帰を行う．
- テストデータ (data/regression_test.csv) に対して上で作った線形回帰モデルを適用し，横軸に目的変数の真の値，縦軸に回帰モデルによる予測値を取った散布図を作り，ファイルに保存する．図を作成する場合，matplotlib を用いるとよい．
- 2 つの回帰について精度を計算し，比較せよ．

test/docstring（ボーナス）

自身の書いたコードに対して，適切な docstring と test を書け．

Q3. 最小全域木

最小全域木

（無向）グラフ G とは，頂点の集合 V と，頂点どうしを結ぶ（方向を持たない）辺の集合 E のペアのことである．各辺 $e \in E$ に対して，重みと呼ばれる実数 $w(e)$ が付与されているグラフは重み付きグラフと呼ばれる．

グラフの中で，閉路を持たず，さらに全体が連結であるものを木という．グラフ G に対し，G の全ての頂点を含むような部分グラフで木であるようなものを全域木という．全域木は複数ある可能性があるが，重み付きグラフ G において，辺の重みの合計が最小となるような全域木を最小全域木という．

最小全域木のアルゴリズム

最小全域木を求めるアルゴリズムには 2 つある．

Kruskal algorithm

これは以下のような手法である．

1. 求めたい全域木の辺の集合 E_T を空集合として初期化する．
2. 重みが最小の辺をまず選択し，E_T に付け加える．
3. 以降，それまでチェックしていない辺の中で，重み最小の辺 e を選択する．
   - E_T の辺と e で閉路が形成されなければ，E_T に e を追加する．
   - そうでない場合，e は E_T に追加せず，飛ばす．
4. 全ての頂点が E_T の辺で結ばれるまでこれを繰り返す．

Prim algorithm

これは以下のような手法である．

1. 求めたい全域木の頂点集合 V_T と辺集合 E_T を空集合で初期化する．
2. ランダムに頂点を選び V_T に追加する．
3. V_T に属する頂点と，V_T に属さない頂点を結ぶ辺のうち，重みが最小の辺 e を E_T に追加する．
4. e が $u \in V_T$ と $v \notin V_T$ を結んでいる場合，v を V_T に追加する．
5. V_T がすべての頂点を含むまでこれを繰り返す．

課題

アルゴリズムの実装

Kruskal algorithm，もしくは Prim algorithm のどちらかを用いて最小全域木を求めるコードを python ライブラリとして `graph_mst/`に記述せよ．

ただし，グラフの入力は重みを成分とする隣接行列 (numpy の 2 次元配列) で与えられる．すなわち，例

えば

```
array(
[[0, 2, 0],
 [2, 0, 1],
 [0, 1, 0]]
)
```

の場合，頂点は 3 つであり，それを v_1, v_2, v_3 とするとき，v_1 と v_2 の間の辺の重みが 2，v_2 と v_3 の辺の間の重みが 1 で，それ以外に辺はないことを意味する．（無向グラフなので対称行列になることに注意せよ．）

どのような出力にするかは自由とする．

サンプル CSV からの計算

`data/graph_adj.csv` は，頂点が 50 あるグラフの隣接行列を CSV ファイルで書き出したものである．この CSV ファイルを読み込み，上の最小全域木を求めるアルゴリズムを適用した結果をファイルに出力するコードを `main.py` に書け．結果は頂点番号を 0〜49 とするとき，

```
0,1
1,10
...
```

のように，1 行ごとに，最小全域木の辺を，その端点となる頂点番号をカンマ区切りで書くことによって出力せよ．

test/docstring（ボーナス）

自身のコードに適切なテストと docstring をつけよ．