

全体ゼミでの振り返り

2026年5月29日

923044 高宮悠聖

今週行ったこと(5/29)

- ・ プログラムの内容確認
- ・ アプリ作成

プログラムの内容確認

ソースコード確認

- ・ 内容を変更するために使用するプログラムの内容を確認
 - ▶ コメントアウトで分かりやすいように説明を記述

```
utils.py M visualize.py M
social-hm visualize.py M Data_loader / frame_preprocess
1 import os # OS操作ライブラリ フォルダ作成,ファイル存在確認,パス結合などに使用
2 import pickle # Pythonオブジェクト保存用,Social-LSTMでは,txtデータを前処理後, .cpkl ファイルとして保存などに使用
3 import numpy as np # 数値計算ライブラリ,座標配列管理,歩行軌跡保存,フレーム操作,行列計算
4 import pandas as pd # 表形式データ処理ライブラリ,txt/csv読み込みに使用
5 import random # ランダム処理用,validationデータ選択などに使用
6 import torch # PyTorch本体,AI計算ライブラリ,Tensor計算,LSTM,GPU計算,順逆伝播
7 import math # 数学関数ライブラリ
8 from torch.autograd import Variable # PyTorchの学習可能Tensor,現在のPyTorchではTensorと統合されているため本来不要,ただし古いSocial-LSTM実装では使用
9 from helper import * # helper.py にある便利関数を使用,utils.py の補綴
10
11
12 # DataLoaderクラス
13 # 役割:「歩行データをAI学習できる形に変換」,train.py から呼ばれる
14 # 流れ:txtデータ->DataLoader->sequence->model.pyへ入力
15 class DataLoader():
16     def __init__(self,
17                 f_path, # Social-LSTM のプロジェクトのルートフォルダ位置,絶対パスに変換している
18                 batch_size=5, # 一度に学習する系列数
19                 seq_length=20, # 1系列のフレーム長
20                 num_of_validation=0, # validationに使用するデータ数
21                 forcePreProcess=False, # 前処理を強制再実行
22                 infer=False, # 推論(test)モードか
23                 generate=False # データ生成モード
24             ):
25
26     # テストデータ一覧
27     # 学習後の性能評価用,各txt:frame_id ped_id y x を持つ
28     base_test_dataset = [
29         '/data/test/biwi/biwi_eth.txt', # BIWI dataset
30         '/data/test/crowds/crowds_zara01.txt', # crowds dataset
31         '/data/test/crowds/uni_examples.txt',
32
```

プログラムの内容確認 (ii)

式確認

$$H_t^i(m, n, :) = \sum_{j \in N_i} 1_{mn} [x_t^j - x_t^i, y_t^j - y_t^i] h_{t-1}^j$$

$H_t^i(m, n, :)$:

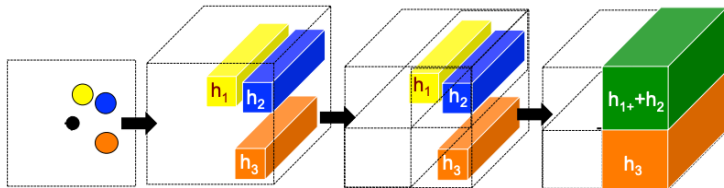
時刻 t のとき i 番目の人が、グリッド位置 m, n 内にいるときの hidden state を集めた周囲情報マップ

$\sum_{j \in N_i}$: i の近くにいる周囲の人 j たち全員を足し合わせる

$x_t^j - x_t^i, y_t^j - y_t^i$: j が i から見て縦横にどれだけ離れてるか

$1_{mn}[\dots]$: 指示関数, j が (m, n) マスにいたら 1, 違えば 0,

h_{t-1}^j : j 番目の人の LSTM 記憶



プログラムの内容確認 (iii)

$$e_t^i = \varphi(x_t^i, y_t^i; W_e)$$

e_t^i : 自分の位置特徴ベクトル

x_t^i, y_t^i : i 番目の人の現在座標

$\varphi()$: 変換関数,線形変換 + 活性化関数(ReLU)

W_e : 重み

$$a_t^i = \varphi(H_t^i; W_a)$$

a_t^i : 周囲状況を表す特徴ベクトル

$\varphi(H_t^i; W_a)$: 周囲情報を特徴ベクトルへ変換

$$h_t^i = \text{LSTM}(h_{t-1}^i, e_t^i, a_t^i; W_l)$$

h_t^i : i 番目の人の更新した新しい hidden state

h_{t-1}^i : 前回の hidden state

e_t^i : 自分の位置特徴ベクトル

a_t^i : 周囲状況を表す特徴ベクトル

プログラムの内容確認 (iv)

$$(\hat{x}, \hat{y})_t^i \sim N(\mu_t^i, \sigma_t^i, \rho_t^i)$$

$(\hat{x}, \hat{y})_t^i$: 予測された未来位置

$N(\mu_t^i, \sigma_t^i, \rho_t^i)$: 正規分布 (ガウス分布)

μ_t^i : 予測位置の中心

σ_t^i : 予測の広がり

ρ_t^i : x 方向と y 方向の関連

$$[\mu_t^i, \sigma_t^i, \rho_t^i] = W_p h_{t-1}^i$$

$[\mu_t^i, \sigma_t^i, \rho_t^i]$: 未来位置分布の設定値

h_{t-1}^i : 前回の hidden state

W_p : 出力用の重み

プログラムの内容確認 (v)

$$L^i(W_e, W_l, W_p) = - \sum_{t=T_{\text{obs}}+1}^{T_{\text{pred}}} \log(P(x_t^i, y_t^i \mid \mu_t^i, \sigma_t^i, \rho_t^i))$$

$L^i(W_e, W_l, W_p)$: 損失関数 (Loss)

W_e : 自分位置 embedding, W_l : LSTM 内部, W_p : 出力変換

$-\sum$: 未来時間全部の誤差を足す,

正解確率は大きいほどいいので $-$ をつけて Loss を最小化するようにしている

$t = T_{\text{obs}} + 1$: 観測終了後から, T_{pred} : 予測終了時刻

$P(x_t^i, y_t^i \mid \mu_t^i, \sigma_t^i, \rho_t^i)$: 本当の位置が予測分布でどれくらい確率高いか

$-\log(P)$: 予測が外れるほど大きくなる

アプリ作成

- ・カラオケにあった「狩歌」というボードゲームが面白かったので、スマホでもできるように作成した
- ・ルール



<https://www.xaquinel.com/works/caruita-basic-set>

アプリ作成 (ii)

WORD HUNTING

2人

3人

4人

P2 : 10

P4 : 3

ありがとう 3点	Forever 4点	夢 1点	夢 1点
ありがとう 3点	夢 1点	Forever 4点	Forever 4点
夢 1点	好き 1点	永遠 2点	Forever 4点
好き 1点	好き 1点	未来 1点	未来 1点

FINISH

P3 : 7

P1 : 9

RESULT

P1 : 9

P2 : 10

P3 : 7

P4 : 3

WINNER
PLAYER2

TITLE