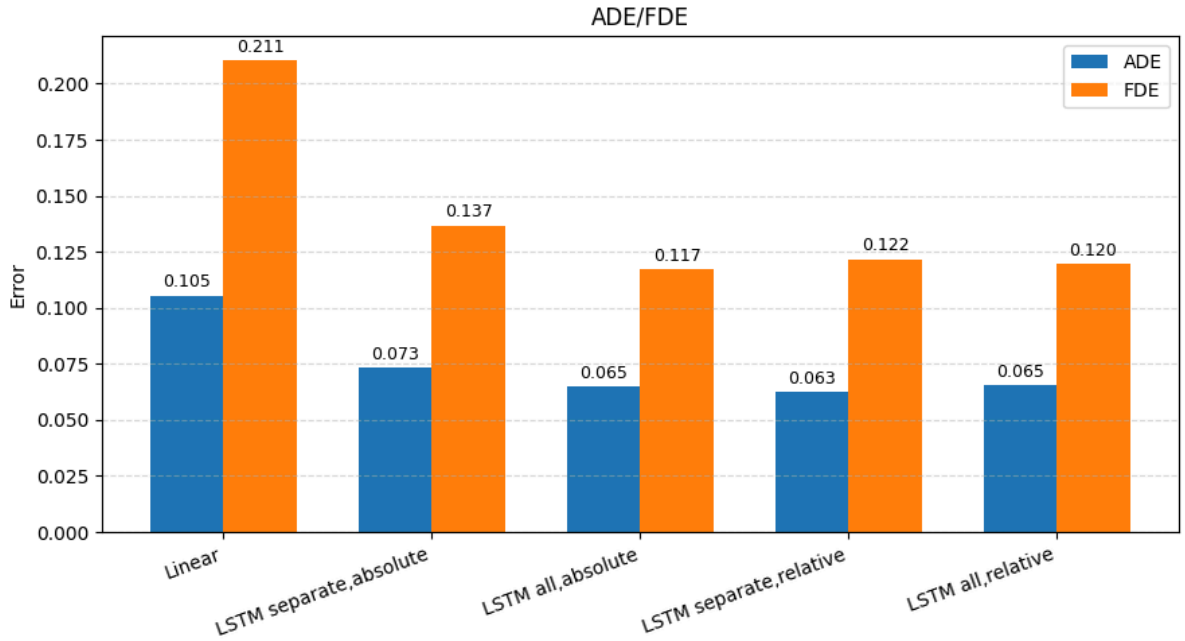
202410178

今村隼人

経営と機械学習レポート



グラフから ADE と FDE では ADE の方が小さいことがわかる。 それぞれ誤差の絶対値を評価しているため平均の誤差よりも最終地点の誤差の方が大きいということになる。これは予測する値のうち入力データから離れるほど予測からずれやすいということとも一致する。

二次元と一次元について、座標を絶対値で入力した場合は二次元で入力した方が精度が高く、相対値では FDE は一次元の方がわずかに精度がよく ADE は二次元の方が精度がよい。このことから二次元でデータを入力した方が精度がよいと予想できる。歩行者の X 軸と Y 軸方向の移動の関係性などもデータに含められるため二次元データの方が精度がよいと予想できる。

xy 別の場合相対値の方が精度が高く二次元データの場合は絶対値の方が精度が高くなっている。絶対値の場合相対値に比べて場所の情報も学習することができるのでより複雑な情報を扱うことができるようになると予測できる。一次元と二次元で絶対値と相対値のどちらの精度が高いのか異なるのは LSTM のモデルと今回のデータ数による説明できる現象の複雑さにより説明する力が制限されているからだと予想できる。

LSTM のいずれのモデルも線形予測よりも精度が高いことがわかる。人の動きは線形ではなくより複雑な動きをしていることが予想できる。

これまでの手法では座標のみを用いて学習していたので、座標に加えて速度と加速度を加えて学習を行うようにした。 具体的には下のようなコードで実装している。

```py
class TrajectoryDatasetWithDerivatives(Dataset):
    def __init__(self, df, obs_len=20, pred_len=30, stride=1, normalize=True):

        self.obs_len = obs_len
        self.pred_len = pred_len
        self.seq_len = obs_len + pred_len
        self.normalize = normalize

        self.seqs = []
        # modeはx,y両方を使う'all'に固定
        mode = 'all'

        for tid, g in df.groupby("traj_id"):
            g = g.sort_values("frame")

            xy = g[["x","y"]].to_numpy(dtype=np.float32)

            if len(xy) < self.seq_len:
                continue

            vel = np.diff(xy, axis=0)
            vel = np.concatenate([np.zeros((1, 2), dtype=np.float32), vel], axis=0)

            acc = np.diff(vel, axis=0)
            acc = np.concatenate([np.zeros((1, 2), dtype=np.float32), acc], axis=0)

            pva = np.concatenate([xy, vel, acc], axis=1)

            # ウィンドウ切り出し
            for s in range(0, len(pva)-self.seq_len+1, stride):
                subseq = pva[s:s+self.seq_len]
                self.seqs.append(subseq)

        # shapeを揃える (人数, 50, 6)
        if len(self.seqs) > 0:
            self.seqs = np.stack(self.seqs, axis=0)
        else:
            D = 6 # pos(2) + vel(2) + acc(2)
            self.seqs = np.zeros((0, self.seq_len, D), dtype=np.float32)

        D = self.seqs.shape[2]

```

```
        if self.normalize and len(self.seqs) > 0:
            obs_data = self.seqs[:,:self.obs_len,:]
            self.mean = obs_data.reshape(-1, D).mean(axis=0, keepdims=True)
            self.std  = obs_data.reshape(-1, D).std(axis=0, keepdims=True) +
1e-6
        else:
            self.mean = np.zeros((1,D), dtype=np.float32)
            self.std  = np.ones((1,D), dtype=np.float32)

```

xy を合わせた二次元のデータとして、速度はある時刻の座標のひとつ前の座標との差分としている。加速度はある時刻の速度のひとつ前の速度との差分としている。これらを合わせた 6 次元のデータとして学習している。これにより単に座標だけを学習しているモデルと比べて大きく精度を上げることができた。これはモデルが過去の軌道の「状態」をより正確に表現できるようになったためだと考えられる。座標のみの入力では、モデルは位置の系列から暗黙的に動き方を抽出しなければならなかった。これに対し、速度と加速度を明示的に入力特徴量として与えることで、モデルは学習の初期段階から歩行者の「勢い」や「向きの変化」といった情報を直接利用できるようになった。結果として、特に軌道が非線形に変化する場面（例：方向転換、加減速）において、未来の動きをより的確に予測することが可能となり、大幅な精度向上に繋がった。

また評価関数として ADE のみを採用しているところを ADE と FDE との和にした。このようにすることで最終的な地点の距離の違いを重く評価するようにした。ほかにも ADE の 2 乗や 0.5 乗なども試したが ADE と FDE の和が最も精度が高かった。これは人が移動するときは行き先があってそこに向かって移動しているという現実の事象をよりうまく表現できているからだと考えられる。