

深層学習入門 第2回

東海大学 情報通信学研究科 情報通信学専攻
南 直輝

今日の内容

- ・タイタニック号の乗客者の年齢や性別などの項目がある表形式データセットを用いて、生存予測をする。(PyTorch を用いる)

データセットをダウンロードし内容を確認する

- support.py を取得し、インポートする。
- support.py にある fetch_titanic 関数を用い、訓練用データとテスト用データに分割する。
- x_train の情報(オブジェクトの型や Column など)を確認する。

```
[1] 1 !curl -O https://raw.githubusercontent.com/khirotaka/tigerfish/master/utis/support.py
```

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
			Dload Upload	Total	Spent	Left	Speed
100	588	100	588	0	0	3398	0
--:--:--	--:--:--	--:--:--	--:--:--	--:--:--	--:--:--	--:--:--	3398

```
[2] 1 import support
```

```
[3] 1 (x_train, y_train), x_test = support.fetch_titanic()
```

```
[4] 1 x_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 623 entries, 857 to 684
Data columns (total 14 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   pclass      623 non-null   int64
1   sex         623 non-null   object
2   age         502 non-null   float64
3   sibsp       623 non-null   int64
4   parch       623 non-null   int64
5   fare        623 non-null   float64
6   embarked    621 non-null   object
7   class       623 non-null   category
8   who         623 non-null   object
9   adult_male  623 non-null   bool
10  deck        151 non-null   category
11  embark_town 621 non-null   object
12  alive       623 non-null   object
13  alone       623 non-null   bool
dtypes: bool(2), category(2), float64(2), int64(3), object(5)
memory usage: 56.4+ KB
```

データセットをダウンロードし内容を確認する 続き

- 先頭の5行を確認する。x_trainは、DataFrame型なので .head() が使える。
- カラム内にあるnullの合計を確認する。

```
[5] 1 x_train.head()
```

	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
857	1	male	51.0	0	0	26.5500	S	First	man	True	E	Southampton	yes	True
52	1	female	49.0	1	0	76.7292	C	First	woman	False	D	Cherbourg	yes	False
386	3	male	1.0	5	2	46.9000	S	Third	child	False	NaN	Southampton	no	False
124	1	male	54.0	0	1	77.2875	S	First	man	True	D	Southampton	no	False
578	3	female	NaN	1	0	14.4583	C	Third	woman	False	NaN	Cherbourg	no	False

```
[6] 1 x_train.isnull().sum()
```

```
pclass      0
sex          0
age         121
sibsp        0
parch        0
fare         0
embarked      2
class        0
who          0
adult_male   0
deck        472
embark_town   2
alive        0
alone        0
dtype: int64
```

データの前処理

- 4つのカラムでnullがあったので、そのカラムを削除する。
- カラムが削除されているか確認する。

```
[7] 1 x_train = x_train.drop(["age", "embarked", "deck", "embark_town"], axis=1)
```

```
[8] 1 x_train.isnull().sum()
```

```
pclass      0
sex          0
sibsp       0
parch       0
fare        0
class       0
who         0
adult_male  0
alive       0
alone       0
dtype: int64
```

データ前処理 続き

- pandas のメソッド、`get_dummies()` で、文字列でカテゴリー分けされた性別などのデータを、男1 女0 のように変換する。
- 変換されたことを確認する。

```
[9] 1 import pandas as pd
```

```
[10] 1 x_train = pd.get_dummies(x_train)
```

```
[11] 1 x_train.head()
```

	pclass	sibsp	parch	fare	adult_male	alone	sex_female	sex_male	class_First	class_Second	class_Third
857	1	0	0	26.5500	True	True	0	1	1	0	0
52	1	1	0	76.7292	False	False	1	0	1	0	0
386	3	5	2	46.9000	False	False	0	1	0	0	1
124	1	0	1	77.2875	True	False	0	1	1	0	0
578	3	1	0	14.4583	False	False	1	0	0	0	1

ライブラリのインポート

- PyTorch ライブラリのインポートする。
- x_train と y_train を tensor型に変換する。

```
[12] 1 import torch
      2 import torch.nn as nn
      3 import torch.optim as optim
      4 from torch.utils.data import Dataset, DataLoader
      5
      6 import numpy as np

[13] 1 data = torch.tensor(x_train.values.astype(np.float32))
      2 print(data[:5])
      3 print(data.shape)
      4 print("-" * 50)
      5 targets = torch.tensor(y_train.values.astype(np.int64))
      6 print(targets[:5])
      7 print(targets.shape)

tensor([[ 1.0000,  0.0000,  0.0000, 26.5500,  1.0000,  1.0000,  0.0000,  1.0000,
          1.0000,  0.0000,  0.0000,  0.0000,  1.0000,  0.0000,  0.0000,  1.0000],
        [ 1.0000,  1.0000,  0.0000, 76.7292,  0.0000,  0.0000,  1.0000,  0.0000,
          1.0000,  0.0000,  0.0000,  0.0000,  0.0000,  1.0000,  0.0000,  1.0000],
        [ 3.0000,  5.0000,  2.0000, 46.9000,  0.0000,  0.0000,  0.0000,  1.0000,
          0.0000,  0.0000,  1.0000,  1.0000,  0.0000,  0.0000,  1.0000,  0.0000],
        [ 1.0000,  0.0000,  1.0000, 77.2875,  1.0000,  0.0000,  0.0000,  1.0000,
          1.0000,  0.0000,  0.0000,  0.0000,  1.0000,  0.0000,  1.0000,  0.0000],
        [ 3.0000,  1.0000,  0.0000, 14.4583,  0.0000,  0.0000,  1.0000,  0.0000,
          0.0000,  0.0000,  1.0000,  0.0000,  0.0000,  1.0000,  1.0000,  0.0000]])
torch.Size([623, 16])
-----
tensor([1, 1, 0, 0, 0])
torch.Size([623])
```

Datasetとネットワークの作成

- Dataset を作成する。
- ネットワークを作成する。自作してみましょう！
 - __init__ : 使う層を定義する
 - forward : データの流れを定義する

```
[14] 1 class TitanicDataset(Dataset):
      2     def __init__(self, x, y):
      3         self.data = x
      4         self.targets = y
      5
      6     def __len__(self):
      7         return len(self.targets)
      8
      9     def __getitem__(self, item):
     10         x = self.data[item]
     11         y = self.targets[item]
     12         return x, y
```

```
[15] 1 train_ds = TitanicDataset(data, targets)
      2 train_loader = DataLoader(train_ds, batch_size=32, shuffle=True)
```

```
[16] 1 # class Model(nn.Module):
      2 #     def __init__(self, in_features, mid_features, n_class):
      3 #         def forward(self, x):
```

入力層 1 層、中間層 2 層、出力層 1 層 の
合計 4 層のネットワークを実装する。

モデルの学習

- 作成したネットワークのインスタンスを生成し、最適化アルゴリズムと損失関数を設定する。
- モデルの学習をする。

```
[17] 1 model = Model(16, 10, 2)
      2 optimizier = optim.Adam(model.parameters()) # 最適化アルゴリズム
      3 criterion = nn.CrossEntropyLoss() # 損失関数

[18] 1 from tqdm.auto import tqdm

[19] 1 epochs = 10
      2
      3 with tqdm(total=epochs) as pbar:
      4     for epoch in range(epochs):
      5         for batch in train_loader:
      6             a, b = batch
      7             optimizier.zero_grad()
      8             out = model(a)
      9             loss = criterion(out, b)
     10             loss.backward()
     11             optimizier.step()
     12             pbar.set_description("loss: {:.4f}".format(loss.detach().item()))
     13             pbar.update(1)
```

loss: 0.4162: 100%  10/10 [00:00<00:00, 10.48it/s]

学習したモデルを評価

- 学習済みのモデルを、テストデータを用いて、f1_score で評価する。

```
[20] 1 x_test = x_test.drop(["age", "embarked", "deck", "embark_town"], axis=1)
      2 x_test = pd.get_dummies(x_test)
      3 x_test = torch.tensor(x_test.values.astype(np.float32))
      4 with torch.no_grad():
      5     result = model(x_test)
      6     predicts = torch.max(result, 1)[1]
```

```
[21] 1 support.calc_score(predicts)    # f1_score, 最大値1、最小値0
```

```
0.8323699421965317
```