1　ニューラルネットワーク入門

A　機械学習の概要

機械学習では一般に、なんらかの値を入力することで適切な出力を返すような式を作ることを目指します。

1. ワンルームの賃料予測であれば、 **敷地面積**，**築年数**，**駅からの距離** などから **月当たり賃料** の予測値を計算する式を作ります。
2. 顧客の店舗回遊に関する情報 **食品棚の訪問回数**，**衣類棚の訪問回数**，**雑貨棚の訪問回数** などから店舗回遊の仕方が類似した顧客をグルーピングし、あるお客さんがどの **グループ** に該当するのかを計算する式を作ることも考えられます。

このような式を「予測式」ということにしましょう。機械学習では、予測式を得るために**データ**を用います。ワンルームの賃料予測の例であれば、

物件id	敷地面積	築年数	駅からの距離	月あたりの賃料
1	25.0	10	3	10.2
2	31.0	9	5	14.8
3	29.0	1	8	11.4
⋮	⋮	⋮	⋮	⋮

のようなデータを準備します。

問題A1　みなさんも自由に機械学習の課題を考えてみましょう。

問題A2

1. 例2について、どんなデータを準備すればよいかを考えてみてください。
2. 例2は例1と異なる点があります。「出力」という言葉をヒントに、なにが異なるのかを明らかにしてください。
3. 「**教師あり学習**」「**教師なし学習**」という言葉の意味を説明し、例1と例2がそれぞれどちらに該当するかを答えてください。

B　ニューラルネットワークとはなにか

機械学習には、予測式を作るための「フレームワーク」がたくさん提案されています。**ニューラルネットワーク**は、この予測式を作るフレームワークの一つです。また一口にニューラルネットワークといっても、

モデル名	用途
多層ニューラルネットワーク, MLP	一般
畳み込みニューラルネットワーク, CNN	主に画像データ
残差ネットワーク, ResNet	主に画像データ
Elmanの回帰結合型ニューラルネットワーク, Elman RNN	主に系列データ
長・短期記憶, LSTM	主に系列データ
Transformer	主に系列データ
YOLO v3	物体検出
UNet	セマンティックセグメンテーション
⋮	⋮

などなど、用途に応じた「モデル」がたくさんあります。

問題B1　予測式を作るフレームワークまたはモデルとして、もしニューラルネットワークのほかに聞いたことがあるものがあれば、挙げてみてください。

C　MLPを動かしてみよう

試しに多層ニューラルネットワーク（以下、MLPと略す）を動かして、ニューラルネットワークを動かすときのイメージを掴んでおきましょう。

課題設定

課題　ワインのさまざまな特徴から味の評価を予測してみよう。

今回は130種類のワインについて、13の特徴とワインの評価がレコードされたデータセットを準備しました。 `data` ディレクトリに `wine.csv` というファイル名で準備してあります。

In [1]:

```
# データセットの読み込み
import pandas as pd

data = pd.read_csv("./data/wine.csv")
data.head(n = 5)
```

Out[1]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	n
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	

味の評価は2段階 `0/1` で与えられていて、 `target` 列に記録されています。

In [2]:

```
# 含まれているラベルの割合の確認
data.target.value_counts()
```

Out[2]:

```
1    71
0    59
Name: target, dtype: int64
```

データの様子の確認

入力する変数のヒストグラム（正確にはカーネル密度推定）と散布図を、味の評価で色分けして図示しておきます。ヒストグラムや散布図からは

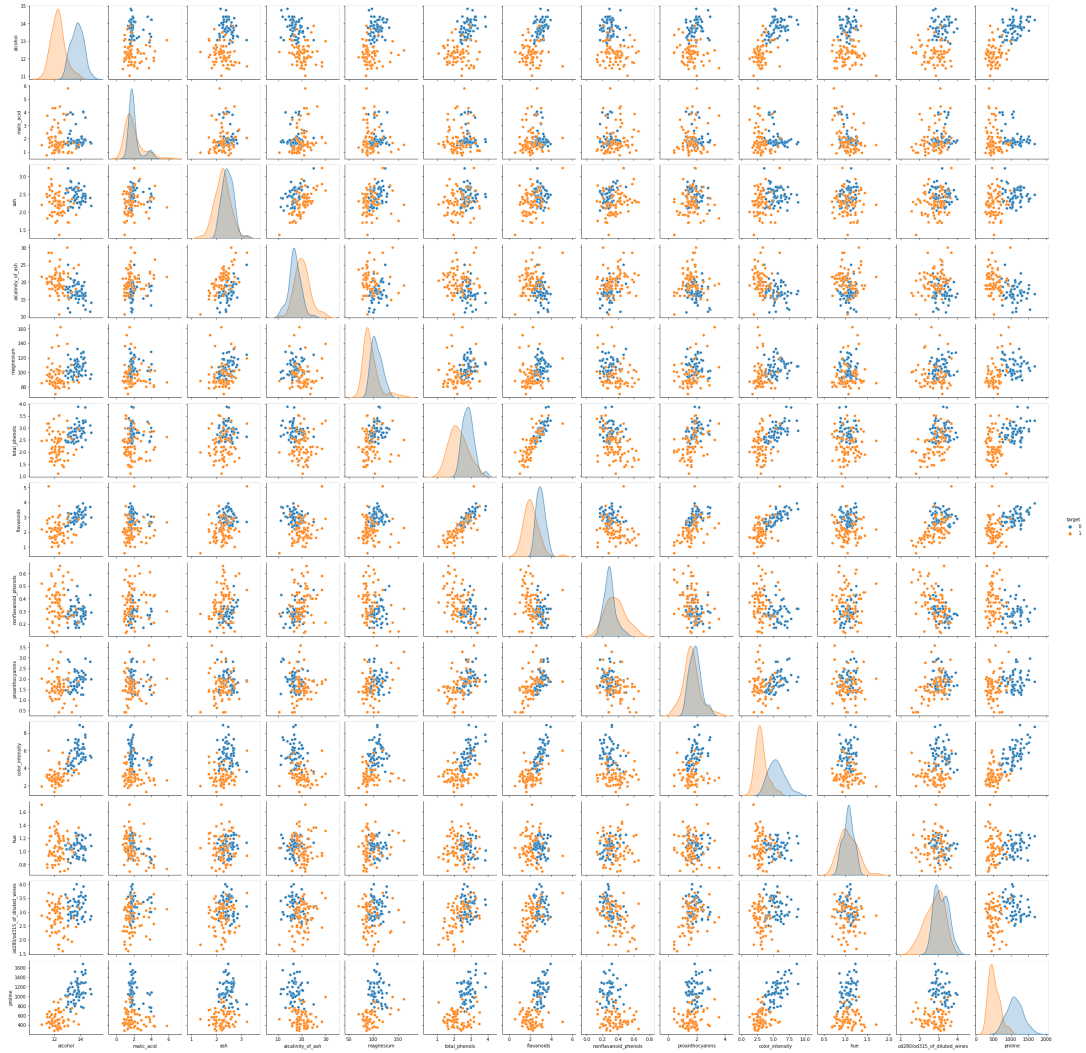
- この予測がどれくらい難しいものなのか
- どのあたりに注目するとうまく分類できそうか

といったことを考えるヒントが得られます。

In [3]:

```
# データの可視化
import matplotlib.pyplot as plt
import seaborn as sns

sns.pairplot(data, hue = "target")
plt.show()
```



前処理

今回は予測式を作ったあとで、データにないような新しいワインに対して、どれくらいうまく予測できるかを評価することにします。しかし「データにないような新しいワイン」に対する予測式の評価をどうやってやるのでしょうか。今回は、まず手元のデータを

- 訓練データ：モデルを学習させて予測式を得るためのデータ
- テストデータ: 学習には用いないデータ

に分割しておきます。次に、訓練データを用いてモデルを学習させます。最後に得られた予測式をテストデータで評価します。これを**holdout検証**といいます。

Remark　予測式を得る計算のことを**学習**といいます。

In [4]:

```
# データを訓練データとテストデータに分割
from sklearn.model_selection import train_test_split

X = data.drop("target", axis = 1).values
y = data["target"].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

X_train.shape, X_test.shape      # 訓練データ104件、テストデータ26件
```

Out[4]:

```
((104, 13), (26, 13))
```

ニューラルネットワークの計算を行う際には、入力する変数の正規化を行うことが一般的です。この理由は講座の今後の授業のなかで説明を与えます。

In [5]:

```
# データを正規化
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()
ss.fit(X_train)

X_train, X_test = ss.transform(X_train), ss.transform(X_test)
```

sklearn を用いたMLPのコーディング

MLPは `sklearn.neural_network` モジュールの `MLPClassifier` クラスを用いて計算できます。設定する値は主に

- アーキテクチャ
- 学習アルゴリズム

の2種類に分類できます。これらの値はMLPの仕組みに関わっているので、次回以降に詳しく解説を与えます。

In [6]:

```
# sklearnを用いたMLPのコーディング
from sklearn.neural_network import MLPClassifier

mlp = MLPClassifier(hidden_layer_sizes = (5, ),     # アーキテクチャ
                    activation = "relu",
                    batch_size = int(104/2),     # 学習アルゴリズム
                    solver = "sgd",
                    learning_rate = "constant",
                    learning_rate_init = 0.1,
                    early_stopping = True,
                    n_iter_no_change = 10,
                    validation_fraction = 0.2)

mlp.fit(X_train, y_train)
```

Out[6]:

```
MLPClassifier(batch_size=52, early_stopping=True, hidden_layer_sizes=(5,),
              learning_rate_init=0.1, solver='sgd', validation_fraction=0.2)
```

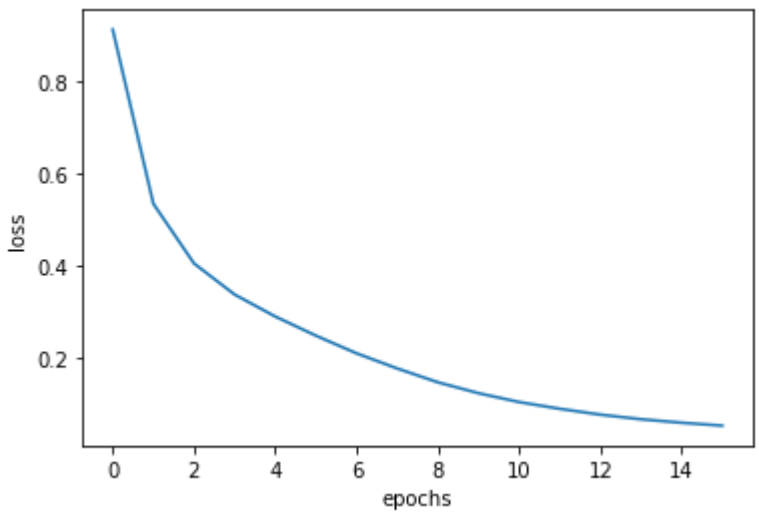
学習曲線

MLPの学習は、訓練データで予測したラベルと実際のラベルの食い違い（**損失**）が小さくなるように、予測式を更新することで行われます。更新1回あたりを**epoch**といいます。横軸にepoch、縦軸に損失の値をとったグラフをかくことでその更新の様子を知ることができます。これを**学習曲線**といいます。

今回はepochが進むごとに順調に損失が小さくなっていることが確認できます。後の授業で詳しく解説しますが、例えば「学習率」 `learning_rate_init` という値が適切に設定できていないと、学習曲線がepochが進むごとに上昇したり、少しずつしか小さくならないことがあります。

In [7]:

```
# 学習曲線
plt.plot(mlp.loss_curve_)
plt.xlabel("epochs")
plt.ylabel("loss")
plt.show()
```



モデルの評価

正解のラベルと予測したラベルをクロス集計表で比較します。このようなクロス集計表を**混同行列**といいます。

In [8]:

```
# テストデータでの混同行列
from sklearn.metrics import confusion_matrix

pred_test = mlp.predict(X_test)
confusion_matrix(y_test, pred_test)
```

Out[8]:

```
array([[14,  0],
       [ 0, 12]])
```

In [9]:

```
# 訓練データでの混同行列
pred_train = mlp.predict(X_train)
confusion_matrix(y_train, pred_train)
```

Out[9]:

```
array([[45,  0],
       [ 0, 59]])
```

D　PyTorchとはなにか

PyTorch はPython言語でニューラルネットワークをコーディングするためのパッケージです。同様のパッケージには Tensorflow , JAX , MXNet などがあります。現在は、多くの人が PyTorch か Tensorflow またはその両方を用いています。

さきほどは sklearn パッケージを用いてMLPの計算を行いました。しかし、 sklearn だとMLP以外のニューラルネットワークのモデルは計算できません。そこで、この講座では PyTorch を用いてニューラルネットワークをコーディングしていきます。

E　今後の目標

ニューラルネットワークは、決定木などの機械学習フレームワークに比べて、様々なタスクに対してモデルが提案されてきました。これは「ニューラルネットワークとは**予測式の柔軟なモデリング**を可能にしたフレームワークである」という性質から来ています。

私たちは、MLPという最もシンプルなモデルから勉強をはじめます。そして MLPをスタート地点に、さまざまなモデルを勉強していきます。各モデルには、タスクが持つ固有の事情に応じた予測式のモデリングのアイディアが含まれています。

そこで、この講座では

- ニューラルネットワークの仕組みをMLPを例に理解すること。
- MLPと比較しながら、さまざまなモデルに含まれるアイディアに親しむこと。

を目標にニューラルネットワークのさまざまなモデルを紹介していきます。

略解

問題A1　自由に考えてみましょう。

問題A2

1. 以下のようになります。

顧客id	食品棚の訪問回数	衣類棚の訪問回数	雑貨棚の訪問回数
1	3	0	1
2	0	10	4
3	5	3	0
⋮	⋮	⋮	⋮

1. 例1では出力したい変数 **月あたりの賃料** を準備するのに対し、例2では出力したい変数 **グループ** は準備しない。つまり、出力したい変数を準備するかしないかが異なる。
2. 出力したい変数をデータに準備して予測式を作る方法を「教師あり学習」、準備せずに予測式を作る方法を「教師なし学習」という。例1は教師あり学習、例2は教師なし学習です。

問題B1　決定木分析、ランダムフォレスト、勾配ブースティング、サポートベクトルマシン、k-means法、階層クラスタリング、主成分分析、t-SNE、UMAPなど。