機械学習入門

経済学部 BX584

k-近傍法（最近傍法）

MNISTデータ

- 手書き数字画像データセット
  - 超有名なデータセット
  - 深層学習のベンチマークにも良く使われる
  - 合計70,000枚の画像
    - 0～9の画像数：6903 7877 6990 7141 6824 6313 6876 7293 6825 6958
  - 28x28ピクセルのグレースケール画像
- 参考URL

http://scikit-learn.org/stable/auto_examples/neural_networks/plot_mnist_filters.html#sphx-glr-auto-examples-neural-networks-plot-mnist-filters-py

しかし…

- MNISTデータを今日説明する最近傍法に使うと時間がかかりすぎる

- 最近傍法の特徴
  - 利点：原理が単純
  - 欠点：計算時間が長い

- そのため、scikit-learnのdigitsデータセットを使うことに
  - 8x8ピクセルの手書き数字画像
  - 画像数も合計2,000枚に達しない

k-近傍法(k-nearest neighbors)

- データの分類に使える手法のひとつ
- モデルを使わない手法！（個々のデータをそのまま使う）
- ラベルを予測したいデータと最も類似したk個の訓練データを求める
- そのk個で多数決を採り、最多数のラベルを答えにする
  - 利点：簡単
  - 欠点：すべてのデータとの類似度を計算する必要がある（計算量大）

scikit-learnのdigitsデータセットの読み込み

```
from sklearn.datasets import load_digits

digits = load_digits()
```

digitsデータセットの可視化

```
import matplotlib.pyplot as plt


# 添え字が100のデータを可視化する
# クラスラベルも表示する
plt.imshow(digits.data[100].reshape(8,8))
print('class', digits.target[100])
```

問1

- 添え字0のデータと添え字10のデータのユークリッド距離を求める
- 添え字0のデータと添え字1のデータのユークリッド距離を求める
- どちらのユークリッド距離のほうが小さいか？

訓練データ/検証データ/テストデータ

- 訓練データ
  - 正解が分かっているとしてよいデータ
- 検証データ
  - ハイパーパラメータを調整するために使うデータ
    - ハイパーパラメータ：人間が手動で調整しないといけないパラメータ
  - 予測と正解をつきあわせて性能を評価
  - 最も良い性能を出すハイパーパラメータの値を探る
- テストデータ
  - 正解を予測するために使うデータ
  - テストデータでの評価は最終的な評価
    - この評価結果を見てハイパーパラメータを変えてはいけない！

digitsデータセットを訓練/検証/テスト・データに分割

```
# 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_train, X_test, y_train, y_test = train_test_split(digits.data,
                                                    digits.target, test_size=0.2,
                                                    random_state=0)

X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train,
                                                      test_size=0.25,
                                                      random_state=0)

print(X_train.shape, X_valid.shape, X_test.shape)
```

すべての訓練データとのユークリッド距離を求める

```
import numpy as np

for i in range(len(X_train)):
    print(np.linalg.norm(X_train[i] - X_valid[0]))
```

ブロードキャストを使うと・・・

```
dist = np.linalg.norm(X_train - X_valid[0], axis=1)


print(y_train[np.argmin(dist)] == y_valid[0])
#この行では何をしているでしょうか？
```

k-近傍法の性質

- 方法としてシンプルなので何をやっているか理解しやすい
  - Facebookの大規模類似度検索ライブラリFaiss

    https://github.com/facebookresearch/faiss
- 計算量が大きい
  - テストデータひとつごとにすべての訓練データをスキャン
    - だからFacebookのライブラリのような高速化が必要

最近傍法からモデルに基づくアプローチへ

- 最近傍法の最大の特徴はモデルに基づかないこと
  - モデルに基づかない＝データそのものしか使わない
    - 最近傍法では、入力と出力を単にペアとして扱い、その間の関連性を考えない。
  - モデルに基づく＝データが何らかの構造を持つと仮定する
- データが持つ構造とは？

例）線形性 ＝ 出力が入力の線形関数として書ける

付録：MNISTデータの扱い方

MNISTデータの読み込み

```
from sklearn.datasets import fetch_mldata


mnist = fetch_openml('mnist_784')
X, y = mnist.data, mnist.target
print(len(X))
print(len(y))
```

MNISTデータの最初の100個を可視化

```
fig, axes = plt.subplots(10, 10)
for x, ax in zip(X, axes.ravel()):
    ax.axis('off') # 軸を消す
    ax.matshow(x.reshape(28, 28), cmap=plt.cm.gray)
plt.show()
# subplotsメソッドは、複数のグラフを一度に描くのに使える。
```