kNN、k近傍法
k-Nearest Neighbors

1. 概要
2. アルゴリズムと距離
3. 例

概要

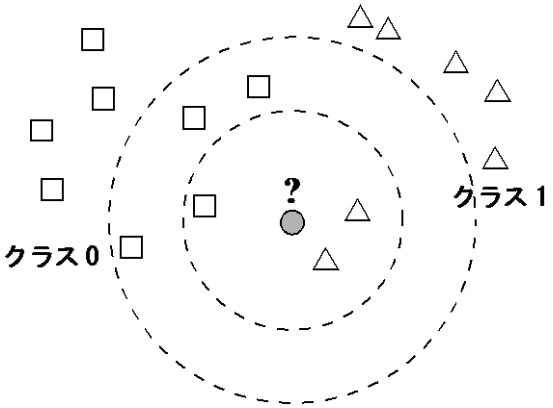
kNN（k-Nearest Neighbors）の特徴

- 教師つき学習の一種
- アルゴリズムは単純であるが，性能は比較的高い。
- ただし，ノンパラメトリック手法であるため、クラス分類線を式で表現できない。
  分類の仕方が，どのような意味や重要性を有するかの意味付けが難しい。
- 単に分類したいという用途であるならば，有力な手法である。

アルゴリズムと距離

アルゴリズムの考え方

1. 四角印のデータはクラス0，三角印のデータはクラス1に属していて，そのラベル名および特徴量（右図では位置）は既知とする（教師つき学習）。
2. 新たにデータ（丸印）を得たとして，このクラスを判定したい。
3. このデータの位置から近い距離から順にk個の既知データを取得する。
4. k個のクラスの多数決で，丸印のクラスを推定する。
5. k=3ならば丸印はクラス1，k=6ならばクラス0となる。



留意事項

- 3.の距離は幾つかの種類がある。ここに，距離はdistance, metricとも表現されるが，数学で距離空間はmetric spaceを用いられる。この場合，距離は大きいか小さいかの比較ができればいいので，一般に用いられるユークリッド距離（定規で測るようなもの）以外にもあり，付録にsklearnが用意しているものを示す。
- kが偶数のとき，クラス0，1が同数の場合が考えられる。この場合，棄却，確率的に決めるなどの方法がある。この計算を避けて，kを奇数とすることもある。

sklearn.neighbors.DistanceMetric
http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.DistanceMetric.html

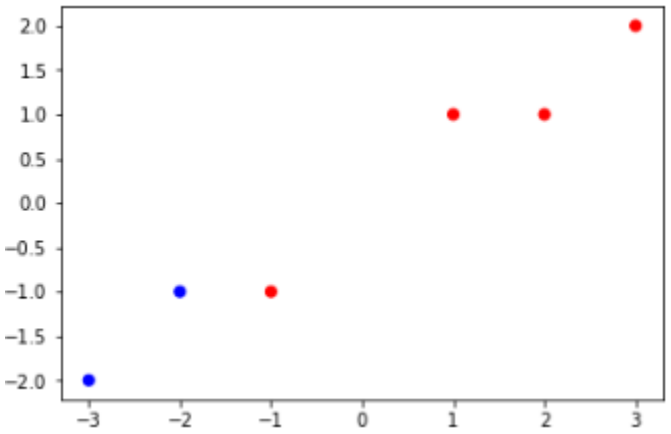
例：kNNの分類を見る

- 下記のデータに対しkNNによるクラス分類の結果を示す。
- neigh.kneighbors()より，distanceはX[0],X[1],...の順に，近傍のk個[自分自身，他の2個]がどれであるか，その際の距離が示される。

kNN_Exam

```
X = np.array([ [-3.0, -2.0], [-2.0, -1.0], [-1.0, -1.0], [1.0, 1.0], [2.0, 1.0], [3.0, 2.0]])
y = np.array([0, 0, 1, 1, 1, 1])
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(X, y)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=3, p=2,
           weights='uniform')
```



距離はミンコフスキーを用いているが，p=2よりユークリッド距離となる。
距離に対する重み(weight)は一様(uniform)である。

例：kNNの分類を見る

```
: 1 X_test = np.array([ [-1.5, -1.0], [0.0, 0.5]])
  2 print('Estimated class: ', neigh.predict(X_test))
```

```
Estimated class:  [0 1]
```

新たにデータを取得したとして，それらのクラスを推定する。
散布図を見て，結果の妥当性を見ること。

```
: 1 distances, indices = neigh.kneighbors(X_test)
  2 print(indices)
  3 print(distances)
```

```
[[1 2 0]
 [3 2 4]]
[[ 0.5         0.5         1.80277564]
 [ 1.11803399  1.80277564  2.06155281]]
```

k=3とおいたので
- テストデータの0番目：近いのが，もとのデータの0，1，2番目
- テストデータの1番目：近いのが，もとのデータの3，2，4番目
- 散布図を見て，この妥当性を確認すること。

indicesで示された隣接データとの距離

例：Iris データ

Xは2次元

- iris変量の内，sepal（がく）と petal（花びら）の長さ[cm]を見るため 0番目と2番目のデータをXに格納

kNN_Iris

```
# Load iris data, 150 sample
iris = datasets.load_iris()
X = iris.data[:, [0, 2]] # Length
y = iris.target

# Split into training and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=2/15, random_state=123)
```

kNNのkの値で結果は変わるか？

- k=3, 7, 9 のとき，結果は同じで下記のとおり

Accuracy = 0.85

	precision	recall	f1-score	support
0	1.00	1.00	1.00	7
1	0.71	0.83	0.77	6
2	0.83	0.71	0.77	7
avg / total	0.86	0.85	0.85	20

例：iris データ

どれが誤っているか？

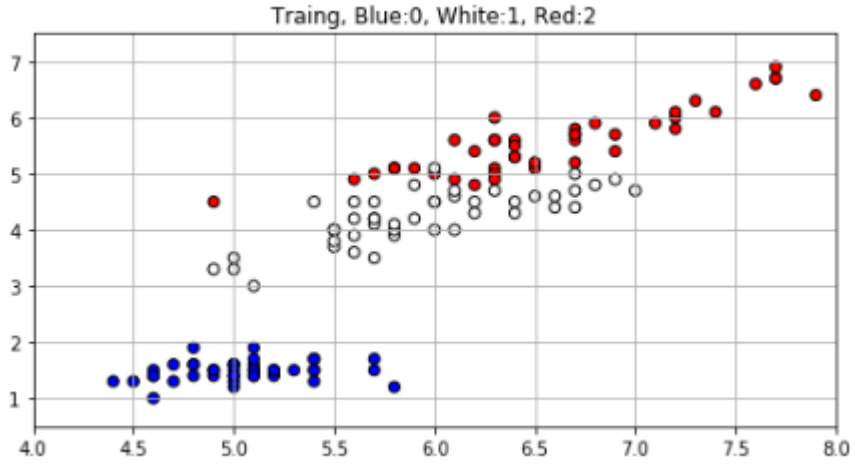
- 右図は，推定したy_trainのクラス
- 燈色が誤りデータ
- 図を見て，どう考える？

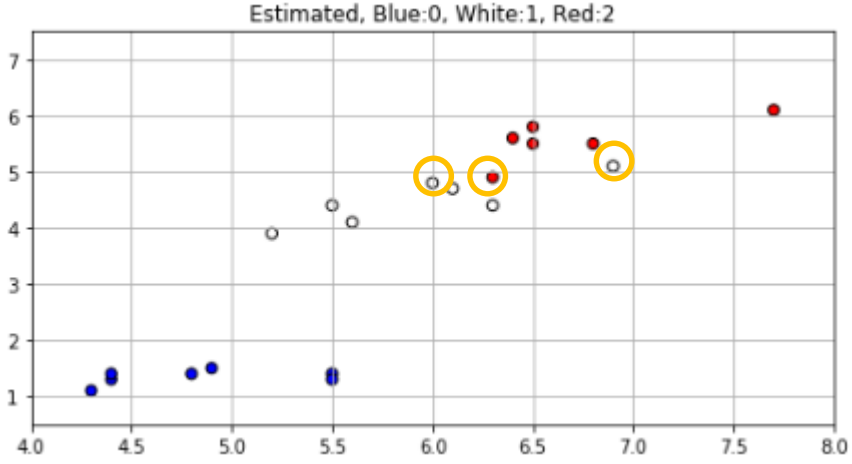
```
print('Real       =',y_test)
print('Estimation =',y_test_est)
```

```
Real       = [1 2 2 1 0 2 1 0 0 1 2 0 1 2 2 2 0 0 1 0]
Estimation = [2 2 2 1 0 1 1 0 0 1 1 0 1 2 2 2 0 0 1 0]
```

```
#上のグラフと比較してkNNの性能を推測する。
print('  0:',X_test[0],'¥n  5:',X_test[5],'¥n 10:',X_test[10])
```

```
 0: [ 6.3  4.9]
 5: [ 6.   4.8]
10: [ 6.9  5.1]
```





その他

- kNNは最近傍法の一種であり，scikit-learnは他の最近傍法の手法を提供している。興味のある方は下記を参照
  - http://scikit-learn.org/stable/modules/neighbors.html

距離（Distance Metric）

- http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.DistanceMetric.html

Metrics intended for real-valued vector spaces:

identifier	class name	args	distance function
“euclidean”	EuclideanDistance	•	`sqrt(sum((x - y)^2))`
“manhattan”	ManhattanDistance	•	`sum(|x - y|)`
“chebyshev”	ChebyshevDistance	•	`max(|x - y|)`
“minkowski”	MinkowskiDistance	p	`sum(|x - y|^p)^(1/p)`
“wminkowski”	WMinkowskiDistance	p, w	`sum(w * |x - y|^p)^(1/p)`
“seuclidean”	SEuclideanDistance	V	`sqrt(sum((x - y)^2 / V))`
“mahalanobis”	MahalanobisDistance	V or VI	`sqrt((x - y)' V^-1 (x - y))`

http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
p : integer, optional (default = 2)

Power parameter for the Minkowski metric. When p = 1, this is equivalent to using manhattan_distance (l1), and euclidean_distance (l2) for p = 2. For arbitrary p, minkowski_distance (l_p) is used.

metric : string or callable, default ‘minkowski’

the distance metric to use for the tree. The default metric is minkowski, and with p=2 is equivalent to the standard Euclidean metric. See the documentation of the DistanceMetric class for a list of available metrics.

備考：ノルム（norm）は，解析学分野で，空間の概念的“長さ”を表し，ベクトル空間では距離に相当する。

参考文献：平井有三，はじめてのパターン認識，森北出版，第10章クラスタリング

付録：分類線（識別線）の引き方

- **作図は，ボロノイ図(Voronoi diagram)のアルゴリズムに基づいている。**
  - http://scikit-learn.org/stable/modules/clustering.html
  - https://www.datarobot.com/blog/classification-with-scikit-learn/　ほんまかいな？
  - ちなみに，ボロノイ図の応用例
    - 最も近い 携帯電話 の基地局を探す
    - 新しい基地局をどこに作ればよいかの指標を得る
    - キタキツネの勢力範囲
    - 有限要素法の領域分割
    - 画像のデータ圧縮
  - 詳しい内容は他の成書をご覧ください。

END