iris

September 25, 2021

1 Data Science Lab 1 - Iris dataset

The goal is to determine wether or not the various classes of Iris are separated.

1.1 Method 1 - distances

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
iris = pd.read_csv("../data/iris.csv")
iris
```

[2]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
..	…	…	…	…	…
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

[150 rows x 5 columns]

1.1.1 General distance

[3]:
```
def distance(v1, v2, metric='minkowski', L=2):
    if metric == "minkowski":
        return np.power(np.power(v1 - v2, L).sum(), 1/L)
    else:
        raise NotImplementedError
```

[4]:
```
iris_center = iris.groupby("species").mean().transpose()
iris_center
```

[4]:
species	setosa	versicolor	virginica
sepal_length	5.006	5.936	6.588
sepal_width	3.428	2.770	2.974
petal_length	1.462	4.260	5.552
petal_width	0.246	1.326	2.026

[5]:
```
v1 = iris[iris['species'] == 'setosa'].iloc[0, :-1]
v2 = iris_center.loc[:, 'setosa']

distance(v1, v2)
```

[5]:
```
0.14135062787267663
```

1.1.2 Intra-class distance

[6]:
```
iris[iris['species'] == 'setosa'].apply(lambda elt: distance(elt, v2), axis=1).
↪max()
```

[6]:
```
1.2480304483465139
```

[7]:
```
def intra_class(df, class_name, cats=None, metric='minkowski'):
    cats = df.columns[-1] if cats is None else cats
    df = df[df[cats] == class_name]
    center = df.drop(cats, axis=1).mean()
    return df.apply(lambda elt: distance(elt, center, metric), axis=1).max()
```

[8]:
```
intra_class(iris, 'setosa')
```

[8]:
```
1.2480304483465139
```

1.1.3 Inter-class distance

[9]:
```
# Directional inter-class distance
def inter_class(df, source_class, target_class, cats=None, metric='minkowski'):
    cats = df.columns[-1] if cats is None else cats
    center = df[df[cats] == target_class].drop(cats, axis=1).mean()
    df = df[df[cats] == source_class]
    return df.apply(lambda elt: distance(elt, center, metric), axis=1).min()
```

[10]:
```
inter_class(iris, 'versicolor', 'setosa')
```

[10]:
```
1.9911755321919762
```

1.1.4 Pair distances

[11]:
```
from itertools import combinations

distances = []
for (class1, class2) in combinations(iris.loc[:, 'species'].unique(), 2):
    intra = intra_class(iris, class1)
    inter = inter_class(iris, class2, class1)
    distances.append([class1, class2, intra, inter, intra < inter])

pd.DataFrame(distances, columns=["class 1", "class 2", "intra", "inter",␣
 ↪"separated"])
```

[11]:

	class 1	class 2	intra	inter	separated
0	setosa	versicolor	1.248030	1.991176	True
1	setosa	virginica	1.248030	3.495137	True
2	versicolor	virginica	1.552569	0.757147	False

[12]:
```
# Direction-sensitive separation tests
distances = []
for (class1, class2) in combinations(iris.loc[:, 'species'].unique(), 2):
    for i in range(2):
        intra = intra_class(iris, class1)
        inter = inter_class(iris, class2, class1)
        distances.append([class1, class2, intra, inter, intra < inter])
        class1, class2 = class2, class1

pd.DataFrame(distances, columns=["class 1", "class 2", "intra", "inter",␣
 ↪"separated"])
```

[12]:

	class 1	class 2	intra	inter	separated
0	setosa	versicolor	1.248030	1.991176	True
1	versicolor	setosa	1.552569	2.861271	True
2	setosa	virginica	1.248030	3.495137	True
3	virginica	setosa	2.070507	4.344813	True
4	versicolor	virginica	1.552569	0.757147	False
5	virginica	versicolor	2.070507	0.651306	False

We can conclude that *setosa* and *versicolor* are separated, as well as *setosa* and *virginica*.

However, we cannot assert that *versicolor* and *virginica* are separated.

1.2 Method 2 - vissualisation

[ ]:
```

```