圖形識別 作業一：KNN	曾宏鈞 06160485

使用Scikit-learn調參

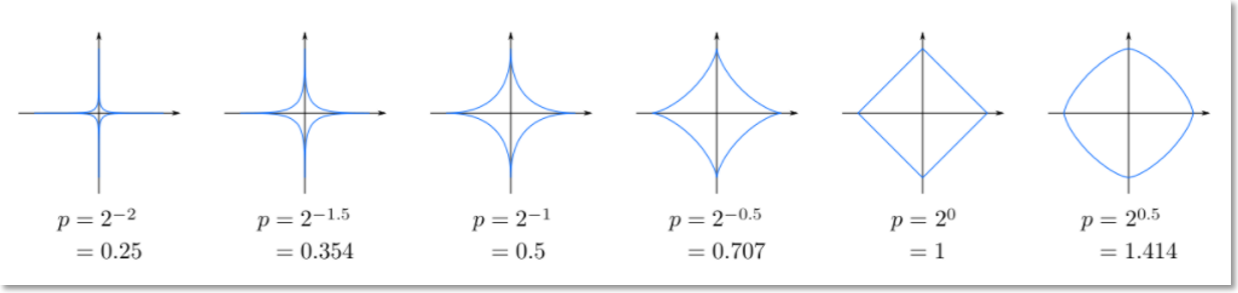
以下名稱簡寫
Minkowski distance(MD)
Neighbors(N)

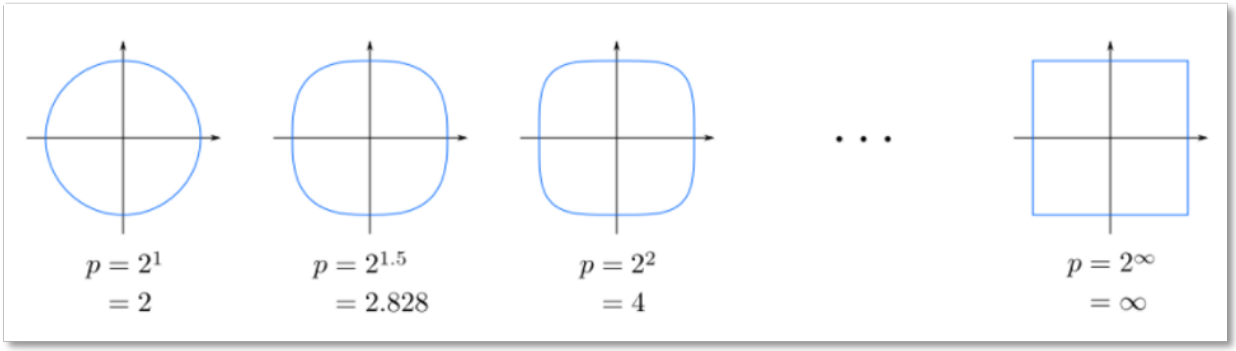
1. 距離(p)

由 Minkowski distance (式1)所定義p值，可以得到不同p所形成的值域範圍。

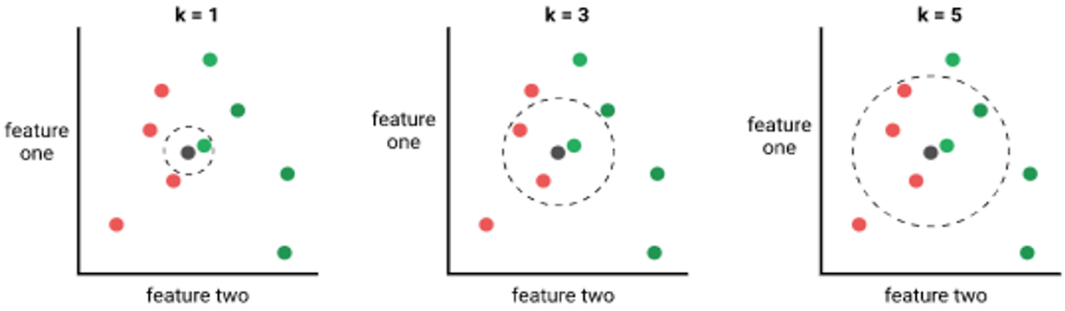
$$d_p(x,y) = \sqrt{\sum_{i=1}^{n}\left(|x_i - y_i|^p\right)^{\frac{1}{p}}}$$

式1、 Minkowski distance




$p = 2^1$
$= 2$
$p = 2^{1.5}$
$= 2.828$
$p = 2^2$
$= 4$
. . .
$p = 2^\infty$
$= \infty$

2. 取的鄰居點數(n)


k = 1
feature one
feature two
k = 3
feature one
feature two
k = 5
feature one
feature two

我們使用Minkowski distance不同的p值、KNN取的鄰居點數(neighbors)n整理成圖1。

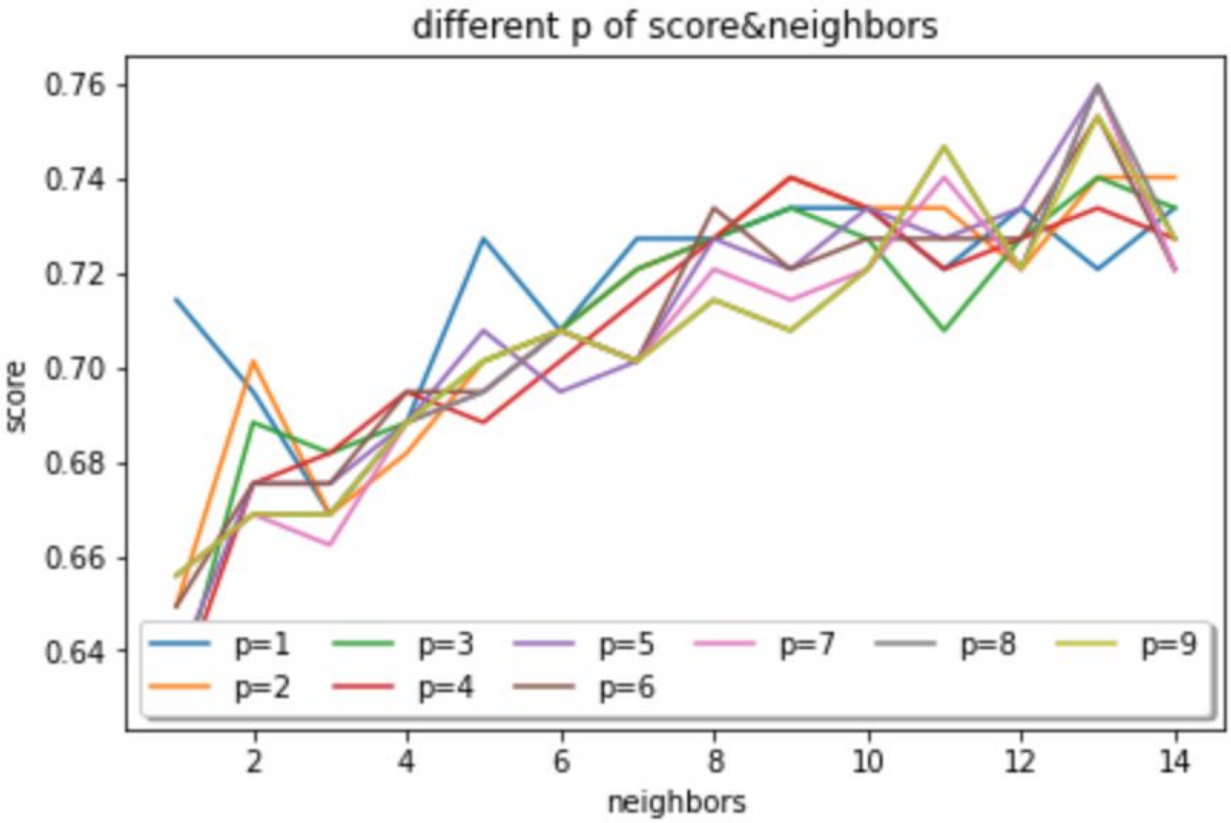
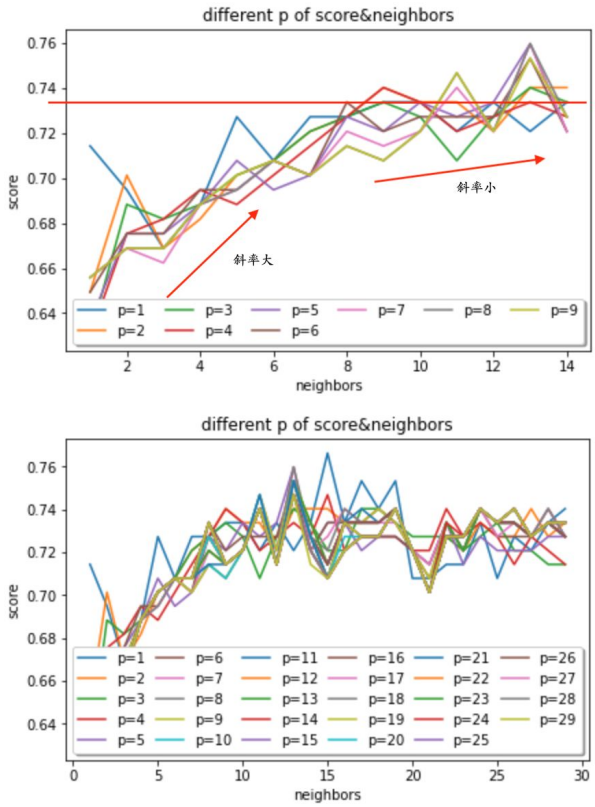


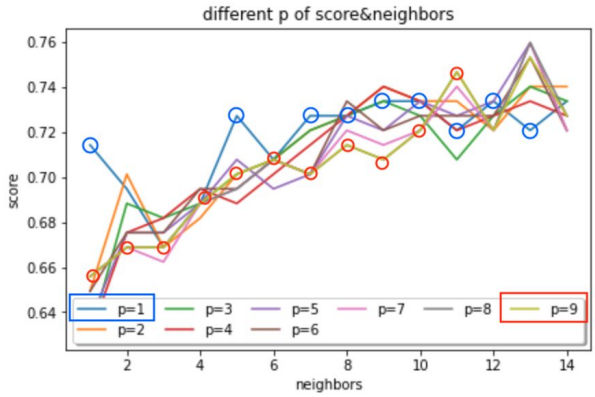
圖1、KNN使用不同Minkowski distance 的參數p及neighbors的準確度圖

結論

<table>
<tr>
<td>

- N值越大，準確度會越高，因為更多點加入平均。
- N越大score越平緩
- 即便增加MD的次方數及增加N數，準確度約收斂在0.735。

</td>
<td>


</td>
</tr>
<tr>
<td>在N小的時候，MD所使用的p值愈大(維度愈高)，會導致準確度較低。
紅色圈(p=9)vs藍色圈(p=1)</td>
<td>


</td>
</tr>
</table>

參考資料

- https://arxiv.org/pdf/1708.04321.pdf
- https://towardsdatascience.com/importance-of-distance-metrics-in-machine-learning-modelling-e51395ffe60d
- https://medium.com/@luigi.fiori.lf0303/distance-metrics-and-k-nearest-neighbor-knn-1b840969c0f4

使用tensorflow實作

Implement KNN class:

```
class TF_KNeighborsClassifier:
    def __init__(self, n_neighbors=2, p=2):
        self.n_neighbors = n_neighbors
        self.p = p
    def fit(self, X, y):
        self.X = tf.convert_to_tensor(X)
        self.y = tf.convert_to_tensor(y)
    def predict(self, X_test):
        y_pred = []
        for x in X_test:
            d = tf.map_fn(self.Manhattan_distance,x)
            min_d_idx = tf.argsort(d)[:self.n_neighbors]
            # find nearest neighbors's label
            nearest_neighbors_label = []
            for i in min_d_idx:
                nearest_neighbors_label.append(self.y[i])
            # find mode
            y, idx, count = tf.unique_with_counts(nearest_neighbors_label)
            #y_pred.append( y[tf.argmax(count)] ) # slower
            y_pred.append( tf.slice(y, begin=[tf.argmax(count, 0)],
size=tf.constant([1], dtype=tf.int64))[0] ) #faster
        y_pred = np.array(y_pred)
        return y_pred
    def score(self,X_test, y_test):
        return accuracy_score(self.predict(X_test), y_test)
    def Manhattan_distance(self,x_test):
        return tf.math.pow( tf.reduce_sum( tf.math.pow(tf.math.abs(
tf.subtract(self.X,x_test) ),self.p) ) , 1.0/self.p)
```

usage

```
knn = TF_KNeighborsClassifier(n_neighbors=3,p=2)
knn.fit(x_train,y_train)
# for predict
y_pred = knn.predict(x_test)
print(y_pred)
# score
result = knn.score(X_test=x_test,y_test=y_test)
result
```