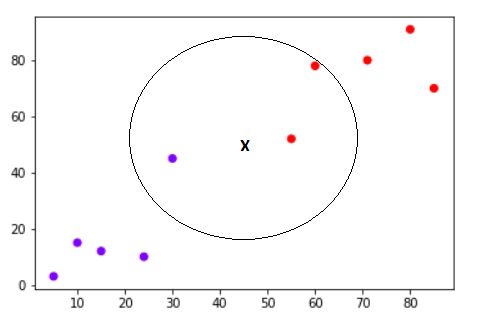
**knn实验iris**

**1.前言**

Knn是一种简单、但是功能强大的分类算法，可以实现复杂的分类效果。Knn是一种有监督的机器学习方法。

它的基本原理，可以用一句古话来总结“近朱者赤，近墨者黑”。根据数据点周围的相近的若干（k，K-nearest data points）数据点的分类标签（majority of the K data points belong），决定该数据点的分类标签。如下图所示，新的数据点x，根据三个最相似近邻的标签，应该分为“红色”。



KNN的主要优点有**，**

1. It is extremely easy to implement
2. As said earlier, it is [lazy learning](https://en.wikipedia.org/wiki/Lazy_learning) algorithm and therefore requires no training prior to making real time predictions. This makes the KNN algorithm much faster than other algorithms that require training e.g SVM, [linear regression](http://stackabuse.com/linear-regression-in-python-with-scikit-learn/), etc.
3. Since the algorithm requires no training before making predictions, new data can be added seamlessly.
4. There are only two parameters required to implement KNN i.e. the value of K and the distance function (e.g. Euclidean or Manhattan etc.)

Knn算法的主要缺点有**，**

1. The KNN algorithm doesn't work well with high dimensional data because with large number of dimensions, it becomes difficult for the algorithm to calculate distance in each dimension.
2. The KNN algorithm has a high prediction cost for large datasets. This is because in large datasets the cost of calculating distance between new point and each existing point becomes higher.
3. Finally, the KNN algorithm doesn't work well with categorical features since it is difficult to find the distance between dimensions with categorical features.

**2.必备条件**

提示：pip install sklearn。

**3.实验过程**

本文给出一个使用scikit learn软件包实现iris数据集分类的例子。

**The Dataset**

在本实例中，我们使用iris数据集。该数据集有四个属性sepal-width（花萼宽度）, sepal-length（花萼长度）, petal-width （花瓣宽度）and petal-length（花瓣长度），我们的目的是根据这四个属性，判定某个植物属于哪个类别，Iris-setosa, Iris-versicolor and Iris-virginica。



首先导入必要的库。

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  import pandas as pd |

装载数据集，数据集的形式是pandas dataframe。

|  |
| --- |
| url = "./iris.csv"  # Assign colum names to the dataset  names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']  # Read dataset to pandas dataframe  dataset = pd.read\_csv(url, names=names) |

查看数据集的前**5行。**

|  |
| --- |
| dataset.head() |

输出结果如下**。**

|  |
| --- |
| sepal-length sepal-width petal-length petal-width Class  0 5.1 3.5 1.4 0.2 Iris-setosa  1 4.9 3.0 1.4 0.2 Iris-setosa  2 4.7 3.2 1.3 0.2 Iris-setosa  3 4.6 3.1 1.5 0.2 Iris-setosa  4 5.0 3.6 1.4 0.2 Iris-setosa |

**Preprocessing**

把数据集分解出属性（X）和标签(y)，使用pandas的纵向切割方法。

|  |
| --- |
| X = dataset.iloc[:, :-1].values  y = dataset.iloc[:, 4].values |

**Train Test Split**

把数据集划分成训练集和测试集。

|  |
| --- |
| from sklearn.model\_selection import train\_test\_split  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20) |

80%的数据划分到训练集，20%的数据划分到测试集。

**Feature Scaling**

原始数据的取值范围变化很大，有必要对数据进行normalization规范化处理，在这里对X进行缩放。

|  |
| --- |
| The majority of classifiers calculate the distance between two points by the Euclidean distance.  If one of the features has a broad range of values, the distance will be governed by this particular feature. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance. |

欧式距离需要在规范化的数据上运算，才能给出正确的结果（读者自行构造一个实例）。梯度下降算法（神经网络的训练方法）在规范化的数据上收敛得更快。

|  |
| --- |
| from sklearn.preprocessing import StandardScaler  scaler = StandardScaler()  scaler.fit(X\_train)  X\_train = scaler.transform(X\_train)  X\_test = scaler.transform(X\_test) |

**Training and Predictions**

对模型进行训练和使用模型进行预测。

|  |
| --- |
| from sklearn.neighbors import KNeighborsClassifier  classifier = KNeighborsClassifier(n\_neighbors=5)  classifier.fit(X\_train, y\_train) |

在测试集上进行预测。

|  |
| --- |
| y\_pred = classifier.predict(X\_test) |

**Evaluating the Algorithm**

为了评估一个模型的优劣，我们可以使用confusion matrix, precision, recall 以及f1 score等指标和手段。这些技术的确切含义，请参考“覃雄派、陈跃国、杜小勇《数据科学概论》”教材的相关内容。

|  |
| --- |
| from sklearn.metrics import classification\_report, confusion\_matrix  print(confusion\_matrix(y\_test, y\_pred))  print(classification\_report(y\_test, y\_pred)) |

输出结果如下。

|  |
| --- |
| The output of the above script looks like this:  [[11 0 0]  0 13 0]  0 1 6]]  precision recall f1-score support  Iris-setosa 1.00 1.00 1.00 11  Iris-versicolor 1.00 1.00 1.00 13  Iris-virginica 1.00 1.00 1.00 6  avg / total 1.00 1.00 1.00 30 |

**Comparing Error Rate with the K Value**

在knn算法中，有一个k的选择问题，什么样的一个k能够获得较好的分类效果呢（准确率）？

我们可以把各种可能的k的取值，及其对应的分类误差率（error rate）绘制在一张图上。

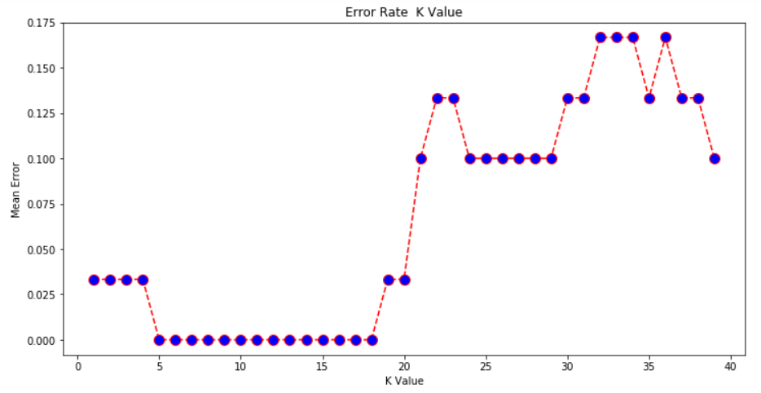
在这里，我们选择的k的取值范围为[1,40]。

|  |
| --- |
| error = []  # Calculating error for K values between 1 and 40  for i in range(1, 40):  knn = KNeighborsClassifier(n\_neighbors=i)  knn.fit(X\_train, y\_train)  pred\_i = knn.predict(X\_test)  error.append(np.mean(pred\_i != y\_test)) |

计算了k=1-40的error rate以后，我们可以绘制图形来看看。

|  |
| --- |
| plt.figure(figsize=(12, 6))  plt.plot(range(1, 40), error, color='red', linestyle='dashed', marker='o',  markerfacecolor='blue', markersize=10)  plt.title('Error Rate K Value')  plt.xlabel('K Value')  plt.ylabel('Mean Error') |

输出结果如下。



K取5-18的任何一个值，都是可以的。

**4.扩展阅读**

KNN is a simple yet powerful classification algorithm. It requires no training for making predictions, which is typically one of the most difficult parts of a machine learning algorithm. The KNN algorithm have been widely used to find document similarity and pattern recognition. It has also been employed for developing recommender systems and for dimensionality reduction and pre-processing steps for computer vision, particularly face recognition tasks.

读者还可以参考如下文档，对kmeans算法进行深入了解。

A Complete Guide to K-Nearest-Neighbors with Applications in Python and R. <https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/>。

**5.参考资料**

[1] <http://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/>, 2018.