KNN Classification

quantra

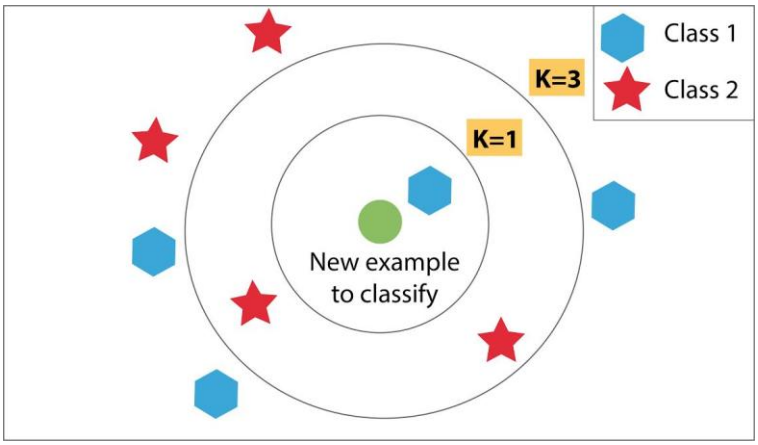
Table of Contents

KNN Classification

K nearest neighbours (KNN) is one of the simplest algorithms used in machine learning that is easy to understand but has been successful in a large number of problems including handwriting recognition, email classification, etc. K nearest neighbours is a non-parametric technique and are used in both Regression and classification problem. Here we are discussing only KNN classification.

Nearest Neighbours classification is a type of instance-based learning or non-generalizing learning. It does not attempt to construct a general internal model but compares new problem with instances seen in training, which has been stored in memory. The purpose of the K nearest neighbours (KNN) classification is to use a data in which the data points are separated into different classes to classify new data points based on a similarity measures (e.g. distance function). Classification of the object is done by a majority vote of its neighbours, the object is assigned to the class which has the most nearest neighbours. As you can increase the number of nearest neighbour meaning value of k, accuracy might increase but the computation cost also increases.

Let's consider the task of classifying a green circle between class 1 and class 2. Consider the case of KNN based on 1-nearest neighbour. It is clear that in this case, KNN will classify the green circle in class 1. Now let's increase the number of nearest neighbours to 3 i.e., 3-nearest neighbour. As you can see in the figure there is 'two' class 2 objects and 'one' class 1 object inside the circle. KNN will classify a green circle in class 2 object as it forms the majority.



One of the important things to do in KNN classification is assigned each point a weight which is usually calculated using its distance. For example, inverse distance weighting, in which each point has a weight equal to the inverse of its distance to the point to be classified. This means that nearer neighbours have a higher vote than the more distant ones.

KNN Classification Algorithm:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn import neighbors, datasets

n_neighbors = 15

# import some data to play with
iris = datasets.load_iris()
X = iris.data[:, :2]  # we only take the first two features. We could
                      # avoid this ugly slicing by using a two-dim dataset
y = iris.target

h = .02  # step size in the mesh

# Create color maps
cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
cmap_bold = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])

for weights in ['uniform', 'distance']:
    # we create an instance of Neighbours Classifier and fit the data.
    clf = neighbors.KNeighborsClassifier(n_neighbors, weights=weights)
    clf.fit(X, y)

    # Plot the decision boundary. For that, we will assign a color to each
    # point in the mesh [x_min, x_max]x[y_min, y_max]
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
```

```
    # Put the result into a color plot
    Z = Z.reshape(xx.shape)
    plt.figure()
    plt.pcolormesh(xx, yy, Z, cmap=cmap_light)

    # Plot also the training points
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap_bold)
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.title("3-Class classification (k = %i, weights = '%s')"
              % (n_neighbors, weights))

plt.show()
```

Output of Nearest Neighbours Classification Model: