K 近邻算法

SYSU & SPF 机器学习研修班系列讲义 *1*

主讲人 ： 付星宇

中山大学数学学院

K 近邻综述

K 近邻(K Nearest Neighbor，KNN)，是机器学习里最基本的分类(Classification)与回归(Regression)算法之一，常应用在推荐系统(Recommendation System)的构造中。

KNN 的核心思想为："物以类聚，人以群分"，如何理解呢？ 想象如下的一个场景：假设我们有一个鲍鱼养殖场 T，里面养殖了很多 A,B 两个品种的鲍鱼，并且我们很清楚养殖场 T 中每只鲍鱼的品种。现在有一个科学家从别处拿来了一只鲍鱼 p，我们想利用 KNN 算法判断鲍鱼 p 最可能是 A,B 中的哪个品种，KNN 算法会这样工作 ：

Step1· 在养殖场 T 中，找到 K 个和鲍鱼 p 最相似的鲍鱼(K Nearest Neighbor)

Step2· 这 K 个挑出来的鲍鱼中，A,B 哪类鲍鱼最多，就预测鲍鱼 p 是哪一类(Majority Voting)

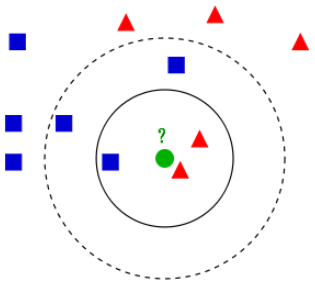
KNN 算法非常简单，很好理解，我们平时日常生活中都会采用这种算法帮助我们做决策。现将上述鲍鱼的例子抽象为形式化语言，表述如下：

现有 A,B 两种类别，有训练集 T，其中 T = {(x1,y1),(x2,y2),...,(xn,yn)}，xi∈R^n ，代表各个训练实例的特征，yi∈{A,B}，代表各个训练实例所属的类别。现在输入一个未知类别的待分类实例 x∈R^n，做如下操作：

Step1· 在某种度量方法下，在 T 中找到 K 个与 x 最相似的 xi

Step2· 预测 x 的类别 y 为找出的 K 个 yi 中出现频率最高的类别

用一副图来直观感受一下 KNN 的决策过程：



假设绿点是待分类的鲍鱼，那当 K 取 3 时，绿点会被分类为红三角类。当 K 取 5 时，绿点会分类为蓝方块类。由此可见 K 值的选取对预测结果会有比较大的影响。

但是我们到现在还有两个问题没有解决 ：

*1)*如何度量两个实例之间的相似度

*2)*如何找到与待分类实例最相似的 *K* 个

下面我们分别解决这两个问题。

相似度的计算

1· 特征向量(Feature Vector)

何为谓特征向量呢？ 还是以鲍鱼养殖场为例子，考虑每一只鲍鱼，那么我们可以测量每只鲍鱼的横向尺寸，纵向尺寸，厚度，年龄，光泽度等等特征，这些特征要尽可能精确描述这只鲍鱼，把所有这些特征放在一起，构成一个 *n* 维实向量。这样，每一只鲍鱼就映射为一个R^n中的向量 *x* ：

$$鲍鱼 \mapsto x$$

我们称这个R^n中的向量 *x* 为鲍鱼的特征向量，所有鲍鱼特征向量构成的集合成为特征空间*(Feature Space)*。

在任何情况下使用 *KNN* 算法，首先都应该把各个实例转化为特征向量的形式，以进行距离计算。

事实上，在机器学习中，选用何种特征描述一个实例是非常非常重要的。在业界，特征处理所耗费的时间往往占机器学习项目的 *80%*的用时。特征选择的好坏，对最后的预测效果起了非常重要的影响，而采用何种机器学习算法进行对数据的学习，往往不是最重要的。

2· 距离度量(Metric)

在之前那个鲍鱼养殖场的例子中，我们希望计算两只鲍鱼的相似度，实际上是计算两只鲍鱼特征向量的某种距离，如何理解呢？ 我们不可能直接算两个鲍鱼之间的相似度，只能通过两个特征向量之间的关系来计算相似度，而特征向量之间的相似度，则可以理解为特征向量在特征空间中的距离。

在R^n中，常有一下一些距离度量方法：

1) 欧式距离： $L_2(x_1,x_2)=\sqrt{\sum_{i=1}^{n}\left(x_1^{(i)}-x_2^{(i)}\right)^2}$ ， x1,x2∈R^n

2) 余弦相似度： $cos(x_1,x_2)=\frac{x_1\cdot x_2}{|x_1|\cdot|x_2|}$ ， x1,x2∈R^n （常用于 NLP）

查找 K 近邻

下面我们解决如何找到 K 近邻的问题，一共介绍两种办法：线性扫描，KD 树搜索。

1. 线性扫描(Linear Scan)

线性扫描是一种最简单的查找 K 近邻的办法，我们遍历训练集 T 中每一个已标记的实例，分别计算其与未标记点的距离。所有距离计算完成之后，对距离进行排序，选取距离最小的前 K 个训练集中的实例作为 K 近邻。假设 T 中有 n 个训练实例，当用线性扫描查找 K 近邻时，我们需要计算 n 次距离，当 n 特别大的时候，这个计算量是不可容忍的。

举个栗子：假设嘀嘀打车用 KNN 算法进行订单分配，我在中山大学东门用滴滴打车 APP 叫出租。订单发出之后，APP 自动把我的订单信息发送给前 K 近的出租车。 如果我们采用线性扫描的策略进行 K 近邻查找，那么 APP 将会计算我和全中国所有滴滴出租车的距离，再取前 K 个最近的出租车，这显然是不可行的。 事实上，我们只需计算新港西路附近所有滴滴出租与我的距离就好了。

2. KD 树

当特征向量之间的度量采用欧式距离时，我们可以采用 KD 树这种数据结构进行训练集中样本的存储。训练集的 KD 树构造完成之后，当我们查找未分类实例在训练集中的 K 近邻时，速度会大大提升。

用 KD 树实现 KNN 算法分为两大部分：

1) 构造训练集的 KD 树

（见《统计学习方法》 算法 3.2）

2) K 近邻查找算法

这里我们给出最近邻查找算法的描述（Wikipedia），（K 近邻查找类似）：

Nearest neighbour search [edit]

The nearest neighbour search (NN) algorithm aims to find the point in the tree that is nearest to a given input point. This search can be done efficiently by using the tree properties to quickly eliminate large portions of the search space.

Searching for a nearest neighbour in a *k*-d tree proceeds as follows:

1. Starting with the root node, the algorithm moves down the tree recursively, in the same way that it would if the search point were being inserted (i.e. it goes left or right depending on whether the point is lesser than or greater than the current node in the split dimension).
2. Once the algorithm reaches a leaf node, it saves that node point as the "current best"
3. The algorithm unwinds the recursion of the tree, performing the following steps at each node:
    1. If the current node is closer than the current best, then it becomes the current best.
    2. The algorithm checks whether there could be any points on the other side of the splitting plane that are closer to the search point than the current best. In concept, this is done by intersecting the splitting hyperplane with a hypersphere around the search point that has a radius equal to the current nearest distance. Since the hyperplanes are all axis-aligned this is implemented as a simple comparison to see whether the distance between the splitting coordinate of the search point and current node is lesser than the distance (overall coordinates) from the search point to the current best.
        1. If the hypersphere crosses the plane, there could be nearer points on the other side of the plane, so the algorithm must move down the other branch of the tree from the current node looking for closer points, following the same recursive process as the entire search.
        2. If the hypersphere doesn't intersect the splitting plane, then the algorithm continues walking up the tree, and the entire branch on the other side of that node is eliminated.
4. When the algorithm finishes this process for the root node, then the search is complete.

思考题

-1 当距离度量为余弦相似度时，如何采用 KD 树查找 K 近邻？(Hint: Normalization)

(Quite Useful in Practice , especially in NLP)

-2 如何用 KD 树搜索 K(K≥2)近邻？ (Hint: Wikipedia)

-3　KNN 是一种懒惰学习策略，即对训练集不会有任何处理，没有显式的学习过程。查阅 LVQ(Learning Vector Quantization)的资料，学习改进的“不懒惰”KNN 算法。