KNN K-近邻算法

数学基础

距离：什么是距离？什么是向量距离？（一维的，二维的，多维的呢？）
距离是指（两物体）在空间或时间上相隔或间隔的长度。

L1 距离(曼哈顿距离)

曼哈顿距离对应L1-范数，也就是在欧几里得空间的固定直角坐标系上两点所形成的线段对轴产生的投影的距离总和。

$$I_1(I_{11},I_{12},\ldots,I_{1n})与I_2(I_{21},I_{22},\ldots,I_{2n})$$

之间的曼哈顿距离：

$$d_1(I_1,I_2)=\sum_p|I_1^P-I_2^p|$$

L2 距离（欧式距离）

最常见的两点之间或多点之间的距离表示法，又称之为欧几里得度量，它定义于欧几里得空间中。n维空间中两个点

$$I_1(I_{11},I_{12},\ldots,I_{1n})与I_2(I_{21},I_{22},\ldots,I_{2n})$$

间的欧氏距离：

$$d_2(I_1,I_2)=\sqrt{\sum_p(I_1^P-I_2^p)^2}$$

First:Why?

question 1

电影名称	打斗次数	接吻次数	电影类型
California Man	3	104	Romance
He's Not Really into Dudess	2	100	Romance
Beautiful Woman	1	81	Romance
KevinLongblade	101	10	Action
Robo Slayer3000	99	5	Action
Amped ii	98	2	Action
未知	18	90	Unknown

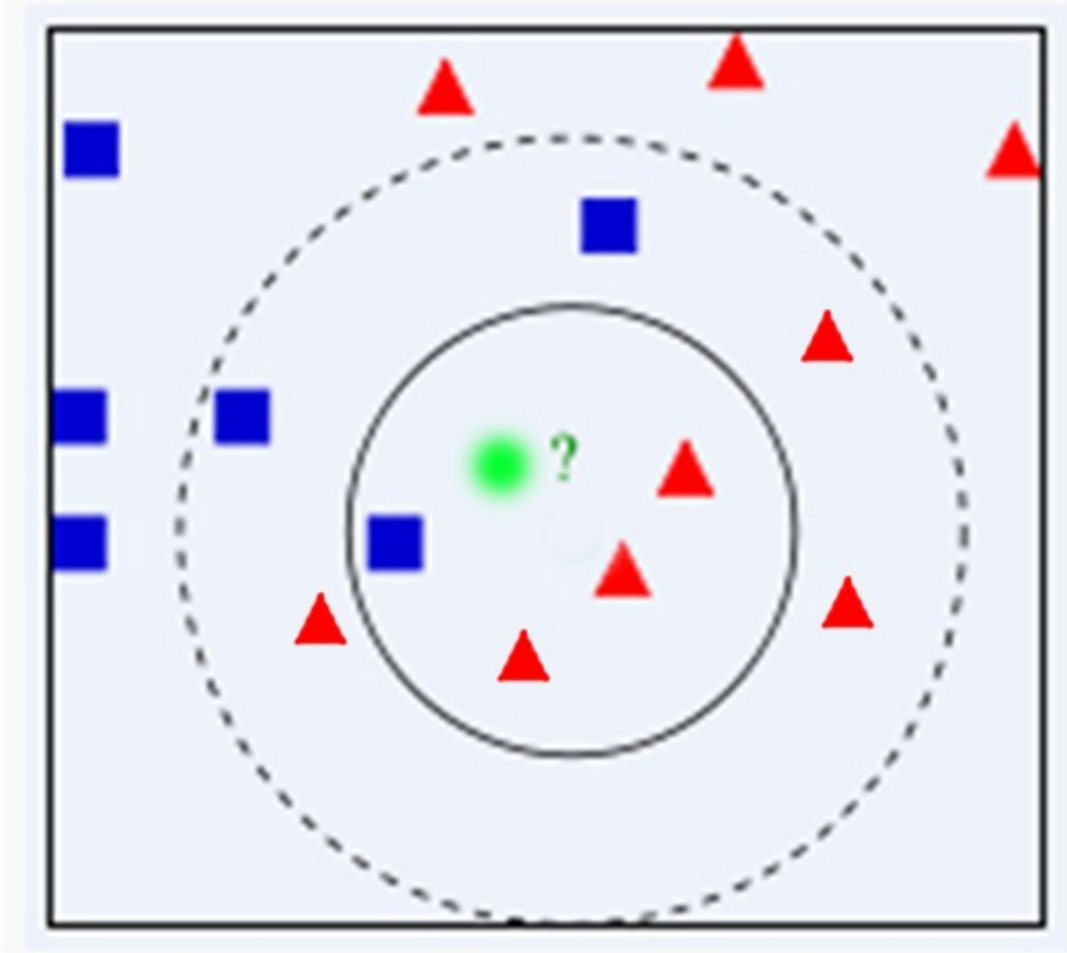
如何判断这个未知的电影属于哪一类？动作类，还是浪漫类？

question 2

如何判断这几颗豆子属于哪一类？

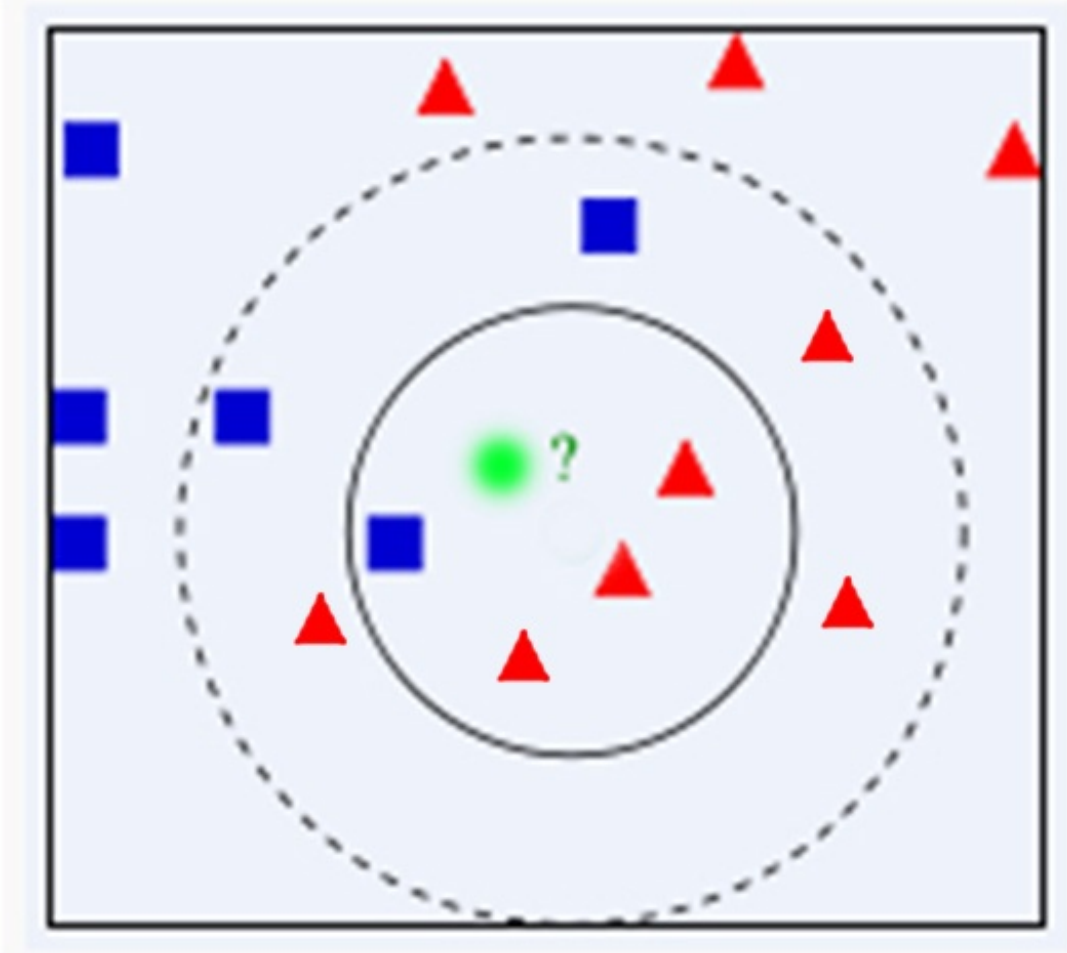
Second:What?

提供一种思路，即：未知的豆离哪种豆最近就认为未知豆和该豆是同一种类。由此，我们引出最近邻算法的定义：为了判定未知样本的类别，以全部训练样本作为代表点，计算未知样本与所有训练样本的距离，并以最近邻者的类别作为决策未知样本类别的唯一依据。但是，最近邻算法明显是存在缺陷的，我们来看一个例子。
问题：有一个未知形状X(图中绿色的圆点)，如何判断X是什么形状?



显然，通过上面的例子我们可以明显发现最近邻算法的缺陷——对噪声数据过于敏感，为了解决这个问题，我们可以可以把位置样本周边的多个最近样本计算在内，扩大参与决策的样本量，以避免个别数据直接决定决策结果。由此，我们引进K-最近邻算法。

K-最近邻算法是最近邻算法的一个延伸。基本思路是：选择未知样本一定范围内确定个数的K个样本，该K个样本大多数属于某一类型，则未知样本判定为该类型。



这里，我们将样本的特征组合成向量，求距离即求向量间的距离，也即L2范数。
K近邻算法并不是一个学习算法，可以称为一种懒惰算法。因为K近邻算法只需要输入一次训练样本集，后续需要判断的样本都和训练样本求取距离，无需重新训练。

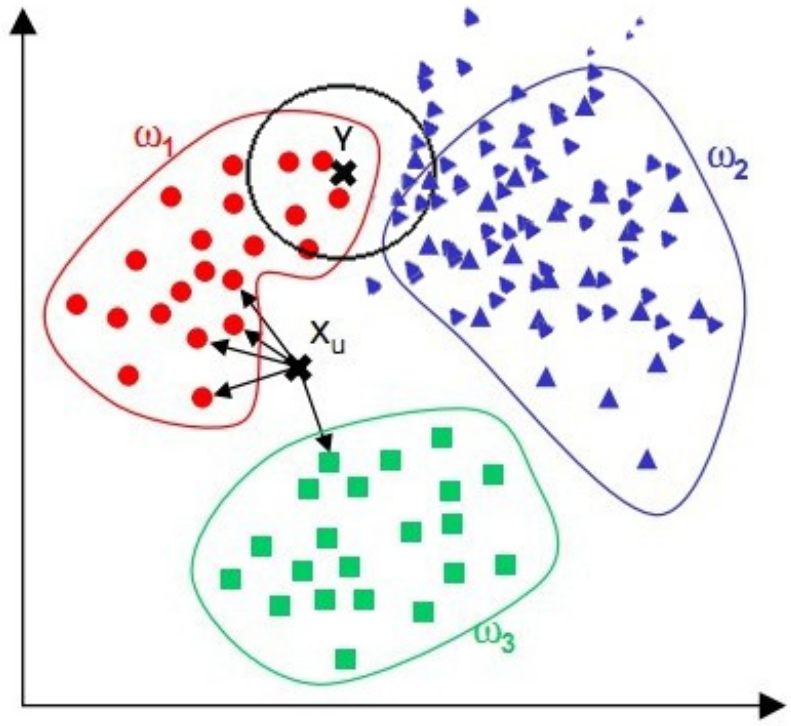
Third:How ?

基础KNN算法实现

1. 初始化距离为最大值
2. 计算未知样本和每个训练样本的距离dist
3. 得到目前K个最临近样本中的最大距离maxdist
4. 如果dist小于maxdist，则将该训练样本作为K-最近邻样本
5. 重复步骤2、3、4，直到未知样本和所有训练样本的距离都算完
6. 统计K个最近邻样本中每个类别出现的次数
7. 选择出现频率最大的类别作为未知样本的类别

缺陷 1

这个算法现在是有缺陷的？大家想想会有什么问题？
观察下面的例子，我们看到，对于位置样本X，通过KNN算法，我们显然可以得到X应属于红点，但对于位置样本Y，通过KNN算法我们似乎得到了Y应属于蓝点的结论，而这个结论直观来看并没有说服力。



策略：增加距离权重

由上面的例子可见：该算法在分类时有个重要的不足是，当样本不平衡时，即：一个类的样本容量很大，而其他类样本数量很小时，很有可能导致当输入一个未知样本时，该样本的K个邻居中大数量类的样本占多数。 但是这类样本并不接近目标样本，而数量小的这类样本很靠近目标样本。这个时候，我们有理由认为该位置样本属于数量小的样本所属的一类，但是，KNN却不关心这个问题，它只关心哪类样本的数量最多，而不去把距离远近考虑在内，因此，我们可以采用权值的方法来改进。和该样本距离小的邻居权值大，和该样本距离大的邻居权值则相对较小，由此，将距离远近的因素也考虑在内，避免因一个样本过大导致误判的情况。

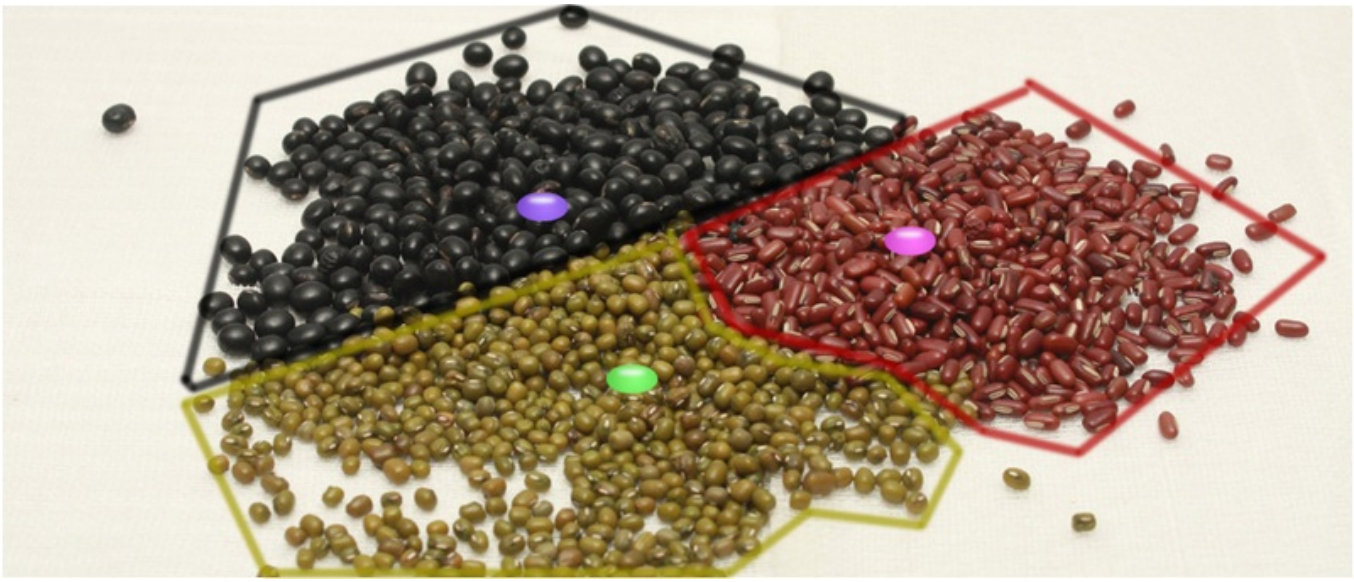
缺陷 2

从算法实现的过程大家可以发现，该算法存两个严重的问题，第一个是需要存储全部的训练样

本，第二个是需要进行繁重的距离计算量。对此，提出以下应对策略。

策略1：分组快速近邻搜索

其基本思想是：将样本集按近邻关系分解成组，给出每组质心的位置，以质心作为代表点，和未知样本计算距离，选出距离最近的一个或若干个组，再在组的范围内应用一般的knn算法。由于并不是将未知样本与所有样本计算距离，故该改进算法可以减少计算量，但并不能减少存储量



策略2：压缩近邻搜索

利用现在的样本集，采取一定的算法产生一个新的样本集，该样本集拥有比原样本集少的多的样本数量，但仍然保持有对未知样本进行分类的能力。

基本思路

1. 定义两个存储器，一个用来存放生成的样本集，称为output样本集；另一个用来存放原来的样本集，称为original样本集。
2. 初始化：output样本集为空集，原样本集存入original样本集，从original样本集中任意选择一个样本移动到output样本集中；
3. 在original样本集中选择第i个样本，并使用output样本集中的样本对其进行最近邻算法分类，若分类错误，则将该样本移动到output样本集中，若分类正确，不做任何处理；
4. 重复3步骤，直至遍历完original样本集中的所有样本，output样本集即为压缩后的样本集。 通过这种方式也能减少算法的计算量，但仍然无法减少存储量。

K-折交叉验证

这是一种优化算法，我们前面已经实现了K-NN算法，但是为了提高泛化能力和降低过拟合。基本思想是：把在某种意义下将原始数据(dataset)进行分组,一部分做为训练集(train set),另一部分做为验证集(validation set or test set),首先用训练集对模型进行训练,再利用验证集来测试模型的泛化误差。另外，现实中数据总是有限的，为了对数据形成重用，从而提出k-折叠交叉验证。

具体步骤

1. 把训练数据D 分为K份，用其中的(K-1)份训练模型，把剩余的1份数据用于评估模型的质量.
2. 将这个过程在K份数据上依次循环。
3. 并对得到的K个评估结果进行合并，如求平均或投票。
   如下图所示的10折交叉验证，训练数据D被分为了
   $$D_1, D_2, D_3 \ldots D_{10}$$
   ，每次取其中9份数据作为训练集，1份作为测试集，最终将循环后所有的评估结果取平均。

问题

Q1:K应该取多少？

交叉的折数(fold)取多少一直没有准确的答案。往大了说这是个历史遗留问题，10这个数字也就被一直沿用了下来。一般有两种流行的取值：**(i) K=10(ii)K=n**，n指的是训练数据的总数，这种方法也叫做留一法(LOOCV)。
让我们思考交叉验证的两种极端情况：
完全不使用交叉验证是一种极端情况，即K=1。在这个时候，所以数据都被用于训练，模型很容易出现过拟合，因此容易是低偏差、高方差(low bias and high variance)。
留一法是K折的另一种极端情况，即K=n。随着K值的不断升高，单一模型评估时的方差逐渐加大而偏差减小。但从总体模型角度来看，反而是偏差升高了而方差降低了。
所以当K值在1到n之间的游走，可以理解为一种方差和偏差妥协的结果。以K=10为例，在训练时我们的训练集数量仅为训练数据的90%。对比不使用交叉验证的情况，这会使得偏差上升，但对于结果的平均又会降低模型方差，最终结果是否变好取决于两者之间的变化程度。而这种直觉上的解释，并不总是有效。如Hastie曾通过实验证明 K折交叉验证比留一法的方差更小，这和我们上面的结论不一样。
另一个值得一提的看法是，交叉验证需要思考场景，而不是普适的。其中关系最大的就是评估模型的稳定性。在2015年的一项研究中，作者发现留一法有最好或者接近最好的结果，在他们的实验中K=10和K=5的效果都远不如留一法或者K=20。
对于稳定模型来说，留一法的结果较为统一，值得信赖。对于不稳定模型，留一法往往比K折更为有效。换句话说，那就是留一法结果较为可靠。

Q2:K是不是取的越大越好呢？

这个也不是，还是看经验和实际需要。

Result

cs231n中给出的结果
1.使用L1距离，我们将会获得38.6%的准确率。
2.使用L2距离，我们将会获得35.4%的准确率。
实际代码测试结果：

[8]:
```
#现在需要先实现 predict_labels 函数（其实也是补全），运行下面的代码
#  （还是在上面路径的 k_nearest_neighbor.py里）
#  K=1，其实就是一个最近邻算法。
y_test_pred = classifier.predict_labels(dists, k=1)

# 输出预测结果（正确的比例）
num_correct = np.sum(y_test_pred == y_test)
accuracy = float(num_correct) / num_test
print 'Got %d / %d correct => accuracy: %f' % (num_correct, num_test, accuracy)
```

```
Got 137 / 500 correct => accuracy: 0.274000
```

你的输出应该有 `27%` 的准确率 . 让我们把 `k` 改的更大试试, `k = 5` :

[ ]:
```
y_test_pred = classifier.predict_labels(dists, k=5)
num_correct = np.sum(y_test_pred == y_test)
accuracy = float(num_correct) / num_test
print 'Got %d / %d correct => accuracy: %f' % (num_correct, num_test, accuracy)
```

```
Got 139 / 500 correct => accuracy: 0.278000
```

Talk

1. k-近邻算法中的K的选择？
2. L1与L2距离选择。考虑两个指标之间的差异很有意思。特别是，当涉及两个矢量之间的差异时，L2距离比L1距离更加不可原谅。
   也就是说，L2距离更喜欢许多中等分歧与一个大分歧。
3. k折交叉验证的的k的选取？

参考来源：

1. CSDN博客
2. 机器学习实战
3. cs231n系列课程