机器学习与深度学习面试系列十（KNN）

什么是KNN（K近邻算法）？

所谓K近邻算法，单从名字来猜想，可以简单粗暴的认为是：K个最近的邻居，当K=1时，算法便成了最近邻算法，即寻找最近的那个邻居。

用官方的话来说，所谓K近邻算法，即是给定一个训练数据集，对新的输入实例，在训练数据集中找到与该实例最邻近的K个实例（也就是上面所说的K个邻居）。对于分类问题，取这K个实例的多数属于的那个类。对于回归问题，可以取它们的加权平均值。

近邻的距离度量有哪些？分别适用于什么场景？

1. **欧氏距离**，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，如点 $x = (x_1, x_2, \ldots, x_n)$ 和 $y = (y_1, y_2, \ldots, y_n)$ 之间的距离为：
$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$
2. **标准化欧氏距离**，标准化欧氏距离是针对简单欧氏距离的缺点而作的一种改进方案。标准欧氏距离的思路：既然数据各维分量的分布不一样，那先将各个分量都"标准化"到均值、方差相等。
$$d(x, y) = \sqrt{\sum_{i=1}^{n} (\frac{x_i - y_i}{\sigma_k})^2}$$
3. **曼哈顿距离**，我们可以定义曼哈顿距离的正式意义为L1-距离或城市区块距离，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。
$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$
4. **马氏距离**(Mahalanobis Distance)，可以看作是欧氏距离的一种修正，修正了欧式距离中各个维度尺度不一致且相关的问题。 $d(x, y) = \sqrt{(x - y)^T \Sigma^{-1} (x - y)}$ 。其中， Σ 是这个分布的协方差。当 $\Sigma = \mathbf{I}$ 时，马氏距离退化为欧氏距离。具体可以参考这篇文章：zhuanlan.zhihu.com/p/46...
5. **余弦相似度**。用来衡量两个向量的相关性（夹角的余弦）。向量 $\mathbf{x}, \mathbf{y}$ 的余弦相似度为：$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{|\mathbf{x}| \cdot |\mathbf{y}|}$ ，其实就是向量的内积除以向量的数量积。
6. **闵可夫斯基距离**(Minkowski distance)， 两个向量（点）的 p 阶距离：$d(\mathbf{x}, \mathbf{y}) = (|\mathbf{x} - \mathbf{y}|^p)^{1/p}$ ，当 $p = 1$ 时就是曼哈顿距离，当 $p = 2$ 时就是欧氏距离。
7. **汉明距离**。两个等长字符串s1与s2之间的汉明距离定义为将其中一个变为另外一个所需要作的最小替换次数。例如字符串"1111"与"1001"之间的汉明距离为2。应用：信息编码（为了增强容

错性，应使得编码间的最小汉明距离尽可能大）。

8. **Pearson相关系数**。衡量两个随机变量的相关性。随机变量 $\boldsymbol{X}, \boldsymbol{Y}$ 的Pearson相关系数为：$\rho_{X,Y} = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$，就是协方差矩阵除以标准差之积。范围：[-1,1]，绝对值越大表示（正/负）相关性越大。

简单说来，各种“距离”的应用场景简单概括为，

- 空间：欧氏距离，
- 路径：曼哈顿距离，
- 以上统一形式:闵可夫斯基距离，
- 加权：标准化欧氏距离，
- 排除量纲和依存：马氏距离，
- 向量差距：夹角余弦，
- 编码差别：汉明距离，
- 相关：Pearson相关系数。

K值选择大小有什么影响？如何选择K值？

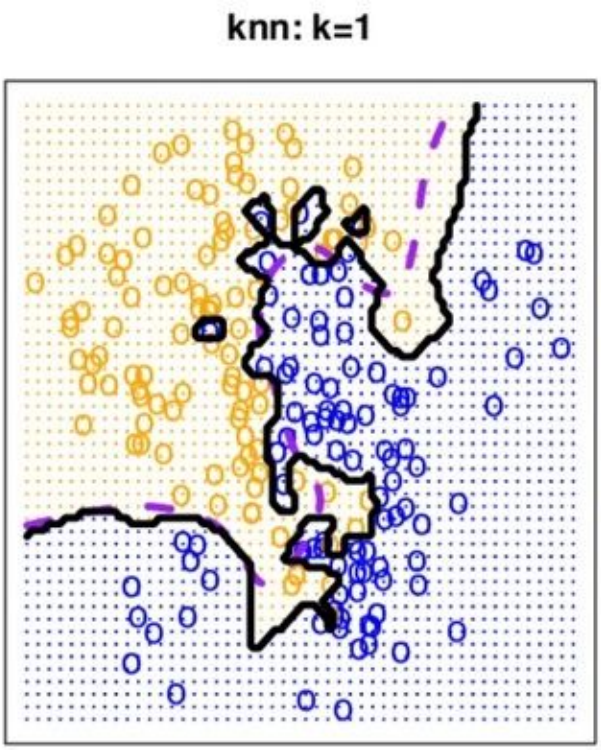
1. 如果选择较小的K值，就相当于用较小的领域中的训练实例进行预测，“学习”近似误差会减小，只有与输入实例较近或相似的训练实例才会对预测结果起作用，与此同时带来的问题是“学习”的估计误差会增大，换句话说，**K值的减小就意味着整体模型变得复杂，容易发生过拟合；**
2. 如果选择较大的K值，就相当于用较大领域中的训练实例进行预测，其优点是可以减少学习的估计误差，但缺点是学习的近似误差会增大。这时候，与输入实例较远（不相似的）训练实例也会对预测器作用，使预测发生错误，且**K值的增大就意味着整体的模型变得简单。**
3. K=N，则完全不足取，因为此时无论输入实例是什么，都只是简单的预测它属于在训练实例中最多的累，模型过于简单，忽略了训练实例中大量有用信息。

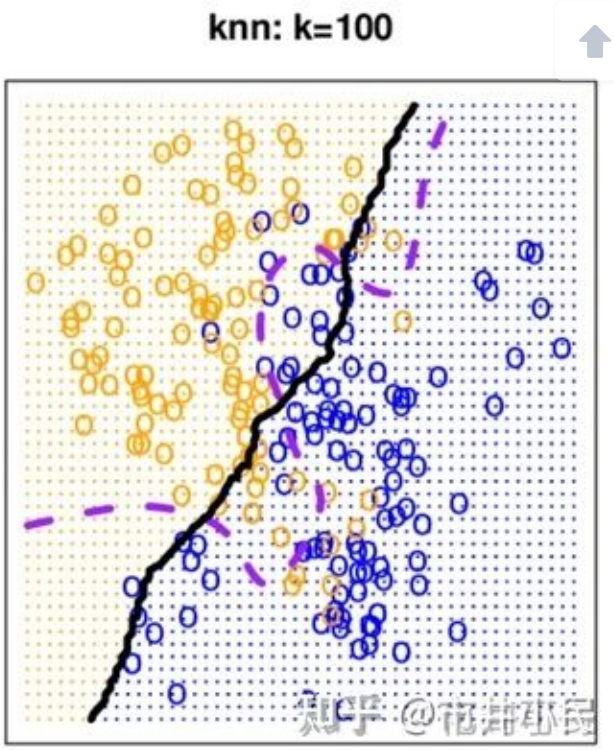
在实际应用中，K值一般取一个比较小的数值，**例如采用交叉验证法（简单来说，就是一部分样本做训练集，一部分做测试集）来选择最优的K值。**

KNN的决策边界是怎样的?

KNN的决策边界一般不是线性的，也就是说KNN是一种非线性分类器。

K越小越容易过拟合，当K=1时，这时只根据单个近邻进行预测，如果离目标点最近的一个点是噪声，就会出错，**此时模型复杂度高，稳健性低，决策边界崎岖。**
但是如果K取的过大，这时与目标点较远的样本点也会对预测起作用，就会导致欠拟合，**此时模型变得简单，决策边界变平滑。**
借用 zhuanlan.zhihu.com/p/79... 的图来说明：





KNN算法优缺点?

优点

1. 简单易用，相比其他算法，KNN算是比较简洁明了的算法。即使没有很高的数学基础也能搞清楚它的原理。
2. 模型训练时间快，KNN算法是惰性学习模型。
3. 预测效果好。
4. 对异常值不敏感

缺点

1. 对内存要求较高，因为该算法存储了所有训练数据
2. 预测阶段可能很慢
3. 对不相关的功能和数据规模敏感

KNN分类和Kmeans的区别?

KNN属于监督学习，类别是已知的，通过对已知分类的数据进行训练和学习，找到这些不同类的特征，再对未分类的数据进行分类。

Kmeans属于非监督学习，事先不知道数据会分为几类，通过聚类分析将数据聚合成几个群体。聚类不需要对数据进行训练和学习。