CS131 Lecture11：目标识别

by：斯坦福大学计算机科学系

github: https://github.com/zhaoxiongjun/CS131_notes_zh-CN （包含中英文版课件及相关课程视频）

1 Mean-shift

在此之前，我们讨论了凝聚聚类和 K 均值算法在图像分割中的应用。本课程的重点是 Mean-shift，一种分析密度函数以确定其最大值的模式搜索技术；Mean-shift 算法的主要应用之一是图像分割。在下面的章节中，详细介绍了 Mean-shift 算法中涉及的不同步骤。

1.1 优化

要正确地将点指定给簇，必须在每个点初始化一个窗口并将其移动到最密集的区域。此过程可能导致大量的冗余或非常相似的计算。可以通过并行计算窗口移动或减少必须在数据上移动的窗口数来提高算法的速度；这是通过使用一种称为“吸引盆地”的方法来实现的。

并行化：移动不同窗口所需的计算彼此独立，可以跨多个处理器拆分，并同时计算。通过并行化多个处理器或机器上的平均移位，可以在不损失精度的情况下实现较大的速度增加。

吸引盆地：靠近路径的点和窗口的停止点很可能会被分配到同一个簇。因此，这些点可以提前分配，而无需计算平均移位操作并在其位置初始化窗口；这减少了计算时间。每次更新到窗口位置后，使用以下方法分配附近的点：将移位端点半径 R 内的所有点分配到同一簇。这是因为窗户刚被移到一个密度更高的区域，因此靠近这个区域的点很可能都属于同一个簇。

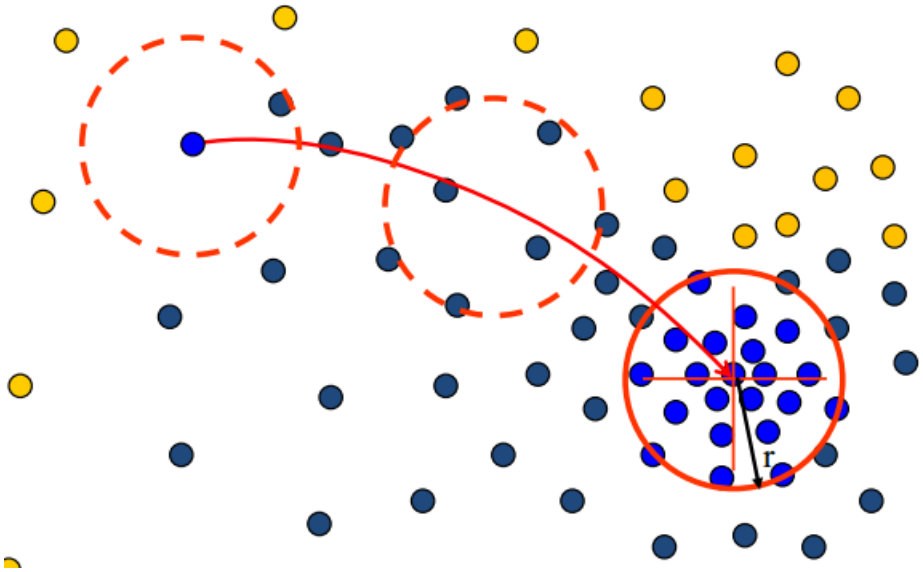

图 1：添加在移位端点半径 R 内的点

将移位路径半径 C 内的所有点指定给同一个簇。想象一下，在移位路径附近的某个点初始化一个窗口。因为所有窗口都朝着最密集区域的方向移动，所以很可能此窗口将朝着相同的方向移动，并且该点最终将分配给同一个簇。通常选择半径 c，使 c 小于等于 r。

在选择 r 和 c 的值时存在一个折中。值越小，越早分配的附近点越少，计算成本就越高。但是，如果不使用此方法计算平均移位，则得到的集群分配的误差将更小。值越大，分配的附近点越多，速度增加越快，但最终的群集分配对标准平均值偏移的精确度也会降低。

1.2 技术细节

为了正确地移动窗口，必须首先确定附近密度最高的区域来计算移动矢量。这可以用多元核密度估计来实现，这是一种估计随机变量概率密度函数的方法。

给定 n 个数据点$\mathbf{x} \in R^D$，利用径向对称（Comaniciu&Meer，2002）核 k（x）的多元核密度估计由下式得出：

$$\hat{f}_K = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right),$$

其中 h 是定义内核半径的带宽参数。径向对称核 k（x）：

$$K(x) = c_k k(||x||^2),$$

其中c_k表示标准化常数。

选择合适的 h 值是实现精确密度估计的关键。选择太小的值会导致半径较小，并且可能会在数据中产生噪声。选择太大的值会包含许多远点，并导致较少的簇。

多元核密度估计的结果导数:

$$\nabla \hat{f}(x) = \frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^{n} g\left(\left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right) \right] \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\left\| \frac{\mathbf{x}-\mathbf{x}_i}{h} \right\|^2 \right)}{\sum_{i=1}^{n} g\left(\left\| \frac{\mathbf{x}-\mathbf{x}_i}{h} \right\|^2 \right)} - \mathbf{x} \right], \quad (3)$$

where $g(x) = -K'(x)$ denotes the derivative of the selected kernel profile.

The first term in equation (3), $\frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^{n} g\left(\left\| \frac{\mathbf{x}-\mathbf{x}_i}{h} \right\|^2 \right) \right]$, is proportional to the density estimate at x. The second term, $\left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\left\| \frac{\mathbf{x}-\mathbf{x}_i}{h} \right\|^2 \right)}{\sum_{i=1}^{n} g\left(\left\| \frac{\mathbf{x}-\mathbf{x}_i}{h} \right\|^2 \right)} - \mathbf{x} \right]$, is the mean-shift vector that points towards the direction of maximum density.

1.3 Mean-shift 步骤

对于给定的点x_t，计算以下步骤以到达集群的中心。

1．计算平均位移向量 m[从上面的等式（3）计算项 2]:

$$\mathbf{m} = \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\left\| \frac{\mathbf{x}-\mathbf{x}_i}{h} \right\|^2 \right)}{\sum_{i=1}^{n} g\left(\left\| \frac{\mathbf{x}-\mathbf{x}_i}{h} \right\|^2 \right)} - \mathbf{x} \right]$$

2．使用 mean-shift 向量平移密度窗口：

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{m}(\mathbf{x}_i^t)$$

3．重复步骤 1 和 2 直到收敛

1.4 核函数

内核 k（x）是一个非负函数，它对 x 的所有值都积分为 1。这些要求确保了内核密度估计将产生概率密度函数。
流行的内核函数的例子包括:

- 均匀（矩形）：
  - $K(x) = \frac{1}{2}$ where $|x| \leq 1$ and 0 otherwise
- 高斯
  - $K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2}$
- Epanechnikov (parabolic)
  - $K(x) = \frac{3}{4}(1 - x^2)$ where $|x| \leq 1$ and 0 otherwise

1.5 Mean-shift 结论

Mean-shift 算法有许多要考虑的利弊。

优点：

- 非常通用且独立于应用程序。
- 无模型。mean-shift 不假设数据集群的任何先验形状。
- 仅取决于定义窗口大小的单个参数。如果采用吸引池，则需要额外的参数（R 和 C）。
- 查找多种模式。分布函数的模式是局部极大值，这些极大值的位置是簇的中心。
- 对数据中的异常值具有鲁棒性。如果离群点距离所有其他点都远于窗口大小，则平均值偏移不会试图将离群点强制到现有集群。

缺点：

- 输出取决于窗口大小，确定适当的窗口大小并不简单。
- 计算所有数据点的平均偏移相对昂贵。
- 与特征空间的维数不匹配。

2 目标识别

2.1 目标识别任务

目标识别领域可以分解为许多不同的视觉识别任务。

分类：分类是教计算机根据对象的类别正确地标记图像。它的重点是回答问题，“这是一个特定对象的图像吗？”“这个图像包含一个特定的对象吗？”通常，分类问题回答的是 yes 或 no。例如，典型的分类问题可能会询问图像是否包含建筑物，或者图像是湖泊。

图像搜索：此识别任务涉及搜索特定对象的照片集合；例如谷歌照片。

组织照片集：对象识别有助于根据照片的位置、活动的相似性、照片中出现的人和其他可识别的对象来组织照片集。

检测：检测的重点是“图像中的特定对象在哪里？”传统的检测方法是搜索包含相关对象的边界框。使用分割技术，也可以更具体地从图像中的像素中选择对象，这就是所谓的精确定位。

检测还可以用于查找几何和语义属性。例如，检测任务包括询问“红绿灯亮了吗？”，“两栋楼的夹角是多少？”。检测发现的其他常见属性包括对象之间的距离，以及图像的对象视图。

单实例识别：单实例识别不是搜索一般的对象分类，而是试图识别图像中的特定对象或地标。例如，我们可能想确定图片中是包含金门大桥还是仅仅包含一座大桥。我们可能想找一盒特定牌子的麦片。

活动或事件识别：活动或事件识别用于检测照片中发生的内容。例如，我们可以问人们在做什么，或者活动是不是婚礼。

2.2 挑战

建立良好的目标识别方法面临的挑战包括图像和类别匹配。由于计算机只能看到图像的像素值，因此目标识别方法必须考虑到大量的方差。

分类号：研究表明，人类可以识别大约 10000 到 30000 种物体。目前，最好的视觉识别方法可以处理大约 200 个类别用于检测，1000 个类别用于分类。许多研究人员正致力于建立图像数据集，以提高对象识别能力；对更多类别进行识别是许多竞争的主题！

视点变化：有许多潜在的方法来查看一个对象，并且视点的变化会导致一个对象看起来非常不同。

光照：不同级别的光，特别是弱光或不同的光方向，将导致阴影移动，对象的细节变得模糊。

尺度：属于一个类别的对象可以有多种大小。如果一个分类只训练特定大小的对象，那么它将无法识别不同大小的相同对象。解决这种偏差的一种方法是找出一种方法，使数据集更好地解释尺度变化。

变形：对象可以改变形式，导致看起来非常不同，但仍然被认为是同一个对象。例如，一个人可以用多种姿势拍照，但如果他们弯腰或双臂交叉，仍被视为一个人。

遮挡：对象可能被遮挡，从而隐藏其特征几何体的各个方面。虽然遮挡对艺术家来说是一个很好的工具（见 Magritte），但对计算机来说却是一个挑战。

背景噪音：背景和前景中的纹理、颜色和形状之间的相似性可能使通过在对象中混合和/或使其看起来不同而难以检测到对象。

类内变化：在一类对象中可能存在显著的形状变化事件。例如，从一个凳子到一个躺椅，从一个豆荚袋到一个抽象的艺术长凳，所有的东西都可以被认为是一把椅子。

3 K-最近邻

3.1 监督学习

我们可以使用机器学习来学习如何根据图像的特征来标记图像。监督学习的目标是使用现有的数据集找到以下方程式：

$$y = f(\mathbf{x})$$

在上面的方程中，y 是输出，f 是预测函数，x 是图像的一组特征。

监督学习分为两个阶段：训练阶段和测试阶段。在训练阶段，我们将方程 f 拟合到训练数据集，它将图像特征集与已知的标识符标签匹配。f 可以通过最小化预测误差，即 y 和 f（x）之间的差来估计。

在第二阶段，即测试阶段，我们使用一个新的测试示例来评估我们的方程 y=f（x），这是 f 以前从未见过的。我们可以比较 f（x）和它的预期输出 y，以评估预测函数是否对新图像起作用。如果预测功能对新图像不起作用，它就不是很有用！

使用监督机器学习的一种方法是学习分类任务的决策规则。决策规则将输入空间划分为由决策边界分隔的决策区域。

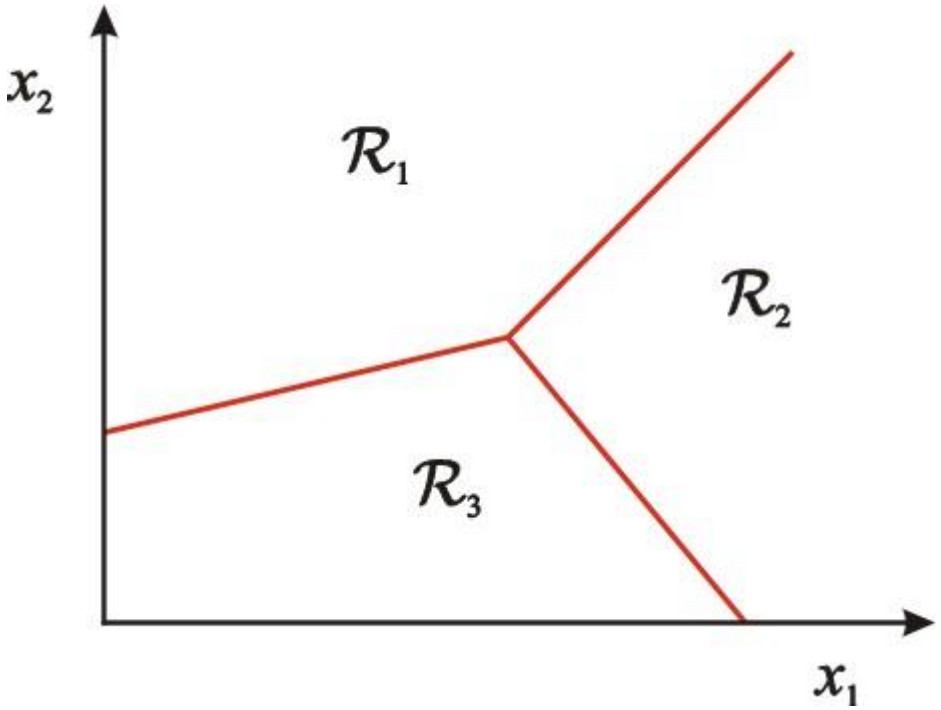

图 3:决策区域和决策边界(红色)三类,R1, R2 和 R3 在二维的特征空间

3.2 近邻分类器

最近邻分类器是一种基于对象最近邻的分类算法。我们给一个新的测试数据点分配与其最近的训练数据点相同的标签。
利用特征间的欧几里德距离找到最近邻。对于一组训练数据，其中x_n和x_m是第 n 个和第 m 个数据点，距离方程写成：

$$Dist(X^n, X^m) = ||X^n - X^m||^2 = \sqrt{\sum_i (X_i^n - X_i^m)^2}$$

最近邻分类器的定义允许围绕训练集中的每个数据点形成复杂的决策边界。

3.3 k 近邻分类器

我们可以通过查看 k 个最近邻而不是仅仅查看最近邻来增强最近邻分类器。k 近邻分类器计算 k 个近邻，并根据在这组近邻上计算的分数来标记新对象。一种常用的度量方法是将一个新对象分配给其大多数近邻所属的类别。启发式方法被用来打破联系，并根据最有效的方法进行评估。

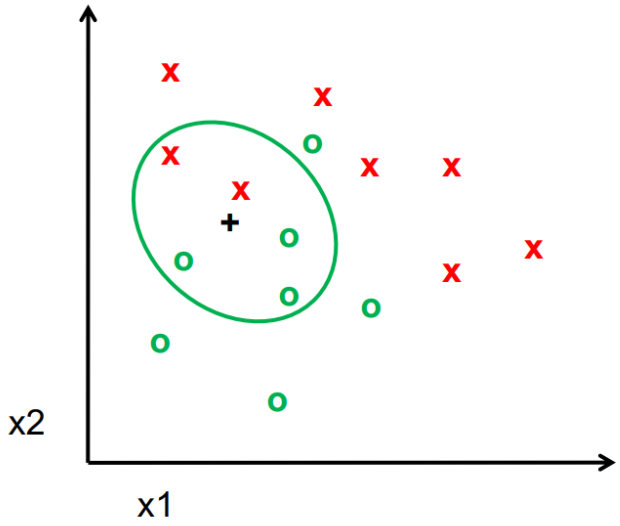

图 4:对于+数据点，绿色圆圈表示它是 k 近邻，对于 k=5。由于五分之三的近邻是绿色圆圈，我们的测试数据点将被归类为绿色圆圈。
与最近邻分类器一样，k 最近邻算法允许使用相对简单的方程来处理复杂的决策边界！

3.4 K 近邻算法的优点

k-nn 是一种非常简单的算法，它是一个很好的尝试。此外，k-nn 具有非常灵活的决策边界。使用 k-nn 的另一个原因是，在无穷多个例子中，1-nn 可证明的误差最多是 bayes 最优误差的两倍（证明超出了此类的范围）。

3.5 K 近邻算法的问题

3.5.1 k 的值如何选择

在使用 k 近邻算法时，选择合适的 k 值是非常重要的。如果 k 值太小，则算法对噪声点太敏感。如果 k 的值太高，则邻域可能包含来自其他类的点。类似地，随着 k 的增加，决策边界将变得更加平滑，并且对于较小的 k 值，将有更小的区域来考虑。

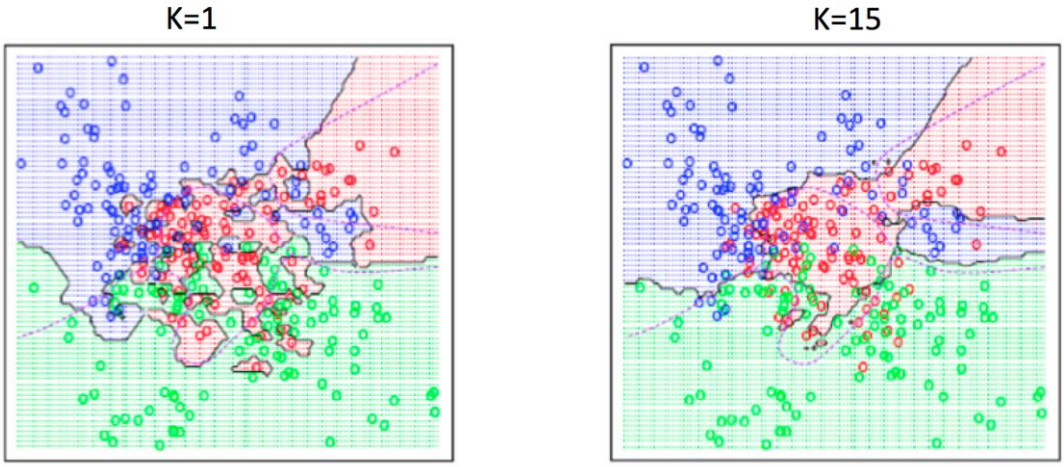

图 1:比较 K = 1 与 K = 15 的结果

解决方案：交叉验证！找到 k 的适当值的一种方法是从训练集创建一个持久性交叉验证/开发集。从那里，我们将调整开发集上的超参数（例如，k），以最大限度地训练，然后“测试”开发集的准确性。然后我们将从训练集更改验证集并重复，直到找到 k 的最佳值。

3.5.2 欧氏度量

k 近邻算法可能出现的另一个问题是，使用欧几里德测度可能会产生反直观的结果。请参见下面的示例:

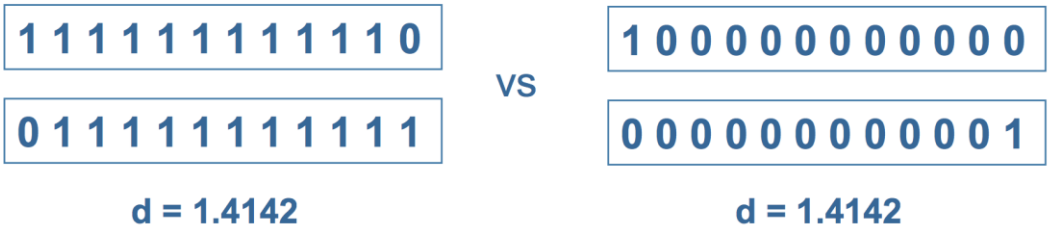

图 3：不同的值，但是使用欧几里德度量将它们等同于相同的值

解决方案：正则化！将矢量归一化为单位长度将保证使用正确的欧氏度量。

3.5.3 维度灾难

在使用 k 近邻算法时，我们需要记住的一个问题是如何处理更大的维数。当维数增加时，我们需要覆盖更大的空间来寻找最近的邻居。这使得计算最近点的时间越来越长，算法运行速度也会越来越慢。这也意味着我们需要更多的例子来训练。目前，维度问题还没有一个最佳的解决方案。

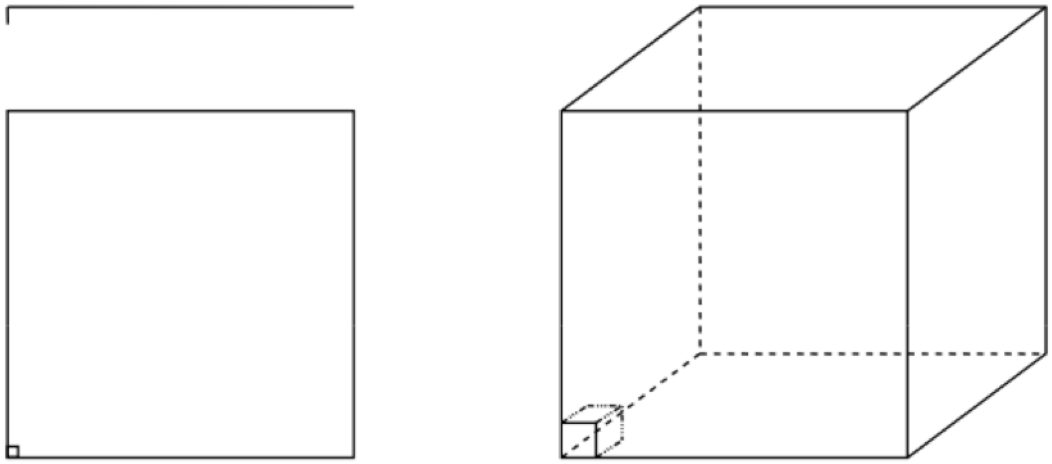

图 4:大尺寸需要较长的计算时间

问题：假设单位超立方体中有 5000 个点均匀分布，我们要应用 5-nn（最近邻）。假设我们的查询点位于原点：

在一维中，我们必须平均走 5/5000=0.001 的距离才能捕捉到 5 个最近的邻居

在二维空间中，我们必须得到一个包含 0.001 体积的正方形

在 D 维中，我们必须得到$(0.001)^{1/d}$

注意：k 近邻只是众多分类器中的一个。尽管如此，为数据选择“最佳”模型并不总是容易的。最好考虑一下你能把你的模型推广到什么程度（也就是说，你的模型从训练过的数据推广到一个新的集合的程度如何？）.

3.6 偏差-方差权衡

泛化误差的关键是通过找到正确的参数数目/类型来获得较低的泛化误差。泛化误差有两个主要成分：偏差和方差。偏差定义为所有训练集的平均模型与真实模型之间的差异。方差定义为从不同训练集估计的模型之间的差异。我们需要在偏差和方差之间找到正确的平衡，因此，偏差-方差权衡。参数太少的模型由于偏差太大（灵活性不够）而不准确。同样，参数过多的模型由于方差过大（对样本太敏感）而不准确。不正确的接头类型如下：

欠拟合：模型太“简单”，无法表示所有相关的类特征。

- 高偏差低方差
- 高训练误差和高测试误差

过拟合：模型过于“复杂”，拟合数据中不相关的特征（噪声）。

- 低偏差高方差
- 高训练误差和高测试误差

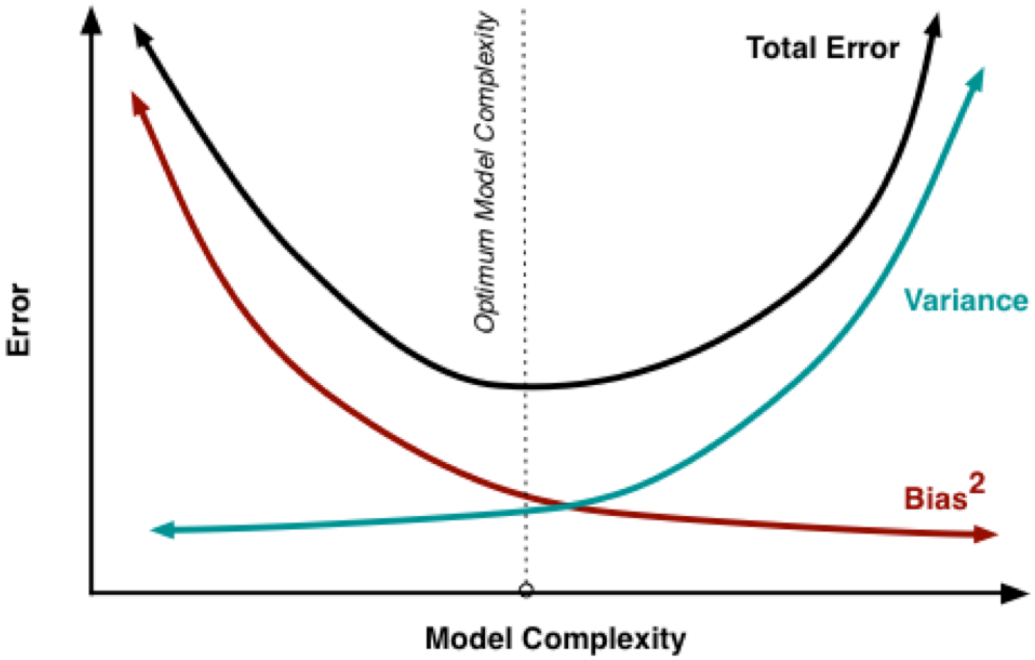

图 5:描绘了偏差和方差之间的权衡