# 机器学习

6.1 介绍

机器学习算法是解决来自许多学科的问题的通用工具，没有详细的领域特定的知识。 它们已被证明在大量语境中非常有效，包括计算机视觉，语音识别，文档分类，自动驾驶，计算科学和决策支持。

核心问题：机器学习的核心问题是从带有标签的数据中学习出一个好的分类规则。这个问题由兴趣域X，（称它为实域空间，比如邮件信息，病人病例等等），而一个分类任务，比如将电子邮件进行垃圾分类识别哪些是垃圾邮件，哪些不是垃圾邮件或者确定哪些患者将需要药物治疗。我们通常将我们的实域空间表示为X={0,1}d或者X = Rd ，对应d个布尔值或者实值特征描述的数据。电子邮件消息的特征可以是存在或不存在各种类型的单词，而用于患者记录的特征可以是各种医学测试的结果。为了执行学习任务，我们的学习算法给定一组标记的训练示例，其是X中的点以及它们的正确分类。该训练数据可以是电子邮件消息的集合，每个标记都可以指出垃圾邮件或非垃圾邮件，或一组患者，每个患者标记为是否对给定的治疗反应良好的情况。我们的算法旨在使用训练示例以产生将在新数据上执行的分类规则。机器学习的一个关键特征，它区别于其他算法任务，我们的目标是泛化：使用一组数据，以便对我们还没有看到的新数据执行良好。我们专注于二分类，其中感兴趣的域中的项目被分类为两个类别，如在上面的医疗和垃圾邮件检测示例中。

怎么样学习：一个高效的解决这个问题的办法是使用许多的算法在训练数据上可以找到一个“简单”的规则同时有一个好的表现在训练数据之上。比如在邮件分类上，我们可以找到一些高度的指示性的词语，使得训练数据中的每个垃圾邮件都有这个词汇，而不是垃圾邮件的则一个也没有。在这样的情况下，规则就是“如果有这写词语就是垃圾邮件，没有就不是”这就是一个简单的规则且可以在训练数据上有则良好的表现。或者我们可以发现一种加权的方式来使用正负加权的方式来对这些词语进行加权，使得电子又见小夕中的词语的总加权在训练数据上是正的，在垃圾邮件是上负的。只要训练数据代表了以后需要预测的数据，那么我们有在训练数据上表现良好的规则，同时也会在未来预测的数据上有良好的表现。我们需要精确地描述我们的意思是“简单”，以及它对于训练数据是否代表未来数据意味着什么。事实上，我们将看到几个复杂性的概念，包括位计数和VC维度，这将允许我们做这种形式的数学语句。这些语句可以被视为形式化的Occam’s razor的直观哲学概念。

形式化问题：为了形式化学习问题，假设实例空间上存在一些概率分布D，使得（a）是随机从这个分布D中取出来的独立的点组成的集合S，而(b)则是我们需要预测的新的点，也是我们认为我们的训练数据未来数据的代表。让C\* 表示为目标概念集，它是X中的正的二分类。比如说，医疗患者中对药物治疗有良好反应的所有患者的集合，或者垃圾邮件中所有垃圾邮件的集合。所以，在我们的训练集的每一个点都是根据它是否属于C∗和我们的目标标记是产生一组h⊆X，称为我们的假设，这是接近C∗相对于分布D. H真正的错误是 在 表示D上面的对称差，和概率质量。换个说法就是说h的真实误差是其对从D随机抽取的数据点进行不正确分类的概率。我们的目标是让h的正式误差变小。其中h的训练误差为 ，他是集合S中的一部分，这里的h和C\*是不一样的。那么.训练误差也称为经验误差。请注意，即使是由随机从D点，一个假设h具有较低的训练误差甚至完全等同C∗的训练样本，它是可能的，但有很高的真实误差。这就是所谓的过度拟合训练数据。例如，假设h，就是在正例上市，这相当于一个规则，将训练样本和预测积极的一个例子，当且仅当它已经在训练样本表现的很好时，训练误差为0。然而，这种假设可能会有真正的高误差，因此将高度过度拟合训练数据。更普遍的是，过度关注因为算法通常会优化训练样本。学习和分析算法的设计，我们将解决过度拟合的问题。

为了能够正式的分析过拟合问题，我们引入一个假设类的概念，也被称为一个概念类或集合系统。一个关于H的假设，H是X子集的集合，称为假设。例如，区间在 类是集从 中收集的，在的线性分隔的类的集合为：



也就是说，它是集合的所有集合在RD是线性可分的补充。在这种情况下，X是在平面F也就是说，它是集合的所有集合在 是线性可分的补充。比如X是平面上的4个点的集合{(-1,-1),(-1,1),(1,-1),(1,1)}, 线性分隔符的类包含X的 个可能子集中的14个。给出一个假设H和训练数据集合S，我们通常在算法上要做的是在H中找到最接近于c的假设，为了解决过拟合问题，我们认为如果S与H的一些性质相比足够大，则以高概率所有 具有接近其真实误差的训练误差，使得如果我们找到训练误差低的假设，我们可以有信心 它的真实误差也会很低。

在给出我们的第一个结果之前，我们注意到将假设表示为{-1,1}的值可以作为指示器



这时的h的真实误差为 ，并且训练误差为

6.2过拟合和一致收敛性

我们现在提出两个方式，解释如何防止过度拟合。给定一个分类假设H，第一个结果只需要给定一个就可以了，所以只要我们的训练数据集比 大，那么我们产生的 将不会有0的训练误差，而真实误差会比 大。这意味着在高概率下，我们的算法发现与训练数据上的目标假设是一致的任何假设将具有低的真实误差。第二个结果表明，如果训练数据集比，那么就在对于任何关于H的假设中不可能会产生训练误差和真实误差超过，这意味着如果我们在训练误差较低的H中找到假设，我们可以确信，即使其训练误差不为零，其真实误差也将很低。

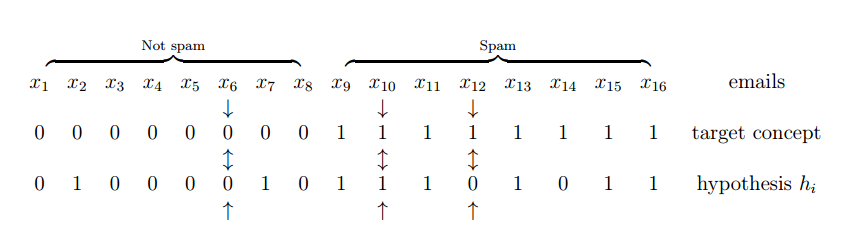
基本思想如下。 如果我们考虑h具有大的真实误差，并且根据D随机选择元素 ，则存在x将属于对称差的合理机会。 如果我们选择足够大的训练样本S，每个点从D中独立选取X，则S与完全不相交的机会将非常小。 这仅仅是针对单个假设h，但是当H是有限的时候，我们现在可以对大的真实误差的所有应用联合约束。 我们在下面形式化。

定理6.1 产生一个H假设类，并且让 ，如果S的训练集大小为



是从分布D绘制的，则在H中每h的概率大于或等于，具有真误差 具有训练误差 。等效地，以大于或等于 的概率，训练误差为零的每个 具有小于的真误差。

证明：让 …是从H中选出来的，且他们的真实误差大于或者等于。这是假设使我们不想要输出的。考虑选择一个样例S

 图6.1：假设 中有1/4的概率预测是不正确的。因此，训练集在为|S|，那么是假设正确的可能性为 (大小为n)，让 作为事件并让 同S保持一致性。那么每一个的真实误差都大于等于



换句话说，我们用来描述一个大小为n的样本S时，在对S进行预测而不出错的情况那么他们的偏差是，连续n次的出现在最后，那就是 。所以他们的联合边界在i次后就是

事实上，具有大于或等于 的真误差的H中的任何假设具有训练误差零的概率最多为 ，将n替换为定理中的大小那么最多就是

这在机器学习中被称为PAC学习理论，即如果我们能找到一个 做为样本，那么我们可能得到一个比较正确的值。

定理6.1解决了在H中存在具有零训练误差的假设的情况。 如果最好的 在S上有5％的错误？ 那我们可以相信它的真实误差是低的吗，并说它至多10％的错误率？ 为此，我们想要一个定理6.1的模拟，说明一个足够大的训练集S，每个的训练误差在±的真实误差。 这样的情况被称为一致收敛，因为我们要求训练集误差在H中的所有集合上均匀地收敛到它们的真实误差。为了直观地看出为什么这样的语句对于足够大的S和单个假设hi应该是真的， 在10％的位置上不同的字符串，并随机选择大样本的位置。 样品中不同的位置数量将接近10％。

为了证明了一致收敛的边界，我们使用一个尾不等式伯努利独立随机变量的和（比如，抛硬币）。以下是特别方便，在附录部分12.4.11 Chernoff边界的变化。

定理 6.2(Hoeffding边界)  是从{0,1}中取值的，且互相独立，p表示x=1的概率。 （比如，扔n次硬币，s是人头数出现的和），让



定理6.2保证了6.1的一致收敛性。

定理6.3（一致收敛性）让H是一个分类 ，其中一个训练集的大小是

 ，并用来描述分布D，那么它的概率就应该大于等于 ，每一个H中的h就满足 。

证明：首先，修正一些，让 成为h假设错误的指示器指出在样本S中的第j个。是独立的从{0,1}中选择的，它的概率是中的h的真实误差，是h的训练误差。Hoeffding边界保证了事件 中的 是小于等于 。根据一致边界从 中选出的 ，存在之间的不同是真实误差和训练误差之差大于 或者小于等于，使用n个从定理中的值，上述不等式的右侧最为理想的是 。

定理6.3证明我们的训练样本的优化的方法，即使我们不能找到一个规则的零训练误差。如果我们的训练集足够大，概率高，S上的良好性能将转化为D的良好性能。

注意：定理6.1和6.3需要答以是有限的是有意义的。第6.9节中增长函数和VC维数的概念，将定理推广到某些无限假设类6.3。

6.3 例证和Occam’s Razor

我们现在举一些例子来说明定理6.1和6.3的使用，并使用这些定理给一个正式的连接到奥卡姆剃刀的概念。

6.3.1 析取学习

假设一个空间域为 并且我们的目标概念可以被析取或是和“或（or）”来表示特征，比如 或直接表示为。比如，我们想要尝试预测一个邮件信息是不是垃圾邮件，我们的特征对应于存在或不存在不同的垃圾邮件的可能指示符，则这将相应于这样的指示，即存在这些指示符的一些子集，使得每个垃圾邮件具有至少一个垃圾邮件，并且每个非垃圾邮件都没有以上没有特征。事实上，让H表示这个分类的析取，其中 ,因此，通过定理6.1，只要找到一个一致的分离过的样本大小S 为



我们怎样才能有效地找到一个一致的析取存在呢？这里是一个简单的算法。

简单的析取学习：给出一个样本S，不需要其他的特征，将他们全部的S中消极的样本设置为1，输出的概念h表示他们的或运算的特征表示。

引理 6.4： 这个样本析取学习产生的析取h表示一致性是和S一样的，无论目标是否需要析取。

证明：假设目标概念 是一个析取目标，对于中的任何 ，任何否定的例子中，通过定义OR，的值不会被设置为1.因此h中的也是一样的。由于h包含在c \*中列出的所有变量，这确保h将正确地预测S中的所有正例子上的正值。此外，h将正确地预测S中的所有负例子上的负值，因为通过设计在任何反例中设置为1 。 因此，h在S中的所有示例是正确的。

因此，结合引理6.4与定理6.1，就可以得到一个有效的PAC分类器了。

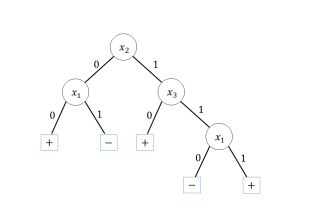
6.3.2 Occam’s razor

Occam’s razor是奥古斯特的威廉在公元1320年左右所说的概念，一般来说，人们应该更喜欢对更复杂的解释。18为什么要这样做，我们可以正式声称为什么这是一个好主意？ 如果我们每个人对于哪些解释比其他解释更简单不同意怎么办？ 事实证明，我们可以使用定理6.1来做一个数学陈述的Occam’s razor解决这些问题。

首先，我们的意思是一个规则是“简单的”？让我们假设我们每个人都有一些描述规则，使用位（因为我们是计算机科学家）的方法，也称为描述语言，我们每个人可以不同，但是我们可以肯定地说，一个事实是在任何给定的描述语言中，至多有 个规则可以使用少于b个位来描述（因为 因此，通过将H设置为可以以少于b位描述的所有规则的集合并插入定理6.1，得到以下：

定理 6.5（Occam’s razor）修正任何描述语言，并考虑从分布D中抽取的训练样本S.至少概率有，任何同h和S一样的规则都可以被描述为这种语言使用少量的bit位且 ， 。等同地，以至少1-δ的概率，可以在少于b个位中描述的所有规则将具有。

比如， 并且忽略了低位项 ，这代表着如果数字为需要写出我们规则的一致性同训练数据的10%，那么我们就可以相信在D中只有10%的误差，这个定理也许令人惊讶的是，它意味着我们每个人都有不同的描述规则的方法，但都使用Occam’s razor.。请注意，这个定理并没有说复杂的规则必然是坏的，甚至是给定的两个规则与复杂的规则必然是更糟糕的数据一致。它所说的是，Occam’s razor是一个很好的方式，在简单的规则是不可能愚弄我们，因为没有那么多简单的规则。

图6.2 一种决策树有三个节点和4个叶子，这个数的一致性布尔函数是

6.3.3 应用：决策树学习

一种流行的机器学习的实用方法是学习决策树，见图6.2。而求最小决策树适合给定的训练样本是NPhard，还有一些启发式算法中所使用的practice.19假设我们运行这样一个启发式的一个训练集和输出树的k个节点。这样的树可以使用 比特：log 2(d)比特来给出根中的特征的索引， 比特来指示每个子是否是叶，并且如果是， 标签，然后分别是 和位来描述左子树和右子树，其中 是左子树中的节点数量，是节点数量右子树中节点的数量。 因此，通过定理6.5，如果我们可以产生一个具有少于 节点的一致树，我们可以确信真实误差是低的。

6.4 正则化：惩罚的复杂性

定理6.3和6.5提出了一下想法。假设没有与训练数据完全一致的简单规则，但我们注意到有非常简单的规则，训练误差为20％，然后一些更复杂的训练误差为10％的规则，等等。 在这种情况下，也许我们应该优化一些训练误差和简单性的组合。 这是正则化的概念，也称为复杂性惩罚。

具体来说，一个正规化是一个惩罚惩罚更复杂的假设。鉴于我们的定理到目前为止，一个自然的措施，假设的复杂性是多少位，我们需要写下来。现在考虑修正一些描述语言，并且让 表示可以在该语言中的i比特中描述的那些假设，因此 。让 ，重新认识6.3的边界，我们知道它的概率至少 ，所有的 都满足 。现在，依靠这些联合边界都是在i中的，事实上 ，并且事实上 ,给出推论。

推论6.6 这些描述语言，且考虑训练数据集S是从分布D中选出的，它们的概率大于等于，所有的h满足

，

其中大小（h）表示在给定语言中描述h所需的比特数。

推论6.6给了我们正在寻找的权衡。 它告诉我们不是寻找低训练误差的规则，而是可能想要在显示的公式中搜索具有低右侧的规则。 如果我们可以找到一个这个数量很小的，我们可以确信真正的错误也将是低。

6.5 在线学习和感知算法

到目前为止，我们一直在考虑通常所说的批处理学习场景。 你被赋予一个“批次”的数据|训练样本S，你的目标是使用它产生一个假设h，在新数据上具有低误差，假设S和新数据都是从一些固定的 分布D.我们现在切换到更具挑战性的在线学习场景，其中我们去除了从固定概率分布中采样数据的假设，或者根据任何概率过程。

具体地，在线学习情景如下进行。 在每个时间t = 1,2…

1. 该算法以任意示例 呈现，并且被要求对其标签 进行预测。
2. 该算法被告知示例 的真实标签，并且如果那么就错误。

学习算法的目标是使总共尽可能少的错误。 例如，考虑一个电子邮件分类器，当新的电子邮件到达时必须将其分为“重要”或“它可以等待”。 然后，用户查看电子邮件并通知算法是否不正确。 我们可能不想将电子邮件建模为固定概率分布中的独立随机对象，因为它们通常是对先前电子邮件的回复，并且彼此建立。 因此，在线学习模型将比该设置的批处理模型更合适。

直观地，在线学习模型比批模型更难，因为我们已经去除了我们的数据包括从固定概率分布的独立抽取的要求。 实际上，我们将很快看到，在在线模型中具有良好性能的任何算法可以被转换为在批模型中具有良好性能的算法。 尽管如此，在线模型有时可能是一个更干净的模型，用于设计和分析算法。

6.5.1 样例：析取学习

作为一个简单的例子，让我们重温在在线模型中学习分离的问题。 我们可以通过从假设 ，并且将其用于预测来解决这个问题。 我们将保持不变量，目标析取中的每个变量也在我们的假设中，这在开始时是清楚的。 这确保了可能的唯一错误是对于其中 是正的，但是 是负的示例x。 当发生这样的错误时，我们简单地从h中移除任何设置为1的变量。 由于这些变量不能在目标函数中（因为x是负的），所以我们保持不变量并从h中删除至少一个变量。 这意味着该算法在与分离符合的任何一系列示例上总共犯了最多的错误。

事实上，我们可以通过找出并没有确定性算法可以保证少于错误来这种限制。

定理6.7：对于任何确定性算法A，存在实例σ和析取的序列，使得A对由 标记的序列σ会产生错误。

6.5.2 二分算法

如果我们不关心运行时间，那么保证对属于任何给定类H的目标至多做出 错误的简单算法称为二分算法。 该算法简单地维持版本空间 ，其包括与到目前为止所看到的每个示例上的标签一致的所有，并且基于对这些函数的多数表决来预测。 每个错误都保证将版本空间V的大小减小至少一半（因此名称），因此错误的总数最多为。 注意，这可以被看作在H下写一个函数所需的位数。

6.5.3 感知算法

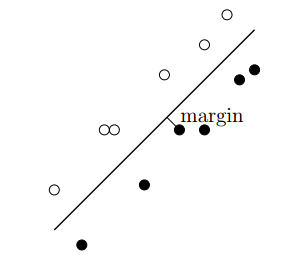
感知器算法是学习在d维空间线性分离器的有效算法，取决于数据的分离的边缘。具体来说，假设目标函数可以描述为一个向量 这样每个正例x我们 和每一个负例。请注意，如果我们认为这个例子X作为空间中的点，然后 是x到超平面的距离为 。因此，我们可以认为我们的假设，说明存在一个线性分离器通过正例的起源在一边，另一边的所有负面的例子，所有的距离至少为 ，该量γ称为分离余量。如图6.3：

图6.3 线性分割符的边距

感知器算法的保证将是在到目前为止看到的所有示例xt中，错误的总数最多 ，其中 。 因此，如果存在通过原点的超平面，其相对于包围数据的最小球的半径将正实例与负实例正确地分开大的余量，则错误的总数将较小。 该算法非常简单，并如下进行。

感知算法：开始我们将权重向量设置为 ，那么当t=1,2，…时就可以做到：

1. 给出一个样例 ，并预测
2. 如果预测是错的，就对它进行更新：
   1. 如果是正值，就将
   2. 如果是负值，就将

虽然简单，但是Perceptron算法对其错误总数有很好的保证。

定理6.8 在任何的序列中 如果存在向量 ，使得对于正的样例，对于负的样例(比如，一个线性分类距离 )，感知算法最有产生 个错误，其中。

为了得到这个边界值，若我们将所有的实体 都乘上100，我们可以吧素有实体的权重 乘上100，他依然满足条件，那么边界是可以缩放的，比如我们的测量单位。

证明：考虑一些 ，我们将跟踪两个变量， 和 ，首先，每次我们产生来了一个错误，就至少增加1，这是因为如果 是一个正例，那么



然后，对每一个错误，我们都认为最多增加 ，首先考虑错误的正的例子。若果在正的例子 中产生了一个错误，我们就可以得到



若在中间过程中，我们产生了一个错误，就代表着 ，若是在负例当中产生的我们就有了



注意到它是很重要，我们就只更新这些错误。

所以，如我们有M个错误，那么 ，并且 ，或者等价的 。我们就可以的到 ，也就是说我们预测的w，且  将不会比w更长，这意味着我们有了：



的需求

6.5.4 扩展：可部分数据和hinge-loss(最大边际目标函数)

假设上面存在的分类所有示例得到了 ，比如将所有的电子邮件正确的分类为重要的和不重要的。在现实中是几乎不存在的，也就是说我们得到的

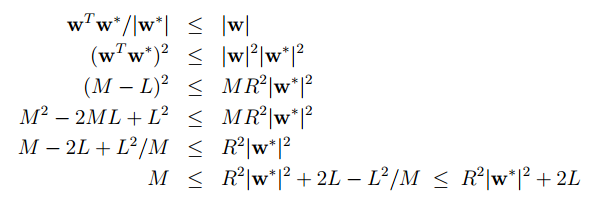
并不是很好的怎么办？我们可以看到这些证明有什么用：若果这些例子中的 并没正确的分类，当证明第二部分有效的时候，那么第一部分的分解(w的点积随 增加)，但是如果并没有频繁的改变，那么 只是一个 “较小错误”，那么我们也就只能产生更多的错误。

为了让它更加形式化，将hinge-loss定义为 上的权重 为最大值 .换句话说如果我们的需求是 ，则hinge-loss为0；否则，hinge-loss是 的值，同样的如果上的权重的hinge-loss在负例中最大值为.给定标记的样例S的序列，定义他们的所有hinge-loss为 是hinge-loss上样本集S上的标签的和。就可以得到以下推理。

定理6.9 在任何一个样例S中的序列  感知算法最多会产生 个错误，其中 。

证明：之前在更新感知算法中每次更新最多使增加 ，所以在算法上产生M个错误，我们就有了 。

难以再说的是短发的每次更新 至少增加1，相反的在正例方面我们将通过来增加(也可能是负例)，他至少是 。同样的在负例中我们同样通过来增加，同样的至少需要。如果我们将他们所有的错误加起来，我们得到了他们的值，然后我们就得到了 ，其实在里使用的hinge-loss其实都不是负的，所以S中总和总是大于w所产生的错误。

 最后做一些代数变换。令 ，因此我们就得到了：

6.6 核函数

假设及时最好的的权重 都不怎样？例如，可以使用可替代的线性分割符判定边界，重要和不重要的邮件之间的边界看起来更像一个圆圈，如图6.4所示。

解决这样的问题有一种强而有力的方法就是使用所谓的核函数，或者有的时候称其为“核技巧”，这只是一种想法。假设有这样一个函数 ，称为

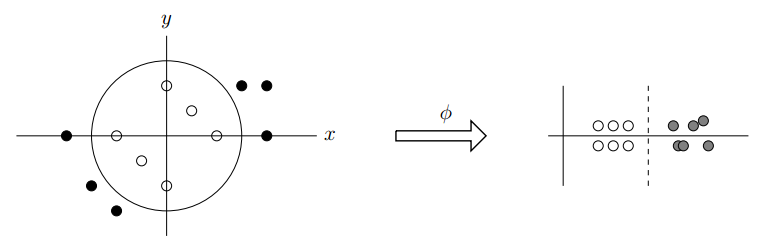
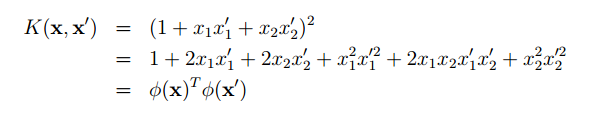
“核”，对数据点，对于某些函数 ，其中可能 ，我们也就有了 。事实上，在这种情况下，如果我们可以编写Perceptron算法，以便它只通过点积与数据交互，然后用K的调用替换每个点积，我们可以表现得好像我们已经执行了显式的函数φ 而不必实际计算φ。

图 6.4 输入的 是不可线性分割的数据，但是可以被 分割， ，对应的核函数为：

 比如，考虑一个核函数为，其中 。在这样情况下将 映射到 空间维数中。比如 我们就有了(使用表示x的第i个坐标)

对于其中的，注意到此空间中的线性分隔符可以对应于更复杂的判定边界，例如原始空间中的椭圆。比如，对于的超平面 对应的原始空间中的圆 ，如图6.4

这是因为如果高维空间中，存在一个比如想定理6.9一样的小的边界，则算法就可以在执行后得到一个良好的结果，产生很少的错误，但是好的结果我们没必要对进行映射计算。

所以如何将感知算法视为只需要通过点积和数据就可以改变呢？请注意，始终是数据的线性组合。比如，如果我们对第一，第二，第五示例都做出了错误，他们分别是正，正，负的，我们可以用 来表示。所以如果我们用这样的方式去跟踪w的变化，那么我们就可以用来预测新的示例 ，我们可以把它写为 ，因此，如果我们用“K”去替换每一个点积，我们就可以明确的执行 映射的算法，这就是所谓的内核化。

示例上的许多不同的成对函数是合法的内核函数。 一种创建内核函数的简单方法是通过以下定理将其他内核函数组合在一起。

定理 6.10 假设 是核函数，那么

1.对于任何常数c≥0，是一个合法的内核。 事实上，对于任何标量函数f，函数 是一个合法的内核。

2. 是一个合法函数

3. 也是合法函数

你将在练习6.9中证明定理6.10，注意，这立即意味着通过使用 是合法核，事实上这个函数就是在说 是一个合法函数，函数 是一个合法的内核函数，然后将它们相加，然后乘以它本身k倍。 另一个流行的内核是高斯内核，定义为：

如果我们将内核视为相似性的度量，则该内核将两个数据对象之间的相似性定义为随着它们之间的平方距离而按指数减小的量。高斯核可以表示为真核函数，首先将其写为 ， ，然后采用的泰勒公式展开，应用规则定理6.10。 从技术上讲，这最后一步需要考虑无数许多规则的应用，并允许无限维向量空间。

6.7 在线批量转换

假设我们有一个具有良好错误限制的在线算法，例如感知算法。 我们可以使用它在分配（批处理）学习设置中获得保证吗？ 直观地，答案应该是肯定的，因为在线设置只是更难。 事实上，这种直觉是正确的。 我们在这里介绍了这种在线到批量转换的两种自然方法。

转换过程：随机停止。假设我们有一个具有错误约束M的在线算法A.假设我们在大小为 的样本S上单次运行算法。 令 为A在第t个示例上出错的事件的指示符随机变量。 由于对于任何集合S， ，我们肯定有 ，其中期望是从的随机抽取的 。 通过期望的线性，并且将两边除以，我们因此具有：



令 表示由算法A用于对第t个示例进行预测的假设。 由于第t个例子是从D中随机抽取的，因此我们有 。 这意味着如果我们从1到|S|随机选择t，比如，在随机时间停止算法，则在S的抽取和t的选择上取得的随机性所得到的预测规则的预期误差为最多 ， 如等式（6.1）所给出。 因此，我们有：

定理6.11 (在线随机停止批处理)如果具有错误约束M的在线算法A在大小为的样本S上运行。 并且在1和 之间的随机时间停止，假设h的期望误差产生满足。