**1. 前言**

这个世界不缺少专家，我只是期待他们中有一位能把事情说清楚。

机器学习很火。  
机器学习专家很贵。  
所有大型互联网公司都驾着机器学习的马车朝着人工智能前进。

然而今天哪怕是互联网从业者，大部分也是不知道机器学习到底是什么的。机器如何学习？机器学到的是什么？为什么机器经过学习能够神奇的预测用户的喜好、股票的涨跌？人们好奇又渴望。  
这里所说的从业者可能是开发工程师，可能是产品经理，也可能是运营，他们与机器学习专家们在同一家公司工作，参与同一个项目，但机器学习算法对他们仍然像黑魔法一样，神秘又疑惑。

这样的局面未免让人沮丧，毕竟如果相对论都可以在高等教育中得到普及，有什么领域是复杂到没办法好好说清楚的呢。据我有限的观察，造成这个局面的原因无非两种：

1. 不少专业人士乐于将机器学习包装得晦涩曲折，以享用他人迷惑眼神中的优越感。
2. 很少人把机器学习以直接的、让人容易理解的方式说出来。有那么几个在这样做的人，面向的也是专业领域学习者而非一般的科普受众。

我鄙视第一种人类。  
我希望所有的写作者都能够追逐Richard Stevens的光芒，把复杂的东西变简单，追求简洁明了，追求直接易懂。

这个系列文章，我将试着为开发工程师，产品经理、设计师、所有希望了解学习机器学习的人，介绍机器学习的原理、方法和实战技巧。  
我追求它尽可能好理解的同时，也会保持它的准确度和实用度。理论方面，以周志华的[《机器学习》](https://link.zhihu.com/?target=https%3A//book.douban.com/subject/26708119/)西瓜书，林軒田機器學習系列课程([Foundations](https://link.zhihu.com/?target=https%3A//www.youtube.com/playlist%3Flist%3DPLXVfgk9fNX2I7tB6oIINGBmW50rrmFTqf)[Techniques](https://link.zhihu.com/?target=https%3A//www.youtube.com/playlist%3Flist%3DPLXVfgk9fNX2IQOYPmqjqWsNUFl2kpk1U2))，Andrew Ng's [Machine Learning](https://link.zhihu.com/?target=https%3A//www.coursera.org/learn/machine-learning) 为学习资料，结合我个人的理解及日常与朋友同事的讨论。实用实战方面，我将以手机淘宝中第一款\*\*\*\* DAU (Daily Active User) 导购产品——有好货为例子，如果你从事导购或者电商相关工作那么对例子中的场景一定非常熟悉。如果你对导购并不了解也不用担心，讲解的重点仍是机器学习原理和方法的普世应用，理解了原理方法之后可以在任何适合机器学习的场景中进行实践。

这是这个系列的第一篇，看完这篇您将知道

* 什么是机器学习？
* 机器学到的到底是什么？
* 什么样的问题适合用机器学习来解决？

**2. 什么是机器学习？机器学到的到底是什么？**

**2.1 什么是机器学习？**

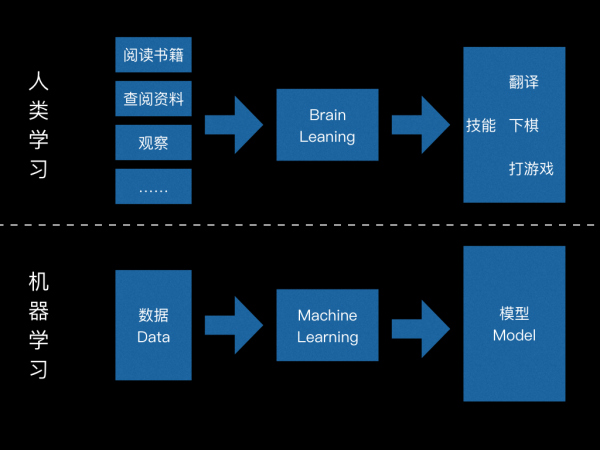
在讨论机器学习之前，我们首先看看人类是如何学习的。如图1上半部分所示，人类通过阅读书籍、查阅资料，观察得到信息，这些信息经过人脑学习，最后习得了某种技能。

机器学习也是类似，只不过机器学习的输入是**数据（Data）**，学到的结果叫**模型（Model）***(备注1)*。从数据中学得模型这个过程通过执行某个**学习算法（Learning Algorithm）**来完成。

* 数据（Data）
* 学习算法（Learning Algorithm）
* 模型（Model）

是机器学习的三个要素。

图1：



当然，上面的类比可能还比较抽象。下面就以“有好货”这个产品为例子，来理解一下机器学习的概念。

首先介绍下有好货（图2所示）。有好货是手机淘宝的一款导购产品，在首页第一屏就能看到产品入口。这款产品在2015-2016我们用了一年的时间，将每日用户数从\*\*\*\*做到了\*\*\*\*，成为第一个每日用户数破\*\*\*\*的导购产品。这里面的一大秘诀就是将个性化推荐技术、机器学习技术与产品设计做了深层次的融合，以个性化推荐和机器学习为内核驱动产品设计的演进和决策。（有好货的故事我将另写一个系列）

这里以有好货的“瀑布流”页面（图2中间）为例。这个页面是一个完全个性化的页面，不同的用户进入到有好货瀑布流页看到的商品推荐是不同的。  
有限的屏幕空间，我们希望给每个用户展现他最有可能点击的商品。那什么商品是当前用户最可能点击的呢？这个预测就由机器学习完成。  
类比图一的概念，这个场景下的**数据**、**学习算法**和**模型**分别对应着：

* **数据**：输入数据包括：用户对商品的浏览、点击历史行为数据以及相应的用户商品特征数据。这些历史数据记录了什么样的用户点击了什么样的商品，什么样的用户对什么样的商品看了没有点击。我们认为这些历史数据中蕴含了某种规律，希望机器学习能把这种规律挖掘出来，在将来面对新的用户和商品时，就能预测是否会点击。
* **学习算法**：机器学习算法有很多，逻辑回归、随机森林都适用于这个场景，但这里我们先不对具体的算法作展开，暂时读者只需要概念性的知道，基于输入数据执行“学习算法”便可产生模型（模型就代表了学习算法从数据中挖掘出的规律）。
* **模型**：学得模型之后，面对新的用户和商品，模型就能作出相应的判断，用户会点击还是不会点击。利用模型的这个“技能”，我们便可以做到给每个用户推荐他最感兴趣的商品了。

总而言之，机器学习是一门研究“学习算法”的学问，“学习算法”基于历史的经验数据产生模型，进而使计算机有了对新情况进行预判和预测的“技能”（比如预测用户的喜好或股票的涨跌）。

图2：



**2.2 机器学到的到底是什么？**

理解了机器学习的概念，我们知道机器学习无非三个要素(1)数据，(2)学习算法，(3)模型

1. **数据**很好理解，当我们希望预测用户是否会点击某个商品，就把历史上用户对商品的点击浏览行为喂给机器学习算法，希望从历史数据中中挖掘出**某种规律**。
2. **学习算法**有很多，上面提到过的逻辑回归、随机森林只是众多算法中的两种。事实上对各种不同学习算法的讨论是机器学习书籍的重点，一章介绍一种，就厚厚一本书了。读者不必着急，这部分我们将慢慢展开。
3. 在这一小节，我想重点讨论的是，我们说机器学习学得的模型可以预测用户是否会点击某个商品，可是**模型在机器内部到底是怎么表示的呢？机器学到的模型到底是什么？**许多人觉得机器学习非常神秘，是因为人类习得的技能并没有一个直观的展示形式，因此很难想象机器学到的模型到底是什么，其实答案非常简单：

机器学到的模型是一个**映射**。

映射，在数学的许多分支就等价于函数（*备注2*）。而函数，我们再熟悉不过了，给定一个（输入集合中的）元素，函数唯一对应（输出集合的）一个输出值。

比如函数f(x)=x^2，给定任意实数x，x的平方就是函数的输出。

比如函数f(x)=w_1*x_1+w_2*x_2+...+w_d*x_d+b，当w_1,w_2,...,w_d是确定的，那么给定一组x_1,x_2,...,x_d，就能唯一确定一个输出值f(x)。（事实上这个就是最简单的一种机器学习模型——线性模型）。

而在有好货的例子中，机器学习学到的模型就是这样一个函数：

给定一个用户和商品，这个函数就能够唯一输出一个分数，表示用户点击该商品的可能性。

这就是机器学习的秘密。

**3. 什么样的问题适合用机器学习来解决？**

不少计算机科学专业的同学可能会有些疑惑，计算机科学在本科阶段教授了大量的算法——字符串匹配算法、排序算法、贪心、动态规划，算法导论厚厚一千多页，可这些都不属于机器学习的范畴，机器学习也不是计算机科学本科的必修课。

那到底算法导论中的算法跟机器学习算法有什么区别呢？  
**什么样的问题适合用机器学习来解决？什么情况需要使用机器学习呢？**  
答案是：

难以用规则解决的问题，可以尝试用机器学习来解决。（*备注3*）

算法导论中经典的排序问题，无论解法是快排还是归并排序，解法已经是一个确定的规则。但是机器学习问题，比如垃圾邮件识别，比如辨识一张图片中的物体是不是树叶，就很难用规则来解决。前者的规则难以穷举，后者则根本很难描述辨别树叶的规则。

因此，仍然以规则堆砌的观念来看待算法的朋友们注意了，  
永远不要跟机器学习专家说：“加条规则呗”  
永远不要跟机器学习专家说：“加条规则呗”  
永远不要跟机器学习专家说：“加条规则呗”  
God Bless You~

**3.1 适合用机器学习解决的问题的必要条件**

另外，在2.2节我们提到：当我们希望预测用户是否会点击某个商品，就把历史上用户对商品的点击浏览行为喂给机器学习算法，因为我们认为历史数据中隐藏着用户是否会点击商品的**某种规律**。

这其实道出了能用机器学习解决的问题需要具备这样的必要条件

有大量数据，并且数据中有隐藏的某种规律或模式

如果某些问题没有任何的规律，比如抛硬币，那么无论有多少数据也是不行的。

**3.2 小测试**

读到这里，不如试试看你对机器学习理解的怎么样了。  
判断下面这些问题适不适合用机器学习解决。能不能用机器学习解决。

问题：预测下一次六合彩的中奖号码。  
答案：不能用机器学习解决，因为跟投硬币一样摇奖是随机的，并没有规律。

问题：判断一个图形是否是圆。  
答案：无需用机器学习解决，因为有明显的规则。

问题：预测股票的涨跌。  
答案：可以用机器学习辅助交易并盈利。要是你发现自己能很好的解这个问题，请跟我做朋友吧 ：）

问题：预测一个10岁的小朋友长大了会喜欢的女孩子的类型。  
答案：可能不能用机器学习解决，因为缺少“大量数据”这点必要条件。

**4. 小结和预告**

这是系列文章的第一篇，我们首先介绍了机器学习的基本概念，机器学习的三个要素：数据，学习算法，还有模型。

然后我们揭示了机器学到的模型，本质上就是一个映射，或者函数。

最后我们总结了机器学习适合解决的问题，是那些难以用规则解决的问题。并且机器学习的必要前提不仅是有大量的数据，而且需要数据中确实存在隐藏的某种规律，机器学习才能帮的上忙。

希望我有把事情说清楚，有任何疑惑或者问题，欢迎留言。我回答后会把FAQ附在每篇文章的后面。下一篇将细化具体的机器学习原理，可能会引出一个入门级的机器学习模型。您有什么希望了解学习的内容，也可以留言。

**赛前采访**

在开始这个系列的第二篇文章之前，非常荣幸得采访到霹霹博士。霹霹目前在阿里巴巴认知计算实验室做AI方向的艰深探索，我第一次跟他交流机器学习跟数据挖掘的区别时得知，数据挖掘领域鼻祖级的人物[韩家炜(Jiawei Han)](https://link.zhihu.com/?target=https%3A//en.wikipedia.org/wiki/Jiawei_Han)竟然是他师祖。

* 八汰（也就是我）：听说你师祖是数据挖掘领域的鼻祖级人物？这是个什么关系？
* 霹霹：韩家炜，我老板是他学生的学生，他的风格就是特别能把事情解释清楚。不过我没有跟他一起学习过。

* 八汰：我第一次跟你讨论问题就发现你有那种把问题说的特别明白的能力，你自己有意识到这点吗？这跟你的老师或者师祖有没有什么关系？
* 霹霹：我之前只意识到我老板有这个风格。。没意识到我也被影响了。我不是跟你说过吗，我老板对 Paper 的要求是高中生也能读懂。
* 八汰：这让人特别喜欢喜欢跟你聊天。小姑娘应该也喜欢跟你聊天。
* 霹霹：。。。那倒没有。
* 八汰：不过这是另一个问题了，我们还是应该focus在机器学习的话题上。
* 霹霹：0 0

* 八汰：我上学时一直是学的计算机科学，在接触机器学习算法之前对算法的认识就是《算法导论》那种所谓的“传统算法”，关于“传统算法”和机器学习算法的本质不同，你是怎么理解的？
* 霹霹：传统的算法设计是根据给定的输入和目标，设计求解的计算过程。机器学习中涉及的算法则是用一种参数化的方式设计这一求解过程。因为机器学习的目标是generalization，我们希望得到一个在未来能多用几次的模型；而传统算法在大多数情况下都是为具体问题设计的，力求一次性解决问题。机器学习通常会基于一些假设，带有一种强烈的玄学色彩。

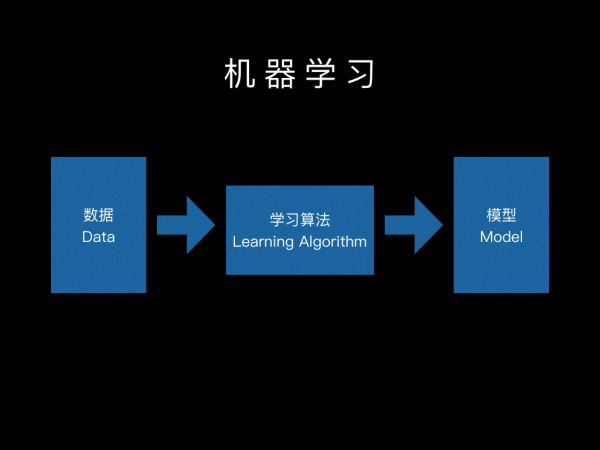
* 八汰：有没有推荐的入门或进阶的书、博客或论文。最好是能像你师祖那样特别能把事情解释清楚的。
* 霹霹：我找一下，我之前入门读的是一本data mining的书，中英文两本一起看的，感觉还可以。数据挖掘原理（[中文](https://link.zhihu.com/?target=https%3A//book.douban.com/subject/1103515/)[英文](https://link.zhihu.com/?target=https%3A//www.amazon.com/Principles-Adaptive-Computation-Machine-Learning/dp/026208290X)），这个。其实现在Machine Learning的书特别多了，[西瓜书](https://link.zhihu.com/?target=https%3A//book.douban.com/subject/26708119/)、[Deep Learning textbook](https://link.zhihu.com/?target=http%3A//www.deeplearningbook.org/)都挺好的。

* 八汰：最后一个问题，你是我认识的算法圈里长的最帅的人之一，我可以爆你的照片以骗取点击吗？
* 霹霹：-，-|||~ 不可以。
* 八汰：好的。

**正文**

好，现在正式进入正文。在《写给大家看的机器学习书》的第一篇，我们了解了机器学习的基本概念，机器学习的三个要素——数据、学习算法和模型（如图1所示）。

图1：



在这个系列的第二篇，我将首先借有好货这个真实的应用场景，让大家看看数据长什么样，了解现实中的机器学习输入数据是怎么来的。

接着，我们需要引出模型的的符号化表示和定义。数学符号的引入一方面有利于后续进入到具体的学习算法的讨论时有更高的效率，另一方面这也是每个学习者准确理解机器学习绕不过去的一环。

然后，在理解了输入数据，熟悉了模型的符号化表示后，我们将对图1所示的相对粗略的机器学习流程做进一步的细化，细化后如图3所示。**如果说学完这一篇你只能带走一点知识的话，那就带走图3吧。**一图胜千言，以后别人再问你机器学习是什么，大胆说出让机器学习专家也吃惊的精准理解吧：）

在这篇文章中，我们将接触到不少机器学习最常用到的名词、术语。不要小看名字的力量，一个东西一旦你知道了它的名字，就更容易注意到它的存在，并且掌握它。

**1. 数据（Data）**

**1.1 数据长什么样**

图2：



我们仍然以有好货产品的“瀑布流”页面（图2中间）为例，来看看机器学习的输入数据长什么样。

有好货瀑布流页是一个完全个性化的页面，不同的用户进入到有好货瀑布流页，看到的商品推荐是不同的。  
我们希望在有限的屏幕空间内给每个用户展示他最有可能点击的商品，因此机器学习的目标是要让学得的模型能够预判用户是否会点击某个商品。  
要学得这样的模型，输入数据简单来说大约长这样（表1所示）：

表1



像这样的学习算法的输入数据，叫**“训练数据”(Training Data)**。  
训练数据的每一行称为一个**“训练样本”（Training Sample）**，通常大家就简称**“样本”(Sample)**。

我们注意到，每个样本有三个属性——年龄、性别、商品价格，代表了我们认为用户是否会点击某个商品主要由年龄、性别、商品价格三个因素共同决定（当然这里我们做了简化，实际上影响的因素远不止三个）。  
这里的年龄、性别、商品价格我们称之为**“特征”（Feature）**。

在这个场景中，我们希望学得的模型可以用来预判用户是否会点击某个商品，因此光有年龄、性别、商品价格这样的特征信息还不够，还需要知道每个训练样本用户是否会点击。“是否点击”这个信息，称为样本的**标注（Lable）**。

**1.2 训练数据怎么来的**

了解了训练数据的长相，一定有人会问：训练数据是怎么来的呢？

其实也很简单，这个大数据时代，绝大多数互联网产品都会把用户的行为数据——包括浏览历史、点击历史记录下来，我们称为**日志（Log）**。

从日志数据中就能知道每个用户点过什么商品（对应标注为1的样本），看了什么商品却没有点（对应标注为-1的样本），再关联上用户的特征数据（年龄、性别）和商品的特征数据（价格），就得到学习算法所需要的训练数据了。

**1.3 机器学习问题的分类**

需要指出的是，并不是所有的机器学习问题都需要标注。事实上，根据训练数据是否有标注，机器学习问题大致划分为**监督学习（Supervised Learning）**和**无监督学习（Unsupervised Learning）**两大类。

* 监督学习：每个输入样本都有标注，这些标注就像老师的标准答案一样”监督“着学习的过程。而监督学习又大致分成两类：**分类（Classification）**和**回归（Regression）**：
  + 分类问题：标注是离散值，比如用户”点击“和”不点击“。如果标注只有两个值，则称为二分类，如果标注有多个值，则称为多分类。
  + 回归问题：标注是连续值，比如如果问题是预测北京市房屋的价格，价格作为标注就是一个连续值，属于回归问题。
* 无监督学习：训练样本没有标注，无监督学习解决的典型问题是**聚类（clustering）**问题。比如对一个网站的用户进行聚类，看看这个网站用户的大致构成，分析下每类用户群的特点是什么。

此外，机器学习还有其他的类别，比如半监督学习、增强学习，我们将慢慢涉及。

**2. 模型 (Model)**

还记得在第一篇我们说"机器学到的模型是一个映射"，这是一个很好理解又非常准确的表述。

可是在后续的学习中，尤其是进入到具体的学习算法的讨论时，文字表达的效率不如符号表示来的高，所以现在我们就要引出模型的符号化表示和定义。这不仅能大大的提升后续讨论学习算法时的效率，也是准确理解机器学习绕不过去的一环。

其实也很简单，映射包括输入和输出，在这里输入就是用户的年龄、性别、商品价格，输出就是用户是否会点击，好，我们开始吧。

**2.1 映射的输入**

* 输入样本用符号x表示，第i个样本记作x_i。
* 每个样本有三个特征，于是样本x_i又可以写成一个向量x_i=(x_{i1}, x_{i2}, x_{i3})（ x_{i1} 在这里指年龄、x_{i2}指性别、x_{i3}指商品价格），称为**“特征向量” (Feature Vector)**。
* 所有的特征向量的集合就是总的输入集合，这个例子中本质上是个三维向量张成的三维空间，称为**“样本空间” (Sample Space)**或“输入空间”，记作\cal{X}。任意的输入x都是这个3维样本空间\cal{X}中的一个向量，用符号表示就是x \in \cal{X}（读作x属于\cal{X}）。

2.2 映射的输出

* 输出的样本标注用符号y表示，第i个样本的标注记作y_i。
* 在这个例子中标注只有-1和1两种取值，我们用一个一维向量y=(y_{i1})就可以表示。同样的，这个一维向量张成的空间就是**“标注空间”(Label Space)**或“输出空间”，记作\cal{Y}。任意的输出y都属于\cal{Y}，用符号表示就是y \in \cal{Y}。

**2.3 映射的表示**

于是机器学习模型就是输入空间\cal{X}到输出空间\cal{Y}的一个映射，将映射用符号g表示，则模型记作g:\cal{X}\rightarrow\cal{Y}。

学得这个模型之后我们对于新的样本要预测用户是否会点击，只需要将样本x (x \in \cal{X})传入映射函数g，得到的输出y=g(x) (y \in \cal{Y})就是对用户是否会点击的预测。

**3. 机器学习概念图的细化**

上面我们讲了训练数据，也讲了模型，现在我们知道学习算法的目标就是要学一个映射函数g 。一旦学到g之后对于任意的输入x，都可以计算出预测结果y就等于g(x) 。

这小节我们将图1所示的相对粗略的机器学习概念做进一步的细化，细化后如图3所示。  
**如果说学完这一篇你只能带走一点知识的话，那就带走这张图吧。**一图胜千言，希望你看到图3就能回忆起整篇内容。

**3.1 数据(Data)部分的细化**

首先，我们假设有一个完美的映射f，它不仅对训练数据中的所有样本都能够正确的预测用户是否点击，对于遇到的新的样本也是一样（用符号表示就是对于任意样本(x_i,y_i)，映射f都能够使得y_i=f(x_i)）。

如果我们能得到这样一个完美的映射，问题就解决了，不过很可惜，这样一个完美的模型只有上帝才知道，它代表了学习的最理想的目标，人们称它为**“上帝真相”(Ground Truth)**，有时也叫做**目标函数(Target Function)**。

既然我们不知道，也无法知道Ground Truth f，我们怎么能做到以它为目标进行学习呢？不要忘了我们还有训练数据，既然训练数据中所有样本都满足f，当训练样本足够多时我们不妨认为海量的样本就反映了Ground Truth ![f
](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAoAAAAOCAYAAAAWo42rAAAApklEQVQoFX2RAQ2DMBREf6cADZVANgfMwbTgAS1IGBLGHOABB927sDWlH3bJpT/X67/+1lJKdkQz6+EDjtq/UDiEEFrEG3znzZNuA4a+3DuLXTC2pXEXrUiobhF21LrjhvKUatDBudZ3HbejpkFe3zovR0ZNO2fHr6gj0DVIdHopYGggkv+EOvqKcYIOtfGOY3QuCYoBunyET9j8i9Zz6P0GTCurwwedwrLoH3omdgAAAABJRU5ErkJggg==)的样子，我们称这种情况为“训练数据来自于f”。（如图3下半的左侧部分所示）

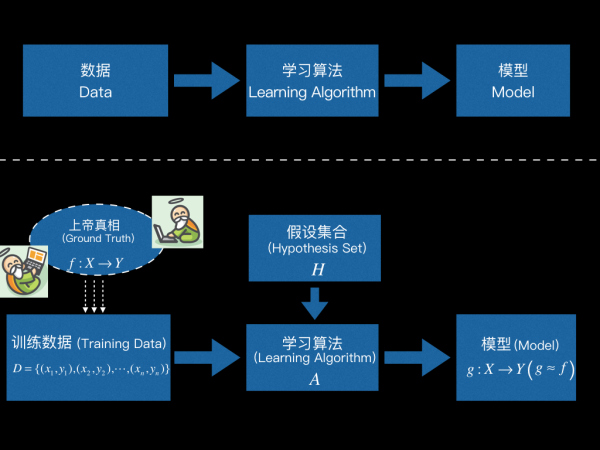
3.2 学习算法和模型部分的细化

假设了Ground Truth f的存在，那么学习算法要做的就是找出某个映射，这个映射尽可能得接近f。在实际的训练过程中，学习算法会有一个**假设集合(Hypothesis Set，记作\cal{H})**，这个集合包含所有候选的映射函数。学习算法做的事情就是从中选出最好的g，使得g越接近f越好。（如图3下半的中间部分和右侧侧部分所示）

**综上**

综上，我们把机器学习流程重新细化成下图所示：

图3：



* 输入数据\cal{D}=\{(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)\}
  ，来自于Ground Truth。
* Learning Algorithm根据训练数据，从Hypothesis Set中选出最优的那个映射g: \cal{X} \to \cal{Y}作为最终学得的模型，使得g越接近f越好(g \approx f)。

到这里，我们看着图3就可以用一句话来定义机器学习

Machine Learning：Use training data to compute model g that approximates Ground Truth f.

**题记 —— 我们为何出发**

在开始这个系列文章的第三篇之前，为了对初次见面的朋友更友好，将这个题记放在前面。

哪怕所有的初心最终都被遗忘，至少现在的我们足够认真。

——阿真

*机器学习很火。  
机器学习专家很贵。  
所有大型互联网公司都驾着机器学习的马车朝着人工智能前进。*

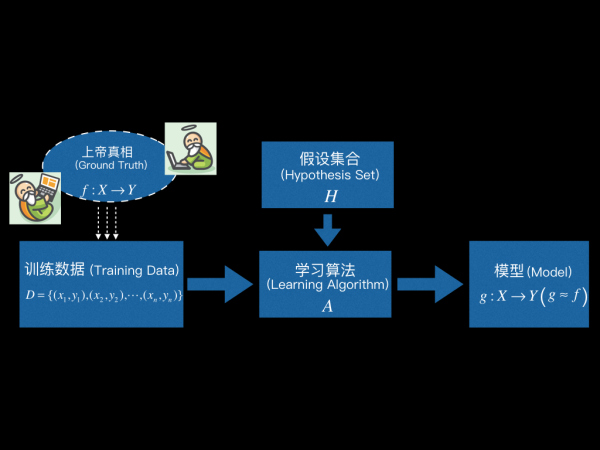
*然而今天哪怕是互联网从业者，大部分也是不知道机器学习到底是什么的。机器如何学习？机器学到的是什么？为什么机器经过学习能够神奇的预测用户的喜好、股票的涨跌？人们好奇又渴望。*

*这里所说的从业者可能是开发工程师，可能是产品经理，也可能是运营，他们与机器学习专家们在同一家公司工作，参与同一个项目，但机器学习算法对他们仍然像黑魔法一样，神秘又疑惑。这样的局面未免让人沮丧，毕竟如果相对论都可以在高等教育中得到普及，有什么领域是复杂到没办法好好说清楚的呢。*

*这个系列文章，我将试着为开发工程师，产品经理、设计师、所有希望了解学习机器学习的人，介绍机器学习的原理、方法和实战技巧。*

**希望把所有的复杂变明了，所有的阴天破乌云，希望世界和平，阿门！**

**本篇综述**



前两篇我们已经学习了机器学习的概念和组成：

学习算法 (Learning Algorithm) 根据训练数据，从假设集合 (Hypothesis Set) 中选出最优的那个映射g: \cal{X} \to \cal{Y}作为最终学得的模型，使得g越接近f越好（g \approx f）。

这一篇，我们要具体地讲一个学习算法，把它用在有好货这个场景，看看这个算法到底是怎么样从用户日志中寻找规律，学得模型的。一旦学得模型以后就可以对未来做出预测 —— 预测用户是否会点击某个商品，预测人生的巅峰、世界的和平、一起学习机器学习的女朋友...

这个学习算法的名字叫 **PLA**，全称 Perceptron Learning Algorithm。其中 Perceptron 译作**感知机**，它是人工神经网络中最基础的两层神经网络模型。学好Perceptron Learning Algorithm，你离入门人工神经网络也就不远啦。

具体来说，这一篇我们将首先介绍PLA的假设集合，看看PLA的假设集合中等着被挑选的候选函数长什么样。看过PLA假设集合的函数表示之后，重要的是理解PLA假设集合的直观解释，事实上之所以把PLA作为第一个学习算法，就是因为它有着非常直观的理解方式。

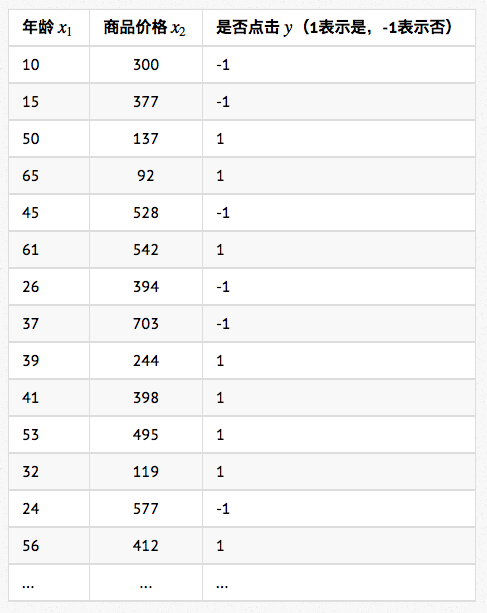
接着将看到PLA的细节，PLA 是一个相当简洁的算法，算法过程仅有4步，我写的Python 编码含注释不到30行。学习PLA的重点应该放在理解PLA算法的第三步，理解PLA之所以被设计成这样，背后的含义其实朴素又直观。

然后，我们将讨论学习算法两个很重要的问题——学习算法能最终停止并学到东西吗？如果能，学习算法需要运行多久呢？这时我们将惊讶于经典PLA的两点性质：（1）算法可能永远也无法运行结束，会迷失在茫茫的训练数据中永远找不到出口。（2）哪怕知道PLA最终能找到出口，我们也无法事先知道学习需要花多久。不过别急于换台，我们紧接着就会给出一版升级的PLA算法，只需在经典PLA的基础上增加简单两步，就可以解决上面的问题。

好，我们开始吧。

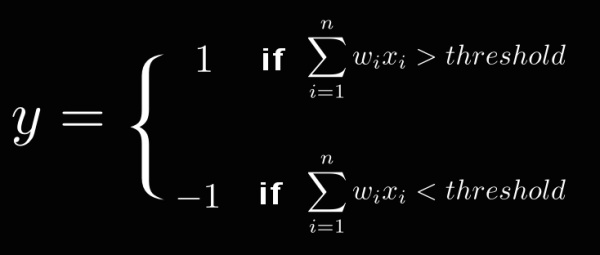
**1. PLA 的 假设集合 (Hypothesis Set)**

首先，我们从有好货的用户日志中得到训练数据如下(考虑数据安全，数据当然是我Mock的)：



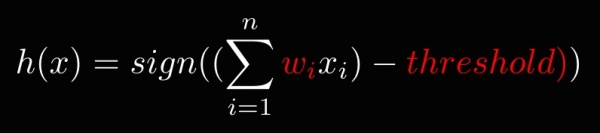
如第二篇第2小节所述，每一条样本表示为x = (x_1, x_2)，其中x_1表示年龄，x_2表示商品价格。样本标注用y表示，y=1表示用户点击，y=-1表示用户没点击。（这里我们只选了年龄和商品价格两个维度的特征，当然是为了方便讲解和作图做得简化，实际应用中会使用到多得多的特征）

PLA 的基本想法是每个特征都有个权重w_i ，表示该特征的重要程度。综合所有的特征和权重计算一个最终的分数，如果分数超过某个阈值 (threshold)，就表示用户会点击，否则则不会点击。  
我们将特征的权重记作w=(w1,w2)，w_1代表了年龄这维特征的重要性，w_2代表了商品价格这维特征的重要性。于是判断一个用户会不会点击，就变成了下面这个函数：



如果\sum_{i=1}^{n} w_i x_i大于threshold ，那么我们判断用户会点击。如果\sum_{i=1}^{n} w_i x_i小于threshold，我们判断用户不会点击。

该函数经过简单变换，可以表示得更简洁。如下所示，其中sign(x)是求表达式的符号，当x大于0时，sign(x)就等于1。当x小于0时，sign(x)就等于-1：



注意，在算法完成学习之前，函数中的w_i和threshold是未知的，不同的w_i和threshold值对应了不同的函数，事实上**所有可能的w_i和threshold所代表的函数集合，构成了PLA的假设集合(Hypothesis Set)，叫做 Perceptron Hypothesis 。**而 PLA 算法要做的，就是根据训练数据找到最优的w和threshold。

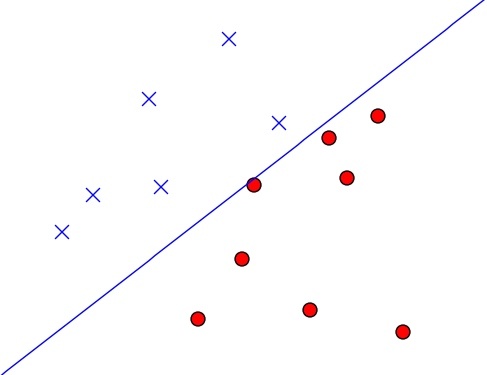
**2. Perceptron Hypothesis 的直观解释**

OK，上小节我们知道了 PLA 的假设集合 (Hypothesis Set) 就是所有可能的h(x)，不过函数表示理解起来还是有点抽象，那么 Perceptron Hypothesis 有没有直观的理解方式呢？有！

在有好货的例子里，x=(x_1,x_2)只有两维特征，于是我们先将函数简化：

https://pic1.zhimg.com/v2-1d4ab0c8b39a50b7f2a6000b661a237c_b.jpg

接着把所有训练样本x=(x_1,x_2)画到二维平面上，红圈表示点击，蓝叉表示没点击，如下图所示。：



由于方程w1 \cdot x1 + w2 \cdot x2 - threshold = 0就表示平面上的一条直线，且我们知道直线右边所有的点都有w1 \cdot x1 + w2 \cdot x2 - threshold > 0，左边所有的点都有w1 \cdot x1 + w2 \cdot x2 - threshold < 0 , 于是可以推出

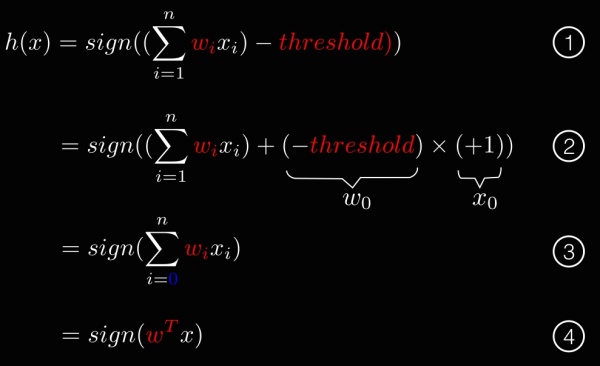
PLA假设集合中任意一个确定的h(x)，都可视作一条直线将平面分隔成了两个区域。线的左边有h(x)=-1，右边有h(x)=1。

所以，学习算法希望选中的模型g，就是如上图所示这样的一条线，它正好将训练数据分成两个区域，右边都是红圈，左边都是蓝叉。也正因为这样，Perceptron 是一种线性分类器（Linear Classifiers）。

**3. PLA 怎么选一条最好的线**

直观理解了 PLA 的 Hypothesis Set 之后，现在我们来看看 PLA 具体是如何确定最优的w和threshold，得到最优的那条线的。

我们首先需要对公式做一些简化变形：



上图第2步： 将- threshold记作w_0，并给所有的x增加一个维度x_0，所有的x_0设为1  
上图第3步：将表达式后面一项收进\sum，于是\sum_{i=1}^{n}变成\sum_{i=0}^{n}  
上图第4步： 将\sum_{i=0}^{n} w_i x_i表示成向量相乘的形式，即w^Tx

这个变形的目的是把参数threshold统一收进w，于是 PLA 找到最优的那条线就等价于找到最优的参数w。于是 PLA 算法就可以写得相当简洁，仅有如下4步：

* Step 1: 随便找一条线，即任意找一个 n 维向量![w_0
  ](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAABIAAAAKCAYAAAC5Sw6hAAAA+UlEQVQoFYWSgQ2CMBBFKXEAwgi6gYkbMIKuIBvoDI7gCoYNYATjBrKByAb4/qWXNBjlkm+v93/v+ovZNE0ZsQdHcNI+1ir215gX5G9QOD9f8xDCFsEI7qAGHgeSpzYccr50cr6uKIwIHzS8kDeJQDdSzaNF12uDVg68aU+9Mxsk4nX1dczNinIHnNmWBpjlqNWALKeoCSJ1yCaS6jayagGvxiknqx4jfGWNYmVwhnUHUnHFELe9gXslWp0r9EZ2E7o2QN4VLajjXm/RqfgnSmskARPPM+Gvw/qSupWHBvWpNSeW1hsCe9Mo1H+rC/wsHfziE8viBr3fB/29hgPuktjmAAAAAElFTkSuQmCC)，赋初值另w = w_0
* Step 2: 如果这条线正好把训练数据正确切分，Luckly 训练结束！！此时w代表的h(x)就是我们学得的模型g
* Step 3: 如果有任意一个样本没有被切分正确，即存在任意(x', y')，使得sign(w^T x') \neq y'，此时我们对w代表的线做一点点修正，另w_{t+1} = w_t + y'x'
* Step 4: 跳转到Step 2

就是这么简单，PLA的基本思路就是：先随便找一条线，如果没能正确切分，就修正一点点，直到所有的红圈都在右边，蓝叉都在左边。  
Python 编码含注释不到30行（[包含画图展示的完整代码可访问我的 Github](http://link.zhihu.com/?target=https%3A//github.com/hancyxhx/perceptron-learning-algorithm) ）：

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**def** **PLA**(trainingData):

w **=** np**.**mat([1,2127,205])**.**transpose() *# Step 1: 向量w赋初值*

**while** True:

(status, x, y) **=** noMistakePoint(trainingData, w)

**if** status **==** 'YES': *# Step 2: 切分正确，学习完成*

**return** w

**else**:

w **=** w **+** y**\***x *# Step 3: 修正w*

**def** **noMistakePoint**(training\_data, w):

'''训练数据中是否有点被切分错误'''

status **=** 'YES'

**for** (x, y) **in** training\_data:

**if** mSign(w**.**transpose() **\*** x) **<>** sign(y):

status **=** 'NO'

**return** (status, x, y)

**return** status, None, None

sign **=** **lambda** x:1 **if** x **>** 0 **else** **-**1 **if** x **<** 0 **else** **-**1

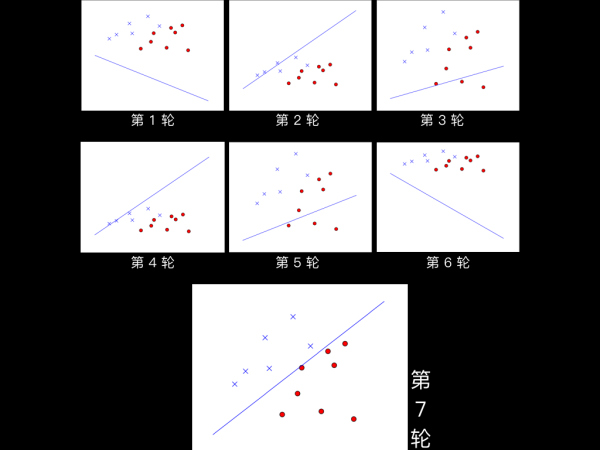
**def** **mSign**(m):

'''判断某个矩阵的[0][0]元素正负.大于0返回1，否则返回-1'''

x **=** m**.**tolist()[0][0]

**return** 1 **if** x **>** 0 **else** **-**1 **if** x **<** 0 **else** **-**1

每对训练数据进行一次循环遍历我们称为“一轮学习”，当w的初值设为w_0=(1,2127,205)时，PLA仅需 7 轮就成功找到了能够正确切分训练数据的线（如下图所示），最终PLA得到的w=(-1, 2169, -216)，即学得的模型为g(x) = sign(2169 \cdot x_1 - 216 \cdot x_2 -1)。

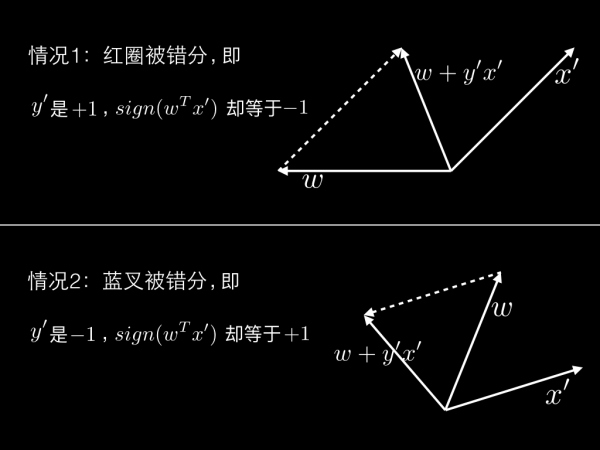


PLA 算法很简单，学习的关键在于理解算法的第3步，怎么理解当发现一个样本(x', y')没被切分正确时，对 w 的修正是w_{t+1} = w_t + y'x'呢？这个修正的含义是什么呢？  
分两种情况来分析：

情况1：被错分的点是红圈。也就是说y'是+1，sign(w^T x')却等于-1，使得sign(w^T x') \neq y'。学过向量乘的同学都知道，两个向量w^T和x'相乘小于0，说明它们的夹角大于90°。如果希望它们相乘大于0，需要把夹角变小，而另w = w+y'x'（即w = w + x'）之后正是另两个向量的夹角变小了——如下图情况1所示。

情况2同理：被错分的点是蓝叉。也就是说y'是-1，sign(w^T x')却等于+1，使得sign(w^T x') \neq y'。说明w^T和x'的夹角小于90°，如果希望它们相乘小于0，需要把夹角变大，而另w = w+y'x'（即w = w - x'）之后正是另两个向量的夹角变大了——如下图情况2所示。

这就是每次修正直线背后的含义.



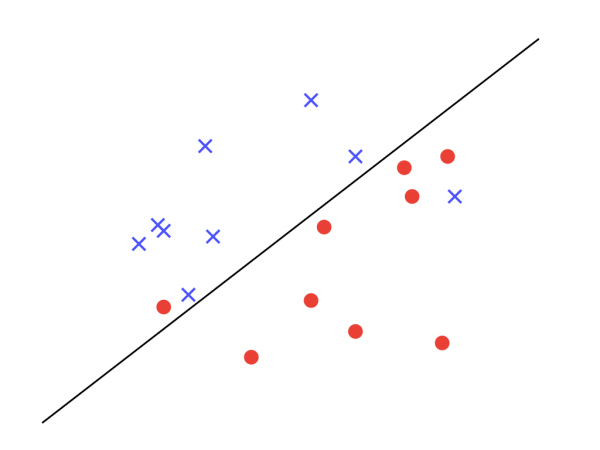
**4. PLA 一定会停吗？PLA 多久会停？**

当 PLA 停止计算时，就说明它找到了一条线能将所有的训练数据切分正确。那么问题来了，PLA 一定会停吗？它一定能找到这样的一条直线吗？

答案是：不一定！  
事实上，如果训练数据本身就不存在任何一条线能将其正确切分，比如如下图所示，那么 PLA 无论如何也无法找到理想的直线。学习将进入永不停止的循环。

训练数据至少存在一条直线能将其切分开，称为训练数据是线性可分的 (Linear Seperable) 。因此：

训练数据是线性可分的 (Liner Seperable) ，是 PLA 能够学习的必要条件。



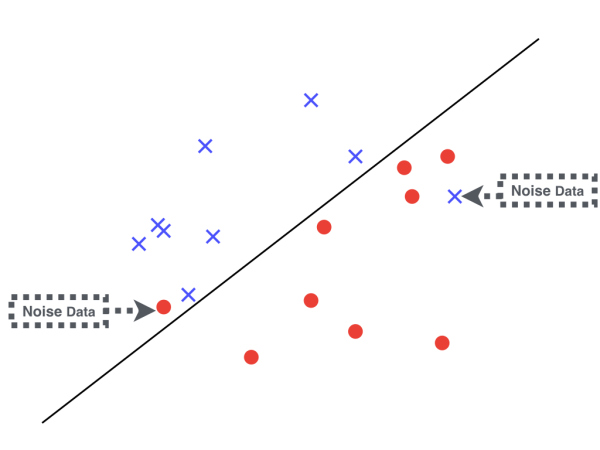
好，如果假设训练数据是线性可分的，PLA 的计算需要多久才能停呢？这里省略具体推导过程（[有兴趣可观看机器学习基石课程的推导](http://link.zhihu.com/?target=https%3A//www.youtube.com/watch%3Fv%3DOkrrz0IYoSE%26index%3D8%26list%3DPLXVfgk9fNX2I7tB6oIINGBmW50rrmFTqf)），推导可得 PLA 最多需要R^2 / \rho^2轮就会停，其中R^2 = \max_{n} ||x_n|| ^2，\rho = \min_{n} y_n \frac{w_t^T}{||w_f||} x_n。注意\rho的公式中用到了“上帝真相”f，而f是未知的，这就意味着哪怕我们知道PLA在线性可分的训练数据上会停止，但具体多久会停止这个值无法计算出来 ：(

**5. PLA 搞鸡毛啊！**

这个时候有的朋友要骂娘啦，整了半天 PLA 的训练数据必须线性可分，哪怕线性可分需要计算多久也不知道。PLA 搞鸡毛啊！  
不要急！！大招马上出来。

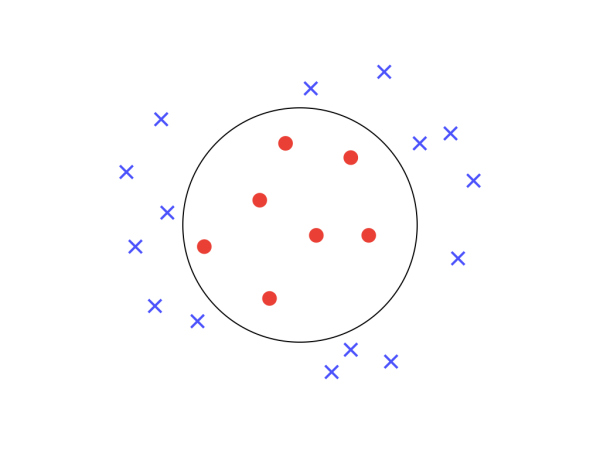
我们再看看下面这张图，这里的训练数据不是线性可分，是由其中两个点导致的。在现实中，哪怕原本的训练数据产生于某个“上帝真相”并且是线性可分的，在收集数据处理数据的过程中不可避免的会引入一些脏数据，这部分错误的训练数据我们称为噪声(Noise)。针对这种噪声数据引起的，原本线性可分的训练数据变成了不是线性可分的情况，有一个升级版的 PLA 算法，只需要增加简单的两步节能解决问题：

* Step 1: 随便找一条线，即任意找一个 n 维向量w_0，赋初值另w_0。**同时用一个变量w_{best}表示在训练数据上表现最好的线，初始有w_{best} = w_0。**
* Step 2: 如果这条线正好把训练数据正确切分，Luckly 训练结束！！此时w代表的h(x)就是我们学得的模型g。
* Step 3: 如果有任意一个样本没有被切分正确，即存在任意(x', y')，使得sign(w^T x') \neq y'，此时我们对w代表的线做一点点修正，另w_{t+1} = w_t + y'x'
  。
* Step 4:**如果修正后的w_{t+1}比w_{best}表现的更好，那么修改w\_{best} = w\_{t+1}。**
* Step 5: 如果训练轮数小于N_{train}，跳转到Step 2。否则训练结束，此时w_{best}代表的h(x)就是我们学得的模型g。



升级版的 PLA 算法来自于实际经验，当噪声较小时，经过一定次数的训练之后并把表现最好的线记下来，也能得到一个不错解。上面说w_{t+1}和w_{best}表现的好坏，是指这条线在训练数据上犯了多少错，即\sum_{i=1}^{n} [ y \neq sign(w^T x) ]。  
至此，升级版的 PLA 算法无需要求训练数据线性可分，训练最多也会在N_{train}轮内结束。：）  
试试动手写写看升级版的 PLA 算法吧。

当然，如果你的训练数据不是线性可分是下图这种情况，作为线性分类器的 PLA 是无能为力的，这需要后面要讲的非线性模型来搞定。期待一下吧。



本文主要回顾下几个常用算法的适应场景及其优缺点！（提示：部分内容摘自网络）。

机器学习算法太多了，分类、回归、聚类、推荐、图像识别领域等等，要想找到一个合适算法真的不容易，所以在实际应用中，我们一般都是采用启发式学习方式来实验。通常最开始我们都会选择大家普遍认同的算法，诸如SVM，GBDT，Adaboost，现在深度学习很火热，神经网络也是一个不错的选择。假如你在乎精度（accuracy）的话，最好的方法就是通过交叉验证（cross-validation）对各个算法一个个地进行测试，进行比较，然后调整参数确保每个算法达到最优解，最后选择最好的一个。但是如果你只是在寻找一个“足够好”的算法来解决你的问题，或者这里有些技巧可以参考，下面来分析下各个算法的优缺点，基于算法的优缺点，更易于我们去选择它。

## 偏差&方差

在统计学中，一个模型好坏，是根据偏差和方差来衡量的，所以我们先来普及一下偏差(bias)和方差(variance)：

* 偏差：描述的是预测值（估计值）的期望E’与真实值Y之间的差距。偏差越大，越偏离真实数据。

Bias[f^(x)]=E[f^(x)]−f(x) (1)(1)Bias[f^(x)]=E[f^(x)]−f(x)

* 方差：描述的是预测值P的变化范围，离散程度，是预测值的方差，也就是离其期望值E的距离。方差越大，数据的分布越分散。

Var[f^(x)]=E[(f^(x)−E[f^(x)])2](2)(2)Var[f^(x)]=E[(f^(x)−E[f^(x)])2]

模型的真实误差是两者之和，如公式[(3)](http://www.csuldw.com/2016/02/26/2016-02-26-choosing-a-machine-learning-classifier/#mjx-eqn-3)(3)：

E[(y−f^(x))2]=Bias[f^(x)]2+Var[f^(x)]+σ2(3)(3)E[(y−f^(x))2]=Bias[f^(x)]2+Var[f^(x)]+σ2

通常情况下，如果是小训练集，高偏差/低方差的分类器（例如，朴素贝叶斯NB）要比低偏差/高方差大分类的优势大（例如，KNN），因为后者会发生过拟合（overfiting）。然而，随着你训练集的增长，模型对于原数据的预测能力就越好，偏差就会降低，此时低偏差/高方差的分类器就会渐渐的表现其优势（因为它们有较低的渐近误差），而高偏差分类器这时已经不足以提供准确的模型了。

当然，你也可以认为这是生成模型（如NB）与判别模型（如KNN）的一个区别。

为什么说朴素贝叶斯是高偏差低方差?

以下内容引自知乎：

首先，假设你知道训练集和测试集的关系。简单来讲是我们要在训练集上学习一个模型，然后拿到测试集去用，效果好不好要根据测试集的错误率来衡量。但很多时候，我们只能假设测试集和训练集的是符合同一个数据分布的，但却拿不到真正的测试数据。这时候怎么在只看到训练错误率的情况下，去衡量测试错误率呢？

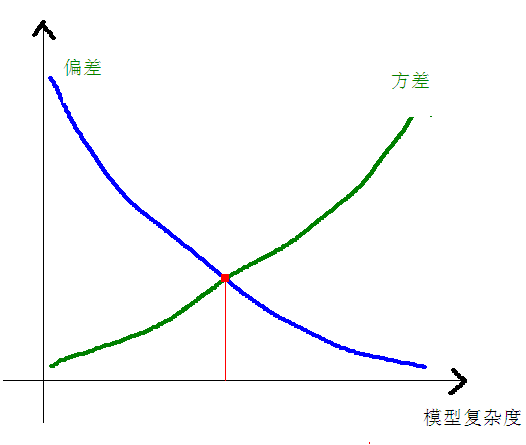
由于训练样本很少（至少不足够多），所以通过训练集得到的模型，总不是真正正确的。（就算在训练集上正确率100%，也不能说明它刻画了真实的数据分布，要知道刻画真实的数据分布才是我们的目的，而不是只刻画训练集的有限的数据点）。而且，实际中，训练样本往往还有一定的噪音误差，所以如果太追求在训练集上的完美而采用一个很复杂的模型，会使得模型把训练集里面的误差都当成了真实的数据分布特征，从而得到错误的数据分布估计。这样的话，到了真正的测试集上就错的一塌糊涂了（这种现象叫过拟合）。但是也不能用太简单的模型，否则在数据分布比较复杂的时候，模型就不足以刻画数据分布了（体现为连在训练集上的错误率都很高，这种现象较欠拟合）。过拟合表明采用的模型比真实的数据分布更复杂，而欠拟合表示采用的模型比真实的数据分布要简单。

在统计学习框架下，大家刻画模型复杂度的时候，有这么个观点，认为Error = Bias + Variance。这里的Error大概可以理解为模型的预测错误率，是有两部分组成的，一部分是由于模型太简单而带来的估计不准确的部分（Bias），另一部分是由于模型太复杂而带来的更大的变化空间和不确定性（Variance）。

所以，这样就容易分析朴素贝叶斯了。它简单的假设了各个数据之间是无关的，是一个被**严重简化了的模型**。所以，对于这样一个简单模型，大部分场合都会Bias部分大于Variance部分，也就是说高偏差而低方差。

在实际中，为了让Error尽量小，我们在选择模型的时候需要平衡Bias和Variance所占的比例，也就是平衡over-fitting和under-fitting。

偏差、方差、模型复杂度三者之间的关系使用下图表示会更容易理解：

[](http://www.csuldw.com/assets/articleImg/bias_variance.png)

当模型复杂度上升的时候，偏差会逐渐变小，而方差会逐渐变大。

## 常见算法优缺点

### 1.****朴素贝叶斯****

朴素贝叶斯属于生成式模型（关于生成模型和判别式模型，主要还是在于是否需要求联合分布），比较简单，你只需做一堆计数即可。如果注有条件独立性假设（一个比较严格的条件），朴素贝叶斯分类器的收敛速度将快于判别模型，比如逻辑回归，所以你只需要较少的训练数据即可。即使NB条件独立假设不成立，NB分类器在实践中仍然表现的很出色。它的主要缺点是它不能学习特征间的相互作用，用mRMR中R来讲，就是特征冗余。引用一个比较经典的例子，比如，虽然你喜欢Brad Pitt和Tom Cruise的电影，但是它不能学习出你不喜欢他们在一起演的电影。

**优点**：

* 朴素贝叶斯模型发源于古典数学理论，有着坚实的数学基础，以及稳定的分类效率。
* 对小规模的数据表现很好，能个处理多分类任务，适合增量式训练；
* 对缺失数据不太敏感，算法也比较简单，常用于文本分类。

**缺点**：

* 需要计算先验概率；
* 分类决策存在错误率；
* 对输入数据的表达形式很敏感。

### 2.****Logistic Regression（逻辑回归）****

逻辑回归属于判别式模型，同时伴有很多模型正则化的方法（L0， L1，L2，etc），而且你不必像在用朴素贝叶斯那样担心你的特征是否相关。与决策树、SVM相比，你还会得到一个不错的概率解释，你甚至可以轻松地利用新数据来更新模型（使用在线梯度下降算法-online gradient descent）。如果你需要一个概率架构（比如，简单地调节分类阈值，指明不确定性，或者是要获得置信区间），或者你希望以后将更多的训练数据快速整合到模型中去，那么使用它吧。

**Sigmoid函数**：表达式为公式[(4)](http://www.csuldw.com/2016/02/26/2016-02-26-choosing-a-machine-learning-classifier/#mjx-eqn-4)(4).

f(x)=11+e−x(4)(4)f(x)=11+e−x

**优点：**

* 实现简单，广泛的应用于工业问题上；
* 分类时计算量非常小，速度很快，存储资源低；
* 便利的观测样本概率分数；
* 对逻辑回归而言，多重共线性并不是问题，它可以结合L2正则化来解决该问题；

**缺点**：

* 当特征空间很大时，逻辑回归的性能不是很好；
* 容易**欠拟合**，一般准确度不太高
* 不能很好地处理大量多类特征或变量；
* 只能处理两分类问题（在此基础上衍生出来的softmax可以用于多分类），且必须**线性可分**；
* 对于非线性特征，需要进行转换；

### ****3.线性回归****

线性回归是用于回归的，它不像Logistic回归那样用于分类，其基本思想是用**梯度下降法**对最小二乘法形式的误差函数进行优化，当然也可以用normal equation直接求得参数的解，结果为：

w^=(XTX)−1XTy(5)(5)w^=(XTX)−1XTy

而在LWLR（局部加权线性回归）中，参数的计算表达式为:

w^=(XTWX)−1XTWy(6)(6)w^=(XTWX)−1XTWy

由此可见LWLR与LR不同，LWLR是一个非参数模型，因为每次进行回归计算都要遍历训练样本至少一次。

**优点**： 实现简单，计算简单；  
**缺点**： 不能拟合非线性数据.

### 4.最近邻算法——KNN

KNN即最近邻算法，其主要过程为：

1. 计算训练样本和测试样本中每个样本点的距离（常见的距离度量有欧式距离，马氏距离等）；

2. 对上面所有的距离值进行排序(升序)；

3. 选前k个最小距离的样本；

4. 根据这k个样本的标签进行投票，得到最后的分类类别；

如何选择一个最佳的K值，这取决于数据。一般情况下，在分类时较大的K值能够减小噪声的影响，但会使类别之间的界限变得模糊。一个较好的K值可通过各种启发式技术来获取，比如，交叉验证。另外噪声和非相关性特征向量的存在会使K近邻算法的准确性减小。近邻算法具有较强的一致性结果，随着数据趋于无限，算法保证错误率不会超过贝叶斯算法错误率的两倍。对于一些好的K值，K近邻保证错误率不会超过贝叶斯理论误差率。

**KNN算法的优点**

* 理论成熟，思想简单，既可以用来做分类也可以用来做回归；
* 可用于非线性分类；
* 训练时间复杂度为O(n)；
* 对数据没有假设，准确度高，对outlier不敏感；

**缺点**

* 计算量大（体现在距离计算上）；
* 样本不平衡问题（即有些类别的样本数量很多，而其它样本的数量很少）效果差；
* 需要大量内存；

### 5.决策树

决策树的一大优势就是易于解释。它可以毫无压力地处理特征间的交互关系并且是非参数化的，因此你不必担心异常值或者数据是否线性可分（举个例子，决策树能轻松处理好类别A在某个特征维度x的末端，类别B在中间，然后类别A又出现在特征维度x前端的情况）。它的缺点之一就是不支持在线学习，于是在新样本到来后，决策树需要全部重建。另一个缺点就是容易出现过拟合，但这也就是诸如随机森林RF（或提升树boosted tree）之类的集成方法的切入点。另外，随机森林经常是很多分类问题的赢家（通常比支持向量机好上那么一丁点），它训练快速并且可调，同时你无须担心要像支持向量机那样调一大堆参数，所以在以前都一直很受欢迎。

决策树中很重要的一点就是选择一个属性进行分枝，因此要注意一下信息增益的计算公式，并深入理解它。

信息熵的计算公式如下:

H=−∑i=1np(xi)log2p(xi)(7)(7)H=−∑i=1np(xi)log2p(xi)

其中的n代表有n个分类类别（比如假设是二类问题，那么n=2）。分别计算这2类样本在总样本中出现的概率p1和p2，这样就可以计算出未选中属性分枝前的信息熵。

现在选中一个属性xixi用来进行分枝，此时分枝规则是：如果xi=vxi=v的话，将样本分到树的一个分支；如果不相等则进入另一个分支。很显然，分支中的样本很有可能包括2个类别，分别计算这2个分支的熵H1和H2,计算出分枝后的总信息熵H’ =p1 H1+p2H2,则此时的信息增益ΔH = H - H’。以信息增益为原则，把所有的属性都测试一边，选择一个使增益最大的属性作为本次分枝属性。

**决策树自身的优点**

* 计算简单，易于理解，可解释性强；
* 比较适合处理有缺失属性的样本；
* 能够处理不相关的特征；
* 在相对短的时间内能够对大型数据源做出可行且效果良好的结果。

**缺点**

* 容易发生过拟合（随机森林可以很大程度上减少过拟合）；
* 忽略了数据之间的相关性；
* 对于那些各类别样本数量不一致的数据，在决策树当中,信息增益的结果偏向于那些具有更多数值的特征（只要是使用了信息增益，都有这个缺点，如RF）。

#### 5.1 Adaboosting

Adaboost是一种加和模型，每个模型都是基于上一次模型的错误率来建立的，过分关注分错的样本，而对正确分类的样本减少关注度，逐次迭代之后，可以得到一个相对较好的模型。该算法是一种典型的boosting算法，其加和理论的优势可以使用Hoeffding不等式得以解释。有兴趣的同学可以阅读下笔者后面写的这篇文章[Adaboost - 新的角度理解权值更新策略](http://www.csuldw.com/2016/08/28/2016-08-28-adaboost-algorithm-theory/).下面总结下它的优缺点。

**优点**

* Adaboost是一种有很高精度的分类器。
* 可以使用各种方法构建子分类器，Adaboost算法提供的是框架。
* 当使用简单分类器时，计算出的结果是可以理解的，并且弱分类器的构造极其简单。
* 简单，不用做特征筛选。
* 不易发生overfitting。

关于随机森林和GBDT等组合算法，参考这篇文章：[机器学习-组合算法总结](http://www.csuldw.com/2015/07/22/2015-07-22%20%20ensemble/)

**缺点：**对outlier比较敏感

### 6.SVM支持向量机

支持向量机，一个经久不衰的算法，高准确率，为避免过拟合提供了很好的理论保证，而且就算数据在原特征空间线性不可分，只要给个合适的核函数，它就能运行得很好。在动辄超高维的文本分类问题中特别受欢迎。可惜内存消耗大，难以解释，运行和调参也有些烦人，而随机森林却刚好避开了这些缺点，比较实用。

**优点**

* 可以解决高维问题，即大型特征空间；
* 能够处理非线性特征的相互作用；
* 无需依赖整个数据；
* 可以提高泛化能力；

**缺点**

* 当观测样本很多时，效率并不是很高；
* 对非线性问题没有通用解决方案，有时候很难找到一个合适的核函数；
* 对缺失数据敏感；

对于核的选择也是有技巧的（libsvm中自带了四种核函数：线性核、多项式核、RBF以及sigmoid核）：

* 第一，如果样本数量小于特征数，那么就没必要选择非线性核，简单的使用线性核就可以了；
* 第二，如果样本数量大于特征数目，这时可以使用非线性核，将样本映射到更高维度，一般可以得到更好的结果；
* 第三，如果样本数目和特征数目相等，该情况可以使用非线性核，原理和第二种一样。

对于第一种情况，也可以先对数据进行降维，然后使用非线性核，这也是一种方法。

### 7. 人工神经网络的优缺点

**人工神经网络的优点：**

* 分类的准确度高；
* 并行分布处理能力强,分布存储及学习能力强，
* 对噪声神经有较强的鲁棒性和容错能力，能充分逼近复杂的非线性关系；
* 具备联想记忆的功能。

**人工神经网络的缺点：**

* 神经网络需要大量的参数，如网络拓扑结构、权值和阈值的初始值；
* 不能观察之间的学习过程，输出结果难以解释，会影响到结果的可信度和可接受程度；
* 学习时间过长,甚至可能达不到学习的目的。

### 8、K-Means聚类

之前笔者写过一篇关于K-Means聚类的文章，参见[机器学习算法-K-means聚类](http://www.csuldw.com/2015/06/03/2015-06-03-ml-algorithm-K-means/)。关于K-Means的推导，里面可是有大学问的，蕴含着强大的EM思想。

**优点**

* 算法简单，容易实现 ；
* 对处理大数据集，该算法是相对可伸缩的和高效率的，因为它的复杂度大约是O(nkt)，其中n是所有对象的数目，k是簇的数目,t是迭代的次数。通常k<<n。这个算法通常局部收敛。
* 算法尝试找出使平方误差函数值最小的k个划分。当簇是密集的、球状或团状的，且簇与簇之间区别明显时，聚类效果较好。

**缺点**

* 对数据类型要求较高，适合数值型数据；
* 可能收敛到局部最小值，在大规模数据上收敛较慢
* K值比较难以选取；
* 对初值的簇心值敏感，对于不同的初始值，可能会导致不同的聚类结果；
* 不适合于发现非凸面形状的簇，或者大小差别很大的簇。
* 对于”噪声”和孤立点数据敏感，少量的该类数据能够对平均值产生极大影响。

## 算法选择参考

之前笔者翻译过一些国外的文章，其中有一篇文章中给出了一个简单的算法选择技巧：

1. 首当其冲应该选择的就是逻辑回归，如果它的效果不怎么样，那么可以将它的结果作为基准来参考，在基础上与其他算法进行比较；
2. 然后试试决策树（随机森林）看看是否可以大幅度提升你的模型性能。即便最后你并没有把它当做为最终模型，你也可以使用随机森林来移除噪声变量，做特征选择；
3. 如果特征的数量和观测样本特别多，那么当资源和时间充足时（这个前提很重要），使用SVM不失为一种选择。

通常情况下：【GBDT>=SVM>=RF>=Adaboost>=Other…】，现在深度学习很热门，很多领域都用到，它是以神经网络为基础的，目前笔者自己也在学习，只是理论知识不扎实，理解的不够深入，这里就不做介绍了，希望以后可以写一片抛砖引玉的文章。

算法固然重要，**但好的数据却要优于好的算法**，设计优良特征是大有裨益的。假如你有一个超大数据集，那么无论你使用哪种算法可能对分类性能都没太大影响（此时就可以根据速度和易用性来进行抉择）。

**前言：**

z