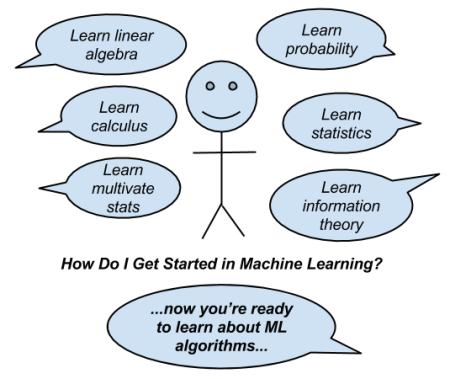
**摘要：**本文面向程序员介绍了一种有别于传统的机器学习攻略，不需要高学历、精通数学、大数据以及超级计算机，也不需要花费太多的时间，只要自上而下专注实际结果，即可简单、高效地实现从开发者到机器学习践行者的飞跃。

**【编者按】**机器学习算法的传统学习路径，是从统计学、概率论、线性代数、微积分等多种数学知识开始，但专业程序员、机器学习爱好者、MachineLearningMastery.com网站大拿[Jason Brownlee博士](http://machinelearningmastery.com/about/)认为这种自下而上的方法停留在算法层面，没有考虑到软件开发和交付，不适合专业程序员，他在[一篇文章](http://machinelearningmastery.com/machine-learning-for-programmers/)中面向程序员介绍了一种有别于传统的机器学习入门攻略，让您能够简单、高效地实现从开发者到机器学习践行者的飞跃。CSDN将其节选翻译，包括传统学习方法为什么失灵，如何**使用现代的方式和“单项最优”的工具与平台来处理实际的端到端的机器学习问题**，以及如何规避常见的错误。原文标题为：面向程序员的机器学习：从开发者到机器学习践行者的飞跃。

**传统方法的大错特错！**

从统计学、概率论、线性代数、微积分等多种数学知识开始，这种自下而上的机器学习教学方法。从表面上看起来，它是缜密的、系统的、听起来似乎是正确的观点，又怎么可能是错误的呢？

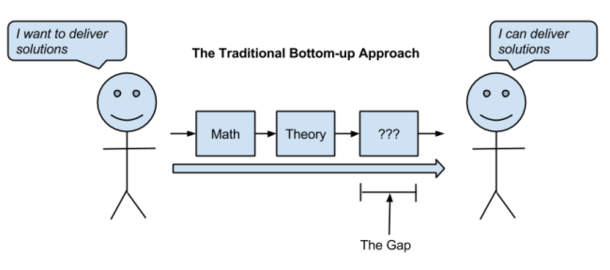


**自下而上式编程模式（或者，如何杀死萌芽中的程序员）**

想象一下，你是一个开发初学者。你已经学会了一些编程方法和语言，并且你正开始学习如何创造单机软件。

你告诉你的朋友和家人，你想从事一个每天都在编程的事业。他们告诉你，在你得到一份程序员工作之前，你需要修得计算机科学学位。

然后，你注册课程并开始修计算机科学学位。几个学期后，你接触到的是越来越多的深奥的代数，微积分和离散数学。你使用的是过时的编程语言。此时，你编程和创造软件的激情也开始动摇。



在实践者道路中，开始进行机器学习的传统方法存在着分歧

也许在某种程度上你不这么认为。往回一看，你意识到没有学到关于现代软件开发实践、语言、工具或者任何在你追求的软件创造和交付中能够使用的方法。

看到与机器学习类似的地方了么？

值得庆幸的是，编程已经有长的历史了，现在已经很流行并且对经济也相当重要，我们已经找到了其它方式给崭露头角的年轻（或资深）程序员所需要的技能，去做他们想做的事情，例如：创造软件。

这并不意味着要教新手程序员可计算性和计算复杂度的理论，或者是算法与数据结构的深层细节。**一些有用的知识（后者提到的算法复杂性与数据结构）可以晚点教。也许有些集中的资料，必须在工程师已经掌握编程知识的前提下才能理解，它们不能单独地拿出来教。**

值得庆幸的是，我们已经把重点放在了软件工程学位上。我们也有像[Codecademy](https://www.codecademy.com/)这种给你学习编写程序的资源了……没错，其实就是编程。

如果一个开发者想“做”机器学习，他真的必须花几年时间和几十甚至成百上千的美元，来获取必要的数学知识和更高的学位么？

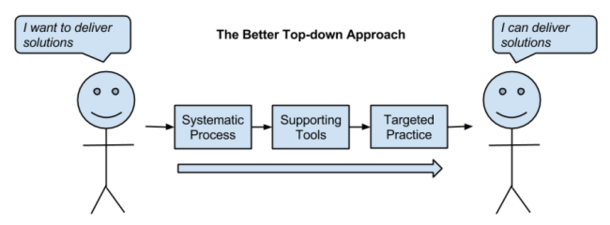
答案当然是不！还有一个更好的方法。

**一个更好的方法**

正如计算机科学，你不能仅仅只是将模式反转为自上而下，并使用相同的教学材料。

原因是，计算机科学课程从来就没有把开发和交付软件的实际问题涵盖到课程内容当中，而且机器学习的课程和书籍也远远不足。它们一般停在了算法方面。

**你需要一个自上而下的方法来学习机器学习。一种你想要专注于实际结果的方法：使用现代的方式和“单项最优”的工具与平台来处理实际的端到端的机器学习问题。**



学习机器学习的一种更好方法，开始学习处理端到端的机器学习问题

上面是我认为的黄砖路。

**1. 一个可重复的有系统流程的结果**

一旦你知道一些工具，那么用机器学习算法解决这个问题就显得比较容易，然后你可能就认为这个问题“完成”了。

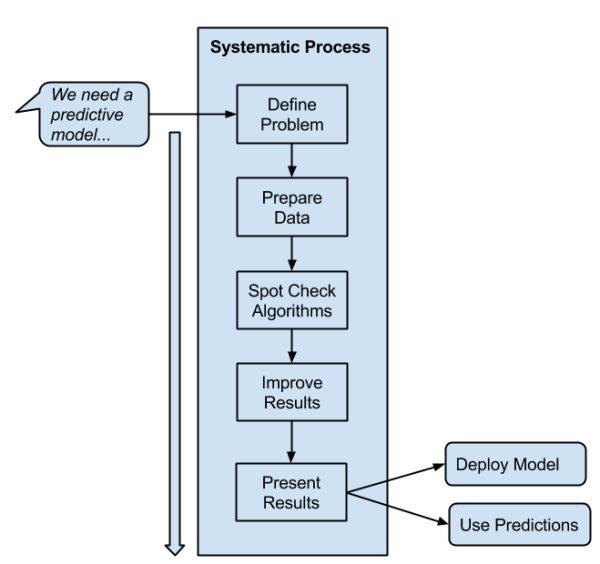
这是很危险的。

你怎么知道你已经完成了？你怎么知道结果就是好的？你又如何知道在数据集上的结果是可靠的呢？

在解决机器学习问题上，你需要系统化学习。这是一个项目，就像一个软件项目，良好的处理过程可以得到一个高质量的，能够在项目之间复用的结果。

考虑这样一个流程，你可以思考一些明确的需求，例如：

* 一个指导你从端到端，从问题规范到演示文稿的演示或到结果部署的流程。就像一个软件项目，你认为你完成了，但事实上你可能没有。要有一个最终交付的心，从一开始就设定一个明确的项目终止条件并集中精力去努力完成它。
* 一个循序渐进的流程，你总是知道下一步该做什么。不知道下一步该做什么是一个项目的杀手。
* 一个保证“好”结果的流程，比如说，优于平均水平或者是比项目要求的结果要好。在项目中，使用已知置信度来可靠地传递良好的结果，但不一定要求是最准确的结果，这是非常常见的。
* 一个不必根据特定工具、编程语言、算法潮流而改变的流程。工具可以不断变化，但过程必须是自适应的。考虑到该领域对算法的痴迷，总会有新的、强大的算法从学术界产生。



选择一个系统的、可重复的过程，你可以始终如一地用它来提供结果

很多伟大的流程，包括一些旧的流程，能够为你的需要提供帮助。

例如：

* [Knowledge Discovery in Databases](http://machinelearningmastery.com/what-is-data-mining-and-kdd/)(KDD)，我已经运用到了[这里](http://machinelearningmastery.com/process-for-working-through-machine-learning-problems/)。
* [CRISP-DM](https://en.wikipedia.org/wiki/Cross_Industry_Standard_Process_for_Data_Mining)
* [OSEMN](http://machinelearningmastery.com/how-to-work-through-a-problem-like-a-data-scientist/)
* 其他…

选择或使用一个对你最好的流程，并且符合上面的要求。

**2. 把“单项最优”工具应用到你的处理过程中**

机器学习工具和库在不断变化，但在任何一个单一的时间点，你必须使用一些东西，让它很好地应用到你提供的结果选择过程中。

你不想评估和选择任何一个过时的算法或者库，你想要所谓的“单项最优”，能给你快速、可靠以及高质量结果的工具或库，并能自动化处理那些和你提供的处理过程一样多的流程。

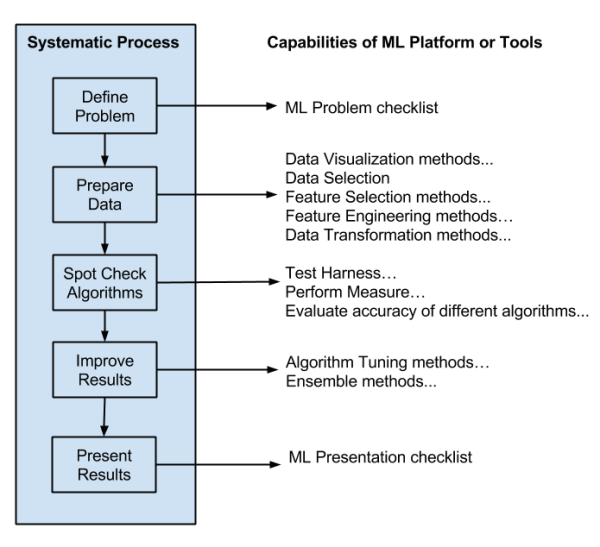
同样，你不得不自己做这些选择。如果你问其他人，你会听到他们的偏见，而他们通常使用的是最新工具。

我也有我自己的偏爱，我喜欢为不同类型的工作使用不同的工具和平台。

例如，在上面列出的情况下，我会建议使用以下最佳的训练工具：

* One-off预测模型：Weka平台，因为我可以加载一个CSV文件，所以在任何时候我都可以不需要任何代码（参见我的映射过程）来设计一个实验并且得到最佳的模型。
* Embedded预测模型：Python中的scikit-learn库，因为我可以在开发模型和部署时使用相同的语言。 IPython以一种很好的方式展示你的管道，并受到广大开发团队的喜爱。MLaaS也是面向更大数据的一个选择。
* Deep-dive模型：R平台扩展包，因为我可以使用整个R平台快速地、自动地尝试很多先进的模型，并设计出更精准的特征选择，特征工程和算法优化实验。

在实际当中，这三个工具在三个场景中的使用取决于具体的实际情况。



把你首选的机器学习工具映射到你的系统流程中使之能够解决实际问题

和开发一样，你需要学习你的工具以充分利用它们。你还需要静观其变，如果那些新的更好的工具可以使用，你应该转到这些工具上，让它们永远的适应到你的重复过程中。

**3. 使用半正式化工作产品进行有针对性的实践**

你可以通过实践取得好的发展——通过开发大量的软件。使用这种熟悉的方法来获得良好的机器学习。你在每个项目中练习的越多，对机器学习（理想的端到端工作问题）的掌握就越好。

**谨慎选择你的实践数据集**

你想挑选的数据集应该是真实的而不是人为的。现在有数以百计的免费数据可以获取，并且它们正在不断地增加复杂性。

* 我建议从[UCI机器学习库](http://archive.ics.uci.edu/ml/)的小内存数据集开始。它们是众所周知的、比较干净的，并且可以作为你新的流程和工具的一个良好开端。
* 在此基础上，我会推荐大内存数据集，比如那些来自Kaggle和KDD杯比赛的数据集。它们本身有些噪声数据，需要你更多灵活的处理和不同的技巧。

坚持使用表格数据，这是给我所有学生的建议。

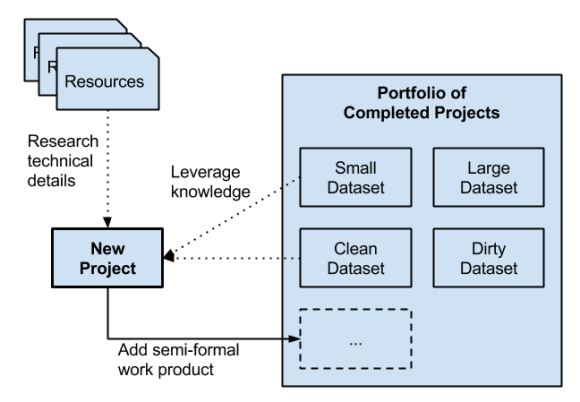
图像和文本数据的处理是新而不同的领域，就它们本身（计算机视觉和自然语言处理）而言，需要你去学习这些领域专业的方法和工具。如果它们是你想要的或是想解决的问题类型，那么最好是从这里开始，而且这里有很多的资源可以获得。

**写出你的结果并建立一个公共的工作包**

创建并保留每个项目的半正式工作产品。我的意思是详细写出你所做的和你所学的一些单独的文件，这样你可以在未来和接下来的项目中再次使用。

这类似于为每个编程项目保留一个目录，并从先前的项目中复用代码和思想。它会大大缩短编程时间，所以我强烈地推荐这个。

保留所有的脚本、代码以及生成的图像，但重要的是要写下你的发现。把它想象成类似代码中的注释。一个单独记录的文件可以是一个简单的PPT或是文本文件，或是与一个会议的演讲或YouTube视频一样详尽地阐述。



工作思路和完整的分离项目，撰写结果并建立项目组合

在一个公共的版本控制库（如GitHub）保存每个项目，这样其他的初学者就可以借鉴你的项目并扩展你的工作。同时把项目链接到你的博客上，如LinkedIn或其它任何地方来证明你增升的技术和能力。

关于这一思想的更多文章，请查看我的相关博文“[Build a Machine Learning Portfolio: Complete Small Focused Projects and Demonstrate Your Skills](http://machinelearningmastery.com/build-a-machine-learning-portfolio/)”。

公共的GitHub库文件很快成为公司招聘过程中，在简历上真正关心技能和成果的方面。

**是的，这种方法是专为开发者提供的**

我们上面制定的方法，是作为开发人员的你可以开始学习机器学习并取得进展而使用的方法。

当然，你可能会对这个方法是否真的适合你而存有一些疑惑。下面，让我来解决你的这些疑虑。

**你不需要去写代码**

你可能是一个web开发者或者类似的开发者，你不需要写大量的代码。你能够使用这个方法开始并运用机器学习。比如[Weka](http://machinelearningmastery.com/how-to-run-your-first-classifier-in-weka/)这样的工具，可以在不需要任何的编程能力条件下，使得设计机器学习实验和建立模型变得非常简单。

编写代码可以让你掌握更多不同的工具和技能，但是这并不是必须的，并且也不是首要的

**你不需要精通数学**

跟开发一样，你不需要懂得关于计算性或大O符号再去编写代码，并上线有用而且可靠的软件，你可以在没有统计学，概率论和线性代数的背景下解决端到端的机器学习问题。

值得注意的是，**我们没有从理论开始，但我们不忽略它。在你需要它的时候，再将方法和算法从中分离出来**。事实上，你会忍受不住的。原因是，解决机器学习问题是让人上瘾的。在追求更好的结果和更准确的预测上，你会使用你能够找到的任何资源。学习对你而言，只是提取智慧中的结晶再将其运用到你的问题上而已。

如果你的目的是掌握理论知识，那么这个方法是慢而低效率的。这就是为什么当通过镜片看世界时会感觉如此不舒服。从机器学习开发者的目标来看，它的意义颇多。

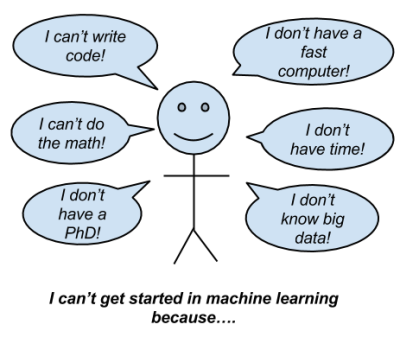
**你不需要一个高文凭**

没有一个门卫会有这方面的知识。这些所有的知识都是可以获得的，并且在现在你完全可以自学。在你开始处理机器学习问题之前，你不需要花掉大量的时间和金钱去拿一个学位。

如果你志在获得更高的学位，为什么不先开始解决机器学习问题，并在几周或几个月后，等你建立了一个完整的小型项目，再回头考虑获得学位的事情。到那时，你不仅会在这个领域的广度上有一个清晰的想法，还会有你非常感兴趣的那部分。

我后来的确回去了，并且拿到了更高的学位。我喜欢做研究，但我更喜欢处理实际问题并交付客户真正关心的更多的结果。我也是在修学位之前就开始处理机器学习问题的，我没有意识到的是，在我面前已经有了这些资源和这条路。

这也是我为何如此激情地说服像你这样的开发者的一个原因，你现在有你所需要的资源了，那么开始行动吧。



这是在机器学习中很容易出现的不行动的借口

**你不需要大数据**

机器学习算法开发在小数据集上理解最佳。数据足够小，你就可以在微软的Excel中查看，可以加载到内存中并且在桌面的工作空间中处理。

大数据!=机器学习。你可以使用大数据建立预测模型，但把它当做是你技能领域的一个专长。在机器学习中，我通常建议我的学生使用小内存数据集进行学习。

如果大数据机器学习是你想要工作的领域，那么从这里开始吧！

**你不需要一台超级计算机**

的确，一些最先进的算法，如深度学习确实需要非常强大的多核GPU。他们是强大的、令人兴奋的算法。但它们也可以在小问题上工作，并且你同样可以用你的台式CPU来计算。

你没有必要在拥有一台大的快速的计算机之前，不开始机器学习。

在你去买一个桌面超级计算机或租金非常贵的[EC2实体](https://aws.amazon.com/)之前，你最好把这些时间花在如何将这些算法用在更小更容易理解的数据集上，并从中获取最大利益。

**你不需要大量的时间**

我们都有忙碌的生活，但如果你真的想要学到点什么，你需要花点时间。

我之前说过，处理机器学习的问题是会让人上瘾的。如果你在机器学习竞赛中被超越了，你会很乐意挤出一个月晚上看电视的时间来将你的算法提高几个百分点。

其实，如果从一开始你就有一个清晰的过程和一个好的工具，那么你可以在一两个小时内处理一个数据集上的端到端问题，也许是花掉一个或两个晚上。事实上，只要其中的几个工具，还有一个完整机器学习项目的文件夹工作空间，你就可以开始解决更大的更有趣的问题了。

接着在你的看板中把它分解成多个小任务，然后指定时间开始。

**开发者如何避免最常见的错误**

自从我发起Machine Learning Mastery以来，我一直在给人以不同的忠告将近两年了。在这段时间里，我看到了五个常见的陷阱，希望你可以避免。

1. **不采取行动**：一切都安排好了，然而我却看到很多开发者都没有采取行动。看电视或阅读新闻比在一个迷人的领域里获得一个新而有价值的技能更容易。真是“牵马到河易，强马饮水难”啊。
2. **挑选的问题太大**：通常，我发现开发者挑选的第一个或第二个数据集太难工作了。问题太大、太复杂或太嘈杂，他们没有准备好去挑战。可怕的是“失败”抹杀了开发者的动机并且放弃这个过程。所以，要挑选那些你能在1小时内就解决的小问题。在你做更大的事情之前，先去做一下这个吧。
3. **从零开始实现算法**：我们有算法实现。它们已经封装好了。至少能满足你在未来几年里想要做的有趣的事情。如果你的目标是学习如何开发和提供可靠的、准确的预测模型，不要花时间去从零开始实现算法，使用库就行。另一方面，如果你想把重点放在实现算法上，那么就要很清楚地把你的目标和重点放在它上面。
4. **没有坚持一个过程**：与敏捷软件开发一样，如果你偏离了这个过程，车轮可以很快地关闭你的项目，结果往往会造成大的混乱。坚持一个过程，从开始到终止，系统地通过并完成端到端问题才是关键。你可以重新审视“你发现的有趣的事情……”作为一个后续的小项目（在你写下的一个“后续工作想法”部分），但也要完成这个过程并且交付它。
5. **不使用资源**：在机器学习领域有很多杰出的论文，书籍和博客文章。你可以利用这些资源来改进你处理的过程，使用的工具和结果的精度。可以使用第三方资源从你的算法和数据集上获取更多的信息。得出算法和问题框架的一个思路。一个智慧的结晶可以改变你项目的流程。记住，如果你采用了自上而下的流程，理论就必须出现在后面。最后再花时间去理解你的终极模型。

温馨提示，不要让任何一个错误发生在你的身上。

**下一步**

我们已经涵盖了很多方法，我希望可以开始让你相信，你可以开始机器学习并且会有所成就。未来，你成为机器学习开发者的梦想会成为现实，并且也很容易实现。

**更新：**

看看下面这个思维导图，你会对文中的重要概念有一个清晰的脉络（感谢Simeon提的建议）。



点击[这里](http://3qeqpr26caki16dnhd19sv6by6v.wpengine.netdna-cdn.com/wp-content/uploads/2015/08/Machine-Learning-For-Programmers-Mind-Map.png)查看原图

**原文链接：**[Machine Learning for Programmers: Leap from developer to machine learning practitioner](http://machinelearningmastery.com/machine-learning-for-programmers/)（译者/刘帝伟 审校/刘翔宇、朱正贵 责编/周建丁）

**关于译者：** [刘帝伟](http://blog.csdn.net/dream_angel_z)，中南大学软件学院在读研究生，关注机器学习、数据挖掘及生物信息领域。