

Issue special-edition | 专题

机器学习原来这么有趣！（二）



亚当·盖特吉 12 个月前

在第一章中我们谈到，机器学习是用泛型算法告诉你一些有关数据的有趣结论，而在这个过程中你不需要写任何与问题有关的特定代码。（如果你还没有读过第一章，[现在先去读吧！](#)）

这一次，我们要来用其中一种泛型算法来做件很酷炫的事情——创造一个看起来像是人类设计的电子游戏关卡！我们会构造一个神经网络，并提供给它已存在的超级马里奥的关卡设计，然后就可以等它自己创造出新的关卡了！

[订阅](#) [往期](#) [登录](#)



和第一章类似，这篇指南是为那些对机器学习感兴趣，但又不知从哪里开始的人而写的。这意味着文中有大量的概括。但是那又如何呢？只要能让读者对机器学习更感兴趣，这篇文章的任务也就完成了。

做出更智能更准确的预测

回到第一章，我们根据房屋各种特征属性创造了一个来估计房价的简单算法。我们给出了一所房子的如下数据：

卧室数量	面积（平方英尺）	地段	成交价
3	2000	Hipsterton	???

我们得到了这个简单的预估函数：

```
def estimate_house_sales_price(num_of_bedrooms, sqft, neighborhood):  
    price = 0  
  
    # 一小撮这个
```

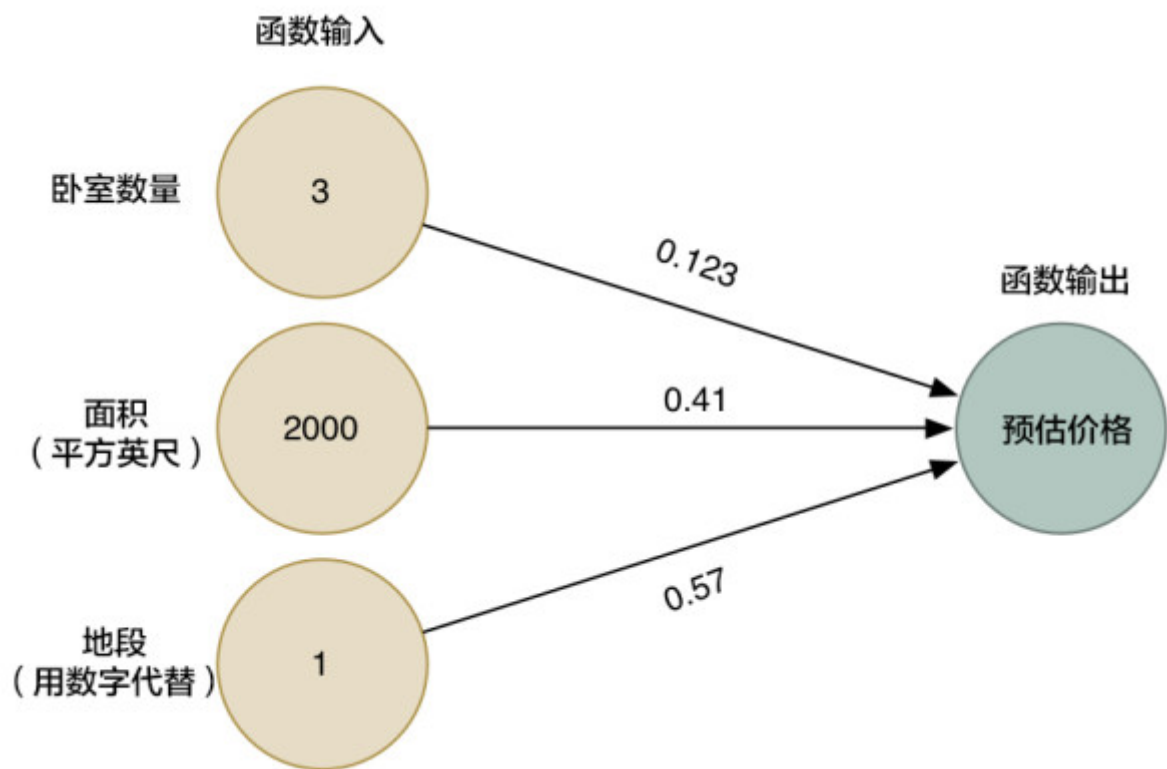
```
# 一大撮这个
price += sqft * **0.41**

# 也许再来一把这个
price += neighborhood * **0.57**

return price
```

换句话说，我们把决定房屋价格的因素乘以它的**权重**，再把这些乘积求和，就可以得到房子的预估价格了。

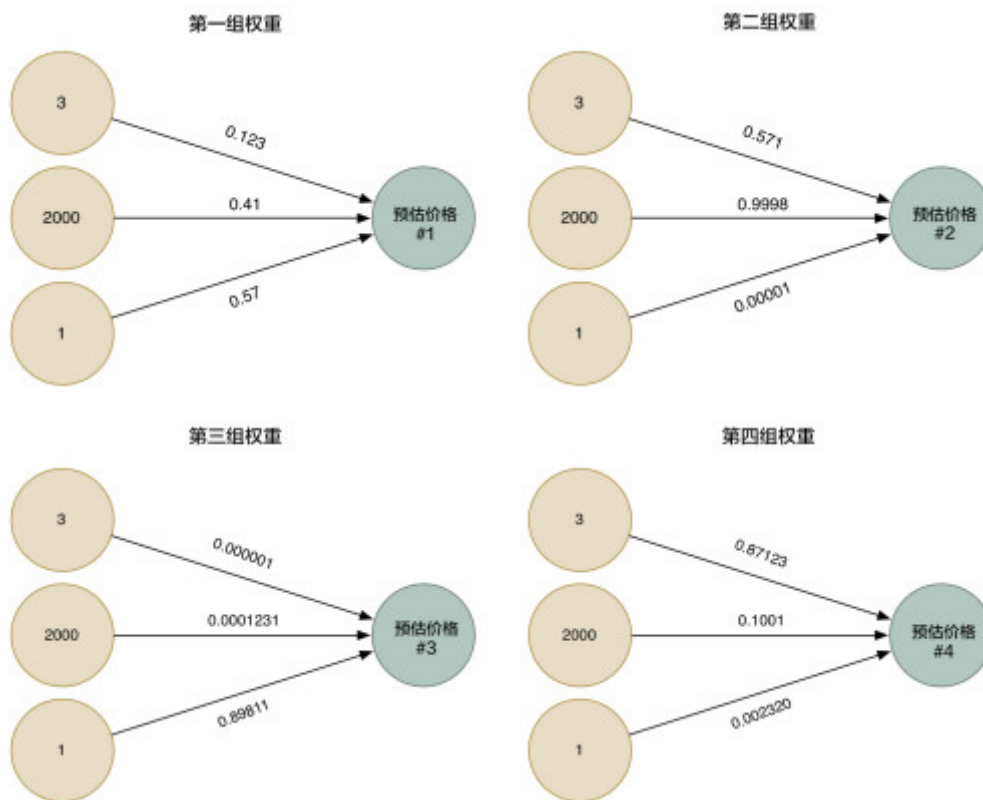
或者不使用代码，我们直接用一个图片来概括这个函数：



箭头表示了函数中的权重。

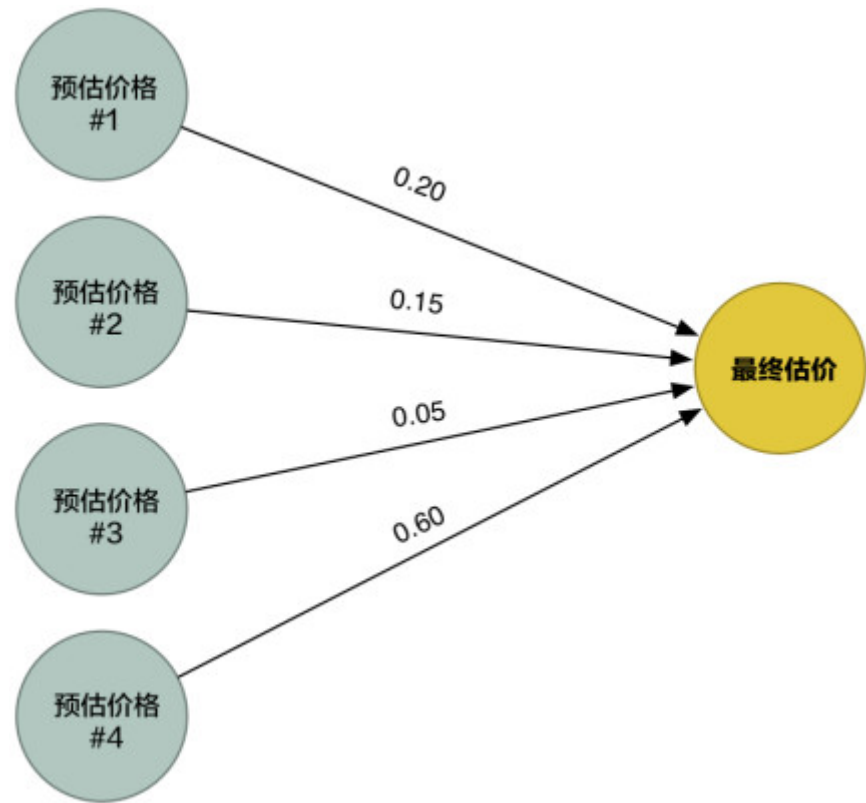
然而，这个算法仅仅能用于处理一些简单的问题，就是那些输入和输出有着**线性关系**的问题。但如果真实价格和决定因素的关系并不是如此简单，那我们该怎么办？比如说，地段对于大户型和小户型的房屋有很大影响，然而对中等户型的房屋并没有太大影响。那我们该

所以为了更加的智能化，我们可以利用不同的权重来多次运行这个算法，收集各种不同情况下的估价。



让我们试着用四种不同的算法来解决该问题。

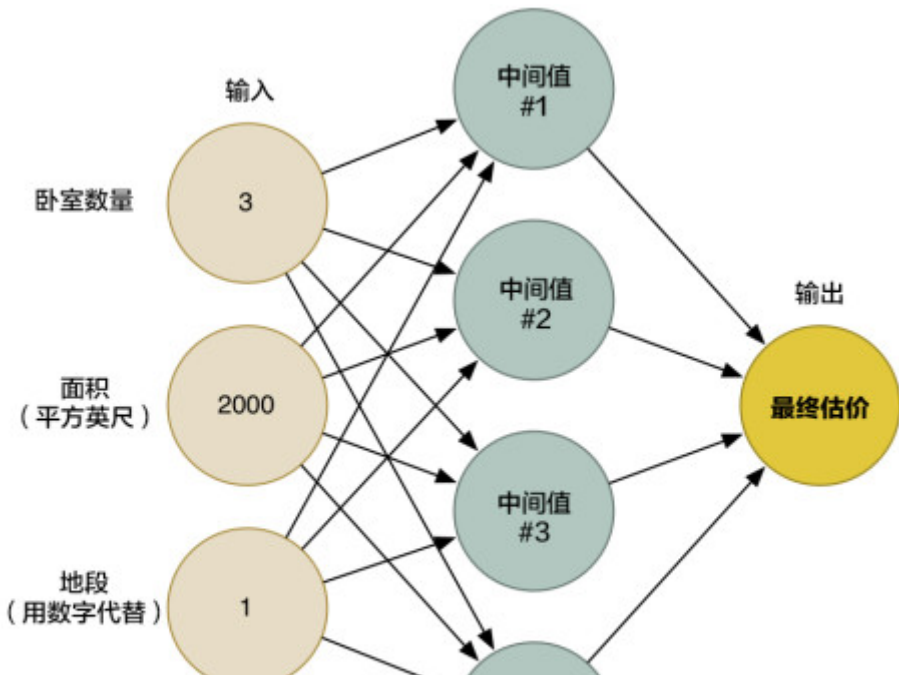
现在，我们有了四种不同的估价方法。我们将这四种估价方法汇总到一个最终估计当中。我们再把们放到同样的算法当中再算一遍（当然这次我们使用的权重不同）！



我们现在就结合了解决同一问题的方法四种不同方法，得到的一个「**超级答案**」。正是由于此，我们才能够用它来模拟更多不同的情况。

神经网络是什么？

我们来把四种不同的预测方法概括到一个图当中：



这就是一张神经网络！每一个节点都知道如何收集一组收据，找到他们的权重，做出对应的输出值（即价格预测）。把这些节点连接到一起，我们就可以模拟更复杂的函数了！

当然了，为了保持简洁性，我跳过了许多内容（例如**特征缩放**和**激活函数**）。但是最重要的是下面的这些内容：

- 我们制造了一个权重×因素的简单函数，我们把这个函数叫做**神经元**。
- 通过连接许许多多的简单**神经元**，我们能模拟那些不能被一个神经元所模拟的函数。

这就好像乐高积木一样！我们不能用一个乐高积木搭成摩天大楼，但是如果有足够多的乐高积木的话，我们能够搭建成任何东西。



也许未来的动物都是由积木搭成的？那只能去未来一探究竟了.....

让我们的神经网络拥有记忆的能力

如果输入相同的数据，我们刚刚看到的那个神经网络总是有着一样的输出。这是因为他没有记忆能力。用编程的语言来说，他是一个**无状态算法**（**stateless algorithms**）。

在许多情况下（例如说估计房价），无状态算法正是你所需要的算法。但是随着时间的增加，这种算法无法在数据中找出规律。❶ 假设我现在让你在电脑上写一个故事。在你开始之前，我需要猜测你最先敲击键盘上的哪个字母。我应该猜哪个呢？

我可以使用的语言（英语）知识来增加我猜对的概率。比如说，你可能会先打单词常见的第一个字母。如果我看一下你过去写过的故事，我能够根据你过去的用词选择来缩小我猜测

我们的模型就像这样：



让我们把这个问题变得更难一点。现在我们假设，在整个文章中间的某一点，我们要猜测你即将输入的下一个字母是什么。这是一个更有趣的问题。

让我们把海明威的著作《[太阳照常升起](#)》的前几个单词当成一个例子：

Robert Cohn was once middleweight boxi

所以，下一个字母是什么？

你可能会猜是「n」，这个词有可能是「boxing」。我们是通过观察还有语言常识来猜测这个词的。同时，「middleweight」这个词也给了我们猜测的额外线索。

换一个角度来说，如果我们知道了在这之前出现的字母，并结合我们的语言常识，要猜对下一个字母就会变得很简单。

为了用神经网络的方法来解决这个问题，我们需要把**状态（state）**加入到我们的模型当中。每次通过神经网络来解决问题的时候，我们将中间的计算结果也保存下来，并作为下次的输入的一部分再次使用。这样一来，我们的模型就能根据以往的输入数据来调整它的猜测。

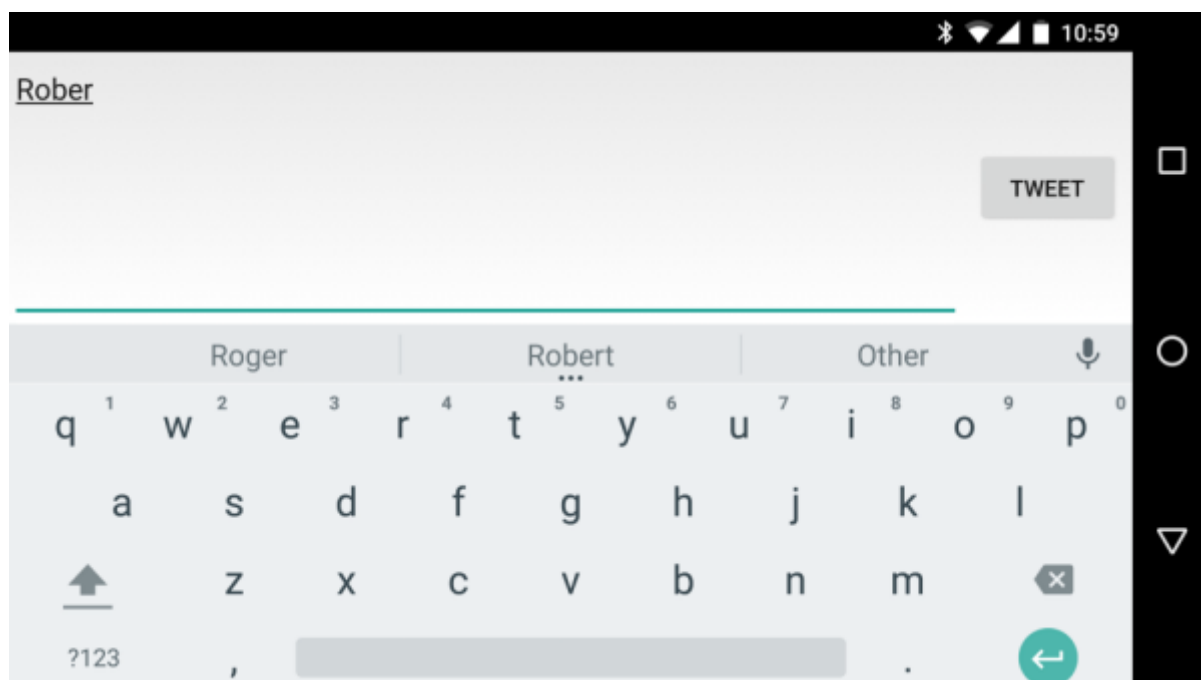
跟踪模型中的每一个状态（state），不仅能够让我们更好预测第一个字母，更能让我们更好的预测到任意位置的下一个字母。

这就是**循环神经网络（Recurrent Neural Network，简称 RNN）**的基本概念。我们每次使用神经网络的时候都会对他进行升级。这使它能跟根据最近浏览的信息更新它的预测。如果数据记忆足够多的话，他甚至能够模拟出长期的规律。

[订阅](#) [往期](#) [登录](#)

猜下一个字母看上去并没有什么实用价值。这么做的意义是什么呢？

根据你的输入，自动补全你想输入的单词，就是一个比较酷炫的应用。



下一个最有可能的字母是「t」。

但是如果我们最大程度地拓展这个想法会怎样？如果我们让我们的模型一直去预测下一个字母，永远不停止，那会怎样？机器会不会自己写出一个完整的故事？

生成一个故事

刚刚我们看到了如何猜测海明威《太阳照常升起》中的下一个字母的。现在让我们来试一试写一个海明威式（写作风格）的故事吧！

让我们用安德烈·卡帕西（Andrej Karpathy）写的[循环神经网络实现](#)来完成我们的目标。安德烈是斯坦福的一位深度学习研究者。有关用 RNNs 生成文字，他写过[一篇非常棒的导论](#)。你可以在 [GitHub](#) 上[查看这个模型的代码](#)。

让我们用《太阳照常升起》中的文字构建我们的模型。全书共有 362,239 个词，使用了 84 个不同的字符（包括标点符号和大小写字母等）。和真实世界的问题相比，这组数据其实已经非常小了。如果想要更好地模仿海明威的文风，我们需要的样本文本长度可能是这本小说长度的几倍。但是这本《太阳照常升起》已经足够展示制作我们模型的过程了。

[订阅](#) [往期](#) [登录](#)

当我们刚开始训练 RNN 的时候，它猜测的并不准确。这是它经过 100 个循环训练之后的结果：

hjCTCnhoofeoxelif edElobe negnk e iohehasenoldndAmdal ayio pe e h'e
btentmuhgehi bcgdltt. gey heho grpiahe.

Ddelnss.eelaishaner" cot AAfhB ht ltny

ehbih a"on bhnte ectrsnae abeahngy

amo k ns aeo?cdse nh a taei.rairrhelardr er deffijha 2

你可以看到的是机器已经知道单词之间应该有空格，这正是我们想要的。

1000 个循环之后，结果看起来更靠谱一点：

hing soor ither. And the caraos, and the crowebel for figttier and ale the room of
me? Streat was not to him Bill-stook of the momansbed mig out ust on the bull, out
here. I been soms

inick stalling that aid.

"Hon't me and acraind on .Hw's don't you for the roed," In's pair."

"Alough marith him." 3

我们的模型已经开始识别出句子的基本构成模式了。它已经学会给句子加上句号，甚至学会了用引号。还成功的拼出了一些单词，但大部分依然在胡扯。

经过几千次交互训练之后，他看起来很棒：

He went over to the gate of the café. It was like a country bed.
"Do you know it's been me."
"Damned us," Bill said.
"I was dangerous," I said. "You were she did it and think I would a fine cape you," I
said.
"I can't look strange in the cab."
"You know I was this is though," Brett said.

“You make it?”

“Sit down,” I said. “I wish I wasn’t do a little with the man.”

“You found it.”

“I don’t know.”

“You see, I’m sorry of chatches,” Bill said. “You think it’s a friend off back and make you really drunk.”

他跑到了咖啡厅的门口。它像是一个乡下的床一样。

「你知道那是我。」

「诅咒我们，」比尔说。

「我曾经很危险，」我说。「你是她做了它并且认为我会一个好的海角你，」我说。

「我不能在出租车里看起来奇怪」

「你知道我是这个虽然，」布拉特说。

「它是一个打架不重要？」

「它让去做它」

「你做的？」

「坐下，」我说。「我希望我没有和那个男人做一点。」

「你找到了它。」

「我不知道。」

「你看，我对 chatches（英文中没有 chatches 这个词）感到抱歉，」比尔说道。

「你认为它是一个朋友后面并且它真的让你醉」

到此时，算法已经收集到了海明威写作的基本风格——简短而直接的对话形式。甚至有一些话语开始能被人类理解。

和原文中的句子作比较：

There were a few people inside at the bar, and outside, alone, sat Harvey Stone.

... ..

[订阅](#) [往期](#) [登录](#)

“Nothing. Just looking for you.”

“Been out to the races?”

“No. Not since Sunday.”

“What do you hear from the States?”

“Nothing. Absolutely nothing.”

“What’s the matter?”

有几个人在里面酒吧间内，哈维·斯通独自在外面坐着。他面前放着一大堆小碟子，他需要刮刮脸了。

「坐下吧，」哈维说，「我正在找你。」

「什么事？」

「没事儿。只不过找你来着。」

「去看赛马啦？」

「没有。星期天以来再没去过。」

「美国有信来吗？」

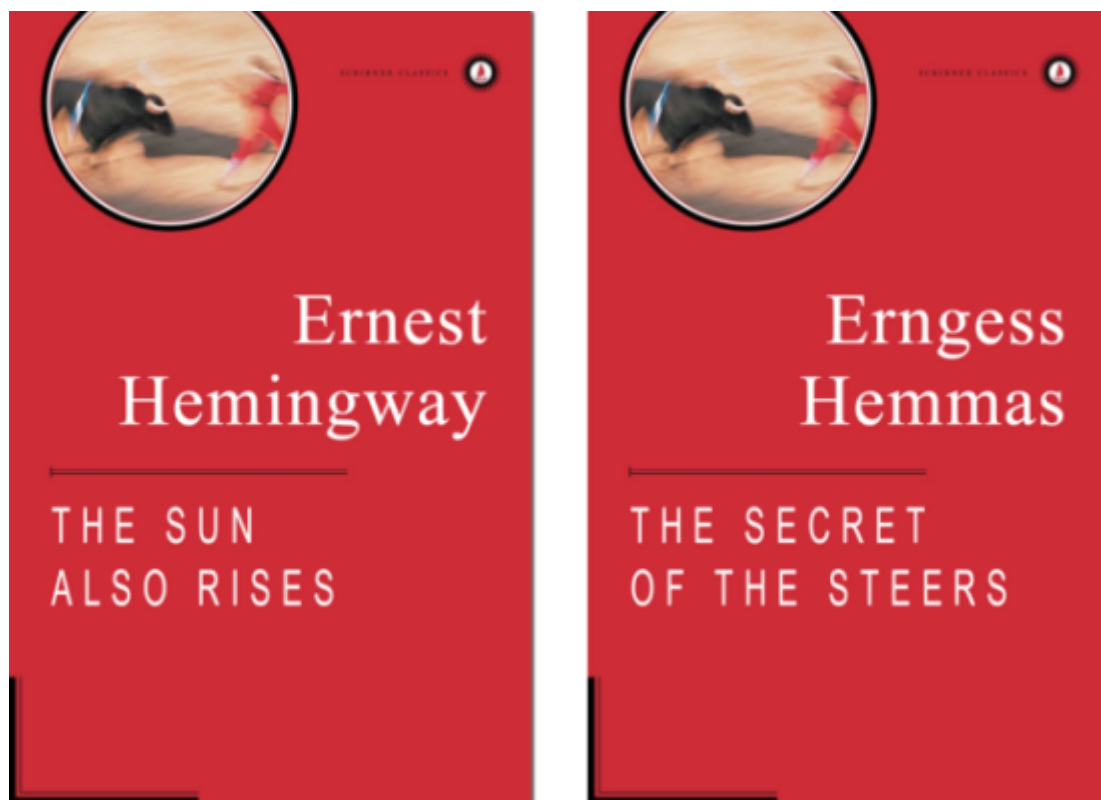
「没有。毫无音信。」

「怎么啦？」

即使我们只是寻找**每个字符之间**的规律，我们的算法也已经用恰当的格式重新出一篇看起来可信的文章。这非常厉害！

我们也不用完全从头生成文本。我们可以把前几个字母放到算法当中去，让它找到后面的几个字母。

让我们一起来模仿海明威著作的封面，来伪造书名和作者玩玩吧！我们规定最开始的字母分别为「Er」、「He」、「The S」。



真书在左，看起来傻乎乎的假书在右（右边的书名意思是——肉用公牛的秘密）。

看起来不错呀！

但是真正让人脑洞大开的部分是，这个算法能够找出任何数据序列中的规律。他可以轻松创作出**食谱**或者是**假的奥巴马的演讲**。但为什么一定要限定是人类语言呢？这个算法也可以用来处理任何有规律可循的数据。

不用马里奥，智造马里奥

2015年，任天堂在 Wii U 平台上发布了**超级马里奥制造**。



每个孩子的梦想！

在这个游戏中，你可以用编辑器创造出你自己的超级马里奥关卡，并把它上传到互联网上和朋友们一起玩。你可以使用游戏中所有经典的道具和敌人来创造你自己的关卡。这就像是一个为成年人设计的乐高积木玩具。

所以问题来了，我们能够使用创作海明威的模型来制作马里奥么？

首先，我们需要一组数据来训练我们的模型。我们会使用马里奥兄弟 1985 年版所有的室外关卡的数据。 ④



这个圣诞节棒极了！谢谢爸爸妈妈！

这个游戏共包含 32 关，而其中 70% 都是风格相似的户外场景。所以我们将使用这些数据。

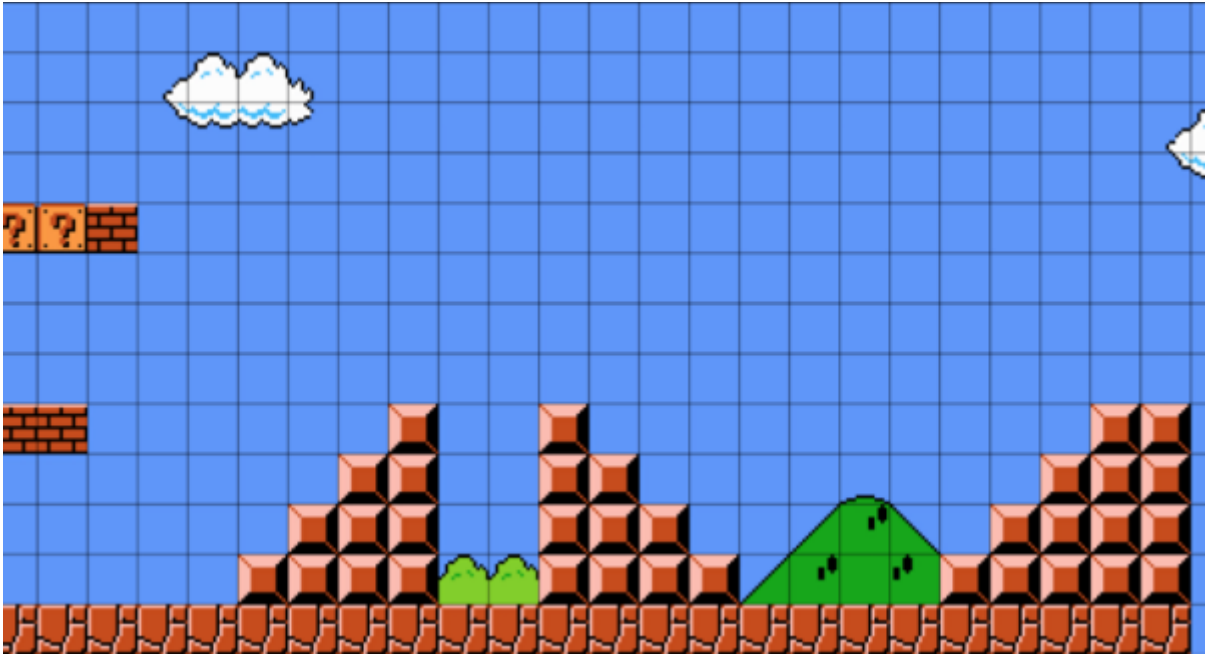
我设计了一个小程序，把原版游戏中所有的关卡设计都提取了出来。超级马里奥兄弟是一个有着 30 年历史的游戏，网上有着丰富的资源来帮助你找出关卡设计在游戏代码中的存储位置。从一个老游戏中提取关卡数据，是一个很有趣的编程练习，你有空可以试一下！

这个就是游戏的第一关（如果你玩过超级马里奥，你应该会记得）：



超级马里奥关卡 1-1

如果我们仔细观察，我们会发现这一关是由一个个简单的小网格类型的物品构成的：



我们可以简单的把这些网格表示成一序列字符，每一个字符都代表着一个物品：

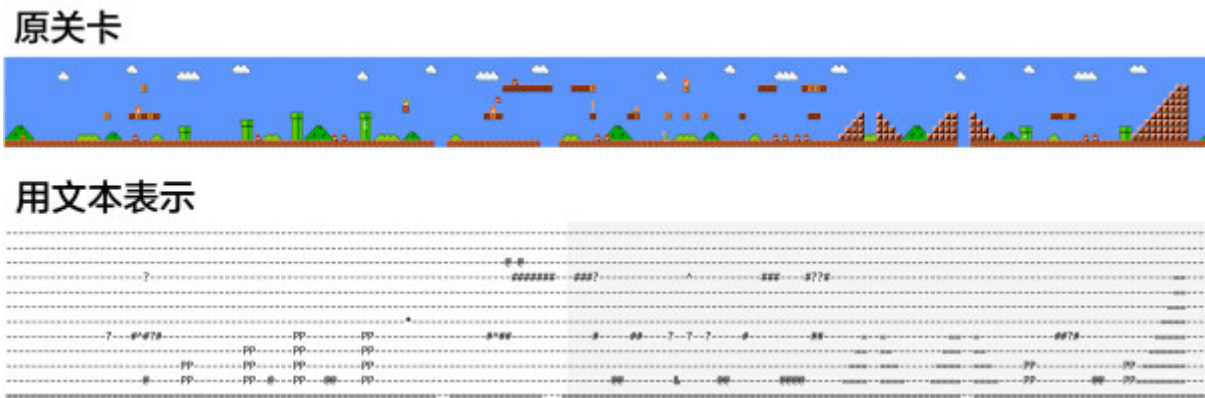
```
-----  
-----  
-----  
#??#-----  
-----  
-----  
-----  
-##-----==-----  
-----==-----==  
-----  
-----  
-----  
=====
```

我们把物品换成了下列字母：

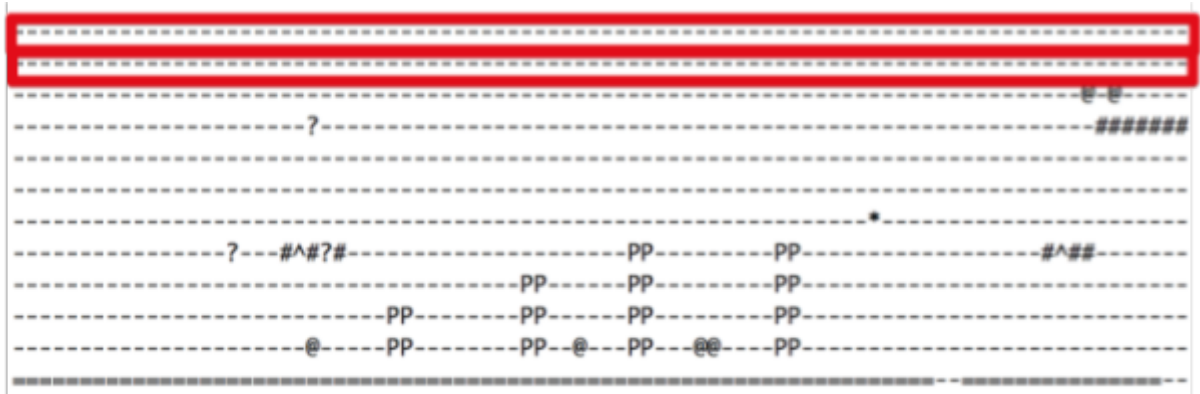
- 「-」代表没有物品
- 「=」代表砖块
- 「#」代表可以打碎的砖块

.....就类似于这样，用不同的字字符代表关卡里的不同的物品。

最后就得到了如下的文本文档：

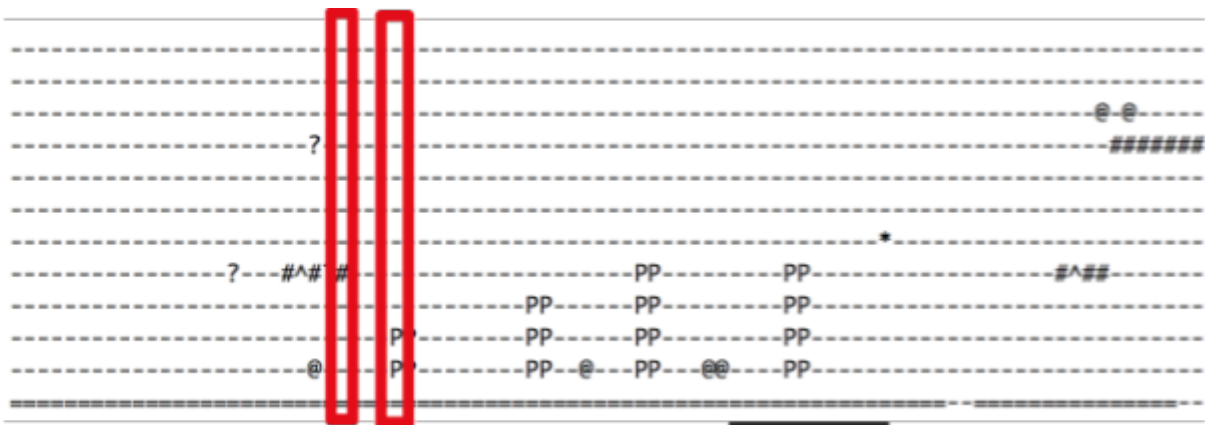


仔细观察这个文本文档，你会发现如果以「行」的顺序观察，并没有什么规律可循：



行一行寻找，并找不到什么规律。你会发现很多行就是空白的。

当你把关卡理解为连续的列的时候，规律就浮现出来了：



一列一列寻找，规律就显现出来了。比如说每一列都以「=」结尾。

为了让算法能找出我们数据中的规律，我们需要把数据以列的形式输入。找出你数据的最有效的表达方法（这个方法也叫作**特征选择**），是使用机器学习算法的重要技巧之一。

为了训练模型，我需要把这个文本档旋转 90 度。这样一来就保证了这些字符被输入进模型之后，模型能够更容易找到其中的规律。

-----=
-----#---=
-----#---=
-----?---=
-----#---=
-----=
-----=
-----@=
-----@=
-----=
-----=
-----=
-----PP=
-----PP=
-----==
-----===
-----====
-----=====

训练我们的模型

和刚刚训练海明威式文章生成器的过程一样，随着训练次数的增加，模型会渐渐被优化。

```
-----
LL+<&=-----P-----
-----
-----T--#--
-----
--==-----=&--T-----
-----
-------$-#-_-
-----==<----
-----b
-
```

它好像理解了应该有很多的「-」和「=」的思路，这已经很好了。但是还没有找到规律。

经过几千个循环之后，它开始变得有模有样了：

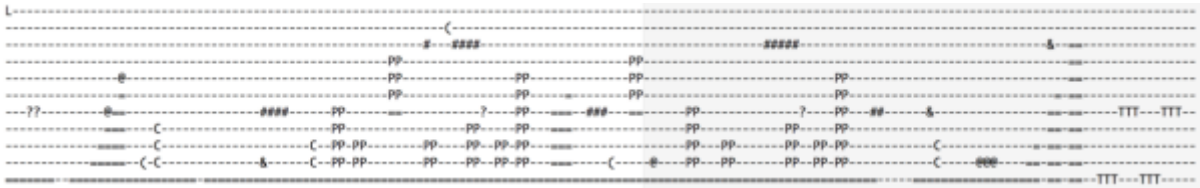
```
--
-----=
-----=
-----PP=
-----PP=
-----=
-----=
-----=
-----?---=
-----=
-----=
```

模型几乎已经知道每一行应该有相同的长度。它甚至开始找出马里奥的一些逻辑：管道（绿色的管子）经常是两格宽，并且至少有两格那么高，所以数据里面的「P」应该以一种 2×2

经过大量的训练之后，模型开始能生成完美的可用数据：

```
-----PP=
-----PP=
-----=
-----=
-----=
---PPP=---
---PPP=---
-----=
```

让我们用我们的模型来创造一整个关卡，并把它们横过来：

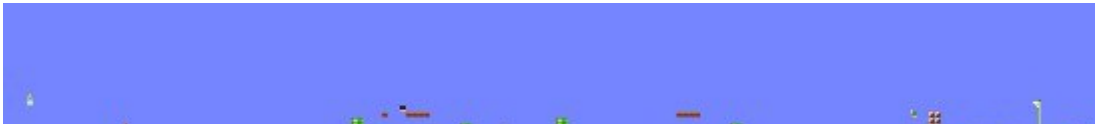


用我们的模型创造的一整关！

数据看起来棒棒哒！并且有以下几个优点值得关注：

- 它把 Lakitu（那个浮在云上的怪）放在了关卡一开始的空中，就像真实的超级马里奥关卡一样。
- 它知道浮在空中的管道是要放在砖块上面的，它不能单单漂在空中。
- 它把敌人放在了恰当的位置。
- 它看上去就超级马里奥的一关一样，因为它是在游戏里存在的原版关卡的基础上创造出来的。

最终，我们把这一个关放到超级马里奥制造里面，来创造出这一关：



自己玩玩看吧！（如果视频无法正常显示的话请[点击这里](#)。）

如果你有超级马里奥制造，你可以[通过网页版书签](#)收藏或者是用关卡代码 `4AC9-0000-0157-F3C3` 来找到这一关。

玩具或是真实世界的应用？

以上我们所使用的循环神经网络算法，就是真实世界中公司用来解决难题的算法。这些难题包括语音识别和文字翻译。解决难题和创建一个游戏关卡的区别，就在于数据量：我们的模型是由极少量的数据生成的。要创建一个非常好的模型，我们需要更多原版超级马里奥兄弟里面的关卡数据。

如果我们像任天堂一样拥有成千上万玩家自己创作的马里奥关卡数据，我们可以制作出一个超棒的模型。但是我们不能——因为任天堂不会把这些数据给我们。天下没有免费的午餐，大公司的数据不会免费给你。

随着机器学习在许多领域越来越重要，好程序与坏程序的区别就在于你拥有多少的数据来训练你的模型。这就是为什么像谷歌和 Facebook 这样的公司如此需要你的数据！

打个比方，谷歌最近开源了 [TensorFlow](#)，这是它用来建立大规模机器学习的工具包。把如此重要，如此实用的技术免费公布出来，对谷歌来说是一个很重量级的决定。你要知道，这可是和谷歌翻译使用的原理是相同的。

但如果你没有海量数据，你仍然没有办法创建一个能和谷歌翻译的匹敌的工具。数据是使谷歌处在行业顶端的源泉。想一想你打开[谷歌地图历史记录](#)或者是 [Facebook 地点记录](#)的时候，你会发现它们记录下来了你去过的每一个地方。

延伸阅读

[订阅](#) [往期](#) [登录](#)

在机器学习中，要解决问题决不只有一种方法。你总是有无数种方法预处理数据，你也有无数种机器学习算法可选。**综合各种方法**之后获得的结果通常会比使用单一方法更好。

以下是一些读者发给我的链接，还有这些有趣的方法可以生成马里奥兄弟关卡：

- **埃米·K.胡佛**（Amy K. Hoover）的团队把**游戏中的每一种物品（管道、地面、平台等）表示成交响乐中的一种声音**。然后再使用一种名为功能性搭建（functional scaffolding）的方法，让系统把每一种物品添加到关卡里去。比如说，你可以先自己制作出你想要的关卡基本样式，然后系统就能通过增加水管和是问号砖块来完善你的创作。
- **史蒂夫·达尔斯科格**（Steve Dahlkog）的团队，则把每一列关卡数据表示为一串 N 元语法（n-gram）的「单词」，在此之上建立模型。相比大型的 RNN，**这种生成关卡的算法更简单**。

作者：**Adam Geitgey**

原文：<https://medium.com/@ageitgey/machine-learning-is-fun-part-2-a26a10b68df3>

译文：<https://zhuanlan.zhihu.com/p/24344720>

译者：巡洋舰科技——赵 95

校对：林沁



亚当·盖特吉

软件工程师，Groupon 工程部总监，带领团队维护 Groupon 网页端。热爱计算机与机器学习。推特帐号 @ageitgey。

发表评论

电子邮件地址不会被公开。 必填项已用*标注

[订阅](#) [往期](#) [登录](#)


评论

姓名 *

电子邮件 *

站点

一个图灵小测试 *

九 + = 16 

发表评论

下一篇

卷首语