Broadview®
www.broadview.com.cn

一站式了解深度学习算法，结合实际工作快速上手！

深度学习算法实践

吴岸城　编著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

深度学习算法实践

吴岸城　编著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书以一位软件工程师在工作中遇到的问题为主线，阐述了如何从软件工程思维向算法思维转变、如何将任务分解成算法问题，并结合程序员在工作中经常面临的产品需求，详细阐述了应该怎样从算法的角度看待、分解需求，并结合经典的任务对深度学习算法做了清晰的分析。

本书在表达上深入浅出，让有志于学习深度学习的读者，能够快速地理解核心所在，并顺利上手实践。

图书在版编目（CIP）数据

深度学习算法实践 / 吴岸城编著. —北京：电子工业出版社，2017.7
ISBN 978-7-121-31793-4

Ⅰ. ①深… Ⅱ. ①吴… Ⅲ. ①机器学习－算法 Ⅳ.①TP181

中国版本图书馆 CIP 数据核字(2017)第 129958 号

策划编辑：刘　皎
责任编辑：郑柳洁
印　　刷：三河市鑫金马印装有限公司
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：720×1000　1/16　印张：13.5　字数：204 千字
版　　次：2017 年 7 月第 1 版
印　　次：2017 年 7 月第 1 次印刷
定　　价：79.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

前　言

随着机器智能的进步，人类预测技能的价值将会降低。原因在于机器预测比人工预测更为低价和优质，正如机器算数肯定比人力算数更为迅速准确。然而，这却并不像许多专家预言的意味着人类工作的末日，因为人类判断技能的价值将得以凸显。用经济学语言表述就是，判断是预测的互补品，因此当预测的成本降低，对判断的需求就会增大。

——*The Simple Economics of Machine Intelligence*，《哈佛商业评论》

当年互联网的大潮席卷一切时，数字通信技术被认为将颠覆商业、改变一切。之后的移动互联网也在某种程度上被认为将颠覆商业。经济学家总体上并没有被当时的互联网泡沫所忽悠。

现如今，有关人工智能的报道铺天盖地，有昔日“新经济”泡沫之势。这一次，基本的经济学原理和框架就足以帮助我们理解和预测这一技术形态对商业产生的影响。技术革命往往会使某些重要活动的成本降低，比如说通信或搜索信息等活动。究其实质，人工智能或者机器智能（Machine Intelligence）是一项预测技术，因此它的经济影响将围绕降低预测成本这个中心来展开。

对于已经身处这个大潮中的开发者、架构师、数据分析人员等，只能去拥抱这项技术。深度学习并不是一项凭空冒出来的技术，它在机器学习之上做了很多优化。本

质上所有的有监督学习都是在探讨怎样无限地逼近目标函数（强化学习另外讨论），而深度网络就是让机器代替人类提取特征的工作变得更有可能真正实现。在经济学家的眼里，现阶段人工智能的本质是从预测（或分类问题）开始，我想通过几个实际的例子来和大家聊聊这个话题；另一方面，我的团队在工作中积累了一些实际经验，我们也希望能将这些经验贡献出来，如果能在某种程度上对读者有所帮助那就最好不过了。以上两点，促成了本书的诞生。

我写的前一本书（《神经网络与深度学习》，电子工业出版社出版）偏向于概念讲解，因为写的时候深度学习并不普及，让大众了解深度学习的基本概念是最急迫的目标。本书大部分内容则偏向于应用，因为无论是降低社会成本，还是发明创造出新算法，都离不开实践；如果还能从实践中思考一些东西，那就是举一反三的能力了——这是我们人类独特的价值。所以在本书后面的强化学习、股票预测等章节，我们都会留一些问题，读者可以亲自实践，用深度学习这个工具创造出更多的价值，更长远地说，为推动强人工智能贡献自己的一份力量。

我一直相信，创造具有意识的 AI，对所有的科技人员都是一种诱惑，尽管有可能造出来就意味着人类的边缘化，但仍然要憋着劲儿去研究如何把它造出来——这简直就不像人类的自由意志，更像背后有一只手在推动着，我想上帝在造人时的心情也不过如此吧。

本书面向有一定基础、在工作中对深度学习有一定实际需求的读者；也面向那些有志于从传统的软件工程领域转型的工程师们。

本书一共分为 6 章。

第 1 章，主要讲从工程思维到算法思维的转变，对于有基础的读者来说稍显啰唆，但很重要，希望读者能仔细阅读。

第 2 章，阐述文本分析、文本深度特征等内容，已有基础的读者可以根据自己的需求部分略过。

第 3 章，主要介绍对话机器人的相关技术和发展。

第 4 章，主要介绍视觉，以人脸检测为例，从传统的 OpenCV 模式识别做人脸检测到用 CNN 网络做人脸表情识别。勾勒 CNN 的传承发展，讲述做图像分类、目标

识别等其他应用。

第 5 章，主要讲区别于一般的有监督学习的另一个问题：强化学习和 DQN 网络实践。

第 6 章，主要讲预测与推荐，以股票为例，并同时讨论了深度学习在推荐领域的应用。

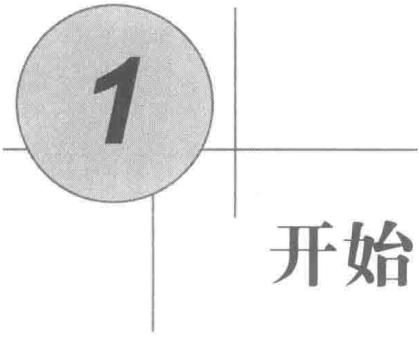
本书的完成得到了团队的大力支持：张帅、郭晓璐提供了图像方面的支持；周维提供了强化学习内容上的支持，在此衷心地感谢他们。

下面是本书用到的环境说明。

为保持一致性，本书所有的代码如无特殊说明都基于 Python 2.7 版本，系统环境为 Ubuntu 16.04，以及所用到的 Keras、TensorFlow、MXNet、Caffe、Openface、openAI、OpenCV、Dlib、NumPy、SciPy、Gensim、Theano 等均为 2016 年 9 月的最新版本。

本书的部分源代码整理在：https://github.com/wac81/Book_DeepLearning_Practice，仅供研究使用，使用时请注明来源，如需用作商业或其他用途，请联系作者取得授权。

目　录



开始

1.1 从传统的软件工程思维转型

2013 年的秋天，这时英特已经从中国一流学府的计算机系毕业 3 年了，他所在的这家公司早已在纳斯达克上市，而英特已经成为这个公司的中层。这 3 年来，英特初出茅庐的那些热情已经被磨砺得差不多了，但他的心底还是在隐隐渴望着什么。对如今的工作，他总觉得缺少些激情，总觉得有点按部就班。从当年的研发工程师到如今的研发管理中层的位置，这种感觉一直如影随形。

今天英特被总经理叫到办公室讨论部门以后的规划，老板看最近大数据已经开始火了，想要英特组建一个大数据的团队，这个团队主要是通过对公司数据仓库的迁移和计算，看看能不能打造出什么新产品。

英特接手此事后，便开始调研，发现如今只要内部开始做大数据的公司，都无一例外地弄了一个算法团队。然而这些算法团队基本上对后期的想法和规划很不清晰，感觉大家都是在摸着石头过河。于是，英特也想配套着弄一个算法团队。

略去大数据团队不说，单说算法团队，英特的公司里就没有这样的人才。这意味着英特需要开始招兵买马了。英特先从一家知名电商公司挖来了一名高级算法人才，

然后照着行业内其他一些公司的做法，组建了相关的数据标注员，并在公司内部挑选出了一些有经验的 Python、C++工程师。

现在英特面临着一个问题：大部分人员都是做工程的研发工程师，现在他们的目标是用编程语言实现算法，那么实现算法与之前做一个工程项目有区别吗？如果还按照之前那样，从工程的角度思考问题，能否实现效果或者能否达到业务部门的要求呢？

英特开始时很乐观，正好这时公司的业务部门给算法团队提了第一个需求——“先把垃圾邮件给过滤下吧，最好对于正式的邮件也做一些分类，比如根据文本内容推荐重要或紧急邮件，对于一般邮件不做重点过滤。”英特接到这个需求后，马上做了些调研，发现这个需求就是一个分类需求，用任何一个分类器都可以比较快地搞定。

那英特如何做呢？按照现有的软件工程的思维，是需求分析→逻辑设计→详细设计→测试→上线，整个过程英特非常熟悉。然而按照这种过程，算法团队的小朋友们还没有干到逻辑设计这一步，就已经被业务部门催促了，“我们只要一个功能你们动作怎么这么慢”。在两个月的煎熬之后，英特总算给出了第一个可以使用的模型，用在对重要邮件、垃圾邮件的分类中。但仅仅过去一天，英特的电话就被打爆了，大家几乎都在抱怨说，重要邮件被分到垃圾邮件中，非重要邮件被推荐了。英特感到很委屈，试图去解释，却又被业务部门讽刺了一顿。重新梳理了整个流程之后，英特得出一个结论：对于算法而言应该有算法的思维。不能完全套用以前做软件工程的那套东西。

1.2 建立算法思维

看看业务部门抱怨的两点：第一，算法团队反应太慢。第二，没有达到预期效果。从中我们隐约可以感觉到传统的软件工程思维并不适合做算法。

下面，我们就和英特一起来梳理下为什么传统的软件工程思维不适合做算法，以及算法的开发流程和步骤应该是怎样的？

1.2.1 算法的开发流程

用算法解决项目中的问题实际上是一种实验思维。实验思维顾名思义，就是我们用初始数据做实验，输出一些实验产物，对于算法来说就是一些向量或矩阵，也就是一堆数字，我们得到这些数字后可以开始反馈到现实业务中，用于比较输出值和真正的业务需求是不是具有一致性。所以做算法的开发流程完全可以抽象为如图 1-1 所示的框架图。

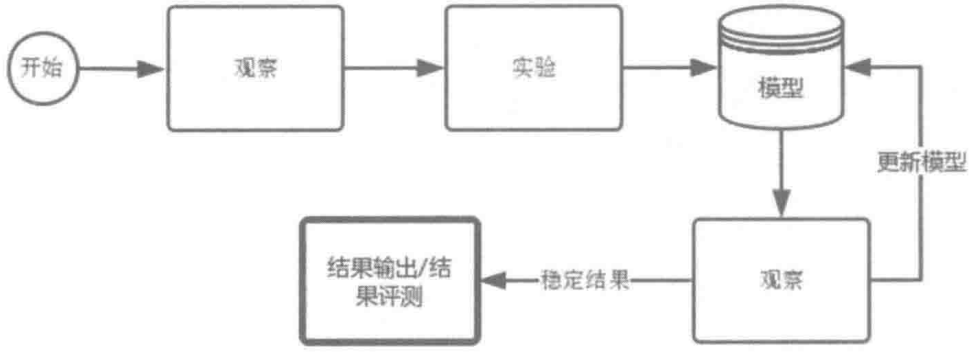


图 1-1 算法开发流程

举个例子，我们开发一个算法时，一般是要用来替代人的一部分工作的，所以我们会先了解人是怎么完成这部分工作的。比如邮件分类，就先来看看人是如何做邮件分类的。

（1）看邮件的标题，很多信息其实在标题上就有呈现；

（2）注意发件人，看用户名是不是一堆无意义的字符；

（3）看邮件内容，邮件内容不用看完，优先去找到一些重点，对于中文的垃圾/重要邮件，会看看有没有“发票”、“讲座”、“小姐”之类的词；

（4）如果邮件带附件，要对附件名进行仔细的甄别。

了解人是怎么来完成邮件分类的，我们就完成了邮件分类算法的前期观察。接着我们将观测到的重要特征抽象出来，这个时候可以人工选择特征，也可以用机器学习的方法抽象一些特征。针对邮件分类问题，我们可以用特征做分类算法，可以采取的分类算法有很多，你既可以选用普通的机器学习算法，比如贝叶斯、SVM、随机森

林等，也可以选用神经网络来进行分类。最后我们将结果回馈到我们观察的事物中去，简单地说，这一步是一个验证和调整的过程，这一步的好坏也直接决定了我们最后的结果是好是坏，这其实也是区分一个有经验的算法工程师和一个初级的算法工程师的标准。

刚才我们简单地了解了算法的开发流程，下面我们再来看看英特是如何做好观察并一步一步完成需求的。

1.2.2 做算法的步骤

根据前文所提到的开发流程，算法组决定根据以下 4 个步骤来分解邮件分类问题。

步骤一：明确有哪些需求

需求部门直接提出了哪些需求？

我们可以发现，直接需求非常简单："先把垃圾邮件给过滤下吧，最好对于正式的邮件也做一些分类，比如根据文本内容推荐重要或紧急邮件，对于一般邮件不做重点过滤。"

通过分解直接需求可以得出两个需求：

（1）过滤垃圾邮件。

（2）分类非垃圾邮件，建议分为重要紧急邮件和一般邮件两类。

其实一般来说，直接需求都不难满足，这里按字面意思设计两个分类器也行，或者设计一个分类器直接将邮件分为垃圾类、重要类、紧急类三类也行。

那么，除了直接需求之外，需求部门的隐含需求有哪些呢？

再次来看看英特被需求部门抱怨的两个点：

（1）慢！

（2）分类错误多！

从这里我们可以看出，需求部门的第一个期望是准确率高，第二个期望是速度能更快。

“大多数需求在提出时，对于准确率都是有要求的！”请大家记住这句话，并默认这个准确率在 95%以上。对于一些诸如图片分类或者预测问题，我们根本达不到这个准确率的时候怎么办？如果限于目前的技术水平达不到，那就尽可能多地提高准确率，去尝试多种算法和模型，因为解决大多数实际问题都不只是用一种算法或者一个模型就能完成的。

如果最后还是达不到，或许可以换种方式引导客户，比如微软之前推出过一个根据人脸识别年龄的 App。对于工业化来说，识别要求非常高，误差也只能在±2 岁之内，而由于每个人的生长环境不同，人种不同，要用一个普遍意义上的模型达到这种区分精度几乎不可能。所以微软将这个功能加到 App 上并赋予它一个娱乐属性，就是要大家别当真了，玩玩就好。从这里我们可以得到启发，如果达不到精度，算法工程师或许可以和产品经理或需求部门商量，在客户引导层面规避对准确度过高的要求。

步骤二：观察需求中涉及的问题

英特观察邮件分类问题的时候，体会了一下其他公司的产品，发现它们的产品也有分类错误的时候，比如企鹅公司做的邮件分类，在垃圾邮件中确实几乎 100%都是垃圾邮件，而正式邮件中偶尔会出现垃圾邮件。但这些错误却不太影响人的主观感受。这是怎么回事呢？原来人们在浏览这些正式邮件时，会自动剔除（跳过）垃圾邮件。而自家的算法组做的模型是非常严格地区分垃圾邮件，有时候把正式邮件也当成垃圾邮件处理了：这可能会导致用户错过非常重要的邮件，因此这种结果对于用户来说简直是不可接受的！这种情况表现到数据观察上，就是精确率低了，而召回率高了。

这里我们引入三个概念：准确率（Accuracy）、召回率（Recall）、精确率（Precision）。

准确率 = 提取出的正确信息条数 / 总样本条数

精确率 = 提取出的正确信息条数 / 提取出的信息条数

召回率 = 提取出的正确信息条数 / 样本中的信息条数

比如说某人的邮箱中有 1000 封正常邮件、500 封垃圾邮件和 100 封重要邮件。现在以找垃圾邮件为目标，模型一共找到 400 封垃圾邮件、5 封重要邮件和 100 封正常邮件，那么三个指标如下。

准确率（垃圾邮件）= 400/（1000+500+100）= 0.25 = 25%

一般来讲我们的准确率不应只针对一类邮件来说，应该对于所有判断正确的邮件而言，这里包含重要邮件、正常邮件，所以准确率的计算如下。

准确率=（400+5+100）/（1000+500+100）= 0.253 = 25.3%

精确率（垃圾邮件）= 400/（400+5+100）= 0.79 = 79%

召回率（垃圾邮件）= 400/500 = 0.8 = 80%

如果有一种模型能同时将精确率和召回率提高那当然是最好的，但如果只能选择其一的话，在这个邮件分类的案例中，我们应该选择精确率提高作为我们的目标。

极端情况下，如果算法组的模型只出了一个结果，而且是准确的，那么精确率就是 100%，但是召回率就很低；而如果模型把所有结果都返回，那么召回率是 100%，但是精确率就会很低。因此在不同的场合中需要自己判断希望精确率比较高还是召回率比较高。如何找到一个比较好的比例呢？如果是读者自己做实验研究，可以绘制 Precision-Recall 曲线来帮助分析（如图 1-2 所示）。

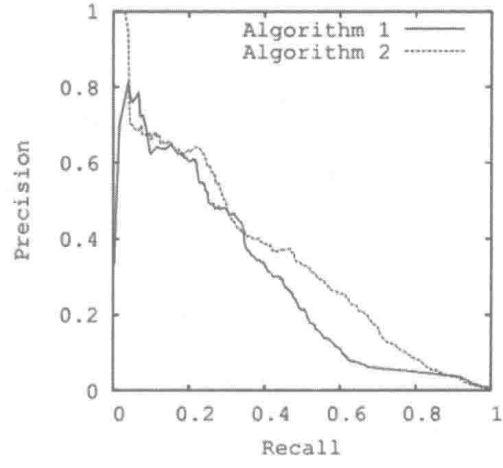


图 1-2　Precision-Recall 曲线

步骤三：开始算法工程的实践

英特在拿到需求后就开始开发算法了，但整体的进度和需求部门期望的进度还是差距很大，这是英特没有跳出传统软件思维的束缚所导致的。

对于需求部门来说，希望算法组非常快地给他们一个结果，这样就可以先试试怎么样？看看缺点在哪里？有了目标才好继续改进。而英特用软件工程的方法把算法当成了一个系统来做，这显然错误地估计了需求部门的隐含需求，也使得自己非常被动。

另外，对于算法的选用，也就是到底使用哪些算法处理业务问题，英特还是觉得有些力不从心，哪些算法可以真正带来效果的提升，光凭想象是没有意义的，一切都需要实践！

在后面的工作中，英特越来越觉得盲目地使用算法，结果往往是消耗了大量时间、精力而效果没有显著提高，对自己非常不利，也对整个算法团队不公平。那么更加务实的方法，就是去理解数据，理解用户，从分析和挖掘需求出发，先解决关键问题，再脚踏实地、一步步优化效果。对数据理解足够深刻之后，算法工程师模型方面的知识才能派上用场。对数据和业务的理解需要时间沉淀，而对于模型的灵活运用需要深厚的理论知识背景。

虽然这本书着重写的是算法，但请大家谨记下面两句话。

> 数据（业务）和算法同等重要，甚至有时候更重要。
> 不要为了用算法而用算法。

步骤四：测试

测试就是如何评价这个算法。现在英特也很头疼测试，在英特观察需求的时候，已经知道了三个用来评价模型好坏的基本参数。

但我们要重新来谈谈有关准确率、精确率和召回率这三个参数的问题。也许从最开始评价模型时，研究者们只发明了准确率一个参数，因为准确率是指对于给定的测试数据集，分类器正确分类的样本数与总样本数之比。在逻辑上这是非常清晰的，这就好比我们一个人区分垃圾邮件，找出真正的垃圾邮件占总邮件的多少，也能说明这个人寻找垃圾邮件的能力。

然后马上就发现这里有问题，我们稍微修改一下上面的数据，比如说某人的邮箱中有 1000 封正常邮件，5 封垃圾邮件，10 封重要邮件。假设现在的一个模型是判别正常邮件的，明眼人一看就知道，即使模型毫无作用，将一共 1015 封邮件都当成正

常邮件处理，这样准确率也能高达 1000/1015 = 0.985 = 98.5%，而这个“啥也没做的”所谓的模型是否真正地反映了处理能力？

看到这里，我们知道了准确率并不能真正评价一个模型的能力，所以研究者们发明了另外几个评价指标来判断，那就是精确率（Precision）、召回率（Recall）和 F_1-Measure。

这里再稍微说下 F_1，F_1 就是 Precision 和 Recall 的调和均值：

$$\frac{2}{F_1} = \frac{1}{P} + \frac{1}{R}$$

把这个式子调整一下就得到：

$$F_1 = \frac{2PR}{P+R}$$

所以我们从上面的例子来看，可以得到：

$$F_1 = 2 \times 0.79 \times 0.8/(0.79 + 0.8) = 0.794 = 79.4\%$$

F_1-Measure 默认精确率和召回率的权重是一样的，但有些场景下，我们可能认为精确率会更加重要，这时可以通过调整参数得到相应式子。

模型的测评指标有了，那如何对模型进行验证呢？有很多测试方法，比如交叉验证等，需要英特及整个算法组在后面积极思考和善加利用。

假设现在已经将模型验证完了，英特也拿到了评测数据，这个模型是否可以投入使用则需要根据这些测评指标来判断，通过不同的参数调整得到相关的测评指标，然后将这些测评指标进行比较，最后选出一个“最为平衡”的模型。注意，是“最为平衡”而不一定是某几个值最高，一切的出发点都以实际业务为准。

1.2.3 英特的总结

在整个算法项目的研发过程中，英特也注意到前人早已对算法的研发流程总结了规律。这些规律已经形成了标准，有两种方法论，一种是 SPSS[1]的 CRISP-DM，另外

1 当然现在 SPSS 已被 IBM 收购了，这里还是称为 SPSS。