机器学习观止
核心原理与实践

Machine Learning

林学森 著

清華大學出版社
北 京

内 容 简 介

本书在写作伊始，就把读者设想为一位虽然没有任何AI基础，但对技术本身抱有浓厚兴趣、喜欢“抽丝剥茧”、探究真相的“有识之士”。有别于市面上部分AI技术书籍从一开始就直接讲解各种“高深莫测”算法的叙述手法，本书尝试先从零开始构建基础技术点，而后“循序渐进”地引领读者前进，最终“直捣黄龙”，赢取最后的胜利。

全书据此分为5篇，共31章，内容基本覆盖了由AI发展历史、数学基础知识、机器学习算法等经典知识点以及深度学习、深度强化学习等较新理论知识所组成的AI核心技术。同时注重“理论联系实践”，通过多个章节重点介绍了如何在工程项目中运用AI来解决问题的诸多经验以及相应的模型算法，以期让读者既能享受到“知其所以然”的乐趣，还能体会到“知其然”的轻松和愉悦。

本书适合对AI感兴趣的读者阅读，从事AI领域工作的研究人员、工程开发人员、高校本科生和研究生都可以从本书中学到机器学习的相关知识。

图书在版编目(CIP)数据

机器学习观止：核心原理与实践 / 林学森著. —北京：清华大学出版社，2021.3
ISBN 978-7-302-55744-9

Ⅰ. ①机… Ⅱ. ①林… Ⅲ. ①机器学习 Ⅳ. ①TP181

中国版本图书馆CIP数据核字(2020)第104785号

责任编辑：李万红 施 猛
封面设计：熊仁丹
版式设计：方加青
责任校对：马遥遥
责任印制：丛怀宇

出版发行：清华大学出版社
网 址：http://www.tup.com.cn，http://www.wqbook.com
地 址：北京清华大学学研大厦A座 邮 编：100084
社 总 机：010-62770175 邮 购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：三河市铭诚印务有限公司
经 销：全国新华书店
开 本：210mm×285mm **印 张：**48.25 **字 数：**1460千字
版 次：2021 年 3 月第 1 版 **印 次：**2021 年 3 月第 1 次印刷
定 价：168.00元

产品编号：086756-01

前 言

这本书前后写了五年，终成稿付梓。

过去五年是以机器学习，特别是深度学习为代表的人工智能技术爆炸性增长的一段关键时期。在本书写作期间，“深度学习三巨头”Yoshua Bengio、Geoffrey Hinton和Yann LeCun共同获得了图灵奖，更是把本轮人工智能浪潮推向了巅峰。可以说，我们大家都在见证，并将在未来几十年里持续见证人工智能的各种突破性研究成果，以及它们在产业界各个角落的落地开花。

那么，本轮人工智能(不论以何种分支为载体)的兴起，真的可以如人类已经经历的几次工业革命一样，给整个社会带来颠覆性的影响吗？

提出这种疑问的人，可能对人工智能在过去几十年的“风雨飘摇”还记忆犹新。

人工智能起源于20世纪50年代，除了Alan Mathison Turing提出的著名的“图灵测试”外，还有几个重要事件，比如：世界上第一台神经网络计算机的建造；1956年在美国Dartmouth College召开的会议上，人们第一次提出人工智能(Artificial Intelligence，AI)术语，并将其确立为一门独立的学科；世界上第一座人工智能实验室，即MIT AI LAB的建立等。

随后的二十年是人工智能的第一次高峰，人们认为只需要较短的时间，机器就会取代人类完成很多工作。然而事实证明，当时的这个观点显然过于乐观了——由于AI研究长期的“雷声大雨点小”，于是1973年一份著名的《莱特希尔报告》(*Lighthill Report*)就成了压死骆驼的最后一根稻草，直接将AI逼进了第一个寒冬(1974—1980年)。

不过这似乎并没有完全浇灭人们对于人工智能的“希望之火”——果然，几年后的1980年，AI又以“专家系统”重出江湖了，而且这一次还是伴随着商业机遇出现的(据悉1985年AI市场规模已经达到了10亿美元级别)。人们在看到新希望的同时，也催生了AI的再次繁荣。

然而好景不长，AI的再次复出仍然没有能够达到人们的预期。正所谓“希望越大，失望越大”，于是，伴随着LISP机(LISP machine)市场的萎缩，AI又迅速地成了一个“弃儿”(1987—1993年)。

至此，AI实际上已经是“四十岁”的“中年人”了。人们常说“四十不惑”，或许AI在经历了这样曲折的“人生阅历”后，也可以逐步成熟稳重起来。事实证明，AI也确实在“凤凰涅槃”，1993年之后的它逐步在多个领域展开了拳脚，并且在学术界和工业界都取得了不小的成绩。特别是步入21世纪以后，计算机运算能力的指数级增长以及大数据的繁荣使得AI“内功”得到了极大的增强，于是以深度学

习为代表的人工智能在多年的理论沉淀后终于得以“厚积薄发”。毫不夸张地说，人工智能已经成为当今最具热度的一个研究方向之一。业界预判AI将是激发新一轮产业革命的突破性技术，如图1所示。

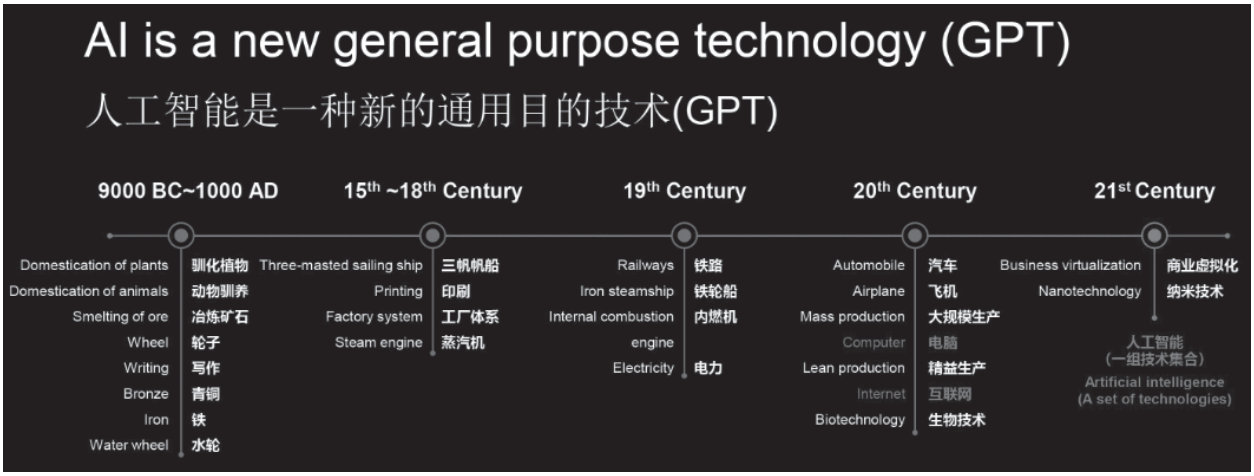

图1　业界预判AI将是激发新一轮产业革命的突破性技术

(材料引用自：华为2018年全联接大会keynote)

那么，人工智能是否会进入下一个“寒冬”，或者说它在本轮热潮中还可以走多远呢？

相信没有人可以给出准确的答案，毕竟我们谁也不可能预知未来。但是，这并不应该成为大家对它热爱与否的前提条件，因为我们相信：

人工智能必将取得全面成功。

或许还需要10年、50年甚至更长的时间，但这个趋势是不可逆转的。与其去纠结人工智能寒冬是否会到来，不如静下心来好好研究一下AI技术是否可以帮助我们解决一些实实在在的工作中的问题。

总的来说，人工智能的发展历史还在继续，而我们这一代人则有幸正在参与它的谱写过程。这也是本书的创作目的之一，希望通过由浅入深的叙述方式，来带领更多的AI爱好者进入这个领域，并在可能的情况下为AI的发展“添砖加瓦”。

致谢

感谢清华大学出版社的编辑，你们的专业态度和处理问题的人性化，是所有作者的“福音”。

感谢我的家人林进跃、张建山、林美玉、杨惠萍、林惠忠、林月明，没有你们的鼓励与理解，就没有本书的顺利出版。

感谢我的妻子张白杨的默默付出，是你工作之外还无怨无悔地照顾着我们可爱的宝宝，才让我有充足的时间和精力来写作。

感谢所有读者的支持，是你们赋予了我写作的动力。

编者

2021 年元旦

目 录

机器学习基础知识篇

经典机器学习篇

深度学习进阶篇

机器学习应用实践及相关原理

机器学习平台篇

5.1 K-NN概述

K-NN的全称为K-Nearest Neighbors(中文常被译作“K最近邻算法”)，是模式识别领域应用相当广泛的一个算法。K-NN经典算法起源于20世纪70年代，它既可以用于分类任务，也能够支持回归任务。K-NN在处理这两种情况时的算法输入都是一样的，即包含特征空间的k个最接近的训练样本，只不过它们在输出结果上有所差异。

1. K-NN分类

当K-NN用于分类任务时，一个对象A的分类结果取决于训练集中与它最相近的k个元素的投票结果(例如少数服从多数)。如果k取值为1，那么意味着只要找寻与A最相近的训练样本就可以确定它的类别，如图5-1所示。

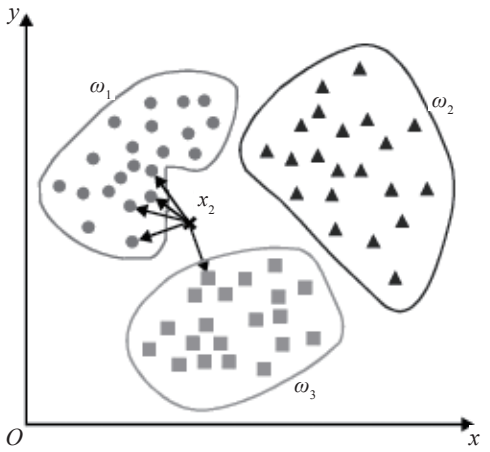

图5-1 K-NN分类图示

2. K-NN回归

当K-NN用于回归任务时，输出结果代表的是该对象A的属性值(取与A最相近的k个样本的平均值)，如图5-2所示。

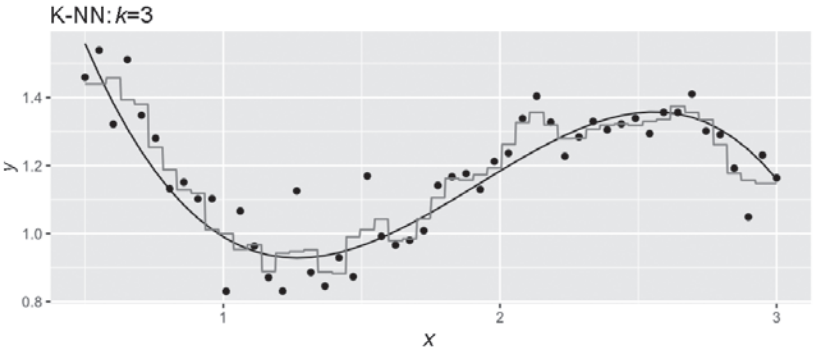

图5-2 K-NN回归

K-NN的另一个显著特点是它不需要训练，所有计算(分类/回归)都是在应用推理的时候才会执行，因而也被人们归于惰性学习法(Lazy Learning)——与之相对的是急性学习法(Eager Learning)。后者是指先利用训练数据进行训练得到一个目标函数，然后应用推理时只需利用训练好的函数进行决策。SVM等算法就属于这种学习方式。

这两种学习方法的区别如下。

(1) 急性学习法需要基于所有样本开展训练过程，因而训练的时间较长，但推理决策时间较短。

(2) 惰性学习法虽然在做最终决策时通常只用到了局部的几个样本，但由于它需要计算推理对象与所有样本之间的距离，复杂度还是达到了$O(n)$。所以惰性学习法不仅需要较大的存储空间，而且决策过程比较慢(没有训练过程)。

5.2 K-NN分类算法

K-NN算法本身并不复杂，它主要由以下三个核心要素组成。

(1) k的取值。

(2) 近邻算法。

(3) 投票机制。

接下来围绕上述三个核心点来展开分析。

1. k的取值

理论上，k可以取合法范围内的任何数值——但需要思考的是，k取不同值会不会影响最终的结果呢？答案是肯定的，而且影响很大。

例如，如图5-3所示的经典案例。

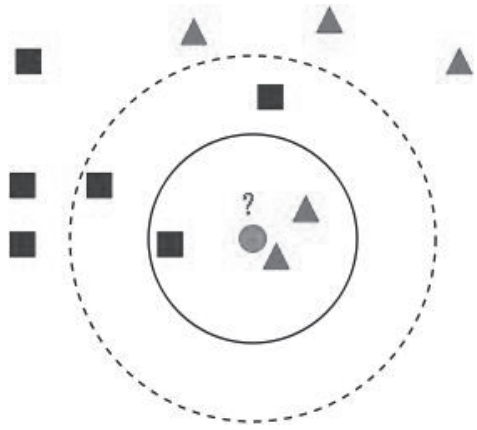

图5-3 k的取值对结果的影响示例

在图5-3的范例中，当k取不同值时，结果如下。

(1) k=3时：不难发现，此时中间的圆圈与两个三角形及一个正方形最近，所以如果按照投票法，它应该归属于三角形类别。

(2) k=5时：此时圆圈与两个三角形及三个正方形最近，因而如果按照投票法，它就变成归属于正方形。

对于k的取值，我们可以思考：

(1) 因为到目前为止还没有成熟的理论来支撑k的选值过程，所以一般的做法是：采用交叉验证的方式来得到一个理论上最佳的k值(类似于模型训练)。

(2) k值较小时。

k值越小，代表参与最终决策的样本数量越少。这种情况下的训练误差可能会较小，但泛化误差会增大，或者说容易发生过拟合。

(3) k值较大时。

k值越大，代表参加最终决策的样本数量越多。其优点是可以减少噪声的影响，因而泛化误差较

小，但缺点是训练误差会增大。

(4) k等于样本数n时。

这是一种极端的情况，此时相当于没有分类。

(5) 根据实践经验来选取k值。

从项目实践经验来看，k的取值一般都是：$k<\sqrt{n}$，n为样本数量。

2. 近邻算法

K-NN的可选近邻算法有很多种，包括但不限于：

1) 曼哈顿(Manhattan)距离

典型公式如下。

$$d_{12}=\sum_{k=1}^{n}\left|x_{1k}-x_{2k}\right|$$

2) 欧几里得(Euclidean)距离

典型公式如下。

$$d_{12}=\sqrt{\sum_{k=1}^{n}\left(x_{1k}-x_{2k}\right)^2}$$

3) 马氏(Mahalanobis)距离

典型公式如下。

$$D\left(X_i,X_j\right)=\sqrt{\left(X_i-X_j\right)^{T}\boldsymbol{S}^{-1}\left(X_i-X_j\right)}$$

其中，$\boldsymbol{S}$为协方差矩阵。

4) 闵可夫斯基(Minkowsky)距离

典型公式如下。

$$d_{12}=\sqrt[p]{\sum_{k=1}^{n}\left|x_{1k}-x_{2k}\right|^{p}}$$

其中，“欧几里得距离”是K-NN中最常用的一种距离度量方法。

3. 投票机制

K-NN的投票机制也有多种选择，例如：

1) 少数服从多数

这是最简单的一种投票方式，但是它没有考虑k个最近邻元素之间的“远近”差异，因而在效果上并不是最佳的。

2) 加权投票

和上述“人人平等”的投票方式不同，加权投票针对不同近邻元素的差异性进行了考量——例如，按照它们与被预测对象的距离远近分别赋予相应的权重，保证距离近的元素对最终决策的影响力更大一些，反之距离远的作用则被有目的性地降低了。

综合来看，距离加权的投票方法是K-NN中采用最广泛的其中一种投票机制。

5.3 K-NN回归算法

K-NN回归算法和K-NN分类算法大致相同，接下来结合一个范例来进行实际的讲解。

假设要帮客户出租一套3室的房子，那么定价多少合适呢？

K-NN可以回答这个问题——可以基于市面上相同或者相近户型的房子的出租价格，来得出一个相对合理的答案，如表5-1所示。

表5-1 市面房租价格

序号	户型	出租价格/元
1	3室	1000
2	3室	1200
3	3室	1150
4	3室	1250
5	5室	2000

假如选择欧氏距离作为度量标准，那么：

1. 对于3室样本

此时欧氏距离公式中的n=1，即

$$\begin{aligned} d_{12} &= \sqrt{\sum_{k=1}^{n}\left(x_{1k}-x_{2k}\right)^2} \\ &= x_1 - x_2 \\ &= 3-3 \\ &= 0 \end{aligned}$$

2. 对于5室样本

同理，可得$d=2$。

再假设k的取值为4，换句话说，我们考虑的是4个最相近的房租样本，那么按照K-NN的算法思想可以得到待出租的3室的价格myPrice：

```
myPrice = (price1 + price2 + price3 + price4) / 4
        = (1000 + 1200 + 1150 + 1250) / 4
        = 1150
```

可以看到上述计算过程中，使用的是针对k个样本取均值的方式——不过和K-NN分类算法中的加权投票法类似，我们在回归场景中也可以考虑基于样本的差异性来区分对待，从而更好地发挥出样本数据的价值。

5.4 K-NN的优缺点

K-NN虽然是一个经典的算法，但它既有优点，同时也有不足之处。

5.4.1 K-NN的优点

K-NN算法的主要优点如下。

(1) 算法简单，易于理解，易于实现。

(2) 不需要训练过程。

(3) 预测精度较高。

(4) 通常对噪声数据不是特别敏感。

(5) 针对多分类问题(即对象具有多个类别标签的情况)，K-NN算法的表现比较突出，甚至超过SVM等算法。

5.4.2 K-NN的缺点

K-NN算法的主要缺点如下。

(1) 属于惰性学习法的一种。在前面已经分析过，这种类型的算法虽然在做最终决策时只用到了局部的几个样本，但由于它对每一个待分类的样本都需要计算其到全体样本数据的距离，才能获取k个最近邻点，所以整体复杂度还是$O(n)$。在实践中，针对这一问题的一个常用解决方法是提前减除掉对分类结果作用不大的部分样本。

(2) 不仅决策过程比较慢(虽然理论上没有训练过程)，而且空间开销也较大。

(3) 可解释性较差。

(4) 当样本不均衡时(即归属于某些类别的样本数量很大，而其他类别样本数量又很小的情况)，针对新样本的预测有可能出现偏差——因为与该样本最近邻的k个数据中，样本数量大的类别很可能会占绝对多数。

(5) 向量的维度越高，那么欧式距离用于计算样本差异的能力就越弱，由此可能导致预测结果的不准确性。

当然，K-NN的上述这些缺点并非完全无法解决。只要做到“知己知彼”，那么就有可能在实际项目中采取有效的手段来规避或者减少这些缺点所带来的影响。

5.5 K-NN工程范例

本节中将通过scikit-learn的一个范例来讲解如何使用K-NN算法来解决一些实际问题。

scikit-learn框架下，与K-NN算法相关的包是sklearn.neighbors，这个包中不仅包含K-NN的经典实现，还包含它的多种扩展实现，如图5-4所示。

neighbors.BallTree	BallTree for fast generalized N-point problems
neighbors.DistanceMetric	DistanceMetric class
neighbors.KDTree	KDTree for fast generalized N-point problems
neighbors.KernelDensity([bandwidth, ...])	Kernel Density Estimation
neighbors.KNeighborsClassifier([...])	Classifier implementing the k-nearest neighbors vote.
neighbors.KNeighborsRegressor([n_neighbors, ...])	Regression based on k-nearest neighbors.
neighbors.LocalOutlierFactor([n_neighbors, ...])	Unsupervised Outlier Detection using Local Outlier Factor (LOF)
neighbors.RadiusNeighborsClassifier([...])	Classifier implementing a vote among neighbors within a given radius

图5-4 K-NN扩展实现

其中，KNeighborsClassifier的原型如下。

class sklearn.neighbors. **KNeighborsClassifier**(*n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None, **kwargs*) ¶ [source]

接下来的代码范例中就是基于KNeighborsClassifier来实现的，核心代码节选以及详细释义如下。

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
```

```
from sklearn import neighbors, datasets
n_neighbors = 15 ##k值选取15
#加载iris(鸢尾花)数据集
iris = datasets.load_iris()
#iris是多维数据集，我们只选取两个维度进行示例
X = iris.data[:, :2]
y = iris.target
```

该数据集是关于鸢尾花的，大致是如图5-5所示的样式。

	花萼长度	花萼宽度	花瓣长度	花瓣宽度	类别
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
5	5.4	3.9	1.7	0.4	0

图5-5 iris数据集

```
h = .02  #通过步进值产生大量被预测对象
#一共有三种类型的iris，因而需要创建三种颜色
cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
cmap_bold = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])
for weights in ['uniform', 'distance']:
```

这两种weights(uniform和distance)，分别对应前面讲解的“少数服从多数”和“加权投票”两种机制。

```
    #紧接着利用scikit-learn提供的KNeighborsClassifier来创建K-NN分类器
    clf = neighbors.KNeighborsClassifier(n_neighbors, weights=weights)
    ##将前面的iris数据与分类器建立关联
    clf.fit(X, y)
    ##结合步进值，产生大量的网格点
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))

    ##利用K-NN分类器对这些网格点执行预测
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    #将预测结果保存下来
    Z = Z.reshape(xx.shape)
    plt.figure()
```

```
    ##根据预测结果，分配不同的颜色值，从而形成decision boundary
    plt.pcolormesh(xx, yy, Z, cmap=cmap_light)
    #绘制所有数据样本
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap_bold,
                edgecolor='k', s=20)
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
     plt.title("3-Class classification (k = %i, weights = '%s')" % (n_neighbors,
weights))
plt.show()
```

最终结果如图5-6所示。

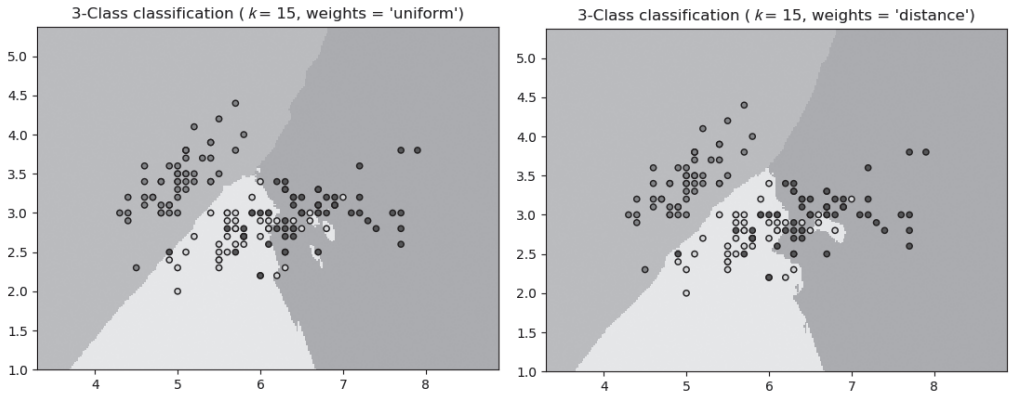

图5-6　基于K-NN算法的工程范例

读者可以仔细比较一下K-NN采用两种不同weights时在分类表现上的异同点。