Deel Learning 500 Questions

July 13, 2019

Contents

Chapter 1

数学基础

Chapter 2

机器学习基础

机器学习起源于上世纪 50 年代，1959 年在 IBM 工作的 Arthur Samuel 设计了一个下棋程序，这个程序具有学习的能力，它可以在不断的对弈中提高自己。由此提出了“机器学习”这个概念，它是一个结合了多个学科如概率论，优化理论，统计等，最终在计算机上实现自我获取新知识，学习改善自己的这样一个研究领域。机器学习是人工智能的一个子集，目前已经发展出许多有用的方法，比如支持向量机，回归，决策树，随机森林，强化方法，集成学习，深度学习等等，一定程度上可以帮助人们完成一些数据预测，自动化，自动决策，最优化等初步替代脑力的任务。本章我们主要介绍下机器学习的基本概念、监督学习、分类算法、逻辑回归、代价函数、损失函数、LDA、PCA、决策树、支持向量机、EM 算法、聚类和降维以及模型评估有哪些方法、指标等等。

2.1 基本概念

2.1.1 大话理解机器学习本质

机器学习 (Machine Learning, ML)，顾名思义，让机器去学习。这里，机器指的是计算机，是算法运行的物理载体，你也可以把各种算法本身当做一个有输入和输出的机器。那么到底让计算机去学习什么呢？对于一个任务及其表现的度量方法，设计一种算法，让算法能够提取中数据所蕴含的规律，这就叫机器学习。如果输入机器的数据是带有标签的，就称作有监督学习。如果数据是无标签的，就是无监督学习。

2.1.2 什么是神经网络

神经网络就是按照一定规则将多个神经元连接起来的网络。不同的神经网络，具有不同的连接规则。例如全连接 (Full Connected, FC) 神经网络，它的规则包括：

1. 有三种层：输入层，输出层，隐藏层。
2. 同一层的神经元之间没有连接。
3. fully connected 的含义：第 N 层的每个神经元和第 N-1 层的所有神经元相连，第 N-1 层神经元的输出就是第 N 层神经元的输入。
4. 每个连接都有一个权值。

神经网络架构 图2.1就是一个神经网络系统，它由很多层组成。输入层负责接收信息，比如一只猫的图片。输出层是计算机对这个输入信息的判断结果，它是不是猫。隐藏层就是对输入信息的传递和加工处理。

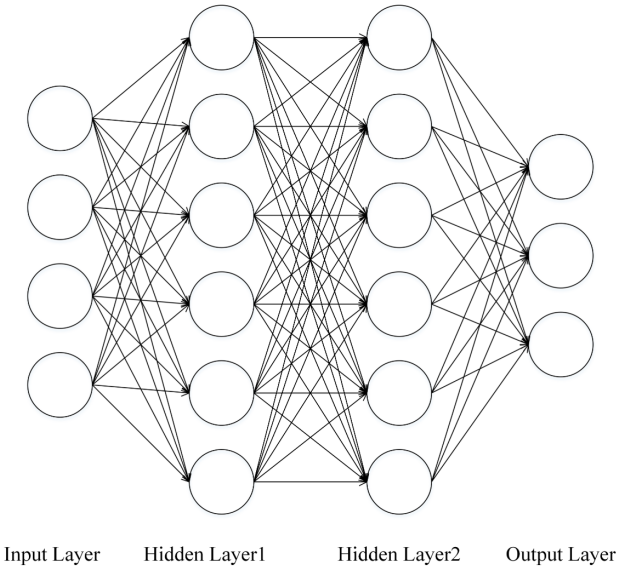
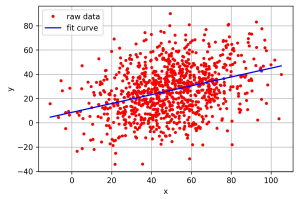


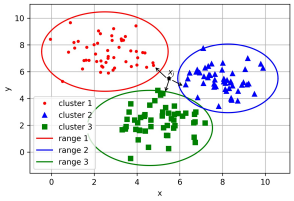
Figure 2.1: 神经网络系统

2.1.3 各种常见算法图示

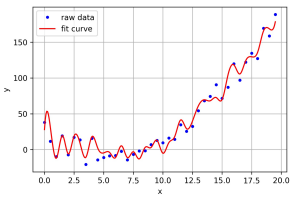
日常使用机器学习的任务中，我们经常会遇见各种算法，2.3是各种常见算法的图示。



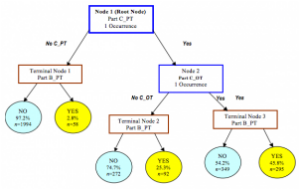
(a) 回归算法



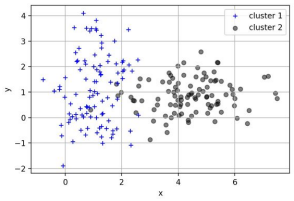
(b) 聚类算法



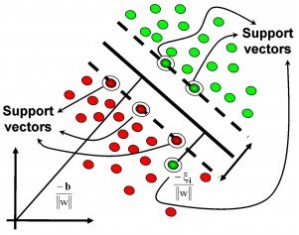
(c) 正则化算法



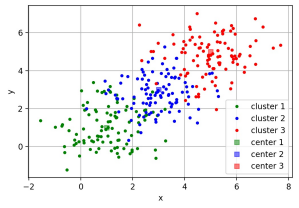
(d) 决策树学习



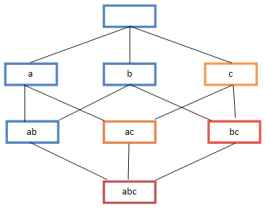
(e) 贝叶斯方法



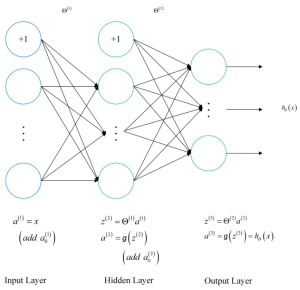
(f) 基于核的算法



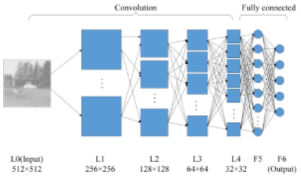
(g) 聚类算法



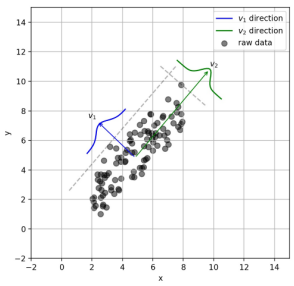
(h) 关联规则学习



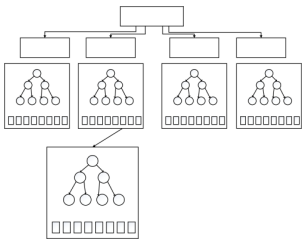
(i) 人工神经网络



(j) 深度学习



(k) 降低维度算法



(l) 集成算法

Figure 2.2: 各种常见算法图示

2.1.4 计算图的导数计算

计算图导数计算是反向传播，利用链式法则和隐式函数求导。

假设 $z = f(u, v)$ 在点 (u, v) 处偏导连续，(u, v) 是关于 t 的函数，在 t 点可导，求 z 在 t 点的导数。

根据链式法则有

$$\frac{dz}{dt} = \frac{\partial z}{\partial u} \cdot \frac{du}{dt} + \frac{\partial z}{\partial v} \cdot \frac{dv}{dt}$$

链式法则用文字描述：“由两个函数凑起来的复合函数，其导数等于里边函数代入外边函数的值之导数，乘以里边函数的导数”。

为了便于理解，下面举例说明：

$$f(x) = x^2, g(x) = 2x + 1$$

则:

$$f[g(x)]' = 2[g(x)] \times g'(x) = 2[2x + 1] \times 2 = 8x + 4$$

2.1.5 理解局部最优与全局最优

笑谈局部最优和全局最优

> 柏拉图有一天问老师苏格拉底什么是爱情？苏格拉底叫他到麦田走一次，摘一颗最大的麦穗回来，不许回头，只可摘一次。柏拉图空着手出来了，他的理由是，看见不错的，却不知道是不是最好的，一次次侥幸，走到尽头时，才发现还不如前面的，于是放弃。苏格拉底告诉他：“这就是爱情。” 这故事让我们明白了一个道理，因为生命的一些不确定性，所以全局最优解是很难寻找到的，或者说根本就不存在，我们应该设置一些限定条件，然后在这个范围内寻找最优解，也就是局部最优解 — 有所斩获总比空手而归强，哪怕这种斩获只是一次有趣的经历。
>
> 柏拉图有一天又问什么是婚姻？苏格拉底叫他到树林走一次, 选一棵最好的树做圣诞树，也是不许回头，只许选一次。这次他一身疲惫地拖了一棵看起来直挺、翠绿，却有点稀疏的杉树回来，他的理由是，有了上回的教训，好不容易看见一棵看似不错的，又发现时间、体力已经快不够用了，也不管是不是最好的，就拿回来了。苏格拉底告诉他：“这就是婚姻。”

优化问题一般分为局部最优和全局最优。其中，

1. 局部最优，就是在函数值空间的一个有限区域内寻找最小值；而全局最优，是在函数值空间整个区域寻找最小值问题。

2. 函数局部最小点是它的函数值小于或等于附近点的点，但是有可能大于较远距离的点。

3. 全局最小点是那种它的函数值小于或等于所有的可行点。

2.1.6 大数据与深度学习之间的关系

首先来看大数据、机器学习及数据挖掘三者简单的定义：

大数据 通常被定义为“超出常用软件工具捕获，管理和处理能力”的数据集。**机器学习** 关心的问题是如何构建计算机程序使用经验自动改进。**数据挖掘** 是从数据中提取模式的特定算法的应用，在数据挖掘中，重点在于算法的应用，而不是算法本身。

机器学习和数据挖掘 之间的关系如下：

数据挖掘是一个过程，在此过程中机器学习算法被用作提取数据集中的潜在有价值模式的工具。

大数据与深度学习关系总结如下：

1. 深度学习是一种模拟大脑的行为。可以从所学习对象的机制以及行为等等很多相关联的方面进行学习，模仿类型行为以及思维。

2. 深度学习对于大数据的发展有帮助。深度学习对于大数据技术开发的每一个阶段均有帮助，不管是数据的分析还是挖掘还是建模，只有深度学习，这些工作才会有可能一一得到实现。

3. 深度学习转变了解决问题的思维。很多时候发现问题到解决问题，走一步看一步不是一个主要的解决问题的方式了，在深度学习的基础上，要求我们从开始到最后都要基于一个目标，为了需要优化的那个最终目标去进行处理数据以及将数据放入到数据应用平台上去，这就是端到端（End to End）。

4. 大数据的深度学习需要一个框架。在大数据方面的深度学习都是从基础的角度出发的，深度学习需要一个框架或者一个系统。总而言之，将你的大数据通过深度分析变为现实，这就是深度学习和大数据的最直接关系。

2.2 机器学习学习方式

根据数据类型的不同，对一个问题的建模有不同的方式。依据不同的学习方式和输入数据，机器学习主要分为以下四种学习方式。

2.2.1 监督学习

特点：监督学习是使用已知正确答案的示例来训练网络。已知数据和其一一对应的标签，训练一个预测模型，将输入数据映射到标签的过程。

常见应用场景：监督式学习的常见应用场景如分类问题和回归问题。

算法举例：常见的有监督机器学习算法包括支持向量机 (Support Vector Machine, SVM)，朴素贝叶斯 (Naive Bayes)，逻辑回归 (Logistic Regression)，K 近邻 (K-Nearest Neighborhood, KNN)，决策树 (Decision Tree)，随机森林 (Random Forest)，AdaBoost 以及线性判别分析 (Linear Discriminant Analysis, LDA) 等。深度学习 (Deep Learning) 也是大多数以监督学习的方式呈现。

2.2.2 非监督式学习

定义：在非监督式学习中，数据并不被特别标识，适用于你具有数据集但无标签的情况。学习模型是为了推断出数据的一些内在结构。

常见应用场景：常见的应用场景包括关联规则的学习以及聚类等。

算法举例：常见算法包括 Apriori 算法以及 k-Means 算法。

2.2.3 半监督式学习

特点：在此学习方式下，输入数据部分被标记，部分没有被标记，这种学习模型可以用来进行预测。

常见应用场景：应用场景包括分类和回归，算法包括一些对常用监督式学习算法的延伸，通过对已标记数据建模，在此基础上，对未标记数据进行预测。

算法举例：常见算法如图论推理算法（Graph Inference）或者拉普拉斯支持向量机（Laplacian SVM）等。

2.2.4 弱监督学习

特点：弱监督学习可以看做是有多个标记的数据集合，次集合可以是空集，单个元素，或包含多种情况（没有标记，有一个标记，和有多个标记）的多个元

素。数据集的标签是不可靠的，这里的不可靠可以是标记不正确，多种标记，标记不充分，局部标记等。已知数据和其一一对应的弱标签，训练一个智能算法，将输入数据映射到一组更强的标签的过程。标签的强弱指的是标签蕴含的信息量的多少，比如相对于分割的标签来说，分类的标签就是弱标签。

算法举例：举例，给出一张包含气球的图片，需要得出气球在图片中的位置及气球和背景的分割线，这就是已知弱标签学习强标签的问题。

在企业数据应用的场景下，人们最常用的可能就是监督式学习和非监督式学习的模型。在图像识别等领域，由于存在大量的非标识的数据和少量的可标识数据，目前半监督式学习是一个很热的话题。

2.2.5 监督学习有哪些步骤

监督学习是使用已知正确答案的示例来训练网络，每组训练数据有一个明确的标识或结果。想象一下，我们可以训练一个网络，让其从照片库中（其中包含气球的照片）识别出气球的照片。以下就是我们在这个假设场景中所要采取的步骤。

- **步骤 1：数据集的创建和分类** 首先，浏览你的照片（数据集），确定所有包含气球的照片，并对其进行标注。然后，将所有照片分为训练集和验证集。目标就是在深度网络中找一函数，这个函数输入是任意一张照片，当照片中包含气球时，输出 1，否则输出 0。
- 步骤 2：数据增强（Data Augmentation）当原始数据搜集和标注完毕，一般搜集的数据并不一定包含目标在各种扰动下的信息。数据的好坏对于机器学习模型的预测能力至关重要，因此一般会进行数据增强。对于图像数据来说，数据增强一般包括，图像旋转，平移，颜色变换，裁剪，仿射变换等。
- 步骤 3：特征工程（Feature Engineering）一般来讲，特征工程包含特征提取和特征选择。常见的手工特征 (Hand-Crafted Feature) 有尺度不变特征变换 (Scale-Invariant Feature Transform, SIFT)，方向梯度直方图 (Histogram of Oriented Gradient, HOG) 等。由于手工特征是启发式的，其算法设计背后的出发点不同，将这些特征组合在一起的时候有可能会产生冲突，如何将组合特征的效能发挥出来，使原始数据在特征空间中的判别性最大化，就需要用到特征选择的方法。在深度学习方法大获成功之后，人们很大一部分不再关注特征工程本身。因为，最常用到的卷积神经网络 (Convolutional Neural Networks, CNNs) 本身就是一种特征提取和选择的引擎。研究者提出的不同的网络结构、正则化、归一化方法实际上就是深度学习背景下的特征工程。
- 步骤 4：构建预测模型和损失将原始数据映射到特征空间之后，也就意味着我们得到了比较合理的输入。下一步就是构建合适的预测模型得到

对应输入的输出。而如何保证模型的输出和输入标签之间的一致性，就需要构建模型预测和标签之间的损失函数，常见的损失函数 (Loss Function) 有交叉熵、均方差等。通过优化方法不断迭代，使模型从最初的初始化状态一步步变化为有预测能力的模型的过程，实际上就是学习的过程。

- 步骤 5：训练选择合适的模型和超参数进行初始化，其中超参数比如支持向量机中核函数、误差项惩罚权重等。当模型初始化参数设定好后，将制作好的特征数据输入到模型，通过合适的优化方法不断缩小输出与标签之间的差距，当迭代过程到了截止条件，就可以得到训练好的模型。优化方法最常见的就是梯度下降法及其变种，使用梯度下降法的前提是优化目标函数对于模型是可导的。
- 步骤 6：验证和模型选择训练完训练集图片后，需要进行模型测试。利用验证集来验证模型是否可以准确地挑选出含有气球在内的照片。在此过程中，通常会通过调整和模型相关的各种事物（超参数）来重复步骤 2 和 3，诸如里面有多少个节点，有多少层，使用怎样的激活函数和损失函数，如何在反向传播阶段积极有效地训练权值等等。
- 步骤 7：测试及应用当有了一个准确的模型，就可以将该模型部署到你的应用程序中。你可以将预测功能发布为 API（Application Programming Interface, 应用程序编程接口）调用，并且你可以从软件中调用该 API，从而进行推理并给出相应的结果。

2.3 分类算法

分类算法和回归算法是对真实世界不同建模的方法。分类模型是认为模型的输出是离散的，例如大自然的生物被划分为不同的种类，是离散的。回归模型的输出是连续的，例如人的身高变化过程是一个连续过程，而不是离散的。

因此，在实际建模过程时，采用分类模型还是回归模型，取决于你对任务（真实世界）的分析和理解。

2.3.1 常用分类算法的优缺点？

接下来我们介绍常用分类算法的优缺点，如表2.1所示。

算法	优点	缺点
Bayes 贝叶斯分类法	1. 所需估计的参数少，对于缺失数据不敏感。 2. 有着坚实的数学基础，以及稳定的分类效率。	1. 需要假设属性之间相互独立，这往往并不成立。（喜欢吃番茄、鸡蛋，却不喜欢吃番茄炒蛋）。 2. 需要知道先验概率。 3. 分类决策存在错误率。
Decision Tree 决策树	1. 不需要任何领域知识或参数假设。 2. 适合高维数据。 3. 简单易于理解。 4. 短时间内处理大量数据，得到可行且效果较好的结果。 5. 能够同时处理数据型和常规性属性。	1. 对于各类别样本数量不一致数据，信息增益偏向于那些具有更多数值的特征。 2. 易于过拟合。 3. 忽略属性之间的相关性。 4. 不支持在线学习。
SVM 支持向量机	1. 可以解决小样本下机器学习的问题。 2. 提高泛化性能。 3. 可以解决高维、非线性问题。超高维文本分类仍受欢迎。 4. 避免神经网络结构选择和局部极小的问题。	1. 对缺失数据敏感。 2. 内存消耗大，难以解释。 3. 运行和调参略烦人。

算法	优点	缺点
KNN K 近邻	1. 思想简单，理论成熟，既可以用来做分类也可以用来做回归； 2. 可用于非线性分类； 3. 训练时间复杂度为 O(n)； 4. 准确度高，对数据没有假设，对 outlier 不敏感；	1. 计算量太大。 2. 对于样本分类不均衡的问题，会产生误判。 3. 需要大量的内存。 4. 输出的可解释性不强。
Logistic Regression 逻辑回归	1. 速度快。 2. 简单易于理解，直接看到各个特征的权重。 3. 能容易地更新模型吸收新的数据。 4. 如果想要一个概率框架，动态调整分类阀值。	1. 特征处理复杂。需要归一化和较多的特征工程。
Neural Network 神经网络	1. 分类准确率高。 2. 并行处理能力强。 3. 分布式存储和学习能力强。 4. 鲁棒性较强，不易受噪声影响。	1. 需要大量参数（网络拓扑、阀值、阈值）。 2. 结果难以解释。 3. 训练时间过长。

算法	优点	缺点
Adaboosting	1. adaboost 是一种有很高精度的分类器。 2. 可以使用各种方法构建子分类器，Adaboost 算法提供的是框架。 3. 当使用简单分类器时，计算出的结果是可以理解的。而且弱分类器构造极其简单。 4. 简单，不用做特征筛选。不用担心 overfitting。	1. 对 outlier 比较敏感

Table 2.1: 常用分类算法的优缺点

2.3.2 分类算法的评估方法

分类评估方法主要功能是用来评估分类算法的好坏，而评估一个分类器算法的好坏又包括许多项指标。了解各种评估方法，在实际应用中选择正确的评估方法是十分重要的。

- 几个常用术语这里首先介绍几个常见的模型评价术语，现在假设我们的分类目标只有两类，计为正例（positive）和负例（negative）分别是：
  - True positives(TP): 被正确地划分为正例的个数，即实际为正例且被分类器划分为正例的实例数；
  - False positives(FP): 被错误地划分为正例的个数，即实际为负例但被分类器划分为正例的实例数；
  - False negatives(FN): 被错误地划分为负例的个数，即实际为正例但被分类器划分为负例的实例数；
  - True negatives(TN): 被正确地划分为负例的个数，即实际为负例且被分类器划分为负例的实例数。

  2.3 是这四个术语的混淆矩阵，做以下说明：

实际类别	预测类别			
		Yes	No	总计
	Yes	TP	FN	P（实际为 Yes）
	No	FP	TN	N（实际为 No）
	总计	P'（被分为 Yes）	N'（被分为 No）	P+N

Figure 2.3: 四个术语的混淆矩阵

1. $P = TP + FN$ 表示实际为正例的样本个数。
2. $True$、$False$ 描述的是分类器是否判断正确。
3. $Positive$、$Negative$ 是分类器的分类结果，如果正例计为 1、负例计为 -1，即 $positive = 1$、$negative = -1$。用 1 表示 $True$，-1 表示 $False$，那么实际的类标 $= TF * PN$，TF 为 $true$ 或 $false$，PN 为 $positive$ 或 $negative$。
4. 例如 True positives(TP) 的实际类标 $= 1*1 = 1$ 为正例，False positives(FP) 的实际类标 $= (-1)*1 = -1$ 为负例，False negatives(FN) 的实际类标 $= (-1)*(-1) = 1$ 为正例，True negatives(TN) 的实际类标 $= 1*(-1) = -1$ 为负例。

- 评价指标
  1. 正确率（accuracy）正确率是我们最常见的评价指标，accuracy = (TP+TN)/(P+N)，正确率是被分对的样本数在所有样本数中的占比，通常来说，正确率越高，分类器越好。
- 错误率（error rate) 错误率则与正确率相反，描述被分类器错分的比例，error rate = (FP+FN)/(P+N)，对某一个实例来说，分对与分错是互斥事件，所以 accuracy =1 - error rate。
- 灵敏度（sensitivity）$sensitivity = TP/P$，表示的是所有正例中被分对的比例，衡量了分类器对正例的识别能力。
- 特异性（specificity) $specificity = TN/N$，表示的是所有负例中被分对的比例，衡量了分类器对负例的识别能力。
- 精度（precision）$precision = TP/(TP + FP)$，精度是精确性的度量，表示被分为正例的示例中实际为正例的比例。
- 召回率（recall）召回率是覆盖面的度量，度量有多个正例被分为正例，$recall = TP/(TP + FN) = TP/P = sensitivity$，可以看到召回率与灵敏度是一样的。
- 其他评价指标

1. 计算速度：分类器训练和预测需要的时间；
2. 鲁棒性：处理缺失值和异常值的能力；
3. 可扩展性：处理大数据集的能力；
4. 可解释性：分类器的预测标准的可理解性，像决策树产生的规则就是很容易理解的，而神经网络的一堆参数就不好理解，我们只好把它看成一个黑盒子。

- 精度和召回率反映了分类器分类性能的两个方面。如果综合考虑查准率与查全率，可以得到新的评价指标 $F1-score$，也称为综合分类率：$F1=\frac{2\times precision\times recall}{precision+recall}$。

  为了综合多个类别的分类情况，评测系统整体性能，经常采用的还有微平均 $F1(micro-averaging)$ 和宏平均 $F1(macro-averaging)$ 两种指标。

  1. 宏平均 $F1$ 与微平均 $F1$ 是以两种不同的平均方式求的全局 $F1$ 指标。
  2. 宏平均 $F1$ 的计算方法先对每个类别单独计算 $F1$ 值，再取这些 $F1$ 值的算术平均值作为全局指标。
  3. 微平均 $F1$ 的计算方法是先累加计算各个类别的 a,b,c,d 的值，再由这些值求出 $F1$ 值。
  4. 由两种平均 $F1$ 的计算方式不难看出，宏平均 $F1$ 平等对待每一个类别，所以它的值主要受到稀有类别的影响，而微平均 $F1$ 平等考虑文档集中的每一个文档，所以它的值受到常见类别的影响比较大。

- ROC 曲线和 PR 曲线

  如图 2.4，ROC 曲线是（Receiver Operating Characteristic Curve，受试者工作特征曲线）的简称，是以灵敏度（真阳性率）为纵坐标，以 1 减去特异性（假阳性率）为横坐标绘制的性能评价曲线。可以将不同模型对同一数据集的 ROC 曲线绘制在同一笛卡尔坐标系中，ROC 曲线越靠近左上角，说明其对应模型越可靠。也可以通过 ROC 曲线下面的面积（Area Under Curve, AUC）来评价模型，AUC 越大，模型越可靠。

  PR 曲线是 Precision Recall Curve 的简称，描述的是 precision 和 recall 之间的关系，以 recall 为横坐标，precision 为纵坐标绘制的曲线。该曲线的所对应的面积 AUC 实际上是目标检测中常用的评价指标平均精度（Average Precision, AP）。AP 越高，说明模型性能越好。

2.3.3 正确率能很好的评估分类算法吗

不同算法有不同特点，在不同数据集上有不同的表现效果，根据特定的任务选择不同的算法。如何评价分类算法的好坏，要做具体任务具体分析。对于决策

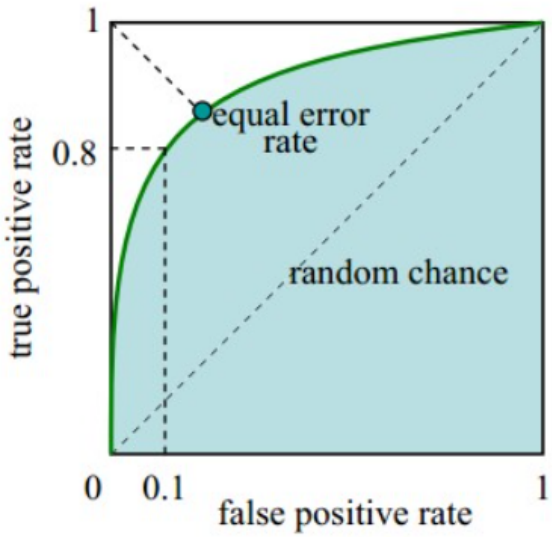


Figure 2.4: ROC 曲线

树，主要用正确率去评估，但是其他算法，只用正确率能很好的评估吗？答案是否定的。正确率确实是一个很直观很好的评价指标，但是有时候正确率高并不能完全代表一个算法就好。比如对某个地区进行地震预测，地震分类属性分为 0：不发生地震、1 发生地震。我们都知道，不发生的概率是极大的，对于分类器而言，如果分类器不加思考，对每一个测试样例的类别都划分为 0，达到 99% 的正确率，但是，问题来了，如果真的发生地震时，这个分类器毫无察觉，那带来的后果将是巨大的。很显然，99% 正确率的分类器并不是我们想要的。出现这种现象的原因主要是数据分布不均衡，类别为 1 的数据太少，错分了类别 1 但达到了很高的正确率缺忽视了研究者本身最为关注的情况。

2.3.4 什么样的分类器是最好的

对某一个任务，某个具体的分类器不可能同时满足或提高所有上面介绍的指标。如果一个分类器能正确分对所有的实例，那么各项指标都已经达到最优，但这样的分类器往往不存在。比如之前说的地震预测，既然不能百分百预测地震的发生，但实际情况中能容忍一定程度的误报。假设在 1000 次预测中，共有 5 次预测发生了地震，真实情况中有一次发生了地震，其他 4 次则为误报。正确率由原来的 $999/1000 = 99.9$ 下降为 $996/1000 = 99.6$。召回率由 $0/1 = 0\%$ 上升为 $1/1 = 100\%$。对此解释为，虽然预测失误了 4 次，但真的地震发生前，分类器能预测对，没有错过，这样的分类器实际意义更为重大，正是我们想要的。在这种情况下，在一定正确率前提下，要求分类器的召回率尽量高。

2.4 逻辑回归

2.4.1 回归划分

广义线性模型家族里，依据因变量不同，可以有如下划分：

1. 如果是连续的，就是多重线性回归。
2. 如果是二项分布，就是逻辑回归。
3. 如果是泊松（Poisson）分布，就是泊松回归。
4. 如果是负二项分布，就是负二项回归。
5. 逻辑回归的因变量可以是二分类的，也可以是多分类的，但是二分类的更为常用，也更加容易解释。所以实际中最常用的就是二分类的逻辑回归。

2.4.2 逻辑回归适用性

逻辑回归可用于以下几个方面：

1. 用于概率预测。用于可能性预测时，得到的结果有可比性。比如根据模型进而预测在不同的自变量情况下，发生某病或某种情况的概率有多大。
2. 用于分类。实际上跟预测有些类似，也是根据模型，判断某人属于某病或属于某种情况的概率有多大，也就是看一下这个人有多大的可能性是属于某病。进行分类时，仅需要设定一个阈值即可，可能性高于阈值是一类，低于阈值是另一类。
3. 寻找危险因素。寻找某一疾病的危险因素等。
4. 仅能用于线性问题。只有当目标和特征是线性关系时，才能用逻辑回归。在应用逻辑回归时注意两点：一是当知道模型是非线性时，不适用逻辑回归；二是当使用逻辑回归时，应注意选择和目标为线性关系的特征。
5. 各特征之间不需要满足条件独立假设，但各个特征的贡献独立计算。

2.4.3 逻辑回归与朴素贝叶斯有什么区别

逻辑回归与朴素贝叶斯区别有以下几个方面：

1. 逻辑回归是判别模型，朴素贝叶斯是生成模型，所以生成和判别的所有区别它们都有。
2. 朴素贝叶斯属于贝叶斯，逻辑回归是最大似然，两种概率哲学间的区别。
3. 朴素贝叶斯需要条件独立假设。
4. 逻辑回归需要求特征参数间是线性的。

2.4.4 线性回归与逻辑回归的区别

线性回归与逻辑回归的区别如下描述：

1. 线性回归的样本的输出，都是连续值，$y \in (-\infty, +\infty)$，而逻辑回归中 $y \in (0, 1)$，只能取 0 和 1。
2. 对于拟合函数也有本质上的差别：
   - 线性回归：$f(x) = \theta^T x = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$
   - 逻辑回归：$f(x) = P(y = 1|x; \theta) = g(\theta^T x)$，其中，$g(z) = \frac{1}{1+e^{-z}}$

可以看出，线性回归的拟合函数，是对 $f(x)$ 的输出变量 y 的拟合，而逻辑回归的拟合函数是对为 1 类样本的概率的拟合。

那么，为什么要以 1 类样本的概率进行拟合呢，为什么可以这样拟合呢？

$\theta^T x = 0$ 就相当于是 1 类和 0 类的决策边界：

当 $\theta^T x > 0$，则 $y > 0.5$；若 $\theta^T x \to +\infty$，则 $y \to 1$，即 y 为 1 类;

当 $\theta^T x < 0$，则 $y < 0.5$；若 $\theta^T x \to -\infty$，则 $y \to 0$，即 y 为 0 类;

这个时候就能看出区别，在线性回归中 $\theta^T x$ 为预测值的拟合函数；而在逻辑回归中 $\theta^T x$ 为决策边界。下表2.2为线性回归和逻辑回归的区别。

	线性回归	逻辑回归
目的	预测	分类
$y^{(i)}$	未知	$(0, 1)$
函数	拟合函数	预测函数
参数计算方式	最小二乘法	极大似然估计

Table 2.2: 线性回归和逻辑回归的区别

下面具体解释一下：

1. 拟合函数和预测函数什么关系呢？简单来说就是将拟合函数做了一个逻辑函数的转换，转换后使得 $y^{(i)} \in (0, 1)$;
2. 最小二乘和最大似然估计可以相互替代吗？回答当然是不行了。我们来看看两者依仗的原理：最大似然估计是计算使得数据出现的可能性最大的参数，依仗的自然是 Probability。而最小二乘是计算误差损失。

2.5 代价函数

2.5.1 为什么需要代价函数

1. 为了得到训练逻辑回归模型的参数，需要一个代价函数，通过训练代价函数来得到参数。
2. 用于找到最优解的目的函数。

2.5.2 代价函数作用原理

在回归问题中，通过代价函数来求解最优解，常用的是平方误差代价函数。假设函数图像如图2.5所示，当参数发生变化时，假设函数状态也会随着变化。

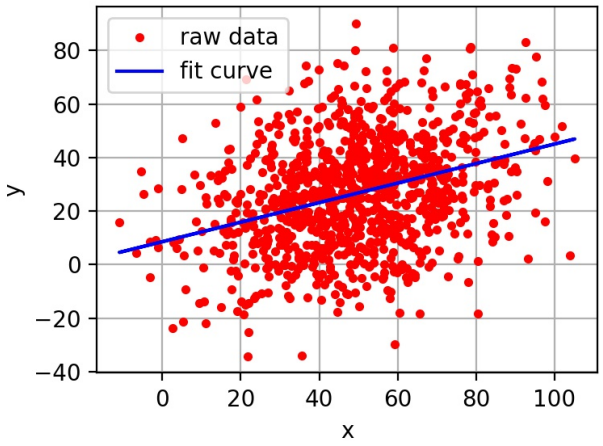


Figure 2.5: $h(x) = A + Bx$ 函数示意图

想要拟合图中的离散点，我们需要尽可能找到最优的 A 和 B 来使这条直线更能代表所有数据。如何找到最优解呢，这就需要使用代价函数来求解，以平方误差代价函数为例，假设函数为 $h(x) = \theta_0 x$。**平方误差代价函数的主要思想**就

是将实际数据给出的值与拟合出的线的对应值做差，求出拟合出的直线与实际的差距。在实际应用中，为了避免因个别极端数据产生的影响，采用类似方差再取二分之一的方式来减小个别数据的影响。因此，引出代价函数：

$$J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)})^2$$

最优解即为代价函数的最小值 $\min J(\theta_0, \theta_1)$。如果是 1 个参数，代价函数一般通过二维曲线便可直观看出。如果是 2 个参数，代价函数通过三维图像可看出效果，参数越多，越复杂。当参数为 2 个时，代价函数是三维图像，如下图2.6所示。

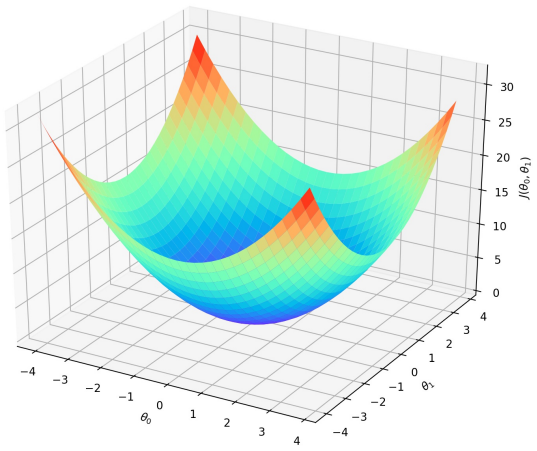


Figure 2.6: 代价函数三维图像

2.5.3 为什么代价函数要非负

目标函数存在一个下界，在优化过程当中，如果优化算法能够使目标函数不断减小，根据单调有界准则，这个优化算法就能证明是收敛有效的。只要设计的目标函数有下界，基本上都可以，代价函数非负更为方便。

2.5.4 常见代价函数

1. 二次代价函数（quadratic cost）：

$$J = \frac{1}{2n} \sum_{x} \|y(x) - a^L(x)\|^2$$

其中，J 表示代价函数，x 表示样本，y 表示实际值，a 表示输出值，n 表示样本的总数。使用一个样本为例简单说明，此时二次代价函数为：

$$J = \frac{(y-a)^2}{2}$$

假如使用梯度下降法（Gradient descent）来调整权值参数的大小，权值 w 和偏置 b 的梯度推导如下：

$$\frac{\partial J}{\partial b} = (a-y)\sigma'(z)$$

其中，z 表示神经元的输入，σ 表示激活函数。权值 w 和偏置 b 的梯度跟激活函数的梯度成正比，激活函数的梯度越大，权值 w 和偏置 b 的大小调整得越快，训练收敛得就越快。

注：神经网络常用的激活函数为 sigmoid 函数，该函数的曲线如下图2.7所示：

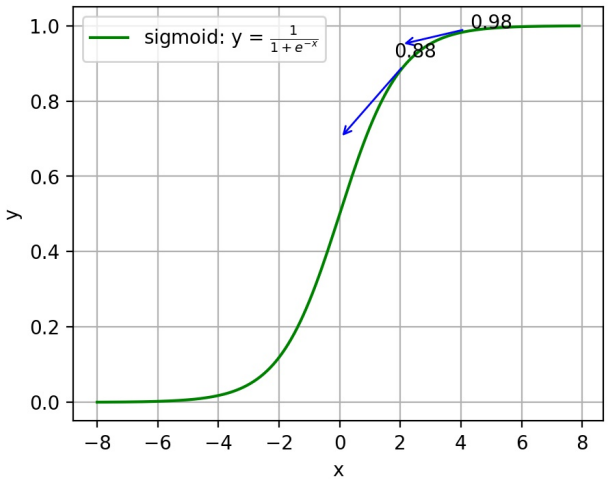


Figure 2.7: sigmoid 函数曲线

如上图所示，对 0.88 和 0.98 两个点进行比较：假设目标是收敛到 1.0。0.88 离目标 1.0 比较远，梯度比较大，权值调整比较大。0.98 离目标 1.0 比较近，梯度比较小，权值调整比较小。调整方案合理。假如目标是收敛到 0。0.88 离目标 0 比较近，梯度比较大，权值调整比较大。0.98 离目标 0 比较远，梯度比较小，权值调整比较小。调整方案不合理。

原因：在使用 sigmoid 函数的情况下, 初始的代价（误差）越大，导致训练越慢。

2. 交叉熵代价函数（cross-entropy）：

$$J = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)]$$

其中，J 表示代价函数，x 表示样本，y 表示实际值，a 表示输出值，n 表示样本的总数。权值 w 和偏置 b 的梯度推导如下：

$$\frac{\partial J}{\partial w_j} = \frac{1}{n}\sum_x x_j(\sigma(z)-y)\,, \frac{\partial J}{\partial b} = \frac{1}{n}\sum_x (\sigma(z)-y)$$

当误差越大时，梯度就越大，权值 w 和偏置 b 调整就越快，训练的速度也就越快。**二次代价函数适合输出神经元是线性的情况，交叉熵代价函数适合输出神经元是 S 型函数的情况。**

3. 对数似然代价函数（log-likelihood cost）：

对数似然函数常用来作为 softmax 回归的代价函数。深度学习中普遍的做法是将 softmax 作为最后一层，此时常用的代价函数是对数似然代价函数。对数似然代价函数与 softmax 的组合和交叉熵与 sigmoid 函数的组合非常相似。对数似然代价函数在二分类时可以化简为交叉熵代价函数的形式。在 tensorflow 中，与 sigmoid 搭配使用的交叉熵函数

```
tf.nn.sigmoid_cross_entropy_with_logits()
```

与 softmax 搭配使用的交叉熵函数

```
tf.nn.softmax_cross_entropy_with_logits()
```

在 pytorch 中：与 sigmoid 搭配使用的交叉熵函数

```
torch.nn.BCEWithLogitsLoss()
```

与 softmax 搭配使用的交叉熵函数

```
torch.nn.CrossEntropyLoss()
```

2.5.5 为什么用交叉熵代替二次代价函数

1. 为什么不用二次方代价函数？由上一节可知，权值 w 和偏置 b 的偏导数为 $\frac{\partial J}{\partial w} = (a-y)\sigma'(z)x$，$\frac{\partial J}{\partial b} = (a-y)\sigma'(z)$，偏导数受激活函数的导数影响，sigmoid 函数导数在输出接近 0 和 1 时非常小，会导致一些实例在刚开始训练时学习得非常慢。

2. 为什么要用交叉熵？交叉熵函数权值 w 和偏置 b 的梯度推导为：

$$\frac{\partial J}{\partial w_j} = \frac{1}{n}\sum_x x_j(\sigma(z)-y) \quad \frac{\partial J}{\partial b} = \frac{1}{n}\sum_x (\sigma(z)-y)$$

   由以上公式可知，权重学习的速度受到 $\sigma(z)-y$ 影响，更大的误差，就有更快的学习速度，避免了二次代价函数方程中因 $\sigma'(z)$ 导致的学习缓慢的情况。

2.6 损失函数

2.6.1 什么是损失函数

损失函数（Loss Function）又叫做误差函数，用来衡量算法的运行情况，估量模型的预测值与真实值的不一致程度，是一个非负实值函数，通常使用 $L(Y, f(x))$ 来表示。损失函数越小，模型的鲁棒性就越好。损失函数是经验风险函数的核心部分，也是结构风险函数重要组成部分。

2.6.2 常见的损失函数

机器学习通过对算法中的目标函数进行不断求解优化，得到最终想要的结果。分类和回归问题中，通常使用损失函数或代价函数作为目标函数。损失函数用来评价预测值和真实值不一样的程度。通常损失函数越好，模型的性能也越好。损失函数可分为经验风险损失函数和结构风险损失函数。经验风险损失函数指预测结果和实际结果的差别，结构风险损失函数是在经验风险损失函数上加上正则项。

下面介绍常用的损失函数：

1. $0-1$ 损失函数

   如果预测值和目标值相等，值为 0，如果不相等，值为 1。

$$L(Y, f(x)) = \begin{cases} 1, & Y \neq f(x) \\ 0, & Y = f(x) \end{cases}$$

一般的在实际使用中，相等的条件过于严格，可适当放宽条件：

$$L(Y, f(x)) = \begin{cases} 1, & |Y - f(x)| \geqslant T \\ 0, & |Y - f(x)| < T \end{cases}$$

2. 绝对值损失函数

   和 $0-1$ 损失函数相似，绝对值损失函数表示为：

$$L(Y, f(x)) = |Y - f(x)|$$

3. 平方损失函数

$$L(Y, f(x)) = \sum_{N} (Y - f(x))^2$$

   这点可从最小二乘法和欧几里得距离角度理解。最小二乘法的原理是，最优拟合曲线应该使所有点到回归直线的距离和最小。

4. 对数损失函数

$$L(Y, P(Y|X)) = -\log P(Y|X)$$

   常见的逻辑回归使用的就是对数损失函数，有很多人认为逻辑回归的损失函数是平方损失，其实不然。逻辑回归它假设样本服从伯努利分布（$0-1$ 分布），进而求得满足该分布的似然函数，接着取对数求极值等。逻辑回归推导出的经验风险函数是最小化负的似然函数，从损失函数的角度看，就是对数损失函数。

5. 指数损失函数

   指数损失函数的标准形式为：

$$L(Y, f(x)) = \exp(-Y f(x))$$

   例如 AdaBoost 就是以指数损失函数为损失函数。

6. Hinge 损失函数

   Hinge 损失函数的标准形式如下：

$$L(y) = \max(0, 1 - ty)$$

   统一的形式：

$$L(Y, f(x)) = \max(0, Yf(x))$$

其中 y 是预测值，范围为 $(-1, 1)$，t 为目标值，其为 -1 或 1。
在线性支持向量机中，最优化问题可等价于

$$\min_{w,b} \sum_{i=1}^{N} \Big(1 - y_i(wx_i + b)\Big) + \lambda\|w\|^2$$

上式相似于下式

$$\frac{1}{m}\sum_{i=1}^{N} l\Big(wx_i + by_i\Big) + \|w\|^2$$

其中 $l(wx_i + by_i)$ 是 Hinge 损失函数，$\|w\|^2$ 可看做为正则化项。

2.6.3 逻辑回归为什么使用对数损失函数

假设逻辑回归模型

$$P(y = 1|x; \theta) = \frac{1}{1 + e^{-\theta^T x}}$$

假设逻辑回归模型的概率分布是伯努利分布，其概率质量函数为：

$$P(X = n) = \begin{cases} 1 - p, n = 0 \\ p, n = 1 \end{cases}$$

其似然函数为：

$$L(\theta) = \prod_{i=1}^{m} P\Big(y = 1|x_i\Big)^{y_i} P\Big(y = 0|x_i\Big)^{1-y_i}$$

对数似然函数为：

$$\begin{aligned} \ln L(\theta) &= \sum_{i=1}^{m} \Big[y_i \ln P(y = 1|x_i) + (1 - y_i) \ln P(y = 0|x_i)\Big] \\ &= \sum_{i=1}^{m} \Big[y_i \ln P(y = 1|x_i) + (1 - y_i) \ln(1 - P(y = 1|x_i))\Big] \end{aligned}$$

对数函数在单个数据点上的定义为：

$$cost(y, p(y|x)) = -y \ln p(y|x) - (1-y) \ln(1 - p(y|x))$$

则全局样本损失函数为：

$$cost(y, p(y|x)) = -\sum_{i=1}^{m} \left[y_i \ln p(y_i|x_i) + (1-y_i) \ln(1 - p(y_i|x_i)) \right]$$

由此可看出，对数损失函数与极大似然估计的对数似然函数本质上是相同的。所以逻辑回归直接采用对数损失函数。

2.6.4 对数损失函数是如何度量损失的

例如，在高斯分布中，我们需要确定均值和标准差。如何确定这两个参数？最大似然估计是比较常用的方法。最大似然的目标是找到一些参数值，这些参数值对应的分布可以最大化观测到数据的概率。因为需要计算观测到所有数据的全概率，即所有观测到的数据点的联合概率。现考虑如下简化情况：

1. 假设观测到每个数据点的概率和其他数据点的概率是独立的。
2. 取自然对数。
   假设观测到单个数据点 $x_i (i = 1, 2, \cdots, n)$ 的概率为：

$$P(x_i; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2} \right)$$

3. 其联合概率为：

$$\begin{aligned} P(x_1, x_2, \cdots, x_n; \mu, \sigma) &= \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x_1 - \mu)^2}{2\sigma^2} \right) \\ &\times \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x_2 - \mu)^2}{2\sigma^2} \right) \\ &\times \cdots \\ &\times \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x_n - \mu)^2}{2\sigma^2} \right) \end{aligned}$$

   对上式取自然对数，可得：

$$\begin{aligned}\ln(P(x_1, x_2, \cdots, x_n; \mu, \sigma)) &= \ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \frac{(x_1-\mu)^2}{2\sigma^2} \\ &+ \ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \frac{(x_2-\mu)^2}{2\sigma^2} \\ &+ \cdots \\ &+ \ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \frac{(x_n-\mu)^2}{2\sigma^2}\end{aligned}$$

根据对数定律，上式可以化简为：

$$\begin{aligned}\ln(P(x_1, x_2, \cdots, x_n; \mu, \sigma)) &= -n\ln(\sigma) - \frac{n}{2}\ln(2\pi) \\ &- \frac{1}{2\sigma^2}[(x_1-\mu)^2 + (x_2-\mu)^2 + \cdots + (x_n-\mu)^2]\end{aligned}$$

然后求导为：

$$\frac{\partial \ln(P(x_1, x_2, \cdots, x_n; \mu, \sigma))}{\partial \mu} = \frac{n}{\sigma^2}[\mu - (x_1 + x_2 + \cdots + x_n)]$$

上式左半部分为对数损失函数。损失函数越小越好，因此我们令等式左半的对数损失函数为 0，可得：

$$\mu = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

同理，可计算 σ。