

mygray0.9

深度学习 500 问

目录

Chapter 1. 数学基础	1
1. 向量和矩阵	1
2. 导数和偏导数	5
3. 特征值和特征向量	7
4. 概率分布与随机变量	8
5. 常见概率分布	12
6. 期望、方差、协方差、相关系数	14
Chapter 2. 机器学习基础	17
1. 基本概念	17
2. 机器学习学习方式	20
3. 分类算法	22
4. 2.9 逻辑回归	33
5. 2.10 代价函数	35
6. 2.11 损失函数	41
7. 梯度下降	45
8. 2.14 线性判别分析 (LDA)	49
9. 主成分分析 (PCA)	55
10. 模型评估	60
11. 决策树	70
12. 支持向量机	72
13. 贝叶斯分类器	78
14. EM 算法	82
15. 降维和聚类	84

CHAPTER 1

数学基础

深度学习通常需要哪些数学基础? ¹ 深度学习里的数学到底难在哪里? 通常初学者都会有一些问题, 在网络推荐及书本推荐里, 经常看到会列出一系列数学科目, 比如微积分、线性代数、概率论、复变函数、数值计算、最优化理论、信息论等等. ² ³ 这些数学知识有相关性, 但实际上按照这样的知识范围来学习, 学习成本会很久, 而且会很枯燥, 本章我们通过选举一些数学基础里容易混淆的一些概念做以介绍, 帮助大家更好的理清这些易混淆概念之间的关系.

1. 向量和矩阵

1.1. 标量、向量、矩阵、张量之间的联系.

1.1.1. 标量. ⁴

定义 1.1. 一个标量表示一个单独的数, 它不同于线性代数中研究的其他大部分对象(通常是多个数的数组). 我们用斜体表示标量. 标量通常被赋予小写的变量名称.

1.1.2. 向量.

定义 1.2. 一个向量表示一组有序排列的数. ⁵ 通过次序中的索引, 我们可以确定每个单独的数. 通常我们赋予向量粗体的小写变量名称, 比如 \mathbf{x} . 向量中的元素可以通过带脚标的斜体表示. 向量 \mathbf{X} 的第一个元素是 \mathbf{X}_1 , 第二个元素是 \mathbf{X}_2 , 以此类推. 我们也会注明存储在向量中的元素的类型(实数、虚数等). ⁶

1.1.3. 矩阵.

定义 1.3. 矩阵是具有相同特征和纬度的对象的集合, 表现为一张二维数据表. 其意义是一个对象表示为矩阵中的一行, 一个特征表示为矩阵中的一列, 每个特征都有数值型的取值. 通常会赋予矩阵粗体的大写变量名称, 比如 \mathbf{A} .

¹删掉了又

²数学, 计算机文献中, 所有的句号都改成了点.

³优化理论改为了最优化理论

⁴英语是在第一次出现概念的时候出现, 所以这儿都删去了.

⁵在数学中, 向量一般指线性空间中方的元素.

⁶存储

1.1.4. 张量.

定义 1.4. 在某些情况下，我们会讨论坐标超过两维的数组。一般地，一个数组中的元素分布在若干维坐标的规则网格中，我们将其称之为张量。使用 A 来表示张量。⁷ 张量 A 中坐标为 (i, j, k) 的元素记作 $A_{(i,j,k)}$ 。

1.1.5. 四者之间关系.

注. 标量是 0 阶张量，向量是一阶张量。举例：

- 标量就是知道棍子的长度，但是你不会知道棍子指向哪儿。
- 向量就是不但知道棍子的长度，还知道棍子指向前面还是后面。
- 张量就是不但知道棍子的长度，也知道棍子指向前面还是后面，还能知道这棍子又向上/下和左/右偏转了多少。

1.2. 张量与矩阵的区别.

- 从代数角度讲，矩阵它是向量的推广。向量可以看成一维的“表格”（即分量按照顺序排成一排），矩阵是二维的“表格”（分量按照纵横位置排列），那么 n 阶张量就是所谓的 n 维的“表格”。张量的严格定义是利用线性映射来描述。
- 从几何角度讲，矩阵是一个真正的几何量，也就是说，它是一个不随参照系的坐标变换而变化的东西。向量也具有这种特性。
- 张量可以用 3×3 矩阵形式来表达。⁸
- 表示标量的数和表示向量的三维数组也可分别看作 1×1 , 1×3 的矩阵。⁹

1.3. 矩阵和向量相乘结果. 若使用爱因斯坦求和约定 (Einstein summation convention)，矩阵 A, B 相乘得到矩阵 C 可以用下式表示：

$$(1) \quad a_{ik} * b_{kj} = c_{ij}, \text{¹⁰}$$

其中， a_{ik}, b_{kj}, c_{ij} 分别表示矩阵 A, B, C 的元素， k 出现两次，是一个哑变量 (Dummy Variables) 表示对该参数进行遍历求和。而矩阵和向量相乘可以看成是矩阵相乘的一个特殊情况，例如：矩阵 B 是一个 $n \times 1$ 矩阵。¹¹

1.4. 向量和矩阵的范数归纳.

⁷删掉了“A”

⁸这儿有错误，矩阵怎么表示张量。

⁹这儿也有错误。

¹⁰去掉了 tag，公式后面也要加逗号。

¹¹ $n \times 1$ 的矩阵改成了 $n \times 1$ 矩阵，这个更符合大家的习惯。

1.4.1. 向量的范数. 定义一个向量为: $a = [-5, 6, 8, -10]$. 任意一组向量设为 $x = (x_1, x_2, \dots, x_N)$. 其不同范数求解如下:

定义 1.5. 向量的 1 范数 向量的各个元素的绝对值之和, 既

$$(2) \quad \|x\|_1 = \sum_{i=1}^N |x_i|, \text{ } ^{12}$$

上述向量 a 的 1 范数结果就是: 29.

定义 1.6. 向量的 2 范数 向量的每个元素的平方和再开平方根, 既

$$(3) \quad \|x\|_2 = \sqrt{\sum_{i=1}^N |x_i|^2}, \text{ } ^{13}$$

上述 a 的 2 范数结果就是: 15.

定义 1.7. 向量的负无穷范数 向量的所有元素的绝对值中最小的, 既

$$(4) \quad \|x\|_{-\infty} = \min |x_i|, \text{ } ^{14}$$

上述向量 a 的负无穷范数结果就是: 5.

定义 1.8. 向量的正无穷范数 向量的所有元素的绝对值中最大的

$$(5) \quad \|x\|_{+\infty} = \max |x_i|,$$

上述向量 a 的正无穷范数结果就是: 10.

定义 1.9. 向量的 p 范数 向量的 p 范数定义为: 15

$$(6) \quad L_p = \|x\|_p = \sqrt[p]{\sum_{i=1}^N |x_i|^p}$$

¹²加上了逗号.

¹³加上了逗号.

¹⁴加上了逗号

¹⁵加了一句向量的 p 范数定义为.

1.4.2. 矩阵的范数. 定义一个矩阵

$$A = \begin{pmatrix} -1 & 2 & -3 \\ 4 & -6 & 6 \end{pmatrix},$$

任意矩阵定义为: $A_{m \times n}$, 其元素为 a_{ij} . 矩阵的范数定义为

$$(7) \quad \|A\|_p := \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p},$$

当向量取不同范数时, 相应得到了不同的矩阵范数.

定义 1.10. 矩阵的 1 范数 (列范数) 矩阵的每一列上的元素绝对值先求和, 再从中取个最大的, (列和最大),

$$(8) \quad \|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^m |a_{ij}|,$$

上述矩阵 A 的 1 范数先得到 $[5, 8, 9]$, 再取最大的最终结果就是: 9.

定义 1.11. 矩阵的 2 范数 矩阵 $A^T A$ 的最大特征值开平方根,

$$(9) \quad \|A\|_2 = \sqrt{\lambda_{\max}(A^T A)},$$

其中, $\lambda_{\max}(A^T A)$ 为 $A^T A$ 的特征值绝对值的最大值. -

上述矩阵 A 的 2 范数得到的最终结果是: 10.0623.

定义 1.12. 矩阵的无穷范数 (行范数) 矩阵的每一行上的元素绝对值先求和, 再从中取个最大的, (行和最大),

$$(10) \quad \|A\|_\infty = \max_{1 \leq i \leq m} \sum_{j=1}^n |a_{ij}|,$$

上述矩阵 A 的行范数先得到 [616], 再取最大的最终结果就是: 16.

定义 1.13. 矩阵的核范数 矩阵的奇异值 (将矩阵 svd 分解) 之和, 这个范数可以用来低秩表示 (因为最小化核范数, 相当于最小化矩阵的秩——低秩), 上述矩阵 A 最终结果就是: 10.9287.

定义 1.14. 矩阵的 L_0 范数¹⁶ 矩阵的非 0 元素的个数, 通常用它来表示稀疏, L_0 范数越小 0 元素越多, 也就越稀疏, 上述矩阵 A 最终结果就是: 6.

¹⁶ L_0 应该是下标

定义 1.15. 矩阵的 L_1 范数¹⁷ 矩阵中的每个元素绝对值之和，它是 L_0 范数的最优凸近似，因此它也可以表示稀疏，上述矩阵 A 最终结果就是：22.

定义 1.16. 矩阵的 F 范数 矩阵的各个元素平方之和再开平方根，

$$(11) \quad \|A\|_F = \sqrt{\left(\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2\right)},$$

它通常也叫做矩阵的 L_2 范数，它的优点在于它是一个凸函数，可以求导求解，易于计算，上述矩阵 A 最终结果就是：10.0995.

定义 1.17. 矩阵的 L_{21} 范数 矩阵先以每一列为单位，求每一列的 F 范数（也可认为是向量的 2 范数），然后再将得到的结果求 L_1 范数（也可认为是向量的 1 范数），很容易看出它是介于 L_1 和 L_2 之间的一种范数，上述矩阵 A 最终结果就是：17.1559.

定义 1.18. 矩阵的 p 范数 矩阵的 p 范数定义为

$$(12) \quad \|A\|_p = \sqrt[p]{\left(\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^p\right)},$$

1.5. 如何判断一个矩阵为正定. 判定一个矩阵是否为正定，通常有以下几个方面：

- 顺序主子式全大于 0；
- 存在可逆矩阵 C 使 $C^T C$ 等于该矩阵；
- 正惯性指数等于 n ；
- 合同于单位矩阵 E （即：规范形为 E ）
- 标准形中主对角元素全为正；
- 特征值全为正；
- 是某基的度量矩阵.

2. 导数和偏导数

2.1. 导数偏导计算.

2.1.1. 导数定义. 导数 (derivative) 代表了在自变量变化趋于无穷小的时候，函数值的变化与自变量的变化的比值.¹⁸ 几何意义是这个点的切线. 物理意义是该时刻的（瞬时）变化率.

在一元函数中，只有一个自变量变动，也就是说只存在一个方向的变化率，这也就是为什么一元函数没有偏导数的原因. 在物理学中有平均速度和瞬时速度之说. 平均速度有

$$v = \frac{s}{t},$$

¹⁷这儿也应该是下标

¹⁸这个定义是牛顿时期的定义，高数上的是用 ϵ, N 语言来写的.

其中 v 表示平均速度, s 表示路程, t 表示时间. 这个公式可以改写为

$$\bar{v} = \frac{\Delta s}{\Delta t} = \frac{s(t_0 + \Delta t) - s(t_0)}{\Delta t},$$

其中 Δs 表示两点之间的距离, 而 Δt 表示走过这段距离需要花费的时间. 当 Δt 趋向于 0 ($\Delta t \rightarrow 0$) 时, 也就是时间变得很短时, 平均速度也就变成了在 t_0 时刻的瞬时速度, 表示成如下形式:

$$v(t_0) = \lim_{\Delta t \rightarrow 0} \bar{v} = \lim_{\Delta t \rightarrow 0} \frac{\Delta s}{\Delta t} = \lim_{\Delta t \rightarrow 0} \frac{s(t_0 + \Delta t) - s(t_0)}{\Delta t}$$

实际上, 上式表示的是路程 s 关于时间 t 的函数在 $t = t_0$ 处的导数. 一般的, 这样定义导数: 如果平均变化率的极限存在, 即有

$$\lim_{\Delta x \rightarrow 0} \frac{\Delta y}{\Delta x} = \lim_{\Delta x \rightarrow 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}$$

则称此极限为函数 $y = f(x)$ 在点 x_0 处的导数. 记作 $f'(x_0)$ 或 $y'|_{x=x_0}$ 或 $\frac{dy}{dx}|_{x=x_0}$ 或 $\frac{df(x)}{dx}|_{x=x_0}$. 通俗地说, 导数就是曲线在某一点切线的斜率.

2.1.2. 偏导数. 既然谈到偏导数 (partial derivative), 那就至少涉及到两个自变量. 以两个自变量为例, $z = f(x, y)$, 从导数到偏导数, 也就是从曲线来到了曲面. 曲线上的一点, 其切线只有一条. 但是曲面上的一点, 切线有无数条. 而偏导数就是指多元函数沿着坐标轴的变化率.

注意. 直观地说, 偏导数也就是函数在某一点上沿坐标轴正方向的变化率. 设函数 $z = f(x, y)$ 在点 (x_0, y_0) 的领域内有定义, 当 $y = y_0$ 时, z 可以看作关于 x 的一元函数 $f(x, y_0)$, 若该一元函数在 $x = x_0$ 处可导, 即有

$$\lim_{\Delta x \rightarrow 0} \frac{f(x_0 + \Delta x, y_0) - f(x_0, y_0)}{\Delta x} = A,$$

函数的极限 A 存在. 那么称 A 为函数 $z = f(x, y)$ 在点 (x_0, y_0) 处关于自变量 x 的偏导数, 记作 $f_x(x_0, y_0)$ 或 $\frac{\partial z}{\partial x}|_{y=y_0}$ 或 $\frac{\partial f}{\partial x}|_{y=y_0}$ 或 $z_x|_{y=y_0}^{x=x_0}$.

偏导数在求解时可以将另外一个变量看做常数, 利用普通的求导方式求解, 比如 $z = 3x^2 + xy$ 关于 x 的偏导数就为 $z_x = 6x + y$, 这个时候 y 相当于 x 的系数.

某点 (x_0, y_0) 处的偏导数的几何意义为曲面 $z = f(x, y)$ 与面 $x = x_0$ 或面 $y = y_0$ 交线在 $y = y_0$ 或 $x = x_0$ 处切线的斜率.

2.2. 导数和偏导数有什么区别? 导数和偏导没有本质区别, 如果极限存在, 都是当自变量的变化量趋于 0 时, 函数值的变化量与自变量变化量比值的极限.¹⁹

¹⁹ 偏导数与任意方向的导数吗?

注. 对导数和偏导数有:

- 一元函数, 一个 y 对应一个 x , 导数只有一个.
- 二元函数, 一个 z 对应一个 x 和一个 y , 有两个导数: 一个是 z 对 x 的导数, 一个是 z 对 y 的导数, 称之为偏导.²⁰
- 求偏导时要注意, 对一个变量求导, 则视另一个变量为常数, 只对改变量求导, 从而将偏导的求解转化成了一元函数的求导.

3. 特征值和特征向量

3.1. 特征值分解与特征向量.

- 特征值分解可以得到特征值 (eigenvalues) 与特征向量 (eigenvectors);
- 特征值表示的是这个特征到底有多重要, 而特征向量表示这个特征是什么.

如果说一个向量 ν 是方阵 A 的特征向量, 将一定可以表示成下面的形式:²¹

$$A\nu = \lambda\nu$$

λ 为特征向量 ν 对应的特征值. 特征值分解是将一个矩阵分解为如下形式:

$$A = Q\Sigma Q^{-1}, \quad ^{22}$$

其中, Q 是这个矩阵 A 的特征向量组成的矩阵, Σ 是一个对角矩阵, 每一个对角线元素就是一个特征值, 里面的特征值是由大到小排列的, 这些特征值所对应的特征向量就是描述这个矩阵变化方向 (从主要的变化到次要的变化排列). 也就是说矩阵 A 的信息可以由其特征值和特征向量表示.

3.2. 奇异值与特征值有什么关系. 那么奇异值和特征值是怎么对应起来的呢? 我们将一个矩阵 A 的转置乘以 A , 并对 $A^T A$ 求特征值, 则有下面的形式:

$$(A^T A)V = \lambda V$$

这里 V 就是矩阵 A 的右奇异向量, 另外还有:

$$\sigma_i = \sqrt{\lambda_i}, u_i = \frac{1}{\sigma_i} AV$$

其中 σ 就是奇异值, u 就是上面说的左奇异向量.²³ 奇异值 σ 跟特征值类似, 在矩阵 Σ 中也是从大到小排列, 而且 σ 的减少特别的快, 在很多情况下, 前 10% 甚至 1% 的奇异值的和

²⁰沿着任意方向都有导数存在的.

²¹向量改成了 ν 这里改成了上下一致的.

²²没有加逗号, Σ 和 \sum 有区别.

²³证明那个哥们也没给

就占了全部的奇异值之和的 99% 以上了. 也就是说, 我们也可以用前 r (r 远小于 mn) 个的奇异值来近似描述矩阵, 即部分奇异值分解:

$$A_{m \times n} \approx U_{m \times r} \Sigma_{r \times r} V_{r \times n}^T.$$

右边的三个矩阵相乘的结果将会是一个接近于 A 的矩阵, 在这儿, r 越接近于 n , 则相乘的结果越接近于 A .

4. 概率分布与随机变量

4.1. 机器学习为什么要使用概率. 事件的概率是衡量该事件发生的可能性的量度. 虽然在一次随机试验中某个事件的发生是带有偶然性的, 但那些可在相同条件下大量重复的随机试验却往往呈现出明显的统计规律.²⁴ ²⁵ 概率论在机器学习中扮演着一个核心角色, 因为机器学习算法的设计通常依赖于对数据的概率假设.

注. 例如在机器学习 (Andrew Ng) 的课中, 会有一个朴素贝叶斯假设, 它就是条件独立的一个例子. 该学习算法对内容做出假设, 用来分辨电子邮件是否为垃圾邮件. 假设无论邮件是否为垃圾邮件, 单词 x 出现在邮件中的概率条件独立于单词 y . 很明显这个假设不是不失一般性的, 因为某些单词几乎总是同时出现. 然而, 最终结果是, 这个简单的假设对结果的影响并不大, 且无论如何都可以让我们快速判别垃圾邮件.

4.2. 变量与随机变量有什么区别.

4.2.1. 随机变量.

定义 4.1. 在一定条件下, 并不总是出现相同结果的现象称为随机现象.

定义 4.2. 随机变量表示随机现象中各种结果的实值函数 (一切可能的样本点). 例如某一时间内公共汽车站等车乘客人数, 电话交换台在一定时间内收到的呼叫次数等, 都是随机变量的实例.

注意. 随机变量与模糊变量的不确定性的本质差别在于, 后者的测定结果仍具有不确定性, 即模糊性.

4.2.2. 变量与随机变量的区别. 当变量的取值的概率不是 1 时, 变量就变成了随机变量; 当随机变量取值的概率为 1 时, 随机变量就变成了变量.²⁶

²⁴数量规律改成了统计规律.

²⁵机器学习除了处理不确定量, 也需处理随机量. 不确定性和随机性可能来自多个方面, 使用概率论来量化不确定性. 这句是病句, 删了.

²⁶感觉有问题.

例 4.1. 当变量 x 值为 100 的概率为 1 的话, 那么 $x = 100$ 就是确定了的, 不会再有变化, 除非有进一步运算. 当变量 x 的值为 100 的概率不为 1, 比如为 50 的概率是 0.5, 为 100 的概率是 0.5, 那么这个变量就是会随不同条件而变化的, 是随机变量, 取到 50 或者 100 的概率都是 0.5, 即 50%.

4.3. 随机变量与概率分布的联系.

定义 4.3. 一个随机变量仅仅表示一个可能取得的状态, 还必须给定与之相伴的概率分布来制定每个状态的可能性. 用来描述随机变量或一族随机变量的每一个可能的状态的可能性大小的方法, 就是概率分布 (probability distribution).

随机变量可以分为离散型随机变量和连续型随机变量. 相应的描述其概率分布的函数是

- 概率质量函数 (Probability Mass Function,PMF): 描述离散型随机变量的概率分布, 通常用大写字母 P 表示.
- 概率密度函数 (Probability Density Function,PDF): 描述连续型随机变量的概率分布, 通常用小写字母 p 表示.

4.4. 离散型随机变量和概率质量函数.

定义 4.4. PMF 将随机变量能够取得的每个状态映射到随机变量取得该状态的概率.

- 一般而言, $P(x)$ 表示时 $X = x$ 的概率.
- 有时候为了防止混淆, 要明确写出随机变量的名称 $P(x=x)$.
- 有时候需要先定义一个随机变量, 然后制定它遵循的概率分布 x 服从 $P(x)$.

PMF 可以同时作用于多个随机变量, 即联合概率分布 (joint probability distribution) $P(X = x, Y = y)$ 表示 $X = x$ 和 $Y = y$ 同时发生的概率, 也可以简写成 $P(x, y)$.

如果一个函数 P 是随机变量 X 的 PMF, 那么它必须满足如下三个条件

- P 的定义域必须是的所有可能状态的集合.
- $\sum_x P(x) \leq 1$.
- $\sum_x P(x) = 1$. 我们把这一条性质称之为归一化的 (normalized).

4.5. 连续型随机变量和概率密度函数.

定义 4.5. 如果一个函数 p 是 x 的 PDF, 那么它必须满足如下几个条件

- p 的定义域必须是 x 的所有可能状态的集合.
- $\int_{-\infty}^{\infty} p(x) dx = 1$. 注意, 我们并不要求 $p(x) = 1$, 因为此处 $p(x)$ 不是表示的对应此状态具体的概率, 而是概率的一个相对大小 (密度).²⁷ 具体的概率, 需要积分去求.

²⁷有问题.

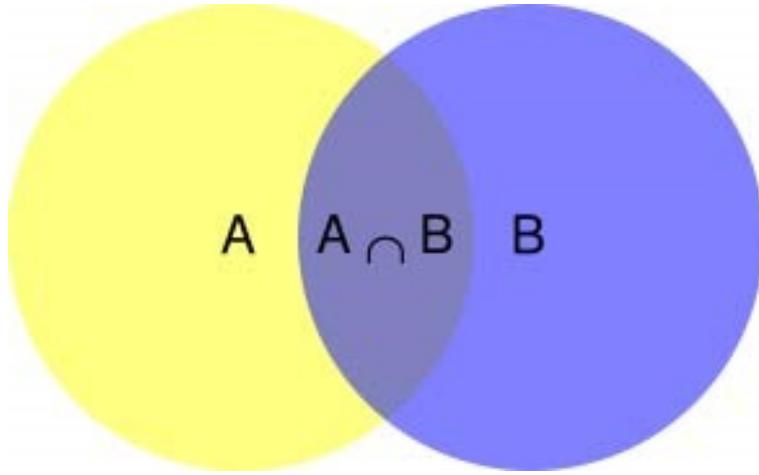


图 1. 文氏图

- $p(x)dx = 1$, 积分下来, 总和还是 1, 概率之和还是 1.

注. PDF $p(x)$ 并没有直接对特定的状态给出概率, 给出的是密度, 相对的, 它给出了落在面积为 x 的无限小的区域内的概率为 $p(x)x$. 由此, 我们无法求得具体某个状态的概率, 我们可以求得的是某个状态 x 落在某个区间 $[a, b]$ 内的概率为 $\int_a^b p(x)dx$.

4.6. 举例理解条件概率.

定理 4.1. 条件概率公式如下:

$$P(A|B) = P(A \cap B)/P(B)$$

注. 在同一个样本空间 Ω 中的事件或者子集 A 与 B , 如果随机从 Ω 中选出的一个元素属于 B , 那么下一个随机选择的元素属于 A 的概率就定义为在 B 的前提下 A 的条件概率.

条件概率文氏图示意如图1所示.

根据文氏图, 可以很清楚地看到在事件 B 发生的情况下, 事件 A 发生的概率就是 $P(A \cap B)$ 除以 $P(B)$.

例 4.2. 一对夫妻有两个小孩, 已知其中一个是女孩, 则另一个是女孩子的概率是多少? (面试、笔试都碰到过).

穷举法: 已知其中一个是女孩, 那么样本空间为男女, 女女, 女男, 则另外一个仍然是女生的概率就是 $1/3$.

条件概率法: $P(|) = P() / P()$, 夫妻有两个小孩, 那么它的样本空间为女女, 男女, 女男, 男男, 则 $P()$ 为 $1/4$, $P = 1 - P() = 3/4$, 所以最后 $1/3$.

这里大家可能会误解, 男女和女男是同一种情况, 但实际上类似姐弟和兄妹是不同情况.

4.7. 联合概率与边缘概率联系区别.

4.7.1. 区别. 联合概率: 联合概率指类似于 $P(X = a, Y = b)$ 这样, 包含多个条件, 且所有条件同时成立的概率. 联合概率是指在多元的概率分布中多个随机变量分别满足各自条件的概率. 边缘概率: 边缘概率是某个事件发生的概率, 而与其它事件无关. 边缘概率指类似于 $P(X = a)$, $P(Y = b)$ 这样, 仅与单个随机变量有关的概率.

4.7.2. 联系. 联合分布可求边缘分布, 但若只知道边缘分布, 无法求得联合分布.

4.8. 条件概率的链式法则. 由条件概率的定义, 可直接得出下面的乘法公式:

定理 4.2. 乘法公式 设 A, B 是两个事件, 并且 $P(A) > 0$, 则有

$$P(AB) = P(B|A)P(A),$$

f

注意.

$$P(ABC) = P(C|AB)P(B|A)P(A),$$

注意. 一般地, 用归纳法可证: 若 $P(A_1 A_2 \dots A_n) > 0$, 则有

$$P(A_1 A_2 \dots A_n) = P(A_n | A_1 A_2 \dots A_{n-1})P(A_{n-1} | A_1 A_2 \dots A_{n-2}) \dots P(A_2 | A_1)P(A_1) = P(A_1) \prod_{i=2}^n P(A_i | A_1 A_2 \dots A_{i-1}),$$

任何多维随机变量联合概率分布, 都可以分解成只有一个变量的条件概率相乘形式.

4.9. 独立性和条件独立性.

4.9.1. 独立性. 两个随机变量 x 和 y , 概率分布表示成两个因子乘积形式, 一个因子只包含 x , 另一个因子只包含 y , 两个随机变量相互独立 (independent).

条件有时为不独立的事件之间带来独立, 有时也会把本来独立的事件, 因为此条件的存在, 而失去独立性.

例 4.3. $P(XY) = P(X)P(Y)$, 事件 X 和事件 Y 独立. 此时给定 Z ,

$$P(X, Y | Z) \neq P(X | Z)P(Y | Z),$$

事件独立时, 联合概率等于概率的乘积. 这是一个非常好的数学性质, 然而不幸的是, 无条件的独立是十分稀少的, 因为大部分情况下, 事件之间都是互相影响的.

4.9.2. 条件独立性. 给定 Z 的情况下, X 和 Y 条件独立, 当且仅当

$$X \perp Y | Z \iff P(X, Y | Z) = P(X | Z)P(Y | Z),$$

X 和 Y 的关系依赖于 Z , 而不是直接产生.

例 4.4. 定义如下事件:

- (1). X : 明天下雨;
- (2). Y : 今天的地面是湿的;
- (3). Z : 今天是否下雨;

Z 事件的成立, 对 X 和 Y 均有影响, 然而, 在 Z 事件成立的前提下, 今天的地面情况对明天是否下雨没有影响.

5. 常见概率分布

5.1. Bernoulli 分布. Bernoulli 分布 (伯努利分布, 0-1 分布) 是单个二值随机变量分布, 单参数 $\phi[0, 1]$ 控制, ϕ 给出随机变量等于 1 的概率. 主要性质有:

$$\begin{aligned} P(x = 1) &= \phi, \\ P(x = 0) &= 1 - \phi, \\ P(x = x) &= \phi^x(1 - \phi)^{1-x}, \end{aligned}$$

其期望和方差为:

$$\begin{aligned} E_x(x) &= \phi, \\ Var_x(x) &= \phi(1 - \phi), \end{aligned}$$

伯努利分布适合对离散型随机变量建模.

5.2. Multinoulli 分布. Multinoulli 分布也叫范畴分布, 是单个 k 值随机分布, 经常用来表示对象分类的分布. 其中 k 是有限值. Multinoulli 分布由向量 $p \in [0, 1]^{k-1}$ 参数化, 每个分量 p_i 表示第 i 个状态的概率, 且 $p_k = 1 - \mathbf{1}^T p$. 这里 $\mathbf{1}^T$ 表示元素全为 1 的列向量的转置, 其实就是对于向量 p 中除了 k 的概率之和. 可以重写为 $p_k = 1 - \sum_0^{k-1} p_i$.

5.3. 二项分布. 通俗点硬币抛多次. 二项分布 (Binomial distribution) 是 n 重伯努利试验成功次数的离散概率分布.

5.4. 多项式分布. 多项式分布 (Multinomial Distribution) 是二项式分布的推广. 二项式做 n 次伯努利实验, 规定了每次试验的结果只有两个, 如果现在还是做 n 次试验, 只不过每次试验的结果可以有多 m 个, 且 m 个结果发生的概率互斥且和为 1, 则发生其中一个结果 X 次的概率就是多项式分布.

5.5. 高斯分布. 高斯分布²⁸. 也叫正态分布 (Normal Distribution), 概率度函数如下:

$$N(x; \mu, \sigma^2) = \sqrt{\frac{1}{2\pi\sigma^2}} e^{(-\frac{1}{2\sigma^2}(x-\mu)^2)},$$

其中, μ 和 σ 分别是均值和方差, 中心峰值 x 坐标由 μ 给出, 峰的宽度受 σ 控制, 最大点在 $x = \mu$ 处取得, 拐点为 $x = \mu \pm \sigma$

正态分布中, $\pm\sigma$ ²⁹、 $\pm 2\sigma$ 、 $\pm 3\sigma$ 下的概率分别是 68.3% 95.5% 99.73%, 这 3 个数最好记住.

此外, 令 $\mu = 0, \sigma = 1$ 高斯分布即简化为标准正态分布:

$$N(x; \mu, \sigma^2) = \sqrt{\frac{1}{2\pi}} e^{(-\frac{1}{2}x^2)},$$

对概率密度函数高效求值:

$$N(x; \mu, \beta^{-1}) = \sqrt{\frac{\beta}{2\pi}} e^{(-\frac{1}{2}\beta(x-\mu)^2)},$$

其中, $\beta = \frac{1}{\sigma^2}$ 通过参数 β 来控制分布精度.

注. 问: 何时采用正态分布? 答: 缺乏实数上分布的先验知识, 不知选择何种形式时, 默认选择正态分布总是不会错的, 理由如下:

1. 中心极限定理告诉我们, 很多独立随机变量均近似服从正态分布, 现实中很多复杂系统都可以被建模成正态分布的噪声, 即使该系统可以被结构化分解.
2. 正态分布是具有相同方差的所有概率分布中, 不确定性最大的分布, 换句话说, 正态分布是对模型加入先验知识最少的分布.

正态分布可以推广到 R^n 空间, 此时称为多维正态分布, 其参数是一个正定对称矩阵 Σ :

$$N(x; \mu, \Sigma) = \sqrt{\frac{1}{(2\pi)^n \det(\Sigma)}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right),$$

对多维正态分布概率密度高效求值:

$$N(x; \mu, \beta^{-1}) = \sqrt{\det(\beta)} (2\pi)^n \exp\left(-\frac{1}{2}(x - \mu)^T \beta (x - \mu)\right),$$

此处, β 是一个精度矩阵.

²⁸高斯改成高斯分布

²⁹ $\pm 1\sigma$ 改成了 $\pm \sigma$

5.6. 指数分布. 深度学习中, 指数分布用来描述在 $x = 0$ 点处取得边界点的分布, 指数分布定义如下:

$$p(x; \lambda) = \lambda I_{x \geq 0} e^{-\lambda x},$$

指数分布用指示函数 $I_{x \geq 0}$ 来使 x 取负值时的概率为零.

5.7. Laplace 分布 (拉普拉斯分布). 一个联系紧密的概率分布是 Laplace 分布 (Laplace distribution), 它允许我们在任意一点 μ 处设置概率质量的峰值

$$\text{Laplace}(x; \mu; \gamma) = \frac{1}{2\gamma} \exp\left(-\frac{|x - \mu|}{\gamma}\right),$$

5.8. Dirac 分布和经验分布. Dirac 分布可保证概率分布中所有质量都集中在一个点上. Diract 分布的狄拉克 δ 函数 (也称为单位脉冲函数) 定义如下:

$$p(x) = \delta(x - \mu), x \neq \mu$$

$$\int_a^b \delta(x - \mu) dx = 1, a < \mu < b$$

Dirac 分布经常作为经验分布 (empirical distribution) 的一个组成部分出现

$$\hat{p}(x) = \frac{1}{m} \sum_{i=1}^m \delta(x - x^{(i)})$$

, 其中, m 个点 x^1, \dots, x^m 是给定的数据集, 经验分布将概率密度 $\frac{1}{m}$ 赋给了这些点.

当我们在训练集上训练模型时, 可以认为从这个训练集上得到的经验分布指明了采样来源.

狄拉克 函数适合对连续型随机变量的经验分布.

6. 期望、方差、协方差、相关系数

6.1. 期望. 在概率论和统计学中, 数学期望 (或均值, 亦简称期望) 是试验中每次可能结果的概率乘以其结果的总和. 它反映随机变量平均取值的大小.

6.1.1. 线性运算.

$$(13) \quad E(ax + by + c) = aE(x) + bE(y) + c.$$

6.1.2. 推广形式.

$$(14) \quad E\left(\sum_{k=1}^n a_k x_k + c\right) = \sum_{k=1}^n a_k E(x_k) + c.$$

6.1.3. 函数期望. 设 $f(x)$ 为 x 的函数, 则 $f(x)$ 的期望为

$$\text{离散函数: } E(f(x)) = \sum_{k=1}^n f(x_k)P(x_k)$$

$$\text{连续函数: } E(f(x)) = \int_{-\infty}^{+\infty} f(x)p(x)dx$$

注. 函数的期望大于等于期望的函数 (Jensen (詹森) 不等式, 即

$$E(f(x)) \geq f(E(x)),$$

注. 一般情况下, 乘积的期望不等于期望的乘积.

注. 如果 X 和 Y 相互独立, 则

$$(15) \quad E(xy) = E(x)E(y).$$

6.2. 方差.

定义 6.1. 概率论中方差用来度量随机变量和其数学期望 (即均值) 之间的偏离程度. 方差是一种特殊的期望.³⁰ 定义为:

$$Var(x) = E((x - E(x))^2)$$

注. 方差性质:

- 1) $Var(x) = E(x^2) - E(x)^2$;
- 2) 常数的方差为 0;
- 3) 方差不满足线性性质;
- 4) 如果 X 和 Y 相互独立, $Var(ax + by) = a^2Var(x) + b^2Var(y)$.

6.3. 协方差.

定义 6.2. 协方差是衡量两个变量线性相关性强度及变量尺度. 两个随机变量的协方差定义为:

$$Cov(x, y) = E((x - E(x))(y - E(y))).$$

方差是一种特殊的协方差. 当 $X = Y$ 时, $Cov(x, y) = Var(x) = Var(y)$.

注. 协方差性质:

- 1) 独立变量的协方差为 0.
- 2) 协方差计算公式:

³⁰数学期望是均值是错的.

$$Cov\left(\sum_{i=1}^m a_i x_i, \sum_{j=1}^m b_j y_j\right) = \sum_{i=1}^m \sum_{j=1}^m a_i b_j Cov(x_i y_j).$$

3) 特殊情况:

$$Cov(a + bx, c + dy) = bdCov(x, y).$$

6.4. 相关系数.

定义 6.3. 相关系数是研究变量之间线性相关程度的量. 两个随机变量的相关系数定义为:

$$Corr(x, y) = \frac{Cov(x, y)}{\sqrt{Var(x)Var(y)}}$$

注. 相关系数的性质:

- 1) 有界性. 相关系数的取值范围是 $[-1, 1]$, 可以看成无量纲的协方差.
- 2) 值越接近 1, 说明两个变量正相关性 (线性) 越强. 越接近 -1, 说明负相关性越强, 当为 0 时, 表示两个变量没有相关性.

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。¹ 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，如监督学习、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

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。

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。

1.2.1. 神经网络架构. 图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。

¹所有的句话都改成了点。

²病句。

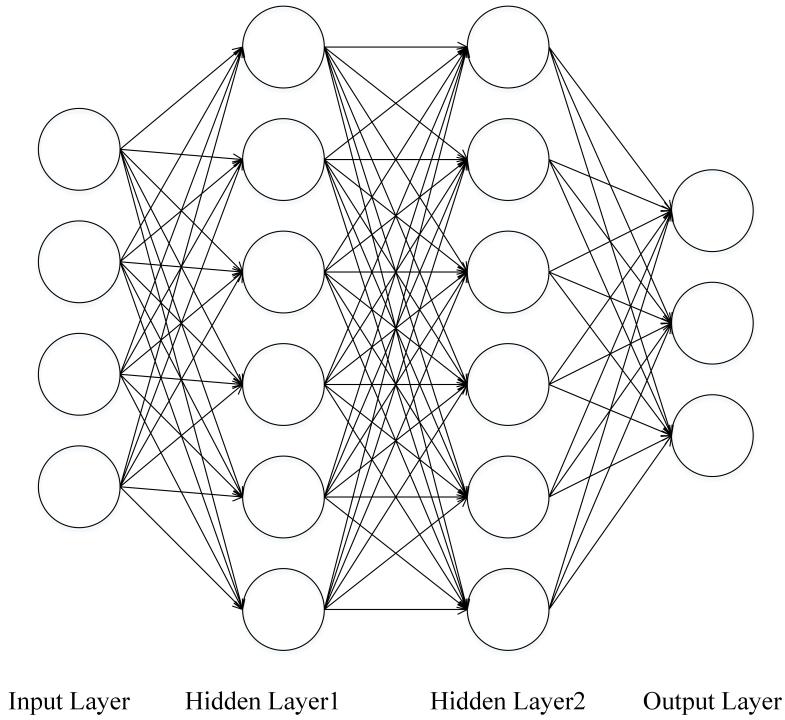


图 1. 神经网络系统

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，图1，图2，图3，图4是各种常见算法的图示。³

表 1: 各种常见算法图示

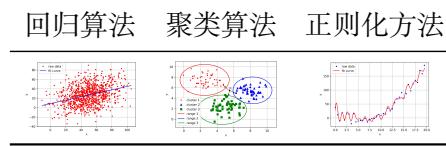


表 2: 各种常见算法图示

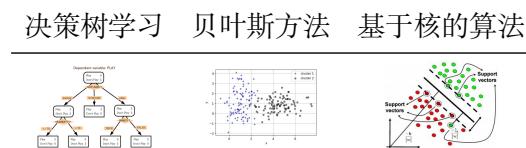


表 3: 各种常见算法图示

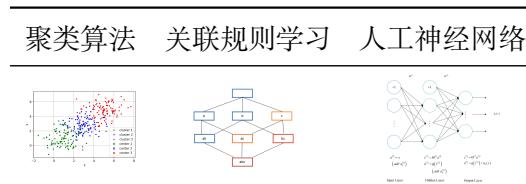


表 4: 各种常见算法图示

导数.⁴ 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, 其导数等于是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:

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

1.5. 理解局部最优与全局最优.

1.5.1. 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.

1.6. 大数据与深度学习之间的关系.

1.6.1. 定义. 首先来看大数据、机器学习及数据挖掘三者简单的定义:

定义 1.1. **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.

定义 1.2. **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.

定义 1.3. **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.

⁴在点 (u, v) 有问题.

1.6.2. 机器学习和数据挖掘之间的关系. 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.

1.6.3. 大数据与深度学习关系总结.

- (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, 这些工作才会有希望一一得到实现.
- (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. 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.

2.1. 监督学习.

- (1). 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.
- (2). 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.
- (3). 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. 非监督式学习.

- (1). 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.
- (2). 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.
- (3). 算法举例: 常见算法包括 Apriori 算法以及 k-Means 算法.

2.3. 半监督式学习.

- (1). 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.
- (2). 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.
- (3). 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.

2.4. 弱监督学习.

- (1). 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.
- (2). 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.

2.5. 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。

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。

3.1. 常用分类算法的优缺点

表 2-1 常用分类算法的优缺点

算 法	优 点	缺 点
--------	--------	--------

算 法	优 点	缺 点
--------	--------	--------

Bayes1) 1)

贝叶斯分类法所需估计的参数少, 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.

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)

算 法	优 点	缺 点
--------	--------	--------

Decision Tree 1)

决策树 不 对于各类型样本数量不一致数据,

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) 短时间处理大量数据,

2) 易于过拟合.

3) 3)

算 法	优 点	缺 点
--------	--------	--------

SVM 1) 1)

支 可 对
持 以 缺
向 以 失
量 解 数
机 决 据
 小 敏
 样 感.
 本 2)
 下 内 存
 机 器 消 耗
 学 学 大,
 习 问 难 以
 的 题. 2) 解
 2) 释.
 提 3)
 高 运 行 和
 泛 行 调 参
 化 性 略 烦
 能. 3)
 可 以 解 决
 以 高 高 维、
 解 高 非 线 性
 决 高 问 题.
 决 超 高 维 文
 决 超 高 维 文

算 法	优 点	缺 点
--------	--------	--------

KNN 1) 1)

K 近 邻 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) 2)

计 算 量 太 大. 对 于 样 本 分 类 不 均 衡 的 问 题, 会 产 生 误 判.

3) 3) 需 要 大 量 的 内 存.

4) 4) 输 出 的 可 解 释 性

5) 5) 训 练 时 间 复

6) 6) 可 以 解 释 性

算 法	优 点	缺 点
--------	--------	--------

Logistlq 特
Re- 速 征
gres- 度 处
sion 快. 理
逻 2) 复
辑 简 杂.
回 单 需
归 易 要
于 理 归
解, 直 一
接 看 化 和
看 到 较
到 各 多 的
个 特 征 工
特征 的 程.
权 重.
3)
能 容 易 地
容 更新 模型
易 地 吸收
更 新 模型
新 吸 收 新
的 收 新 的
数 数据.
据. 4)
如 果

算 法	优 点	缺 点
--------	--------	--------

Neural) 1)
 Net- 分 需
 work 类 要
 神 准 大
 经 确 量
 网 率 参
 络 高 数
 2) (网
 并 络
 行 拓
 处 扑、
 理 阀
 能 值、
 力 阈
 强. 值)
 3) .2)
 分 结
 布 果
 式 难
 存 以
 储 解
 和 释.3)
 学 训
 习 练
 能 时
 力 间
 强. 过
 4) 长.
 鲁 棒
 性 较
 强,
 不 易
 受 噪
 声 影
 响.

算 法	优 点	缺 点
--------	--------	--------

Adaboosting对

ad- out-
aboostier

是 比
一 较
种 敏
有 感

很高
精度
的分
类器

.2)

可以使用各种方法构建子分类器,

Ad-
aboost

算法提供的是框架

.3)

当

		预测类别				
		Yes	No	总计		
实际类别	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		

图 2. 图 2-3 术语的混淆矩阵

算	优	缺
法	点	点

3.2. 2.8.2 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: 1) 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; 2) 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; 3) 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; 4) 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.

表 2-2 四个术语的混淆矩阵

表 2-2 是这四个术语的混淆矩阵, 做以下说明: 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

图 3.

- 1) 正确率 (accuracy) 正确率是我们最常见的评价指标, $\text{accuracy} = (\text{TP} + \text{TN}) / (\text{P} + \text{N})$, 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.
 - 2) 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, $\text{error rate} = (\text{FP} + \text{FN}) / (\text{P} + \text{N})$, 对某一个实例来说, 分对与分错是互斥事件, 所以 $\text{accuracy} = 1 - \text{error rate}$.
 - 3) 灵敏度 (sensitivity) $\text{sensitivity} = \text{TP} / \text{P}$, 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.
 - 4) 特异性 (specificity) $\text{specificity} = \text{TN} / \text{N}$, 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.
 - 5) 精度 (precision) $\text{precision} = \text{TP} / (\text{TP} + \text{FP})$, 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.
 - 6) 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, $\text{recall} = \text{TP} / (\text{TP} + \text{FN}) = \text{TP} / \text{P} = \text{sensitivity}$. 可以看到召回率与灵敏度是一样的.
 - 7) 其他评价指标计算速度: 分类器训练和预测需要的时间; 鲁棒性: 处理缺失值和异常值的能力; 可扩展性: 处理大数据集的能力; 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.
 - 8) 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: $F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$. 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).
- (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的影响.

• ROC 曲线和 PR 曲线

如图2-3, ROC曲线是 (Receiver Operating Characteristic Curve, 受试者工作特征曲线) 的简称, 是一种评估分类器性能的图形。

图 2-3 ROC 曲线

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。

3.3. 2.8.3 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。我们都应该知道，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却忽视了研究者本身最为关注的情况。

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。

4. 2.9 逻辑回归

4.1. 2.9.1 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。

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.

4.3. 生成模型和判别模型的区别. 生成模型: 由数据学习联合概率密度分布 $P(X, Y)$, 然后求出条件概率分布 $P(Y|X)$ 作为预测的模型, 即生成模型: $P(Y|X) = P(X, Y) / P(X)$ (贝叶斯概率). 基本思想是首先建立样本的联合概率密度模型 $P(X, Y)$, 然后再得到后验概率 $P(Y|X)$, 再利用它进行分类. 典型的生成模型有朴素贝叶斯, 隐马尔科夫模型等

判别模型: 由数据直接学习决策函数 $Y=f(X)$ 或者条件概率分布 $P(Y|X)$ 作为预测的模型, 即判别模型. 基本思想是有限样本条件下建立判别函数, 不考虑样本的产生模型, 直接研究预测模型. 典型的判别模型包括 k 近邻, 感知机, 决策树, 支持向量机等. 这些模型的特点都是输入属性 X 可以直接得到后验概率 $P(Y|X)$, 输出条件概率最大的作为最终的类别 (对于二分类任务来说, 实际得到一个 score, 当 score 大于 threshold 时则为正类, 否则为负类).

举例:

判别式模型举例: 要确定一个羊是山羊还是绵羊, 用判别模型的方法是从历史数据中学习到模型, 然后通过提取这只羊的特征来预测出这只羊是山羊的概率, 是绵羊的概率.

生成式模型举例: 利用生成模型是根据山羊的特征首先学习出一个山羊的模型, 然后根据绵羊的特征学习出一个绵羊的模型, 然后从这只羊中提取特征, 放到山羊模型中看概率是多少, 在放到绵羊模型中看概率是多少, 哪个大就是哪个.

联系和区别:

生成方法的特点: 上面说到, 生成方法学习联合概率密度分布 $P(X, Y)$, 所以就可以从统计的角度表示数据.

判别方法的特点: 判别方法直接学习的是决策函数 $Y=f(X)$ 或者条件概率分布 $P(Y|X)$. 不能反映训练数据本

最后, 由生成模型可以得到判别模型, 但由判别模型得不到生成模型.

4.4. 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.

4.5. 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: $f(x) = \theta^T x = \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$

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 $\theta^T x \rightarrow +\infty$, 则 $y \rightarrow 1$, 即 y 为 1 类;

当 $\theta^T x < 0$, 则 $y < 0.5$; 若 $\theta^T x \rightarrow -\infty$, 则 $y \rightarrow 0$, 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 2-3 为线性回归和逻辑回归的区别.

表 2-3 线性回归和逻辑回归的区别

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

下面具体解释一下:

- 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 $y^{(i)} \in (0, 1)$;
- 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.

5. 2.10 代价函数

5.1. 2.10.1 为什么需要代价函数.

- 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.
- 用于找到最优解的目的函数.

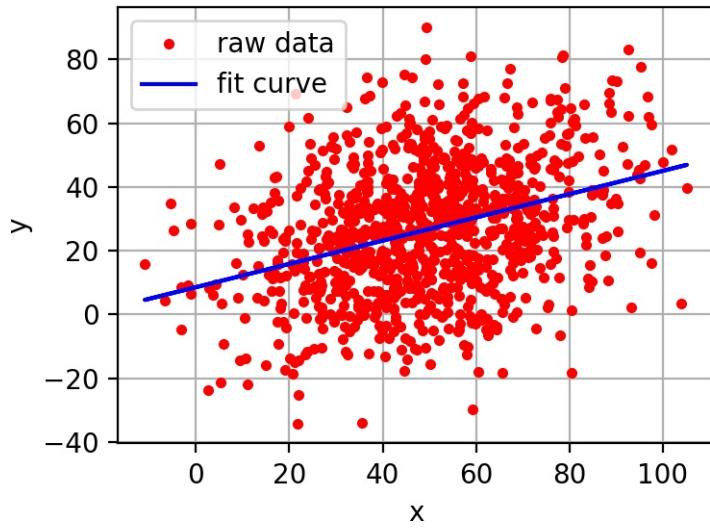


图 4.

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 2-4 所示, 当参数发生变化时, 假设函数状态也会随着变化.

图 2-4 $h(x) = A + Bx$ 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 2-5 所示.

图 2-5 代价函数三维图像

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.

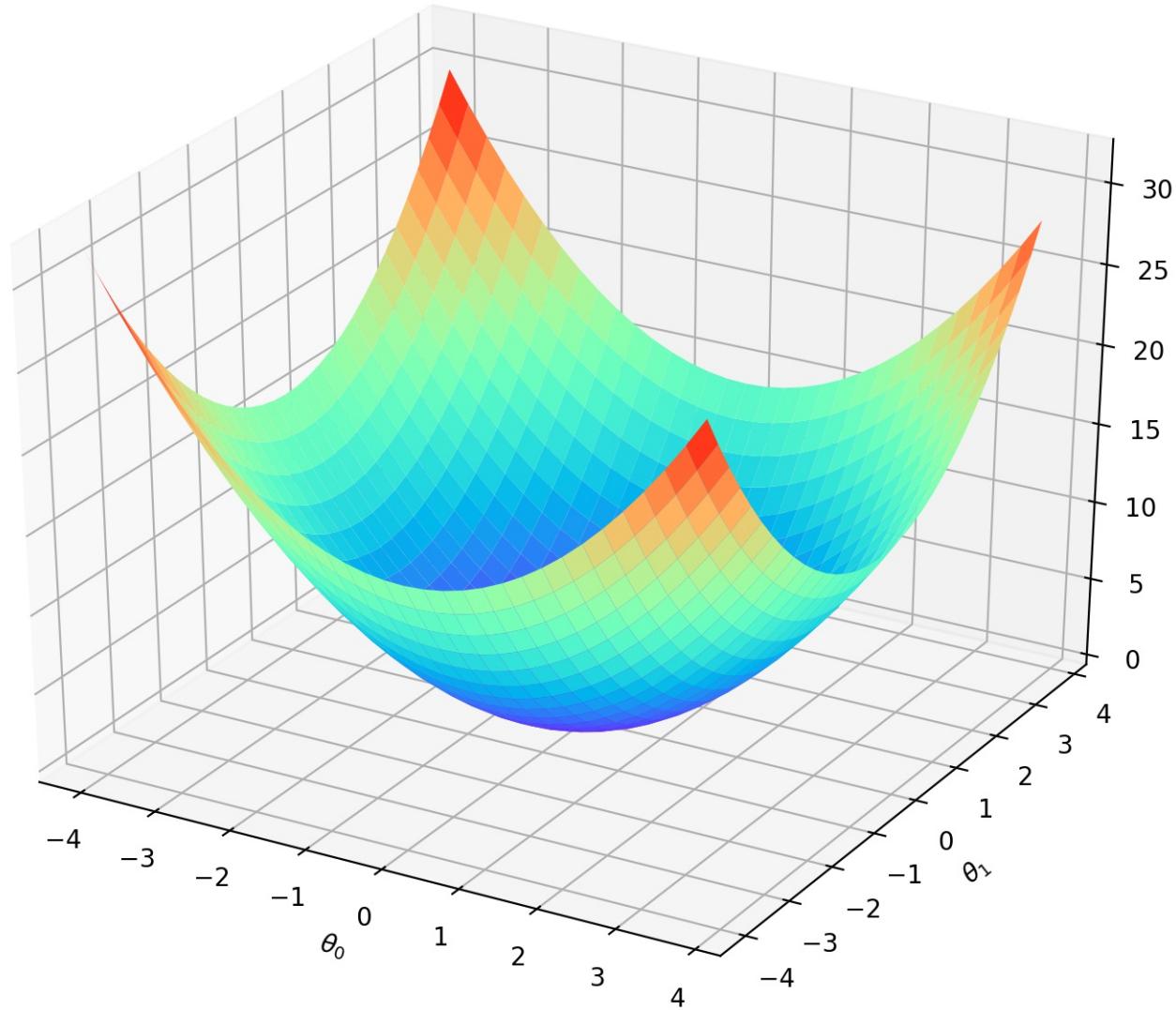


图 5.

5.4. 常见代价函数. (1) 二次代价函数 (quadratic cost):

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

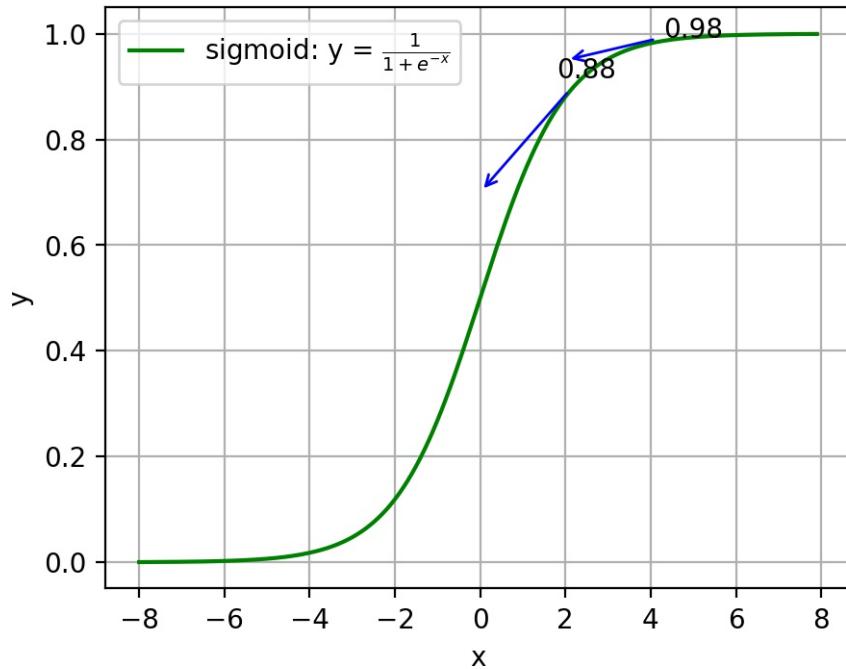


图 6.

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：

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

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 2-6 所示：

图 2-6 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：

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

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()`。

对数似然函数：

我们将似然函数作为机器学习模型的损失函数，并且用在分类问题中。这时似然函数是直接作用于模型的输出的（损失函数就是为了衡量当前参数下 model 的预测值 predict 距离真实值 label 的大小，所以似然函数用作损失函数时当然也是为了完成该任务），所以对于似然函数来说，这里的样本集就成了 label 集（而不是机器学习意义上的样本集 X 了），这里的参数也不是机器学习 model 的参数，而是 predict 值。

其实作为损失函数的似然函数并不关心你当前的机器学习 model 的参数是怎样的，毕竟它此时所接收的输入只有两部分：1、predict.2、label .3、分布模型 (predict 服从的分布)。

显然这里的 label 就是似然函数的观测值，即样本集。而它眼里的模型，当然就是 predict 这个随机变量所服从的概率分布模型。它的目的，就是衡量 predict 背后的模型对于当前观测值的解释程度。而每个样本的 predict 值，恰恰就是它所服从的分布模型的参数。

比如此时我们的机器学习任务是一个 4 个类别的分类任务，机器学习 model 的输出就是当前样本 X 下的每个类别的概率，如 $\text{predict}=[0.1, 0.1, 0.7, 0.1]$ ，而该样本的标签是类别 3，表示成向量就是 $\text{label}=[0, 0, 1, 0]$ 。那么 $\text{label}=[0, 0, 1, 0]$ 就是似然函数眼里的样本，然后我们

可以假设 predict 这个随机变量背后的模型是单次观测下的多项式分布，（因为 softmax 本身是基于多项式分布的）。

回顾：

伯努利分布，也叫做 (0, 1) 分布，贝努利分布可以看成是将一枚硬币（只有正反两个面，代表两个类别）向上扔出，出现某个面（类别）的概率情况，因此其概率密度函数为：

$$f(x) = p^x(1-p)^{1-x} = \begin{cases} p, & x = 1 \\ q, & x = 0 \end{cases}$$

这是理解似然函数做损失函数的关键！另外，贝努利分布的模型参数就是其中一个类别的发生概率。

而二项分布呢，就是将贝努利实验重复 n 次（各次实验之间是相互独立的）。

而多项式分布呢，就是将二项分布推广到多个面（类别）。

所以，单次观测下的多项式分布就是贝努利分布的多类推广！即：

$$f_{multi}(x; p) = \prod_{i=1}^C p_i^{x_i}$$

其中，C 代表类别数.p 代表向量形式的模型参数，即各个类别的发生概率，如 $p=[0.1, 0.1, 0.7, 0.1]$ ，则 $p_1=0.1, p_3=0.7$ 等。即，多项式分布的模型参数就是各个类别的发生概率！x 代表 one-hot 形式的观测值，如 x= 类别 3，则 $x=[0, 0, 1, 0]$.xi 代表 x 的第 i 个元素，比如 x= 类别 3 时， $x_1=0, x_2=0, x_3=1, x_4=0$ 。

想一下，机器学习 model 对某个样本的输出，就代表各个类别发生的概率。但是，对于当前这一个样本而言，它肯定只能有一个类别，所以这一个样本就可以看成是一次实验（观察），而这次实验（观察）的结果要服从上述各个类别发生的概率，那不就是服从多项式分布嘛！而且是单次观察！各个类别发生的概率 predict 当然就是这个多项式分布的参数。

总结一下，对于多类分类问题，似然函数就是衡量当前这个以 predict 为参数的单次观测下的多项式分布模型与样本值 label 之间的似然度。

所以，根据似然函数的定义，单个样本的似然函数即：

$$L = f_{multi}(label; predict)$$

所以，整个样本集（或者一个 batch）的似然函数即：

$$L = \prod_X f_{multi}(label; predict) = \prod_X \prod_{i=1}^C predict(i)^{label(i)}$$

所以在累乘号前面加上 log 函数后，就成了所谓的对数似然函数：

$$L = \sum_X \sum_{i=1}^C label(i) \log(predict(i))$$

而最大化对数似然函数就等效于最小化负对数似然函数，所以前面加个负号就和交叉熵的形式相同的了。

交叉熵定义：对于某种分布的随机变量 $X \sim p(x)$ ，有一个模型 $q(x)$ 用于近似 $p(x)$ 的概率分布，则分布 X 与模型 q 之间的交叉熵即：

$$H(X, q) = - \sum_x p(x) \log q(x)$$

这里 X 的分布模型即样本集 $label$ 的真实分布模型，这里模型 $q(x)$ 即想要模拟真实分布模型的机器学习模型。可以说交叉熵是直接衡量两个分布，或者说两个 model 之间的差异。而似然函数则是解释以 model 的输出为参数的某分布模型对样本集的解释程度。因此，可以说这两者是“同貌不同源”，但是“殊途同归”啦。

tips:

最大似然估计：

给定一堆数据，假如我们知道它是从某一种分布中随机取出来的，可是我们并不知道这个分布具体的参，即“模型已定，参数未知”。例如，我们知道这个分布是正态分布，但是不知道均值和方差；或者是二项分布，但是不知道均值。最大似然估计（MLE，Maximum Likelihood Estimation）就可以用来估计模型的参数。MLE 的目标是找出一组参数，使得模型产生出观测数据的概率最大。

5.5. 2.10.5 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) 为什么要用交叉熵代替二次代价函数权值 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。

6. 2.11 损失函数

6.1. 2.11.1 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.

6.2. 2.11.2 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:

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

(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**

$$L(Y, P(Y|X)) = -\log P(Y|X) = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^M y_{ij} \log(p_{ij})$$

其中, Y 为输出变量, X 为输入变量, L 为损失函数. N 为输入样本量, M 为可能的类别数, y_{ij} 是一个二值指标, 表示类别 j 是否是输入实例 x_i 的真实类别. p_{ij} 为模型或分类器预测输入实例 x_i 属于类别 j 的概率.

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. 形式上等价于二分类的交叉熵损失函数.

(6) **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.

(7) **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:

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

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

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

上式相似于下式

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

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

6.3. 2.11.3 逻辑回归为什么使用对数损失函数.

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

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:

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

对数似然函数为:

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

对数函数在单个数据点上的定义为：

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

则全局样本损失函数为：

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

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。

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 $x_i (i = 1, 2, \dots, n)$ 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：

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

对上式取自然对数，可得：

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

根据对数定律，上式可以化简为：

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

然后求导为：

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

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：

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

同理，可计算 σ 。

7. 梯度下降

7.1. 机器学习中为什么需要梯度下降. 梯度下降是机器学习中常见优化算法之一，梯度下降法有以下几个作用：

- (1) 梯度下降是迭代法的一种，可以用于求解最小二乘问题。
- (2) 在求解机器学习算法的模型参数，即无约束优化问题时，主要有梯度下降法 (Gradient Descent) 和最小二乘法。
- (3) 在求解损失函数的最小值时，可以通过梯度下降法来一步步的迭代求解，得到最小化的损失函数和模型参数值。
- (4) 如果我们需要求解损失函数的最大值，可通过梯度上升法来迭代。梯度下降法和梯度上升法可相互转换。
- (5) 在机器学习中，梯度下降法主要有随机梯度下降法和批量梯度下降法。

7.2. 梯度下降法缺点. 梯度下降法缺点有以下几点：

- (1) 靠近极小值时收敛速度减慢。
- (2) 直线搜索时可能会产生一些问题。
- (3) 可能会“之字形”地下降。

梯度概念也有需注意的地方：

- (1) 梯度是一个向量，即有方向有大小。
- (2) 梯度的方向是最大方向导数的方向。
- (3) 梯度的值是最大方向导数的值。

7.3. 梯度下降法直观理解. 梯度下降法经典图示如下图 2.7 所示：

图 2.7 梯度下降法经典图示

形象化举例，由上图 2.7 所示，假如最开始，我们在一座大山上的某处位置，因为到处都是陌生的，不知道下山的路，所以只能摸索着根据直觉，走一步算一步，在此过程中，每走到一个位置的时候，都会求解当前位置的梯度，沿着梯度的负方向，也就是当前最陡峭的位置向下走一步，然后继续求解当前位置梯度，向这一步所在位置沿着最陡峭最易下山的位置走一步。不断循环求梯度，就这样一步步地走下去，一直走到我们觉得已经到了山脚。当然这样走下去，有可能我们不能走到山脚，而是到了某一个局部的山势低处。由此，从上面的解释可以看出，梯度下降不一定能够找到全局的最优解，有可能是一个局部的最优解。当然，如果损失函数是凸函数，梯度下降法得到的解就一定是全局最优解。

核心思想归纳：

- (1) 初始化参数，随机选取取值范围内的任意数；
- (2) 迭代操作：a) 计算当前梯度；b) 修改新的变量；c) 计算朝最陡的下坡方向走一步；
- d) 判断是否需要终止，如否，返回 a)；

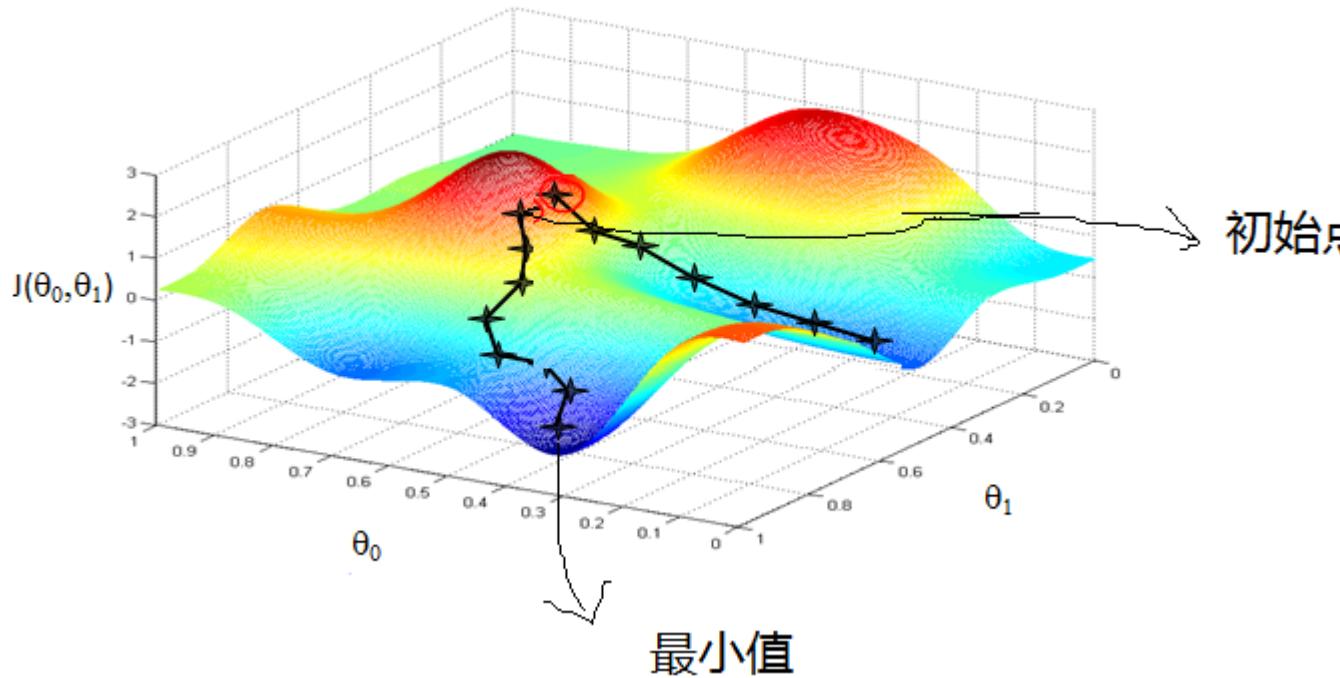


图 7.

(3) 得到全局最优解或者接近全局最优解.

7.4. 梯度下降法算法描述.

梯度下降法算法步骤如下:

(1) 确定优化模型的假设函数及损失函数. 举例, 对于线性回归, 假设函数为:

$$h_{\theta}(x_1, x_2, \dots, x_n) = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n$$

其中, $\theta_i, x_i (i = 0, 1, 2, \dots, n)$ 分别为模型参数、每个样本的特征值. 对于假设函数, 损失函数为:

$$J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{j=0}^m (h_{\theta}(x_0^{(j)}, x_1^{(j)}, \dots, x_n^{(j)}) - y_j)^2$$

(2) 相关参数初始化. 主要初始化 θ_i 、算法迭代步长 α 、终止距离 ζ . 初始化时可以根据经验初始化, 即 θ 初始化为 0, 步长 α 初始化为 1. 当前步长记为 φ_i . 当然, 也可随机初始化.

(3) 迭代计算.

1) 计算当前位置时损失函数的梯度, 对 θ_i , 其梯度表示为:

$$\frac{\partial}{\partial \theta_i} J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{j=0}^m (h_\theta(x_0^{(j)}, x_1^{(j)}, \dots, x_n^{(j)}) - y_j)^2$$

2) 计算当前位置下降的距离.

$$\varphi_i = \alpha \frac{\partial}{\partial \theta_i} J(\theta_0, \theta_1, \dots, \theta_n)$$

3) 判断是否终止. 确定是否所有 θ_i 梯度下降的距离 φ_i 都小于终止距离 ζ , 如果都小于 ζ , 则算法终止, 当然的值即为最终结果, 否则进入下一步. 4) 更新所有的 θ_i , 更新后的表达式为:

$$\theta_i = \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta_0, \theta_1, \dots, \theta_n)$$

$$\theta_i = \theta_i - \alpha \frac{1}{m} \sum_{j=0}^m (h_\theta(x_0^{(j)}, x_1^{(j)}, \dots, x_n^{(j)}) - y_j) x_i^{(j)}$$

5) 令上式 $x_0^{(j)} = 1$, 更新完毕后转入 1). 由此, 可看出, 当前位置的梯度方向由所有样本决定, 上式中 $\frac{1}{m}$ 、 $\alpha \frac{1}{m}$ 的目的是为了便于理解.

7.5. 如何对梯度下降法进行调优. 实际使用梯度下降法时, 各项参数指标不能一步就达到理想状态, 对梯度下降法调优主要体现在以下几个方面:

(1) **算法迭代步长 α 选择.** 在算法参数初始化时, 有时根据经验将步长初始化为 1. 实际取值取决于数据样本. 可以从大到小, 多取一些值, 分别运行算法看迭代效果, 如果损失函数在变小, 则取值有效. 如果取值无效, 说明要增大步长. 但步长太大, 有时会导致迭代速度过快, 错过最优解. 步长太小, 迭代速度慢, 算法运行时间长.

(2) **参数的初始值选择.** 初始值不同, 获得的最小值也有可能不同, 梯度下降有可能得到的是局部最小值. 如果损失函数是凸函数, 则一定是最优解. 由于有局部最优解的风险, 需要多次用不同初始值运行算法, 关键损失函数的最小值, 选择损失函数最小化的初值.

(3) **标准化处理.** 由于样本不同, 特征取值范围也不同, 导致迭代速度慢. 为了减少特征取值的影响, 可对特征数据标准化, 使新期望为 0, 新方差为 1, 可节省算法运行时间.

7.6. 随机梯度和批量梯度区别. 随机梯度下降 (SGD) 和批量梯度下降 (BGD) 是两种主要梯度下降法, 其目的是增加某些限制来加速运算求解. 下面通过介绍两种梯度下降法的求解思路, 对其进行比较. 假设函数为:

$$h_\theta(x_0, x_1, \dots, x_n) = \theta_0 x_0 + \theta_1 x_1 + \dots + \theta_n x_n$$

损失函数为:

$$J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{j=0}^m (h_\theta(x_0^j, x_1^j, \dots, x_n^j) - y^j)^2$$

其中, m 为样本个数, j 为参数个数.

1、批量梯度下降的求解思路如下: a) 得到每个 θ 对应的梯度:

$$\frac{\partial}{\partial \theta_i} J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{m} \sum_{j=0}^m (h_\theta(x_0^j, x_1^j, \dots, x_n^j) - y^j) x_i^j$$

b) 由于是求最小化风险函数, 所以按每个参数 θ 的梯度负方向更新 θ_i :

$$\theta_i = \theta_i - \frac{1}{m} \sum_{j=0}^m (h_\theta(x_0^j, x_1^j, \dots, x_n^j) - y^j) x_i^j$$

c) 从上式可以注意到, 它得到的虽然只是一个全局最优解, 但每迭代一步, 都要用到训练集所有的数据, 如果样本数据很大, 这种方法迭代速度就很慢. 相比而言, 随机梯度下降可避免这种问题.

2、随机梯度下降的求解思路如下: a) 相比批量梯度下降对应所有的训练样本, 随机梯度下降法中损失函数对应的是训练集中每个样本的粒度. 损失函数可以写成如下这种形式,

$$J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{m} \sum_{j=0}^m (y^j - h_\theta(x_0^j, x_1^j, \dots, x_n^j))^2 = \frac{1}{m} \sum_{j=0}^m cost(\theta, (x^j, y^j))$$

b) 对每个参数 θ 按梯度方向更新 θ :

$$\theta_i = \theta_i + (y^j - h_\theta(x_0^j, x_1^j, \dots, x_n^j))$$

c) 随机梯度下降是通过每个样本来迭代更新一次. 随机梯度下降伴随的一个问题是噪音较批量梯度下降要多, 使得随机梯度下降并不是每次迭代都向着整体最优化方向.

小结: 随机梯度下降法、批量梯度下降法相对来说都比较极端, 简单对比如下:

方法	特点
批量梯度下降	a) 采用所有数据来梯度下降.b) 批量梯度下降法在样本量很大的时候, 训练速度慢.
随机梯度下降	a) 随机梯度下降用一个样本来梯度下降.b) 训练速度很快.c) 随机梯度下降法仅仅用一个样本决定梯度方向, 导致解有可能不是全局最优.d) 收敛速度来说, 随机梯度下降法一次迭代一个样本, 导致迭代方向变化很大, 不能很快的收敛到局部最优解.

下面介绍能结合两种方法优点的小批量梯度下降法.

3、小批量 (Mini-Batch) 梯度下降的求解思路如下对于总数为 m 个样本的数据, 根据样本的数据, 选取其中的 $n(1 < n < m)$ 个子样本来迭代. 其参数 θ 按梯度方向更新 θ_i 公式如

下:

$$\theta_i = \theta_i - \alpha \sum_{j=t}^{t+n-1} (h_\theta(x_0^j, x_1^j, \dots, x_n^j) - y^j) x_i^j$$

7.7. 2.12.7 各种梯度下降法性能比较. 下表简单对比随机梯度下降 (SGD)、批量梯度下降 (BGD)、小批量梯度下降 (Mini-batch GD)、和 Online GD 的区别:

	BGD	SGD	Mini-batch GD	Online GD
训练集	固定	固定	固定	实时更新
单次迭代样本数	整个训练集	单个样本	训练集的子集	根据具体算法定
算法复杂度	高	低	一般	低
时效性	低	一般	一般	高
收敛性	稳定	不稳定	较稳定	不稳定

BGD、SGD、Mini-batch GD，前面均已讨论过，这里介绍一下 Online GD.

Online GD 于 Mini-batch GD/SGD 的区别在于，所有训练数据只用一次，然后丢弃. 这样做的优点在于可预测最终模型的变化趋势.

Online GD 在互联网领域用的较多，比如搜索广告的点击率 (CTR) 预估模型，网民的点击行为会随着时间改变. 用普通的 BGD 算法（每天更新一次）一方面耗时较长（需要对所有历史数据重新训练）；另一方面，无法及时反馈用户的点击行为迁移. 而 Online GD 算法可以实时的依据网民的点击行为进行迁移.

8. 2.14 线性判别分析 (LDA)

8.1. 2.14.1 LDA 思想总结. 线性判别分析 (Linear Discriminant Analysis, LDA) 是一种经典的降维方法. 和主成分分析 PCA 不考虑样本类别输出的无监督降维技术不同，LDA 是一种监督学习的降维技术，数据集的每个样本有类别输出.

LDA 分类思想简单总结如下:

1. 多维空间中，数据处理分类问题较为复杂，LDA 算法将多维空间中的数据投影到一条直线上，将 d 维数据转化成 1 维数据进行处理.
2. 对于训练数据，设法将多维数据投影到一条直线上，同类数据的投影点尽可能接近，异类数据点尽可能远离.
3. 对数据进行分类时，将其投影到同样的这条直线上，再根据投影点的位置来确定样本的类别.

如果用一句话概括 LDA 思想，即“投影后类内方差最小，类间方差最大”.

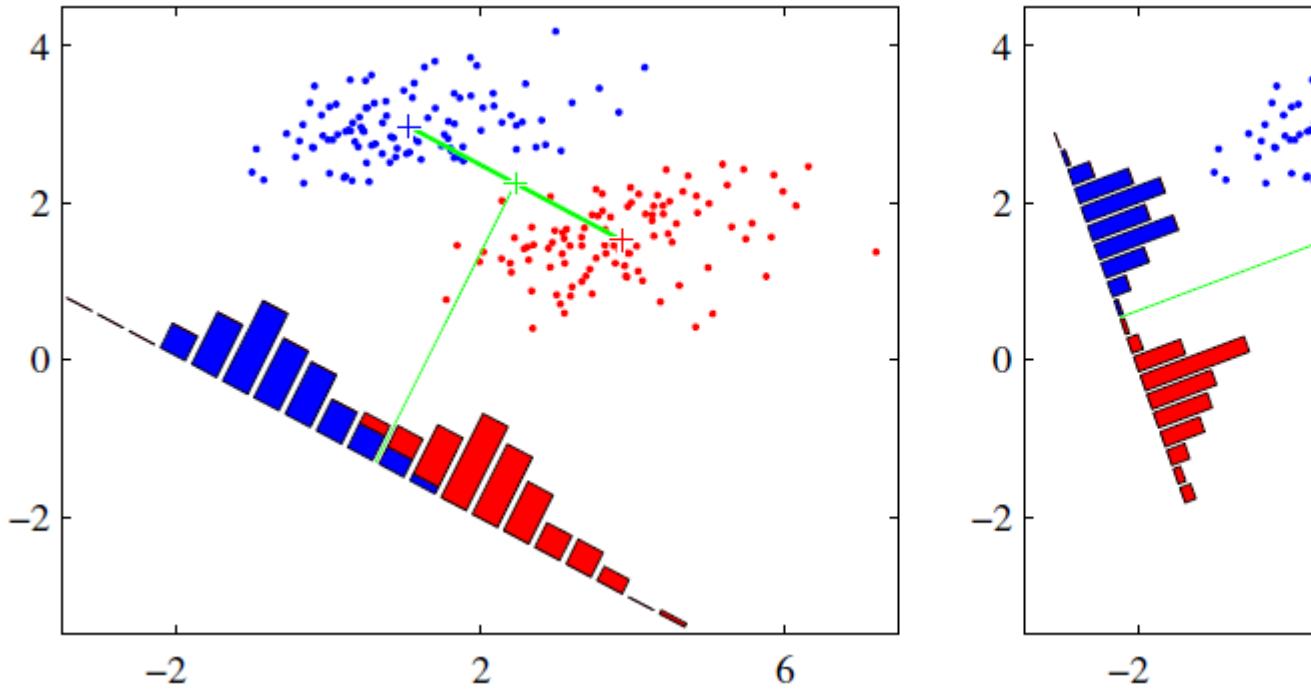


图 8.

8.2. 2.14.2 图解 LDA 核心思想. 假设有红、蓝两类数据，这些数据特征均为二维，如下图所示。我们的目标是将这些数据投影到一维，让每一类相近的数据的投影点尽可能接近，不同类别数据尽可能远，即图中红色和蓝色数据中心之间的距离尽可能大。

左图和右图是两种不同的投影方式。

左图思路：让不同类别的平均点距离最远的投影方式。

右图思路：让同类别的数据挨得最近的投影方式。

从上图直观看出，右图红色数据和蓝色数据在各自的区域来说相对集中，根据数据分布直方图也可看出，所以右图的投影效果好于左图，左图中间直方图部分有明显交集。

以上例子是基于数据是二维的，分类后的投影是一条直线。如果原始数据是多维的，则投影后的分类面是一低维的超平面。

8.3. 2.14.3 二类 LDA 算法原理. 输入：数据集 $D = (\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \dots, (\mathbf{x}_m, \mathbf{y}_m)$ ，其中样本 \mathbf{x}_i 是 n 维向量， $\mathbf{y}_i \in \{0, 1\}$ ，降维后的目标维度 d 。定义

$N_j (j = 0, 1)$ 为第 j 类样本个数；

$X_j (j = 0, 1)$ 为第 j 类样本的集合；

$u_j (j = 0, 1)$ 为第 j 类样本的均值向量；

$\sum_j (j = 0, 1)$ 为第 j 类样本的协方差矩阵.

其中

$$u_j = \frac{1}{N_j} \sum_{\mathbf{x} \in X_j} \mathbf{x} (j = 0, 1) \sum_j = \sum_{\mathbf{x} \in X_j} (\mathbf{x} - u_j)(\mathbf{x} - u_j)^T (j = 0, 1)$$

假设投影直线是向量 \mathbf{w} , 对任意样本 \mathbf{x}_i , 它在直线 w 上的投影为 $\mathbf{w}^T \mathbf{x}_i$, 两个类别的中心点 u_0, u_1 在直线 w 的投影分别为 $\mathbf{w}^T u_0, \mathbf{w}^T u_1$.

LDA 的目标是让两类别的数据中心间的距离 $\|\mathbf{w}^T u_0 - \mathbf{w}^T u_1\|_2^2$ 尽量大, 与此同时, 希望同类样本投影点的协方差 $\mathbf{w}^T \sum_0 \mathbf{w}, \mathbf{w}^T \sum_1 \mathbf{w}$ 尽量小, 最小化 $\mathbf{w}^T \sum_0 \mathbf{w} + \mathbf{w}^T \sum_1 \mathbf{w}$. 定义类内散度矩阵

$$S_w = \sum_0 + \sum_1 = \sum_{\mathbf{x} \in X_0} (\mathbf{x} - u_0)(\mathbf{x} - u_0)^T + \sum_{\mathbf{x} \in X_1} (\mathbf{x} - u_1)(\mathbf{x} - u_1)^T$$

类间散度矩阵 $S_b = (u_0 - u_1)(u_0 - u_1)^T$

据上分析, 优化目标为

$$\arg \max$$

$\mathbf{w}^T \mathbf{J}(\mathbf{w}) = \|\mathbf{w}^T u_0 - \mathbf{w}^T u_1\|_2^2 \frac{\mathbf{w}^T \sum_0 \mathbf{w} + \mathbf{w}^T \sum_1 \mathbf{w}}{\mathbf{w}^T \sum_0 \mathbf{w} + \mathbf{w}^T \sum_1 \mathbf{w}} = \frac{\mathbf{w}^T (u_0 - u_1)(u_0 - u_1)^T \mathbf{w}}{\mathbf{w}^T (\sum_0 + \sum_1) \mathbf{w}} = \frac{\mathbf{w}^T S_b \mathbf{w}}{\mathbf{w}^T S_w \mathbf{w}}$ 根据广义瑞利商的性质, 矩阵 $S_w^{-1} S_b$ 的最大特征值为 $J(\mathbf{w})$ 的最大值, 矩阵 $S_w^{-1} S_b$ 的最大特征值对应的特征向量即为 \mathbf{w} .

8.4. LDA 算法流程总结. LDA 算法降维流程如下:

输入: 数据集 $D = (x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$, 其中样本 x_i 是 n 维向量, $y_i \in C_1, C_2, \dots, C_k$, 降维后的目标维度 d .

输出: 降维后的数据集 \bar{D} .

步骤: 1. 计算类内散度矩阵 S_w . 2. 计算类间散度矩阵 S_b . 3. 计算矩阵 $S_w^{-1} S_b$. 4. 计算矩阵 $S_w^{-1} S_b$ 的最大的 d 个特征值. 5. 计算 d 个特征值对应的 d 个特征向量, 记投影矩阵为 W . 6. 转化样本集的每个样本, 得到新样本 $P_i = W^T x_i$. 7. 输出新样本集 $\bar{D} = (p_1, y_1), (p_2, y_2), \dots, (p_m, y_m)$

异同点 LDA

相同点 1. 两者均可以对数据进行降维; 2. 两者在降维时均使用了矩阵特征分解的思想; 3. 两者都假设数据符合高斯分布.

不同点 有监督的降维方法;

降维最多降到 $k-1$ 维;

可以用于降维, 还可以用于分类;

选择分类性能最好的投影方向；
更明确，更能反映样本间差异；

8.5. LDA 和 PCA 区别.

优缺点	简要说明
-----	------

优点 1.

可

以

使

用

类

别

的

先

验

知

识;

2.

以

标

签、

类

别

衡

量

差

异

性

的

有

监

督

降

维

方

式,

相

对

于

PCA

的

模

糊

性,

其

目

的

更

缺点 1.

LDA

不适合对非高斯分布样本进行降维；

2.

LDA

降维最多降到分类数 $k-1$ 维；

3.

LDA

在样本分类信息依赖方差

8.6. LDA 优缺点.

9. 主成分分析 (PCA)

9.1. 主成分分析 (PCA) 思想总结.

1. PCA 就是将高维的数据通过线性变换投影到低维空间上去.
2. 投影思想: 找出最能够代表原始数据的投影方法. 被 PCA 降掉的那些维度只能是那些噪声或是冗余的数据.
3. 去冗余: 去除可以被其他向量代表的线性相关向量, 这部分信息量是多余的.
4. 去噪声, 去除较小特征值对应的特征向量, 特征值的大小反映了变换后在特征向量方向上变换的幅度, 幅度越大, 说明这个方向上的元素差异也越大, 要保留.
5. 对角化矩阵, 寻找极大线性无关组, 保留较大的特征值, 去除较小特征值, 组成一个投影矩阵, 对原始样本矩阵进行投影, 得到降维后的新样本矩阵.
6. 完成 PCA 的关键是——协方差矩阵. 协方差矩阵, 能同时表现不同维度间的关系以及各个维度上的方差. 协方差矩阵度量的是维度与维度之间的关系, 而非样本与样本之间.
7. 之所以对角化, 因为对角化之后非对角上的元素都是 0, 达到去噪声的目的. 对角化后的协方差矩阵, 对角线上较小的新方差对应的就是那些该去掉的维度. 所以我们只取那些含有较大能量 (特征值) 的维度, 其余的就舍掉, 即去冗余.

9.2. 图解 PCA 核心思想. PCA 可解决训练数据中存在数据特征过多或特征累赘的问题. 核心思想是将 m 维特征映射到 n 维 ($n < m$), 这 n 维形成主元, 是重构出来最能代表原始数据的正交特征.

假设数据集是 m 个 n 维, $(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(m)})$. 如果 $n = 2$, 需要降维到 $n' = 1$, 现在想找到某一维度方向代表这两个维度的数据. 下图有 u_1, u_2 两个向量方向, 但是哪个向量才是我们所想要的, 可以更好代表原始数据集的呢?

从图可看出, u_1 比 u_2 好, 为什么呢? 有以下两个主要评价指标: 1. 样本点到这个直线的距离足够近. 2. 样本点在这个直线上的投影能尽可能的分开.

如果我们需要降维的目标维数是其他任意维, 则: 1. 样本点到这个超平面的距离足够近. 2. 样本点在这个超平面上的投影能尽可能的分开.

9.3. PCA 算法推理. 下面以基于最小投影距离为评价指标推理:

假设数据集是 m 个 n 维, $(x^{(1)}, x^{(2)}, \dots, x^{(m)})$, 且数据进行了中心化. 经过投影变换得到新坐标为 w_1, w_2, \dots, w_n , 其中 w 是标准正交基, 即 $\|w\|_2 = 1$, $w_i^T w_j = 0$.

经过降维后, 新坐标为 w_1, w_2, \dots, w_n , 其中 n' 是降维后的目标维数. 样本点 $x^{(i)}$ 在新坐标系下的投影为 $z^{(i)} = (z_1^{(i)}, z_2^{(i)}, \dots, z_{n'}^{(i)})$, 其中 $z_j^{(i)} = w_j^T x^{(i)}$ 是 $x^{(i)}$ 在低维坐标系里第 j 维的坐标.

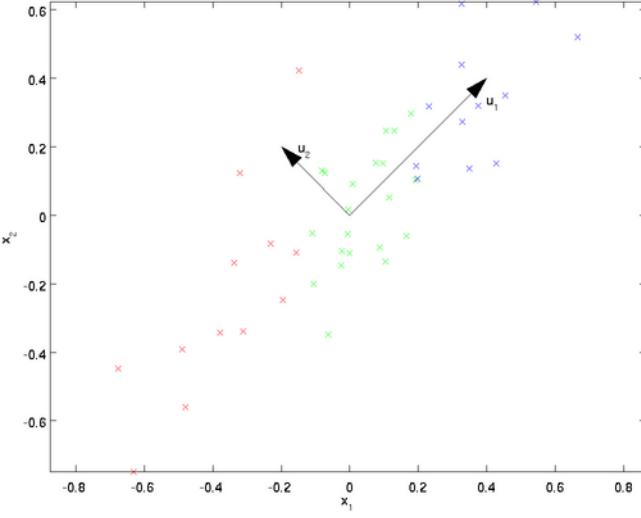


图 9.

如果用 $z^{(i)}$ 去恢复 $x^{(i)}$ ，则得到的恢复数据为 $\hat{x}^{(i)} = \sum_{j=1}^{n'} x_j^{(i)} w_j = W z^{(i)}$ ，其中 W 为标准正交基组成的矩阵。

考虑到整个样本集，样本点到这个超平面的距离足够近，目标变为最小化 $\sum_{i=1}^m \|\hat{x}^{(i)} - x^{(i)}\|_2^2$ 。对此式进行推理，可得：

$$\sum_{i=1}^m \|\hat{x}^{(i)} - x^{(i)}\|_2^2 = \sum_{i=1}^m \|W z^{(i)} - x^{(i)}\|_2^2 = \sum_{i=1}^m (W z^{(i)})^T (W z^{(i)}) - 2 \sum_{i=1}^m (W z^{(i)})^T x^{(i)} + \sum_{i=1}^m (x^{(i)})^T x^{(i)} = \sum_{i=1}^m (z^{(i)})^T$$

在推导过程中，分别用到了 $\bar{x}^{(i)} = W z^{(i)}$ ，矩阵转置公式 $(AB)^T = B^T A^T$ ， $W^T W = I$ ， $z^{(i)} = W^T x^{(i)}$ 以及矩阵的迹，最后两步是将代数和转为矩阵形式。由于 W 的每一个向量 w_j 是标准正交基， $\sum_{i=1}^m x^{(i)} (x^{(i)})^T$ 是数据集的协方差矩阵， $\sum_{i=1}^m x^{(i)} (x^{(i)})^T$ 是一个常量。最小化 $\sum_{i=1}^m \|\hat{x}^{(i)} - x^{(i)}\|_2^2$ 又可等价于

$$\underbrace{\arg \min_W}_{W} - \text{tr}(W^T X X^T W) \text{ s.t. } W^T W = I$$

利用拉格朗日函数可得到

$$J(W) = -\text{tr}(W^T X X^T W) + \lambda(W^T W - I)$$

对 W 求导，可得 $-X X^T W + \lambda W = 0$ ，也即 $X X^T W = \lambda W$ 。 $X X^T$ 是 n' 个特征向量组成的矩阵， λ 为 $X X^T$ 的特征值。 W 即为我们想要的矩阵。对于原始数据，只需要 $z^{(i)} = W^T X^{(i)}$ ，就可把原始数据集降维到最小投影距离的 n' 维数据集。

基于最大投影方差的推导，这里就不再赘述，有兴趣的同仁可自行查阅资料.

9.4. PCA 算法流程总结. 输入： n 维样本集 $D = (x^{(1)}, x^{(2)}, \dots, x^{(m)})$ ，目标降维的维数 n' .

输出：降维后的新样本集 $D' = (z^{(1)}, z^{(2)}, \dots, z^{(m)})$.

主要步骤如下：1. 对所有的样本进行中心化， $x^{(i)} = x^{(i)} - \frac{1}{m} \sum_{j=1}^m j x^{(j)}$. 2. 计算样本的协方差矩阵 XX^T . 3. 对协方差矩阵 XX^T 进行特征值分解. 4. 取出最大的 n' 个特征值对应的特征向量 $w_1, w_2, \dots, w_{n'}$. 5. 标准化特征向量，得到特征向量矩阵 W . 6. 转化样本集中的每个样本 $z^{(i)} = W^T x^{(i)}$. 7. 得到输出矩阵 $D' = (z^{(1)}, z^{(2)}, \dots, z^{(n)})$. 注：在降维时，有时不明确目标维数，而是指定降维到的主成分比重阈值 $k(k \in (0, 1])$. 假设 n 个特征值为 $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ ，则 n' 可从 $\sum_{i=1}^{n'} \lambda_i \geq k \times \sum_{i=1}^n \lambda_i$ 得到.

优缺	简要说明
说	
明	

优点 1. 仅仅需要以方差衡量信息量, 不受数据集以外的因素影响.

2. 各主成分之间正交, 可消除原始数据成

缺点 1. 主成分各个特征维度的含义具有一定模糊性, 不如原始样本特征的解释性强. 2.

方差小的非主成分也

9.5. PCA 算法主要优缺点.

9.6. 降维的必要性及目的. 降维的必要性: 1. 多重共线性和预测变量之间相互关联. 多重共线性会导致解空间的不稳定, 从而可能导致结果的不连贯. 2. 高维空间本身具有稀疏性. 一维正态分布有 68% 的值落于正负标准差之间, 而在十维空间上只有 2%. 3. 过多的变量, 对查找规律造成冗余麻烦. 4. 仅在变量层面上分析可能会忽略变量之间的潜在联系. 例如几个预测变量可能落入仅反映数据某一方面特征的一个组内.

降维的目的: 1. 减少预测变量的个数. 2. 确保这些变量是相互独立的. 3. 提供一个框架来解释结果. 相关特征, 特别是重要特征更能在数据中明确的显示出来; 如果只有两维或者三维的话, 更便于可视化展示. 4. 数据在低维下更容易处理、更容易使用. 5. 去除数据噪声. 6. 降低算法运算开销.

9.7. KPCA 与 PCA 的区别. 应用 PCA 算法前提是假设存在一个线性超平面, 进而投影. 那如果数据不是线性的呢? 该怎么办? 这时候就需要 KPCA, 数据集从 n 维映射到线性可分的高维 $N > n$, 然后再从 N 维降维到一个低维度 $n'(n' < n < N)$.

KPCA 用到了核函数思想, 使用了核函数的主成分分析一般称为核主成分分析 (Kernelized PCA, 简称 KPCA).

假设高维空间数据由 n 维空间的数据通过映射 ϕ 产生.

n 维空间的特征分解为:

$$\sum_{i=1}^m x^{(i)} (x^{(i)})^T W = \lambda W$$

其映射为

$$\sum_{i=1}^m \phi(x^{(i)}) \phi(x^{(i)})^T W = \lambda W$$

通过在高维空间进行协方差矩阵的特征值分解, 然后用和 PCA 一样的方法进行降维. 由于 KPCA 需要核函数的运算, 因此它的计算量要比 PCA 大很多.

10. 模型评估

10.1. 模型评估常用方法? 一般情况下来说, 单一评分标准无法完全评估一个机器学习模型. 只用 good 和 bad 偏离真实场景去评估某个模型, 都是一种欠妥的评估方式. 下面介绍常用的分类模型和回归模型评估方法.

分类模型常用评估方法:

指标	描述
Accuracy	准确率
Precision	精准度/查准率

Recall	召回率/查全率
P-R 曲线	查准率为纵轴, 查全率为横轴, 作图
F1	F1 值
Confusion Matrix	混淆矩阵
ROC	ROC 曲线
AUC	ROC 曲线下的面积

回归模型常用评估方法:

指标	描述
Mean Square Error (MSE, RMSE)	平均方差
Absolute Error (MAE, RAE)	绝对误差
R-Squared	R 平方值

10.2. 2.16.2 误差、偏差和方差有什么区别和联系. 在机器学习中, Bias(偏差), Error(误差), 和 Variance(方差) 存在以下区别和联系:

对于 Error :

- 误差 (error): 一般地, 我们把学习器的实际预测输出与样本的真是输出之间的差异称为“误差”.
- $\text{Error} = \text{Bias} + \text{Variance} + \text{Noise}$, Error 反映的是整个模型的准确度.

对于 Noise:

噪声: 描述了在当前任务上任何学习算法所能达到的期望泛化误差的下界, 即刻画了学习问题本身的难度.

对于 Bias:

- Bias 衡量模型拟合训练数据的能力 (训练数据不一定是整个 training dataset, 而是只用于训练它的那一部分数据, 例如: mini-batch), Bias 反映的是模型在样本上的输出与真实值之间的误差, 即模型本身的精准度.
- Bias 越小, 拟合能力越高 (可能产生 overfitting); 反之, 拟合能力越低 (可能产生 underfitting)
- 偏差越大, 越偏离真实数据, 如下图第二行所示.

对于 Variance:

- 方差公式: $S_N^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2$
- Variance 描述的是预测值的变化范围, 离散程度, 也就是离其期望值的距离. 方差越大, 数据的分布越分散, 模型的稳定程度越差.

- Variance 反映的是模型每一次输出结果与模型输出期望之间的误差，即模型的稳定性.
- Variance 越小，模型的泛化的能力越高；反之，模型的泛化的能力越低.
- 如果模型在训练集上拟合效果比较优秀，但是在测试集上拟合效果比较差劣，则方差较大，说明模型的稳定程度较差，出现这种现象可能是由于模型对训练集过拟合造成的. 如下图右列所示.

10.3. 经验误差与泛化误差. 经验误差 (empirical error): 也叫训练误差 (training error)，模型在训练集上的误差.

泛化误差 (generalization error): 模型在新样本集 (测试集) 上的误差称为“泛化误差”.

10.4. 图解欠拟合、过拟合. 根据不同的坐标方式，欠拟合与过拟合图解不同. 1. **横轴为训练样本数量，纵轴为误差**

如上图所示，我们可以直观看出欠拟合和过拟合的区别：

模型欠拟合：在训练集以及测试集上同时具有较高的误差，此时模型的偏差较大；

模型过拟合：在训练集上具有较低的误差，在测试集上具有较高的误差，此时模型的方差较大.

模型正常：在训练集以及测试集上，同时具有相对较低的偏差以及方差.

2. 横轴为模型复杂程度，纵轴为误差

红线为测试集上的 Error, 蓝线为训练集上的 Error

模型欠拟合：模型在点 A 处，在训练集以及测试集上同时具有较高的误差，此时模型的偏差较大.

模型过拟合：模型在点 C 处，在训练集上具有较低的误差，在测试集上具有较高的误差，此时模型的方差较大.

模型正常：模型复杂程度控制在点 B 处为最优.

3. 横轴为正则项系数，纵轴为误差

红线为测试集上的 Error, 蓝线为训练集上的 Error

模型欠拟合：模型在点 C 处，在训练集以及测试集上同时具有较高的误差，此时模型的偏差较大.

模型过拟合：模型在点 A 处，在训练集上具有较低的误差，在测试集上具有较高的误差，此时模型的方差较大. 它通常发生在模型过于复杂的情况下，如参数过多等，会使得模型的预测性能变弱，并且增加数据的波动性. 虽然模型在训练时的效果可以表现的很完美，基本上记住了数据的全部特点，但这种模型在未知数据的表现能力会大减折扣，因为简单的模型泛化能力通常都是很弱的.

模型正常：模型复杂程度控制在点 B 处为最优.

10.5. 如何解决过拟合与欠拟合. 如何解决欠拟合: 1. 添加其他特征项. 组合、泛化、相关性、上下文特征、平台特征等特征是特征添加的重要手段，有时候特征项不够会导致模型欠拟合。2. 添加多项式特征。例如将线性模型添加二次项或三次项使模型泛化能力更强。例如，FM (Factorization Machine) 模型、FFM (Field-aware Factorization Machine) 模型，其实都是线性模型，增加了二阶多项式，保证了模型一定的拟合程度。3. 可以增加模型的复杂程度。4. 减小正则化系数。正则化的目的是用来防止过拟合的，但是现在模型出现了欠拟合，则需要减少正则化参数。

如何解决过拟合: 1. 重新清洗数据，数据不纯会导致过拟合，此类情况需要重新清洗数据。2. 增加训练样本数量。3. 降低模型复杂程度。4. 增大正则项系数。5. 采用 dropout 方法，dropout 方法，通俗的讲就是在训练的时候让神经元以一定的概率不工作。6. early stopping。7. 减少迭代次数。8. 增大学习率。9. 添加噪声数据。10. 树结构中，可以对树进行剪枝。11. 减少特征项。

欠拟合和过拟合这些方法，需要根据实际问题，实际模型，进行选择。

10.6. 交叉验证的主要作用. 为了得到更为稳健可靠的模型，对模型的泛化误差进行评估，得到模型泛化误差的近似值。当有多个模型可以选择时，我们通常选择“泛化误差”最小的模型。

交叉验证的方法有许多种，但是最常用的是：留一交叉验证、 k 折交叉验证。

10.7. 理解 k 折交叉验证。

1. 将含有 N 个样本的数据集，分成 K 份，每份含有 N/K 个样本。选择其中 1 份作为测试集，另外 $K-1$ 份作为训练集，测试集就有 K 种情况。
2. 在每种情况下，用训练集训练模型，用测试集测试模型，计算模型的泛化误差。
3. 交叉验证重复 K 次，每份验证一次，平均 K 次的结果或者使用其它结合方式，最终得到一个单一估测，得到模型最终的泛化误差。
4. 将 K 种情况下，模型的泛化误差取均值，得到模型最终的泛化误差。

5. 一般 $2 \leq K \leq 10$ 。 k 折交叉验证的优势在于，同时重复运用随机产生的子样本进行训练和验证，每次的结果验证一次，10 折交叉验证是最常用的。
6. 训练集中样本数量要足够多，一般至少大于总样本数的 50%。
7. 训练集和测试集必须从完整的数据集中均匀取样。均匀取样的目的是希望减少训练集、测试集与原数据集之间的偏差。当样本数量足够多时，通过随机取样，便可以实现均匀取样的效果。

10.8. 混淆矩阵. 第一种混淆矩阵:

真实情况 T or F	预测为正例 1, P	预测为负例 0, N
-------------	------------	------------

本来 label 标记为 1, 预测结果真为 T、假为 F TP(预测为 1, 实际为 1) FN(预测为 0, 实际为 1)
 本来 label 标记为 0, 预测结果真为 T、假为 F FP(预测为 1, 实际为 0) TN(预测为 0, 实际也为 0)

第二种混淆矩阵:

预测情况 P or N	实际 label 为 1, 预测对了为 T	实际 label 为 0, 预测对了为 T
预测为正例 1, P	TP(预测为 1, 实际为 1)	FP(预测为 1, 实际为 0)
预测为负例 0, N	FN(预测为 0, 实际为 1)	TN(预测为 0, 实际也为 0)

10.9. 错误率及精度.

1. 错误率 (Error Rate): 分类错误的样本数占样本总数的比例.
2. 精度 (accuracy): 分类正确的样本数占样本总数的比例.

10.10. 查准率与查全率. 将算法预测的结果分成四种情况: 1. 正确肯定 (True Positive,TP) : 预测为真, 实际为真 2. 正确否定 (True Negative,TN) : 预测为假, 实际为假 3. 错误肯定 (False Positive,FP): 预测为真, 实际为假 4. 错误否定 (False Negative,FN): 预测为假, 实际为真

则:

查准率 (Precision) = $TP / (TP+FP)$

理解: 预测出为阳性的样本中, 正确的有多少. 区别准确率 (正确预测出的样本, 包括正确预测为阳性、阴性, 占总样本比例). 例, 在所有我们预测有恶性肿瘤的病人中, 实际上有恶性肿瘤的病人的百分比, 越高越好.

查全率 (Recall) = $TP / (TP+FN)$

理解: 正确预测为阳性的数量占总样本中阳性数量的比例. 例, 在所有实际上有恶性肿瘤的病人中, 成功预测有恶性肿瘤的病人的百分比, 越高越好.

10.11. ROC 与 AUC. ROC 全称是“受试者工作特征” (Receiver Operating Characteristic) .

ROC 曲线的面积就是 AUC (Area Under Curve) .

AUC 用于衡量“二分类问题”机器学习算法性能 (泛化能力) .

ROC 曲线, 通过将连续变量设定出多个不同的临界值, 从而计算出一系列真正率和假正率, 再以假正率为横坐标、真正率为纵坐标绘制成曲线, 曲线下面积越大, 推断准确性越高. 在 ROC 曲线上, 最靠近坐标图左上方的点为假正率和真正率均较高的临界值.

对于分类器, 或者说分类算法, 评价指标主要有 Precision, Recall, F-score. 下图是一个 ROC 曲线的示例.

ROC 曲线的横坐标为 False Positive Rate (FPR)，纵坐标为 True Positive Rate (TPR) . 其中

$$TPR = \frac{TP}{TP + FN}, FPR = \frac{FP}{FP + TN}$$

下面着重介绍 ROC 曲线图中的四个点和一条线. 第一个点 (0,1), 即 $FPR=0$, $TPR=1$, 这意味着 FN (False Negative) =0, 并且 FP (False Positive) =0. 意味着这是一个完美的分类器, 它将所有的样本都正确分类. 第二个点 (1,0), 即 $FPR=1$, $TPR=0$, 意味着这是一个最糟糕的分类器, 因为它成功避开了所有的正确答案. 第三个点 (0,0), 即 $FPR=TPR=0$, 即 FP (False Positive) = TP (True Positive) =0, 可以发现该分类器预测所有的样本都为负样本 (Negative). 第四个点 (1,1), 即 $FPR=TPR=1$, 分类器实际上预测所有的样本都为正样本. 经过以上分析, ROC 曲线越接近左上角, 该分类器的性能越好.

ROC 曲线所覆盖的面积称为 AUC (Area Under Curve), 可以更直观的判断学习器的性能, AUC 越大则性能越好.

10.12. 如何画 ROC 曲线. 下图是一个示例, 图中共有 20 个测试样本, “Class”一栏表示每个测试样本真正的标签 (p 表示正样本, n 表示负样本), “Score”表示每个测试样本属于正样本的概率.

步骤: 1、假设已经得出一系列样本被划分为正类的概率, 按照大小排序. 2、从高到低, 依次将“Score”值作为阈值 threshold, 当测试样本属于正样本的概率大于或等于这个 threshold 时, 我们认为它为正样本, 否则为负样本. 举例来说, 对于图中的第 4 个样本, 其“Score”值为 0.6, 那么样本 1, 2, 3, 4 都被认为是正样本, 因为它们的“Score”值都大于等于 0.6, 而其他样本则都认为是负样本. 3、每次选取一个不同的 threshold, 得到一组 FPR 和 TPR, 即 ROC 曲线上的一点. 以此共得到 20 组 FPR 和 TPR 的值. 4、根据 3、中的每个坐标点, 画图.

10.13. 如何计算 TPR, FPR. 1、分析数据 $y_true = [0, 0, 1, 1]$; $scores = [0.1, 0.4, 0.35, 0.8]$; 2、列表

样本	预测属于 P 的概率 (score)	真实类别
$y[0]$	0.1	N
$y[1]$	0.4	N
$y[2]$	0.35	P
$y[3]$	0.8	P

3、将截断点依次取为 score 值, 计算 TPR 和 FPR. 当截断点为 0.1 时: 说明只要 $score >= 0.1$, 它的预测类别就是正例. 因为 4 个样本的 score 都大于等于 0.1, 所以, 所有样本的预测类别都为 P. $scores = [0.1, 0.4, 0.35, 0.8]$; $y_true = [0, 0, 1, 1]$; $y_pred = [1, 1, 1, 1]$; 正例与反例信息如下:

	正例	反例
正例	TP=2	FN=0
反例	FP=2	TN=0

由此可得: $TPR = TP/(TP+FN) = 1$; $FPR = FP/(TN+FP) = 1$;

当截断点为 0.35 时: $scores = [0.1, 0.4, 0.35, 0.8]$; $y_true = [0, 0, 1, 1]$; $y_pred = [0, 1, 1, 1]$; 正例与反例信息如下:

	正例	反例
正例	TP=2	FN=0
反例	FP=1	TN=1

由此可得: $TPR = TP/(TP+FN) = 1$; $FPR = FP/(TN+FP) = 0.5$;

当截断点为 0.4 时: $scores = [0.1, 0.4, 0.35, 0.8]$; $y_true = [0, 0, 1, 1]$; $y_pred = [0, 1, 0, 1]$; 正例与反例信息如下:

	正例	反例
正例	TP=1	FN=1
反例	FP=1	TN=1

由此可得: $TPR = TP/(TP+FN) = 0.5$; $FPR = FP/(TN+FP) = 0.5$;

当截断点为 0.8 时: $scores = [0.1, 0.4, 0.35, 0.8]$; $y_true = [0, 0, 1, 1]$; $y_pred = [0, 0, 0, 1]$;

正例与反例信息如下:

	正例	反例
正例	TP=1	FN=1
反例	FP=0	TN=2

由此可得: $TPR = TP/(TP+FN) = 0.5$; $FPR = FP/(TN+FP) = 0$;

4、根据 TPR、FPR 值, 以 FPR 为横轴, TPR 为纵轴画图.

10.14. 如何计算 AUC.

- 将坐标点按照横坐标 FPR 排序.
- 计算第 i 个坐标点和第 $i + 1$ 个坐标点的间距 dx .
- 获取第 i 或者 $i + 1$ 个坐标点的纵坐标 y .
- 计算面积微元 $ds = ydx$.

- 对面积微元进行累加，得到 AUC.

10.15. 为什么使用 Roc 和 Auc 评价分类器. 模型有很多评估方法，为什么还要使用 ROC 和 AUC 呢？因为 ROC 曲线有个很好的特性：当测试集中的正负样本的分布变换的时候，ROC 曲线能够保持不变。在实际的数据集中经常会出现样本类不平衡，即正负样本比例差距较大，而且测试数据中的正负样本也可能随着时间变化。

10.16. 直观理解 AUC. 下图展现了三种 AUC 的值：

AUC 是衡量二分类模型优劣的一种评价指标，表示正例排在负例前面的概率。其他评价指标有精确度、准确率、召回率，而 AUC 比这三者更为常用。一般在分类模型中，预测结果都是以概率的形式表现，如果要计算准确率，通常都会手动设置一个阈值来将对应的概率转化成类别，这个阈值也就很大程度上影响了模型准确率的计算。举例：现在假设有一个训练好的二分类器对 10 个正负样本（正例 5 个，负例 5 个）预测，得分按高到低排序得到的最好预测结果为 [1, 1, 1, 1, 1, 0, 0, 0, 0, 0]，即 5 个正例均排在 5 个负例前面，正例排在负例前面的概率为 100%。然后绘制其 ROC 曲线，由于是 10 个样本，除去原点我们需要描 10 个点，如下：

描点方式按照样本预测结果的得分高低从左至右开始遍历。从原点开始，每遇到 1 便向 y 轴正方向移动 y 轴最小步长 1 个单位，这里是 $1/5=0.2$ ；每遇到 0 则向 x 轴正方向移动 x 轴最小步长 1 个单位，这里也是 0.2。不难看出，上图的 AUC 等于 1，印证了正例排在负例前面的概率的确为 100%。

假设预测结果序列为 [1, 1, 1, 1, 0, 1, 0, 0, 0, 0]。

计算上图的 AUC 为 0.96 与计算正例与排在负例前面的概率 $0.8 \times 1 + 0.2 \times 0.8 = 0.96$ 相等，而左上角阴影部分的面积则是负例排在正例前面的概率 $0.2 \times 0.2 = 0.04$ 。

假设预测结果序列为 [1, 1, 1, 0, 1, 0, 1, 0, 0, 0]。

计算上图的 AUC 为 0.88 与计算正例与排在负例前面的概率 $0.6 \times 1 + 0.2 \times 0.8 + 0.2 \times 0.6 = 0.88$ 相等，左上角阴影部分的面积是负例排在正例前面的概率 $0.2 \times 0.2 \times 3 = 0.12$ 。

10.17. 代价敏感错误率与代价曲线. 不同的错误会产生不同代价。以二分法为例，设置代价矩阵如下：

当判断正确的时候，值为 0，不正确的时候，分别为 $Cost_{01}$ 和 $Cost_{10}$ 。

$Cost_{10}$: 表示实际为反例但预测成正例的代价。

$Cost_{01}$: 表示实际为正例但是预测为反例的代价。

代价敏感错误率 = 样本中由模型得到的错误值与代价乘积之和 / 总样本. 其数学表达式为:

$$E(f; D; cost) = \frac{1}{m} \left(\sum_{x_i \in D^+} (f(x_i) \neq y_i) \times Cost_{01} + \sum_{x_i \in D^-} (f(x_i) \neq y_i) \times Cost_{10} \right)$$

D^+ D^- 分别代表样例集的正例子集和反例子集, x 是预测值, y 是真实值.

代价曲线: 在均等代价时, ROC 曲线不能直接反应出模型的期望总体代价, 而代价曲线可以. 代价曲线横轴为 [0,1] 的正例函数代价:

$$P(+)Cost = \frac{p * Cost_{01}}{p * Cost_{01} + (1 - p) * Cost_{10}}$$

其中 p 是样本为正例的概率.

代价曲线纵轴维 [0,1] 的归一化代价:

$$Cost_{norm} = \frac{FNR * p * Cost_{01} + FNR * (1 - p) * Cost_{10}}{p * Cost_{01} + (1 - p) * Cost_{10}}$$

其中 FPR 为假阳率, $FNR=1-TPR$ 为假阴率.

注: ROC 每个点, 对应代价平面上一条线.

例如, ROC 上 (TPR,FPR), 计算出 $FNR=1-TPR$, 在代价平面上绘制一条从 (0,FPR) 到 (1,FNR) 的线段, 面积则为该条件下期望的总体代价. 所有线段下界面积, 所有条件下的学习器的期望总体代价.

10.18. 模型有哪些比较检验方法. 正确性分析: 模型稳定性分析, 稳健性分析, 收敛性分析, 变化趋势分析, 极值分析等. 有效性分析: 误差分析, 参数敏感性分析, 模型对比检验等. 有用性分析: 关键数据求解, 极值点, 拐点, 变化趋势分析, 用数据验证动态模拟等. 高效性分析: 时空复杂度分析与现有进行比较等.

10.19. 为什么使用标准差. 方差公式为: $S_N^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2$

标准差公式为: $S_N = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2}$

样本标准差公式为: $S_N = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$

与方差相比, 使用标准差来表示数据点的离散程度有 3 个好处: 1、表示离散程度的数字与样本数据点的数量级一致, 更适合对数据样本形成感性认知.

2、表示离散程度的数字单位与样本数据的单位一致, 更方便做后续的分析运算.

3、在样本数据大致符合正态分布的情况下, 标准差具有方便估算的特性: 68% 的数据点落在平均值前后 1 个标准差的范围内、95% 的数据点落在平均值前后 2 个标准差的范围内, 而 99% 的数据点将会落在平均值前后 3 个标准差的范围内.

10.20. 类别不平衡产生原因. 类别不平衡 (class-imbalance) 是指分类任务中不同类别的训练样例数目差别很大的情况.

产生原因:

分类学习算法通常都会假设不同类别的训练样例数目基本相同. 如果不同类别的训练样例数目差别很大, 则会影响学习结果, 测试结果变差. 例如二分类问题中有 998 个反例, 正例有 2 个, 那学习方法只需返回一个永远将新样本预测为反例的分类器, 就能达到 99.8% 的精度; 然而这样的分类器没有价值. [# # # 2.16.21 常见的类别不平衡问题解决方法](#) 防止类别不平衡对学习造成的影响, 在构建分类模型之前, 需要对分类不平衡性问题进行处理. 主要解决方法有:

1、扩大数据集

增加包含小类样本数据的数据, 更多的数据能得到更多的分布信息.

2、对大类数据欠采样

减少大类数据样本个数, 使与小样本个数接近. 缺点: 欠采样操作时若随机丢弃大类样本, 可能会丢失重要信息. 代表算法: EasyEnsemble. 其思想是利用集成学习机制, 将大类划分为若干个集合供不同的学习器使用. 相当于对每个学习器都进行欠采样, 但对于全局则不会丢失重要信息.

3、对小类数据过采样

过采样: 对小类的数据样本进行采样来增加小类的数据样本个数.

代表算法: SMOTE 和 ADASYN.

SMOTE: 通过对训练集中的小类数据进行插值来产生额外的小类样本数据.

新的少数类样本产生的策略: 对每个少数类样本 a , 在 a 的最近邻中随机选一个样本 b , 然后在 a 、 b 之间的连线上随机选一点作为新合成的少数类样本.

ADASYN: 根据学习难度的不同, 对不同的少数类别的样本使用加权分布, 对于难以学习的少数类的样本, 产生更多的综合数据. 通过减少类不平衡引入的偏差和将分类决策边界自适应地转移到困难的样本两种手段, 改善了数据分布.

4、使用新评价指标

如果当前评价指标不适用, 则应寻找其他具有说服力的评价指标. 比如准确度这个评价指标在类别不均衡的分类任务中并不适用, 甚至进行误导. 因此在类别不均衡分类任务中, 需要使用更有说服力的评价指标来对分类器进行评价.

5、选择新算法

不同的算法适用于不同的任务与数据, 应该使用不同的算法进行比较.

6、数据代价加权

例如当分类任务是识别小类, 那么可以对分类器的小类样本数据增加权值, 降低大类样本的权值, 从而使得分类器将重点集中在小类样本身上.

7、转化问题思考角度

例如在分类问题时，把小类的样本作为异常点，将问题转化为异常点检测或变化趋势检测问题。异常点检测即是对那些罕见事件进行识别。变化趋势检测区别于异常点检测在于其通过检测不寻常的变化趋势来识别。

8、将问题细化分析

对问题进行分析与挖掘，将问题划分成多个更小的问题，看这些小问题是否更容易解决。

11. 决策树

11.1. 决策树的基本原理. 决策树 (Decision Tree) 是一种分而治之的决策过程。一个困难的预测问题，通过树的分支节点，被划分成两个或多个较为简单的子集，从结构上划分为不同的子问题。将依规则分割数据集的过程不断递归下去 (Recursive Partitioning)。随着树的深度不断增加，分支节点的子集越来越小，所需要提的问题数也逐渐简化。当分支节点的深度或者问题的简单程度满足一定的停止规则 (Stopping Rule) 时，该分支节点会停止分裂，此为自上而下的停止阈值 (Cutoff Threshold) 法；有些决策树也使用自下而上的剪枝 (Pruning) 法。

11.2. 决策树的三要素？ 一棵决策树的生成过程主要分为下 3 个部分：

1、特征选择：从训练数据中众多的特征中选择一个特征作为当前节点的分裂标准，如何选择特征有着很多不同量化评估标准，从而衍生出不同的决策树算法。

2、决策树生成：根据选择的特征评估标准，从上至下递归地生成子节点，直到数据集不可分则决策树停止生长。树结构来说，递归结构是最容易理解的方式。

3、剪枝：决策树容易过拟合，一般来需要剪枝，缩小树结构规模、缓解过拟合。剪枝技术有预剪枝和后剪枝两种。

11.3. 决策树学习基本算法.

11.4. 决策树算法优缺点. 决策树算法的优点：

- 1、决策树算法易理解，机理解释起来简单。
- 2、决策树算法可以用于小数据集。
- 3、决策树算法的时间复杂度较小，为用于训练决策树的数据点的对数。
- 4、相比于其他算法智能分析一种类型变量，决策树算法可处理数字和数据的类别。
- 5、能够处理多输出的问题。
- 6、对缺失值不敏感。
- 7、可以处理不相关特征数据。
- 8、效率高，决策树只需要一次构建，反复使用，每一次预测的最大计算次数不超过决策树的深度。

决策树算法的缺点：

- 1、对连续性的字段比较难预测.
- 2、容易出现过拟合.
- 3、当类别太多时，错误可能就会增加的比较快.
- 4、在处理特征关联性比较强的数据时表现得不太好.
- 5、对于各类别样本数量不一致的数据，在决策树当中，信息增益的结果偏向于那些具有更多数值的特征.

11.5. 熵的概念以及理解.

熵：度量随机变量的不确定性.
定义：假设随机变量 X 的可能取值有 x_1, x_2, \dots, x_n ，对于每一个可能的取值 x_i ，其概率为 $P(X = x_i) = p_i, i = 1, 2, \dots, n$. 随机变量的熵为：

$$H(X) = - \sum_{i=1}^n p_i \log_2 p_i$$

对于样本集合，假设样本有 k 个类别，每个类别的概率为 $\frac{|C_k|}{|D|}$ ，其中 $|C_k||D|$ 为类别为 k 的样本个数， $|D|$ 为样本总数. 样本集合 D 的熵为：

$$H(D) = - \sum_{k=1}^k \frac{|C_k|}{|D|} \log_2 \frac{|C_k|}{|D|}$$

11.6. 信息增益的理解. 定义：以某特征划分数据集前后的熵的差值. 熵可以表示样本集合的不确定性，熵越大，样本的不确定性就越大. 因此可以使用划分前后集合熵的差值来衡量使用当前特征对于样本集合 D 划分效果的好坏. 假设划分前样本集合 D 的熵为 $H(D)$. 使用某个特征 A 划分数据集 D ，计算划分后的数据子集的熵为 $H(D|A)$.

则信息增益为：

$$g(D, A) = H(D) - H(D|A)$$

注：在决策树构建的过程中我们总是希望集合往最快到达纯度更高的子集合方向发展，因此我们总是选择使得信息增益最大的特征来划分当前数据集 D .

思想：计算所有特征划分数据集 D ，得到多个特征划分数据集 D 的信息增益，从这些信息增益中选择最大的，因而当前结点的划分特征便是使信息增益最大的划分所使用的特征.

另外这里提一下信息增益比相关知识： =

信息增益比本质：在信息增益的基础之上乘上一个惩罚参数. 特征个数较多时，惩罚参数较小；特征个数较少时，惩罚参数较大.

惩罚参数：数据集 D 以特征 A 作为随机变量的熵的倒数.

11.7. 剪枝处理的作用及策略. 剪枝处理是决策树学习算法用来解决过拟合问题的一种办法.

在决策树算法中，为了尽可能正确分类训练样本，节点划分过程不断重复，有时候会造成决策树分支过多，以至于将训练样本集自身特点当作泛化特点，而导致过拟合。因此可以采用剪枝处理来去掉一些分支来降低过拟合的风险。

剪枝的基本策略有预剪枝（pre-pruning）和后剪枝（post-pruning）。

预剪枝：在决策树生成过程中，在每个节点划分前先估计其划分后的泛化性能，如果不能提升，则停止划分，将当前节点标记为叶结点。

后剪枝：生成决策树以后，再自下而上对非叶结点进行考察，若将此节点标记为叶结点可以带来泛化性能提升，则修改之。

12. 支持向量机

12.1. 什么是支持向量机. 支持向量：在求解的过程中，会发现只根据部分数据就可以确定分类器，这些数据称为支持向量。

支持向量机（Support Vector Machine, SVM）：其含义是通过支持向量运算的分类器。

在一个二维环境中，其中点 R, S, G 点和其它靠近中间黑线的点可以看作为支持向量，它们可以决定分类器，即黑线的具体参数。

支持向量机是一种二分类模型，它的目的是寻找一个超平面来对样本进行分割，分割的原则是边界最大化，最终转化为一个凸二次规划问题来求解。由简至繁的模型包括：

当训练样本线性可分时，通过硬边界（hard margin）最大化，学习一个线性可分支持向量机；

当训练样本近似线性可分时，通过软边界（soft margin）最大化，学习一个线性支持向量机；

当训练样本线性不可分时，通过核技巧和软边界最大化，学习一个非线性支持向量机；

12.2. 支持向量机能解决哪些问题. 线性分类

在训练数据中，每个数据都有 n 个的属性和一个二分类类别标志，我们可以认为这些数据在一个 n 维空间里。我们的目标是找到一个 $n-1$ 维的超平面，这个超平面可以将数据分成两部分，每部分数据都属于同一个类别。

这样的超平面有很多，假如我们要找到一个最佳的超平面。此时，增加一个约束条件：要求这个超平面到每边最近数据点的距离是最大的，成为最大边距超平面。这个分类器即为最大边距分类器。

非线性分类

SVM 的一个优势是支持非线性分类。它结合使用拉格朗日乘子法（Lagrange Multiplier）和 KKT（Karush Kuhn Tucker）条件，以及核函数可以生成非线性分类器。

12.3. 核函数特点及其作用. 引入核函数目的：把原坐标系里线性不可分的数据用核函数 Kernel 投影到另一个空间，尽量使得数据在新的空间里线性可分。

核函数方法的广泛应用，与其特点是分不开的：

- 1) 核函数的引入避免了“维数灾难”，大大减小了计算量。而输入空间的维数 n 对核函数矩阵无影响。因此，核函数方法可以有效处理高维输入。
- 2) 无需知道非线性变换函数 Φ 的形式和参数。
- 3) 核函数的形式和参数的变化会隐式地改变从输入空间到特征空间的映射，进而对特征空间的性质产生影响，最终改变各种核函数方法的性能。
- 4) 核函数方法可以和不同的算法相结合，形成多种不同的基于核函数技术的方法，且这两部分的设计可以单独进行，并可以为不同的应用选择不同的核函数和算法。

12.4. SVM 为什么引入对偶问题. 1，对偶问题将原始问题中的约束转为了对偶问题中的等式约束，对偶问题往往更加容易求解。

2，可以很自然的引用核函数（拉格朗日表达式里面有内积，而核函数也是通过内积进行映射的）。

3，在优化理论中，目标函数 $f(x)$ 会有多种形式：如果目标函数和约束条件都为变量 x 的线性函数，称该问题为线性规划；如果目标函数为二次函数，约束条件为线性函数，称该最优化问题为二次规划；如果目标函数或者约束条件均为非线性函数，称该最优化问题为非线性规划。每个线性规划问题都有一个与之对应的对偶问题，对偶问题有非常良好的性质，以下列举几个：

- a, 对偶问题的对偶是原问题；
- b, 无论原始问题是凸的，对偶问题都是凸优化问题；
- c, 对偶问题可以给出原始问题一个下界；
- d, 当满足一定条件时，原始问题与对偶问题的解是完全等价的。

12.5. 如何理解 SVM 中的对偶问题. 在硬边界支持向量机中，问题的求解可以转化为凸二次规划问题。

假设优化目标为

$$(1) \quad \begin{aligned} & \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \\ & s.t. y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, i = 1, 2, \dots, m. \end{aligned}$$

step 1. 转化问题：

$$(2) \quad \min_{\mathbf{w}, b} \max_{\alpha_i \geq 0} \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^m \alpha_i (1 - y_i(\mathbf{w}^T \mathbf{x}_i + b))$$

上式等价于原问题，因为若满足 (1) 中不等式约束，则 (2) 式求 \max 时, $\alpha_i(1 - y_i(\mathbf{w}^T \mathbf{x}_i + b))$ 必须取 0，与 (1) 等价；若不满足 (1) 中不等式约束，(2) 中求 \max 会得到无穷大。交换 \min 和 \max 获得其对偶问题：

$$\max_{\alpha_i \geq 0} \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^m \alpha_i (1 - y_i (\mathbf{w}^T \mathbf{x}_i + b))$$

交换之后的对偶问题和原问题并不相等，上式的解小于等于原问题的解。

step 2. 现在的问题是如何找到问题 (1) 的最优值的一个最好的下界？

$$(3) \quad \frac{1}{2} \|\mathbf{w}\|^2 < v 1 - y_i (\mathbf{w}^T \mathbf{x}_i + b) \leq 0$$

若方程组 (3) 无解，则 v 是问题 (1) 的一个下界。若 (3) 有解，则

$$\forall \alpha > 0, \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^m \alpha_i (1 - y_i (\mathbf{w}^T \mathbf{x}_i + b)) < v$$

由逆否命题得：若

$$\exists \alpha > 0, \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^m \alpha_i (1 - y_i (\mathbf{w}^T \mathbf{x}_i + b)) \geq v$$

则 (3) 无解。

那么 v 是问题

(1) 的一个下界。要求得一个好的下界，取最大值即可

$$\max_{\alpha_i \geq 0} \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^m \alpha_i (1 - y_i (\mathbf{w}^T \mathbf{x}_i + b))$$

step 3. 令

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^m \alpha_i (1 - y_i (\mathbf{w}^T \mathbf{x}_i + b))$$

p^* 为原问题的最小值，对应的 w, b 分别为 w^*, b^* ，则对于任意的 $a > 0$ ：

$$p^* = \frac{1}{2} \|\mathbf{w}^*\|^2 \geq L(\mathbf{w}^*, b, \mathbf{a}) \geq \min_{\mathbf{w}, b} L(\mathbf{w}, b, \mathbf{a})$$

则 $\min_{\mathbf{w}, b} L(\mathbf{w}, b, \mathbf{a})$ 是问题 (1) 的一个下界。

此时，取最大值即可求得好的下界，即

$$\max_{\alpha_i \geq 0} \min_{\mathbf{w}, b} L(\mathbf{w}, b, \mathbf{a})$$

核函数	表达式	备注
Linear Kernel 线性核	$k(x, y) = x^t y + c$	
Polynomial Kernel 多项式核	$k(x, y) = (ax^t y + c)^d$	$d \geq 1$ 为多项式的次数
Exponential Kernel 指数核	$k(x, y) = \exp(-\frac{\ x-y\ }{2\sigma^2})$	$\sigma > 0$
Gaussian Kernel 高斯核	$k(x, y) = \exp(-\frac{\ x-y\ ^2}{2\sigma^2})$	σ 为高斯核的带宽, $\sigma > 0$,
Laplacian Kernel 拉普拉斯核	$k(x, y) = \exp(-\frac{\ x-y\ }{\sigma})$	$\sigma > 0$
ANOVA Kernel	$k(x, y) = \exp(-\sigma(x^k - y^k)^2)^d$	
Sigmoid Kernel	$k(x, y) = \tanh(ax^t y + c)$	\tanh 为双曲正切函数, $a > 0, c < 0$

12.6. 常见的核函数有哪些.

12.7. SVM 主要特点. 特点:

- (1) SVM 方法的理论基础是非线性映射, SVM 利用内积核函数代替向高维空间的非线性映射.
- (2) SVM 的目标是对特征空间划分得到最优超平面, SVM 方法核心是最大化分类边界.
- (3) 支持向量是 SVM 的训练结果, 在 SVM 分类决策中起决定作用的是支持向量.
- (4) SVM 是一种有坚实理论基础的新颖的适用小样本学习方法. 它基本上不涉及概率测度及大数定律等, 也简化了通常的分类和回归等问题.
- (5) SVM 的最终决策函数只由少数的支持向量所确定, 计算的复杂性取决于支持向量的数目, 而不是样本空间的维数, 这在某种意义上避免了“维数灾难”.
- (6) 少数支持向量决定了最终结果, 这不但可以帮助我们抓住关键样本、“剔除”大量冗余样本, 而且注定了该方法不但算法简单, 而且具有较好的“鲁棒性”. 这种鲁棒性主要体现在:
增、删非支持向量样本对模型没有影响;
支持向量样本集具有一定的鲁棒性;
有些成功的应用中, SVM 方法对核的选取不敏感

- (7) SVM 学习问题可以表示为凸优化问题，因此可以利用已知的有效算法发现目标函数的全局最小值。而其他分类方法（如基于规则的分类器和人工神经网络）都采用一种基于贪心学习的策略来搜索假设空间，这种方法一般只能获得局部最优解。
- (8) SVM 通过最大化决策边界的边缘来控制模型的能力。尽管如此，用户必须提供其他参数，如使用核函数类型和引入松弛变量等。
- (9) SVM 在小样本训练集上能够得到比其它算法好很多的结果。SVM 优化目标是结构化风险最小，而不是经验风险最小，避免了过拟合问题，通过 margin 的概念，得到对数据分布的结构化描述，减低了对数据规模和数据分布的要求，有优秀的泛化能力。
- (10) 它是一个凸优化问题，因此局部最优解一定是全局最优解的优点。

12.8. SVM 主要缺点。

- (1) SVM 算法对大规模训练样本难以实施

SVM 的空间消耗主要是存储训练样本和核矩阵，由于 SVM 是借助二次规划来求解支持向量，而求解二次规划将涉及 m 阶矩阵的计算 (m 为样本的个数)，当 m 数目很大时该矩阵的存储和计算将耗费大量的机器内存和运算时间。

如果数据量很大，SVM 的训练时间就会比较长，如垃圾邮件的分类检测，没有使用 SVM 分类器，而是使用简单的朴素贝叶斯分类器，或者是使用逻辑回归模型分类。

- (2) 用 SVM 解决多分类问题存在困难

经典的支持向量机算法只给出了二类分类的算法，而在实际应用中，一般要解决多类的分类问题。可以通过多个二类支持向量机的组合来解决。主要有一对多组合模式、一对一组合模式和 SVM 决策树；再就是通过构造多个分类器的组合来解决。主要原理是克服 SVM 固有的缺点，结合其他算法的优势，解决多类问题的分类精度。如：与粗糙集理论结合，形成一种优势互补的多类问题的组合分类器。

- (3) 对缺失数据敏感，对参数和核函数的选择敏感

支持向量机性能的优劣主要取决于核函数的选取，所以对于一个实际问题而言，如何根据实际的数据模型选择合适的核函数从而构造 SVM 算法。目前比较成熟的核函数及其参数的选择都是人为的，根据经验来选取的，带有一定的随意性。在不同的问题领域，核函数应当具有不同的形式和参数，所以在选取时候应该将领域知识引入进来，但是目前还没有好的方法来解决核函数的选取问题。

12.9. 逻辑回归与 SVM 的异同。相同点：

- LR 和 SVM 都是分类算法。
- LR 和 SVM 都是监督学习算法。

- LR 和 SVM 都是判别模型.
- 如果不考虑核函数，LR 和 SVM 都是线性分类算法，也就是说他们的分类决策面都是线性的.
说明：LR 也是可以用核函数的. 但 LR 通常不采用核函数的方法. (计算量太大)

不同点：

1、LR 采用 log 损失，SVM 采用合页 (hinge) 损失. 逻辑回归的损失函数：

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^i \log h_\theta(x^i) + (1 - y^i) \log(1 - h_\theta(x^i))]$$

支持向量机的目标函数：

$$L(w, n, a) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \alpha_i (y_i (w^T x_i + b) - 1)$$

逻辑回归方法基于概率理论，假设样本为 1 的概率可以用 sigmoid 函数来表示，然后通过极大似然估计的方法估计出参数的值.

支持向量机基于几何边界最大化原理，认为存在最大几何边界的分类面为最优分类面.

2、LR 对异常值敏感，SVM 对异常值不敏感.

支持向量机只考虑局部的边界线附近的点，而逻辑回归考虑全局.LR 模型找到的那个超平面，是尽量让所有点都远离他，而 SVM 寻找的那个超平面，是只让最靠近中间分割线的那些点尽量远离，即只用到那些支持向量的样本.

支持向量机改变非支持向量样本并不会引起决策面的变化.

逻辑回归中改变任何样本都会引起决策面的变化.

3、计算复杂度不同. 对于海量数据，SVM 的效率较低，LR 效率比较高

当样本较少，特征维数较低时，SVM 和 LR 的运行时间均比较短，SVM 较短一些. 准确率的话，LR 明显比 SVM 要高. 当样本稍微增加些时，SVM 运行时间开始增长，但是准确率赶超了 LR.SVM 时间虽长，但在可接受范围内. 当数据量增长到 20000 时，特征维数增长到 200 时，SVM 的运行时间剧烈增加，远远超过了 LR 的运行时间. 但是准确率却和 LR 相差无几.(这其中主要原因是大量非支持向量参与计算，造成 SVM 的二次规划问题)

4、对非线性问题的处理方式不同

LR 主要靠特征构造，必须组合交叉特征，特征离散化.SVM 也可以这样，还可以通过核函数 kernel (因为只有支持向量参与核计算，计算复杂度不高). 由于可以利用核函数，SVM 则可以通过对偶求解高效处理.LR 则在特征空间维度很高时，表现较差.

5、SVM 的损失函数就自带正则.

损失函数中的 $1/2\|w\|^2$ 项，这就是为什么 SVM 是结构风险最小化算法的原因!!! 而 LR 必须另外在损失函数上添加正则项!!! **

6、SVM 自带结构风险最小化，LR 则是经验风险最小化.

7、SVM 会用核函数而 LR 一般不用核函数.

13. 贝叶斯分类器

13.1. 图解极大似然估计.

极大似然估计的原理，用一张图片来说明，如下图所示：
例：有两个外形完全相同的箱子，1号箱有99只白球，1只黑球；2号箱有1只白球，99只黑球。在一次实验中，取出的是黑球，请问是从哪个箱子中取出的？

一般的根据经验想法，会猜测这只黑球最像是从2号箱取出，此时描述的“最像”就有“最大似然”的意思，这种想法常称为“最大似然原理”。

13.2. 极大似然估计原理. 总结起来，最大似然估计的目的就是：利用已知的样本结果，反推最有可能（最大概率）导致这样结果的参数值。

极大似然估计是建立在极大似然原理的基础上的一个统计方法。极大似然估计提供了一种给定观察数据来评估模型参数的方法，即：“模型已定，参数未知”。通过若干次试验，观察其结果，利用试验结果得到某个参数值能够使样本出现的概率为最大，则称为极大似然估计。

由于样本集中的样本都是独立同分布，可以只考虑一类样本集 D ，来估计参数向量 $\vec{\theta}$ 。记已知的样本集为：

$$D = \vec{x}_1, \vec{x}_2, \dots, \vec{x}_n$$

似然函数 (likelihood function)：联合概率密度函数 $p(D|\vec{\theta})$ 称为相对于 $\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n$ 的 $\vec{\theta}$ 的似然函数。

$$l(\vec{\theta}) = p(D|\vec{\theta}) = p(\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n | \vec{\theta}) = \prod_{i=1}^n p(\vec{x}_i | \vec{\theta})$$

如果 $\hat{\vec{\theta}}$ 是参数空间中能使似然函数 $l(\vec{\theta})$ 最大的 $\vec{\theta}$ 值，则 $\hat{\vec{\theta}}$ 应该是“最可能”的参数值，那么 $\hat{\vec{\theta}}$ 就是 $\vec{\theta}$ 的极大似然估计量。它是样本集的函数，记作：

$$\hat{\vec{\theta}} = d(D) = \arg \max_{\vec{\theta}} l(\vec{\theta})$$

$\hat{\vec{\theta}}(\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n)$ 称为极大似然函数估计值。

13.3. 贝叶斯分类器基本原理. 贝叶斯决策论通过相关概率已知的情况下利用误判损失来选择最优的类别分类。

假设有 N 种可能的分类标记，记为 $Y = c_1, c_2, \dots, c_N$ ，那对于样本 x ，它属于哪一类呢？

计算步骤如下：

step 1. 算出样本 x 属于第 i 个类的概率，即 $P(c_i|x)$ ；

step 2. 通过比较所有的 $P(c_i|x)$ ，得到样本 x 所属的最佳类别。

step 3. 将类别 c_i 和样本 \mathbf{x} 代入到贝叶斯公式中，得到：

$$P(c_i|\mathbf{x}) = \frac{P(\mathbf{x}|c_i)P(c_i)}{P(\mathbf{x})}.$$

一般来说， $P(c_i)$ 为先验概率， $P(\mathbf{x}|c_i)$ 为条件概率， $P(\mathbf{x})$ 是用于归一化的证据因子。对于 $P(c_i)$ 可以通过训练样本中类别为 c_i 的样本所占的比例进行估计；此外，由于只需要找出最大的 $P(\mathbf{x}|c_i)$ ，因此我们并不需要计算 $P(\mathbf{x})$ 。

为了求解条件概率，基于不同假设提出了不同的方法，以下将介绍朴素贝叶斯分类器和半朴素贝叶斯分类器。

13.4. 朴素贝叶斯分类器。 假设样本 \mathbf{x} 包含 d 个属性，即 $\mathbf{x} = x_1, x_2, \dots, x_d$ 。于是有：

$$P(\mathbf{x}|c_i) = P(x_1, x_2, \dots, x_d|c_i)$$

这个联合概率难以从有限的训练样本中直接估计得到。于是，朴素贝叶斯（Naive Bayesian，简称 NB）采用了“属性条件独立性假设”：对已知类别，假设所有属性相互独立。于是有：

$$P(x_1, x_2, \dots, x_d|c_i) = \prod_{j=1}^d P(x_j|c_i)$$

这样的话，我们就可以很容易地推出相应的判定准则了：

$$h_{nb}(\mathbf{x}) = \arg \max_{c_i \in Y} P(c_i) \prod_{j=1}^d P(x_j|c_i)$$

条件概率 $P(x_j|c_i)$ 的求解

如果 x_j 是标签属性，那么我们可以通过计数的方法估计 $P(x_j|c_i)$

$$P(x_j|c_i) = \frac{P(x_j, c_i)}{P(c_i)} \approx \frac{\#(x_j, c_i)}{\#(c_i)}$$

其中， $\#(x_j, c_i)$ 表示在训练样本中 x_j 与 c_i 共同出现的次数。

如果 x_j 是数值属性，通常我们假设类别中 c_i 的所有样本第 j 个属性的值服从正态分布。我们首先估计这个分布的均值 和方差，然后计算 x_j 在这个分布中的概率密度 $P(x_j|c_i)$ 。

13.5. 举例理解朴素贝叶斯分类器。 使用经典的西瓜训练集如下：

编号	色泽	根蒂	敲声	纹理	脐部	触感	密度	含糖率	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	0.697	0.460	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	0.774	0.376	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	0.634	0.264	是

4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	0.608	0.318	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	0.556	0.215	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	0.403	0.237	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	0.481	0.149	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	0.437	0.211	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	0.666	0.091	否
10	青绿	硬挺	清脆	清晰	平坦	软粘	0.243	0.267	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	0.245	0.057	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	0.343	0.099	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	0.639	0.161	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	0.657	0.198	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	0.360	0.370	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	0.593	0.042	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	0.719	0.103	否

对下面的测试例“测 1”进行分类：

编号	色泽	根蒂	敲声	纹理	脐部	触感	密度	含糖率	好瓜
测 1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	0.697	0.460	?

首先，估计类先验概率 $P(c_j)$ ，有

$$P(=) = \frac{8}{17} = 0.471 \quad P(=) = \frac{9}{17} = 0.529$$

然后，为每个属性估计条件概率（这里，对于连续属性，假定它们服从正态分布）

$$P| = P = | = = \frac{3}{8} = 0.375$$

$$P| = P = | = = \frac{3}{9} \approx 0.333$$

$$P| = P = | = = \frac{5}{8} = 0.625$$

$$P| = P = | = = \frac{3}{9} = 0.333$$

$$P| = P = | = = \frac{6}{8} = 0.750$$

$$P_{|} = P = | = = \frac{4}{9} \approx 0.444$$

$$P_{|} = P = | = = \frac{7}{8} = 0.875$$

$$P_{|} = P = | = = \frac{2}{9} \approx 0.222$$

$$P_{|} = P = | = = \frac{6}{8} = 0.750$$

$$P_{|} = P = | = = \frac{2}{9} \approx 0.222$$

$$P_{|} = P = | = = \frac{6}{8} = 0.750$$

$$P_{|} = P = | = = \frac{6}{9} \approx 0.667$$

$$\begin{aligned} \rho_{0.697|} &= \rho = 0.697| = \\ &= \frac{1}{\sqrt{2\pi} \times 0.129} \exp\left(-\frac{(0.697 - 0.574)^2}{2 \times 0.129^2}\right) \approx 1.959 \end{aligned}$$

$$\begin{aligned} \rho_{0.697|} &= \rho = 0.697| = \\ &= \frac{1}{\sqrt{2\pi} \times 0.195} \exp\left(-\frac{(0.697 - 0.496)^2}{2 \times 0.195^2}\right) \approx 1.203 \end{aligned}$$

$$\begin{aligned} \rho_{0.460|} &= \rho = 0.460| = \\ &= \frac{1}{\sqrt{2\pi} \times 0.101} \exp\left(-\frac{(0.460 - 0.279)^2}{2 \times 0.101^2}\right) \approx 0.788 \end{aligned}$$

$$\begin{aligned} \rho_{0.460|} &= \rho = 0.460| = \\ &= \frac{1}{\sqrt{2\pi} \times 0.108} \exp\left(-\frac{(0.460 - 0.154)^2}{2 \times 0.108^2}\right) \approx 0.066 \end{aligned}$$

于是有

$$P(=) \times P_{|} \times P_{|} \times P_{|} \times P_{|} \times P_{|} \times P_{|} \times p_{0.697|} \times p_{0.460|} \approx 0.063 P(=) \times P_{|} \times P_{|} \times P_{|} \times P_{|} \times P_{|} \times P_{|} \times p_{0.697|} \times p_{0.460|}$$

由于 $0.063 > 6.80 \times 10^{-5}$, 因此, 朴素贝叶斯分类器将测试样本“测 1”判别为“好瓜”.

13.6. 半朴素贝叶斯分类器. 朴素贝叶斯采用了“属性条件独立性假设”，半朴素贝叶斯分类器的基本想法是适当考虑一部分属性间的相互依赖信息. **独依赖估计** (One-Dependence Estimator, 简称 ODE) 是半朴素贝叶斯分类器最常用的一种策略. 顾名思义，独依赖是假设每个属性在类别之外最多依赖一个其他属性，即：

$$P(\mathbf{x}|c_i) = \prod_{j=1}^d P(x_j|c_i, \text{pa}_j)$$

其中 pa_j 为属性 x_i 所依赖的属性，成为 x_i 的父属性. 假设父属性 pa_j 已知，那么可以使用下面的公式估计 $P(x_j|c_i, \text{pa}_j)$

$$P(x_j|c_i, \text{pa}_j) = \frac{P(x_j, c_i, \text{pa}_j)}{P(c_i, \text{pa}_j)}$$

14. EM 算法

14.1. EM 算法基本思想. 最大期望算法 (Expectation-Maximization algorithm, EM)，是一类通过迭代进行极大似然估计的优化算法，通常作为牛顿迭代法的替代，用于对包含隐变量或缺失数据的概率模型进行参数估计.

最大期望算法基本思想是经过两个步骤交替进行计算：

第一步是计算期望 (E)，利用对隐藏变量的现有估计值，计算其最大似然估计值；

第二步是最大化 (M)，最大化在 E 步上求得的最大似然值来计算参数的值.

M 步上找到的参数估计值被用于下一个 E 步计算中，这个过程不断交替进行.

14.2. EM 算法推导. 对于 m 个样本观察数据 $\mathbf{x} = (x^1, x^2, \dots, x^m)$ ，现在想找出样本的模型参数 θ ，其极大化模型分布的对数似然函数为：

$$\theta = \arg \max_{\theta} \sum_{i=1}^m \log P(x^{(i)}; \theta)$$

如果得到的观察数据有未观察到的隐含数据 $\mathbf{z} = (z^{(1)}, z^{(2)}, \dots, z^{(m)})$ ，极大化模型分布的对数似然函数则为：

$$(a) \quad \theta = \arg \max_{\theta} \sum_{i=1}^m \log P(x^{(i)}; \theta) = \arg \max_{\theta} \sum_{i=1}^m \log \sum_{z^{(i)}} P(x^{(i)}, z^{(i)}; \theta)$$

由于上式不能直接求出 θ ，采用缩放技巧：

$$(1) \quad \begin{aligned} \sum_{i=1}^m \log \sum_{z^{(i)}} P(x^{(i)}, z^{(i)}; \theta) &= \sum_{i=1}^m \log \sum_{z^{(i)}} Q_i(z^{(i)}) \frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})} \\ &\geq \sum_{i=1}^m \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})} \end{aligned}$$

上式用到了 Jensen 不等式：

$$\log \sum_j \lambda_j y_j \geq \sum_j \lambda_j \log y_j, \quad \lambda_j \geq 0, \quad \sum_j \lambda_j = 1$$

并且引入了一个未知的新分布 $Q_i(z^{(i)})$.

此时，如果需要满足 Jensen 不等式中的等号，所以有：

$$\frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})} = c, c$$

由于 $Q_i(z^{(i)})$ 是一个分布，所以满足

$$\sum_z Q_i(z^{(i)}) = 1$$

综上，可得：

$$Q_i(z^{(i)}) = \frac{P(x^{(i)}, z^{(i)}; \theta)}{\sum_z P(x^{(i)}, z^{(i)}; \theta)} = \frac{P(x^{(i)}, z^{(i)}; \theta)}{P(x^{(i)}; \theta)} = P(z^{(i)} | x^{(i)}; \theta)$$

如果 $Q_i(z^{(i)}) = P(z^{(i)} | x^{(i)}; \theta)$ ，则第 (1) 式是我们的包含隐藏数据的对数似然的一个下界。如果我们能极大化这个下界，则也在尝试极大化我们的对数似然。即我们需要最大化下式：

$$\arg \max_{\theta} \sum_{i=1}^m \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})}$$

简化得：

$$\arg \max_{\theta} \sum_{i=1}^m \sum_{z^{(i)}} Q_i(z^{(i)}) \log P(x^{(i)}, z^{(i)}; \theta)$$

以上即为 EM 算法的 M 步， $\sum_{z^{(i)}} Q_i(z^{(i)}) \log P(x^{(i)}, z^{(i)}; \theta)$ 可理解为 $\log P(x^{(i)}, z^{(i)}; \theta)$ 基于条件概率分布 $Q_i(z^{(i)})$ 的期望。以上即为 EM 算法中 E 步和 M 步的具体数学含义。

14.3. 图解 EM 算法. 考虑上一节中的 (a) 式，表达式中存在隐变量，直接找到参数估计比较困难，通过 EM 算法迭代求解下界的最大值到收敛为止。

图片中的紫色部分是我们的目标模型 $p(x|\theta)$ ，该模型复杂，难以求解析解，为了消除隐变量 $z^{(i)}$ 的影响，我们可以选择一个不包含 $z^{(i)}$ 的模型 $r(x|\theta)$ ，使其满足条件 $r(x|\theta) \leq p(x|\theta)$ 。

求解步骤如下：

- (1) 选取 θ_1 ，使得 $r(x|\theta_1) = p(x|\theta_1)$ ，然后对此时的 r 求取最大值，得到极值点 θ_2 ，实现参数的更新。
- (2) 重复以上过程到收敛为止，在更新过程中始终满足 $r \leq p$ 。

14.4. EM 算法流程. 输入: 观察数据 $x = (x^{(1)}, x^{(2)}, \dots, x^{(m)})$, 联合分布 $p(x, z; \theta)$, 条件分布 $p(z|x; \theta)$, 最大迭代次数 J

- 1) 随机初始化模型参数 θ 的初值 θ^0 .
- 2) *for* j *from* 1 *to* J :
- a) E 步. 计算联合分布的条件概率期望:

$$Q_i(z^{(i)}) = P(z^{(i)}|x^{(i)}, \theta^j)$$

$$L(\theta, \theta^j) = \sum_{i=1}^m \sum_{z^{(i)}} P(z^{(i)}|x^{(i)}, \theta^j) \log P(x^{(i)}, z^{(i)}; \theta)$$

- b) M 步. 极大化 $L(\theta, \theta^j)$, 得到 θ^{j+1} :

$$\theta^{j+1} = \arg \max_{\theta} L(\theta, \theta^j)$$

- c) 如果 θ^{j+1} 收敛, 则算法结束. 否则继续回到步骤 a) 进行 E 步迭代.
- 输出: 模型参数 θ .

15. 降维和聚类

15.1. 图解为什么会产生维数灾难. 假如数据集包含 10 张照片, 照片中包含三角形和圆两种形状. 现在来设计一个分类器进行训练, 让这个分类器对其他的照片进行正确分类 (假设三角形和圆的总数是无限大), 简单的, 我们用一个特征进行分类:

图 2.21.1.a

从上图可看到, 如果仅仅只有一个特征进行分类, 三角形和圆几乎是均匀分布在这条线段上, 很难将 10 张照片线性分类. 那么, 增加一个特征后的情况会怎么样:

图 2.21.1.b

增加一个特征后, 我们发现仍然无法找到一条直线将猫和狗分开. 所以, 考虑需要再增加一个特征:

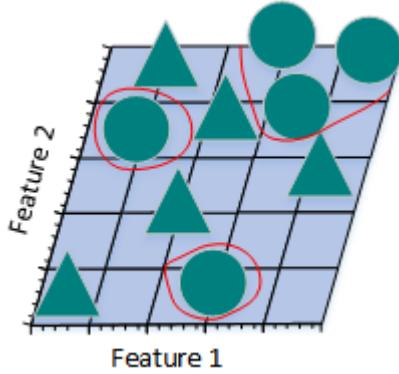
图 2.21.1.c

图 2.21.1.d

此时, 可以找到一个平面将三角形和圆分开.

现在计算一下不同特征数是样本的密度:

- (1) 一个特征时, 假设特征空间时长度为 5 的线段, 则样本密度为 $10 \div 5 = 2$.
- (2) 两个特征时, 特征空间大小为 $5 \times 5 = 25$, 样本密度为 $10 \div 25 = 0.4$.
- (3) 三个特征时, 特征空间大小是 $5 \times 5 \times 5 = 125$, 样本密度为 $10 \div 125 = 0.08$.



以此类推，如果继续增加特征数量，样本密度会越来越稀疏，此时，更容易找到一个超平面将训练样本分开。当特征数量增长至无限大时，样本密度就变得非常稀疏。

下面看一下将高维空间的分类结果映射到低维空间时，会出现什么情况？

图 2.21.1.e

上图是将三维特征空间映射到二维特征空间后的结果。尽管在高维特征空间时训练样本线性可分，但是映射到低维空间后，结果正好相反。事实上，增加特征数量使得高维空间线性可分，相当于在低维空间内训练一个复杂的非线性分类器。不过，这个非线性分类器太过“聪明”，仅仅学到了一些特例。如果将其用来辨别那些未曾出现在训练样本中的测试样本时，通常结果不太理想，会造成过拟合问题。

图 2.21.1.f

上图所示的只采用 2 个特征的线性分类器分错了一些训练样本，准确率似乎没有图 2.21.1.e 的高，但是，采用 2 个特征的线性分类器的泛化能力比采用 3 个特征的线性分类器要强。因为，采用 2 个特征的线性分类器学到的不只是特例，而是一个整体趋势，对于那些未曾出现过的样本也可以比较好地辨别开来。换句话说，通过减少特征数量，可以避免出现过拟合问题，从而避免“维数灾难”。

上图从另一个角度诠释了“维数灾难”。假设只有一个特征时，特征的值域是 0 到 1，每一个三角形和圆的特征值都是唯一的。如果我们希望训练样本覆盖特征值值域的 20%，那么就需要三角形和圆总数的 20%。我们增加一个特征后，为了继续覆盖特征值值域的 20% 就需要三角形和圆总数的 $45\%(0.452^2 \approx 0.2)$ 。继续增加一个特征后，需要三角形和圆总数的 $58\%(0.583^3 \approx 0.2)$ 。随着特征数量的增加，为了覆盖特征值值域的 20%，就需要更多的训练样本。如果没有足够的训练样本，就可能会出现过拟合问题。

通过上述例子，我们可以看到特征数量越多，训练样本就会越稀疏，分类器的参数估计就会越不准确，更加容易出现过拟合问题。“维数灾难”的另一个影响是训练样本的稀疏性并不是均匀分布的。处于中心位置的训练样本比四周的训练样本更加稀疏。

假设有一个二维特征空间，如上图所示的矩形，在矩形内部有一个内切的圆形。由于越接近圆心的样本越稀疏，因此，相比于圆形内的样本，那些位于矩形四角的样本更加难以分类。当维数变大时，特征超空间的容量不变，但单位圆的容量会趋于 0，在高维空间中，大多数训练数据驻留在特征超空间的角落。散落在角落的数据要比处于中心的数据难于分类。

15.2. 怎样避免维数灾难. 有待完善!!!

解决维度灾难问题：

主成分分析法 PCA，线性判别法 LDA
奇异值分解简化数据、拉普拉斯特征映射
Lasso 缩减系数法、小波分析法、

15.3. 聚类和降维有什么区别与联系. 聚类用于找寻数据内在的分布结构，既可以作为一个单独的过程，比如异常检测等等。也可作为分类等其他学习任务的前驱过程。聚类是标准的无监督学习。

1) 在一些推荐系统中需确定新用户的类型，但定义“用户类型”却可能不太容易，此时往往可先对原有的用户数据进行聚类，根据聚类结果将每个簇定义为一个类，然后再基于这些类训练分类模型，用于判别新用户的类型。

2) 而降维则是为了缓解维数灾难的一个重要方法，就是通过某种数学变换将原始高维属性空间转变为一个低维“子空间”。其基于的假设就是，虽然人们平时观测到的数据样本虽然是高维的，但是实际上真正与学习任务相关的是个低维度的分布。从而通过最主要的几个特征维度就可以实现对数据的描述，对于后续的分类很有帮助。比如对于 Kaggle（数据分析竞赛平台之一）上的泰坦尼克号生还问题。通过给定一个乘客的许多特征如年龄、姓名、性别、票价等，来判断其是否能在海难中生还。这就需要首先进行特征筛选，从而能够找出主要的特征，让学到的模型有更好的泛化性。

聚类和降维都可以作为分类等问题的预处理步骤。

但是他们虽然都能实现对数据的约减。但是二者适用的对象不同，聚类针对的是数据点，而降维则是对于数据的特征。另外它们有着很多种实现方法。聚类中常用的有 K-means、层次聚类、基于密度的聚类等；降维中常用的则 PCA、Isomap、LLE 等。

15.4. 有哪些聚类算法优劣衡量标准. 不同聚类算法有不同的优劣和不同的适用条件。可以从以下方面进行衡量判断：1、算法的处理能力：处理大的数据集的能力，即算法复杂度；处理数据噪声的能力；处理任意形状，包括有间隙的嵌套的数据的能力；2、算法是否需要预设条件：是否需要预先知道聚类个数，是否需要用户给出领域知识；

3、算法的数据输入属性：算法处理的结果与数据输入的顺序是否相关，也就是说算法是否独立于数据输入顺序；算法处理有很多属性数据的能力，也就是对数据维数是否敏感，对数据的类型有无要求。

15.5. 聚类和分类有什么区别. **聚类 (Clustering)** 聚类, 简单地说就是把相似的东西分到一组, 聚类的时候, 我们并不关心某一类是什么, 我们需要实现的目标只是把相似的东西聚到一起. 一个聚类算法通常只需要知道如何计算相似度就可以开始工作了, 因此聚类通常并不需要使用训练数据进行学习, 在机器学习中属于无监督学习.

分类 (Classification)

分类, 对于一个分类器, 通常需要你告诉它“这个东西被分为某某类”. 一般情况下, 一个分类器会从它得到的训练集中进行学习, 从而具备对未知数据进行分类的能力, 在机器学习中属于监督学习.

算法名称	可伸缩性	适合的数据类型	高维性	异常数据抗干扰性	聚类形状	算法效率
WAVECLUSTER	很高	数值型	很高	较高	任意形状	很高
ROCK	很高	混合型	很高	很高	任意形状	一般
BIRCH	较高	数值型	较低	较低	球形	很高
CURE	较高	数值型	一般	很高	任意形状	较高
K-PROTOTYPES	一般	混合型	较低	较低	任意形状	一般
DENCLUE	较低	数值型	较高	一般	任意形状	较高
OPTIGRID	一般	数值型	较高	一般	任意形状	一般
CLIQUE	较高	数值型	较高	较高	任意形状	较低
DBSCAN	一般	数值型	较低	较高	任意形状	一般
CLARANS	较低	数值型	较低	较高	球形	较低

15.6. 不同聚类算法特点性能比较.

15.7. 四种常用聚类方法之比较. 聚类就是按照某个特定标准把一个数据集分割成不同的类或簇, 使得同一个簇内的数据对象的相似性尽可能大, 同时不在同一个簇中的数据对象的差异性也尽可能地大. 即聚类后同一类的数据尽可能聚集到一起, 不同类数据尽量分离. 主要的聚类算法可以划分为如下几类: 划分方法、层次方法、基于密度的方法、基于网格的方法以及基于模型的方法. 下面主要对 k-means 聚类算法、凝聚型层次聚类算法、神经网络聚类算法之 SOM, 以及模糊聚类的 FCM 算法通过通用测试数据集进行聚类效果的比较和分析.

15.8. k-means 聚类算法. k-means 是划分方法中较经典的聚类算法之一. 由于该算法的效率高, 所以在对大规模数据进行聚类时被广泛应用. 目前, 许多算法均围绕着该算法进行扩展和改进. k-means 算法以 k 为参数, 把 n 个对象分成 k 个簇, 使簇内具有较高的相似度, 而簇间的相似度较低.k-means 算法的处理过程如下: 首先, 随机地选择 k 个对象, 每个对象初始地代表了一个簇的平均值或中心; 对剩余的每个对象, 根据其与各簇中心的距离, 将它赋给

最近的簇; 然后重新计算每个簇的平均值. 这个过程不断重复, 直到准则函数收敛. 通常, 采用平方误差准则, 其定义如下:

$$E = \sum_{i=1}^k \sum_{p \in C_i} \|p - m_i\|^2$$

这里 E 是数据中所有对象的平方误差的总和, p 是空间中的点, m_i 是簇 C_i 的平均值 [9]. 该目标函数使生成的簇尽可能紧凑独立, 使用的距离度量是欧几里得距离, 当然也可以用其他距离度量.

算法流程: 输入: 包含 n 个对象的数据和簇的数目 k ; 输出: n 个对象到 k 个簇, 使平方误差准则最小. 步骤: (1) 任意选择 k 个对象作为初始的簇中心; (2) 根据簇中对象的平均值, 将每个对象(重新)赋予最类似的簇; (3) 更新簇的平均值, 即计算每个簇中对象的平均值; (4) 重复步骤 (2)、(3) 直到簇中心不再变化;

15.9. 层次聚类算法. 根据层次分解的顺序是自底向上的还是自上向下的, 层次聚类算法分为凝聚的层次聚类算法和分裂的层次聚类算法. 凝聚型层次聚类的策略是先将每个对象作为一个簇, 然后合并这些原子簇为越来越大的簇, 直到所有对象都在一个簇中, 或者某个终结条件被满足. 绝大多数层次聚类属于凝聚型层次聚类, 它们只是在簇间相似度的定义上有所不同.

算法流程:

注: 以采用最小距离的凝聚层次聚类算法为例:

(1) 将每个对象看作一类, 计算两两之间的最小距离; (2) 将距离最小的两个类合并成一个新类; (3) 重新计算新类与所有类之间的距离; (4) 重复 (2)、(3), 直到所有类最后合并成一类.

15.10. SOM 聚类算法. SOM 神经网络 [11] 是由芬兰神经网络专家 Kohonen 教授提出的, 该算法假设在输入对象中存在一些拓扑结构或顺序, 可以实现从输入空间(n 维) 到输出平面(2 维) 的降维映射, 其映射具有拓扑特征保持性质, 与实际的大脑处理有很强的理论联系.

SOM 网络包含输入层和输出层. 输入层对应一个高维的输入向量, 输出层由一系列组织在 2 维网格上的有序节点构成, 输入节点与输出节点通过权重向量连接. 学习过程中, 找到与之距离最短的输出层单元, 即获胜单元, 对其更新. 同时, 将邻近区域的权值更新, 使输出节点保持输入向量的拓扑特征.

算法流程:

(1) 网络初始化, 对输出层每个节点权重赋初值; (2) 从输入样本中随机选取输入向量并且归一化, 找到与输入向量距离最小的权重向量; (3) 定义获胜单元, 在获胜单元的邻近区域调整权重使其向输入向量靠拢; (4) 提供新样本、进行训练; (5) 收缩邻域半径、减小学习率, 重复, 直到小于允许值, 输出聚类结果.

15.11. FCM 聚类算法. 1965 年美国加州大学柏克莱分校的扎德教授第一次提出了‘集合’的概念. 经过十多年的发展, 模糊集合理论渐渐被应用到各个实际应用方面. 为克服非此即彼的分类缺点, 出现了以模糊集合论为数学基础的聚类分析. 用模糊数学的方法进行聚类分析, 就是模糊聚类分析 [12].

FCM 算法是一种以隶属度来确定每个数据点属于某个聚类程度的算法. 该聚类算法是传统硬聚类算法的一种改进.

设数据集 $X = x_1, x_2, \dots, x_n$, 它的模糊 c 划分可用模糊矩阵 $U = [u_{ij}]$ 表示, 矩阵 U 的元素 u_{ij} 表示第 $j(j = 1, 2, \dots, n)$ 个数据点属于第 $i(i = 1, 2, \dots, c)$ 类的隶属度, u_{ij} 满足如下条件:

$$(16) \quad \begin{aligned} \sum_{i=1}^c u_{ij} &= 1 \quad \forall j \\ u_{ij} &\in [0, 1] \quad \forall i, j \\ \sum_{j=1}^c u_{ij} &> 0 \quad \forall i \end{aligned}$$

目前被广泛使用的聚类准则是取类内加权误差平方和的极小值. 即:

$$(min) J_m(U, V) = \sum_{j=1}^n \sum_{i=1}^c u_{ij}^m d_{ij}^2(x_j, v_i)$$

其中 V 为聚类中心, m 为加权指数, $d_{ij}(x_j, v_i) = \|v_i - x_j\|$.

算法流程:

(1) 标准化数据矩阵; (2) 建立模糊相似矩阵, 初始化隶属矩阵; (3) 算法开始迭代, 直到目标函数收敛到极小值; (4) 根据迭代结果, 由最后的隶属矩阵确定数据所属的类, 显示最后的聚类结果.

15.12. 四种聚类算法试验. 选取专门用于测试分类、聚类算法的国际通用的 UCI 数据库中的 IRIS 数据集, IRIS 数据集包含 150 个样本数据, 分别取自三种不同的莺尾属植物 setosa、versicolor 和 virginica 的花朵样本, 每个数据含有 4 个属性, 即萼片长度、萼片宽度、花瓣长度、花瓣宽度, 单位为 cm. 在数据集上执行不同的聚类算法, 可以得到不同精度的聚类结果. 基于前面描述的各算法原理及流程, 可初步得如下聚类结果.

聚类方法	聚错样本数	运行时间/s	平均准确率/ (%)
K-means	17	0.146001	89
层次聚类	51	0.128744	66
SOM	22	5.267283	86
FCM	12	0.470417	92

注：

- (1) 聚错样本数：总的聚错的样本数，即各类中聚错的样本数的和；
- (2) 运行时间：即聚类整个过程所耗费的时间，单位为 s；
- (3) 平均准确度：设原数据集有 k 个类，用 c_i 表示第 i 类， n_i 为 c_i 中样本的个数， m_i 为聚类正确的个数，则 m_i/n_i 为第 i 类中的精度，则平均精度为： $avg = \frac{1}{k} \sum_{i=1}^k \frac{m_i}{n_i}$.