

朴素贝叶斯法 (naive Bayes)

朴素贝叶斯(naive-Bayes) 朴素贝叶斯(naive Bayes) 法是基于贝叶斯定理与特征条件独立假设的分类方法。对于给定的训练数据集，首先基于特征条件独立假设学习输入/输出的联合概率分布；然后基于此模型，对给定的输入 x ，利用贝叶斯定理求出后验概率最大的输出 y 。朴素贝叶斯法实现简单，学习与预测的效率都很高，是一种常用的方法。

学习与分类

基本原理

设输入空间 $\mathcal{X} \subseteq \mathbf{R}^n$ 为 n 维向量的集合，输出空间为类标记集合 $\mathcal{Y} = \{c_1, c_2, \dots, c_K\}$ 。输入为特征向量 $x \in \mathcal{X}$ ，输出为类标记 (class label) $y \in \mathcal{Y}$ 。 X 是定义在输入空间 \mathcal{X} 上的随机向量， Y 是定义在输出空间 \mathcal{Y} 上的随机变量。 $P(X, Y)$ 是 X 和 Y 的联合概率分布。训练数据集

$$T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\} \quad (1)$$

由 $P(X, Y)$ 独立同分布产生。

朴素贝叶斯法通过训练数据集学习联合概率分布 $P(X, Y)$ 。具体地，学习以下先验概率分布及条件概率分布。先验概率分布

$$P(Y = c_k), \quad k = 1, 2, \dots, K \quad (2)$$

条件概率分布

$$P(X = x | Y = c_k) = P\left(X^{(1)} = x^{(1)}, \dots, X^{(n)} = x^{(n)} | Y = c_k\right), \quad k = 1, 2, \dots, K \quad (3)$$

于是学习到联合概率分布 $P(X, Y)$ 。

条件概率分布 $P(X = x | Y = c_k)$ 有指数级数量的参数，其估计实际是不可行的。

朴素贝叶斯法对条件概率分布作了条件独立性的假设。由于这是一个较强的假设，朴素贝叶斯法也由此得名。具体地，条件独立性假设是

$$\begin{aligned} P(X = x | Y = c_k) &= P\left(X^{(1)} = x^{(1)}, \dots, X^{(n)} = x^{(n)} | Y = c_k\right) \\ &= \prod_{j=1}^n P\left(X^{(j)} = x^{(j)} | Y = c_k\right) \end{aligned} \quad (4)$$

朴素贝叶斯法实际上学习到生成数据的机制，所以属于生成模型。条件独立假设等于是说用于分类的特征在类确定的条件下都是条件独立的。这一假设使朴素贝叶斯法变得简单，但有时会牺牲一定的分类准确率。

朴素贝叶斯法分类时，对给定的输入 x ，通过学习到的模型计算后验概率分布 $P(Y = c_k | X = x)$ ，将后验概率最大的类作为 x 的类输出。后验概率计算根据贝叶斯定理进行：

$$P(Y = c_k \mid X = x) = \frac{P(X = x \mid Y = c_k)P(Y = c_k)}{\sum_k P(X = x \mid Y = c_k)P(Y = c_k)} \quad (5)$$

再由独立性假设：

$$P(Y = c_k \mid X = x) = \frac{P(Y = c_k) \prod_j P(X^{(j)} = x^{(j)} \mid Y = c_k)}{\sum_k P(Y = c_k) \prod_j P(X^{(j)} = x^{(j)} \mid Y = c_k)}, \quad k = 1, 2, \dots, K \quad (6)$$

这是朴素贝叶斯法分类的基本公式。于是，朴素贝叶斯分类器可表示为

$$y = f(x) = \arg \max_{c_k} \frac{P(Y = c_k) \prod_j P(X^{(j)} = x^{(j)} \mid Y = c_k)}{\sum_k P(Y = c_k) \prod_j P(X^{(j)} = x^{(j)} \mid Y = c_k)} \quad (7)$$

注意到对所有 y , 分母都是相同的, 因此不用计算:

$$y = \arg \max_{c_k} P(Y = c_k) \prod_j P(X^{(j)} = x^{(j)} \mid Y = c_k) \quad (8)$$

参数估计

极大似然估计

在朴素贝叶斯法中, 学习意味着估计 $P(Y = c_k)$ 和 $P(X^{(j)} = x^{(j)} \mid Y = c_k)$. 可以应用极大似然估计法估计相应的概率。先验概率 $P(Y = c_k)$ 的极大似然估计是

$$P(Y = c_k) = \frac{\sum_{i=1}^N I(y_i = c_k)}{N}, \quad k = 1, 2, \dots, K \quad (9)$$

设第 j 个特征 $x^{(j)}$ 可能取值的集合为 $\{a_{j1}, a_{j2}, \dots, a_{js_j}\}$, 条件概率 $P(X^{(j)} = a_{jl} \mid Y = c_k)$ 的极大似然估计是

$$P(X^{(j)} = a_{jl} \mid Y = c_k) = \frac{\sum_{i=1}^N I(x_i^{(j)} = a_{jl}, y_i = c_k)}{\sum_{i=1}^N I(y_i = c_k)} \quad (10)$$

$$j = 1, 2, \dots, n; \quad l = 1, 2, \dots, S_j; \quad k = 1, 2, \dots, K$$

式中, $x_i^{(j)}$ 是第 i 个样本的第 j 个特征; a_{jl} 是第 j 个特征可能取的第 l 个值; I 为指示函数。

贝叶斯估计

用极大似然估计可能会出现所要估计的概率值为 0 的情况。这时会影响到后验概率的计算结果, 使分类产生偏差。解决这一问题的方法是采用贝叶斯估计。具体地, 条件概率的贝叶斯估计是

$$P_\lambda(X^{(j)} = a_{jl} \mid Y = c_k) = \frac{\sum_{i=1}^N I(x_i^{(j)} = a_{jl}, y_i = c_k) + \lambda}{\sum_{i=1}^N I(y_i = c_k) + S_j \lambda} \quad (11)$$

式中 $\lambda \geq 0$ 。等价于在随机变量各个取值的频数上赋予一个正数 $\lambda > 0$ 。当 $\lambda = 0$ 时就是极大似然估计。常取 $\lambda = 1$, 这时称为拉普拉斯平滑 (Laplace smoothing)。显然, 对任何 $l = 1, 2, \dots, S_j, k = 1, 2, \dots, K$, 有

$$\begin{aligned} P_\lambda(X^{(j)} = a_{jl} | Y = c_k) &> 0 \\ \sum_{l=1}^{S_j} P(X^{(j)} = a_j | Y = c_k) &= 1 \end{aligned} \quad (12)$$

表明式 (4.10) 确为一种概率分布。同样, 先验概率的贝叶斯估计

$$P_\lambda(Y = c_k) = \frac{\sum_{i=1}^N I(y_i = c_k) + \lambda}{N + K\lambda} \quad (13)$$

算法流程

输入: 训练数据 $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$, 其中 $x_i = (x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(n)})^T$, $x_i^{(j)}$ 是第 i 个样本的第 j 个特征, $x_i^{(j)} \in \{a_{j1}, a_{j2}, \dots, a_{js_j}\}$, a_{j1} 是第 j 个特征可能取的第 1 个值, $j = 1, 2, \dots, n, l = 1, 2, \dots, S_j, y_i \in \{c_1, c_2, \dots, c_K\}$; 实例 x ;

输出: 实例 x 的分类。

(1) 计算先验概率及条件概率

$$\begin{aligned} P(Y = c_k) &= \frac{\sum_{i=1}^N I(y_i = c_k)}{N}, \quad k = 1, 2, \dots, K \\ P(X^{(j)} = a_{j1} | Y = c_k) &= \frac{\sum_{i=1}^N I(x_i^{(j)} = a_{j1}, y_i = c_k)}{\sum_{i=1}^N I(y_i = c_k)} \\ j &= 1, 2, \dots, n; \quad l = 1, 2, \dots, S_j; \quad k = 1, 2, \dots, K \end{aligned} \quad (14)$$

(2) 对于给定的实例 $x = (x^{(1)}, x^{(2)}, \dots, x^{(n)})^T$, 计算

$$P(Y = c_k) \prod_{j=1}^n P(X^{(j)} = x^{(j)} | Y = c_k), \quad k = 1, 2, \dots, K \quad (15)$$

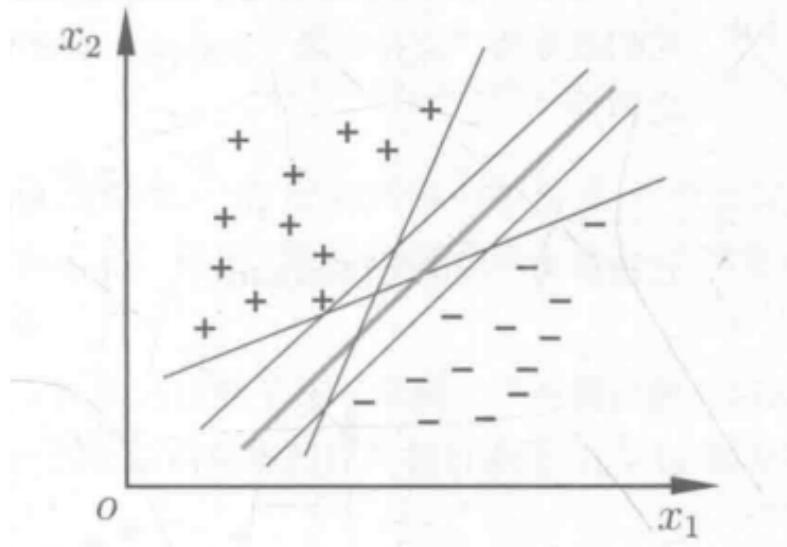
(3) 确定实例 x 的类

$$y = \arg \max_{c_k} P(Y = c_k) \prod_{j=1}^n P(X^{(j)} = x^{(j)} | Y = c_k) \quad (16)$$

支持向量机 (SVM)

线性可分支持向量机

给定训练样本集 $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$, $y_i \in \{-1, +1\}$, 分类学习最基本的想法就是基于训练集 D 在样本空间中找到一个划分超平面, 将不同类别的样本分开。但能将训练样本分开的划分超平面可能有很多, 我们应该努力去找到哪一个呢?



直观上看，应该去找位于两类训练样本“正中间”的划分超平面，因为该划分超平面对训练样本局部扰动的“容忍”性最好。例如，由于训练集的局限性或噪声的因素，训练集外的样本可能比图中的训练样本更接近两个类的分隔界，这将使许多其他划分超平面出现错误，而居中的超平面受影响最小。换言之，这个划分超平面所产生的分类结果是最鲁棒的，对未见示例的泛化能力最强。

在样本空间中，划分超平面可通过如下线性方程来描述：

$$\mathbf{w}^T \mathbf{x} + b = 0 \quad (17)$$

其中 $\mathbf{w} = (w_1, w_2 \cdots w_d)$ 为法向量，决定了超平面的方向； b 为位移项，决定了超平面与原点之间的距离。显然，划分超平面可被法向量 \mathbf{w} 和位移 b 确定，面我们将其记为 (\mathbf{w}, b) 。样本空间中任意点 \mathbf{x} 到超平面 (\mathbf{w}, b) 的距离可写为：

$$r = \frac{|\mathbf{w}^T \mathbf{x} + b|}{\|\mathbf{w}\|} \quad (18)$$

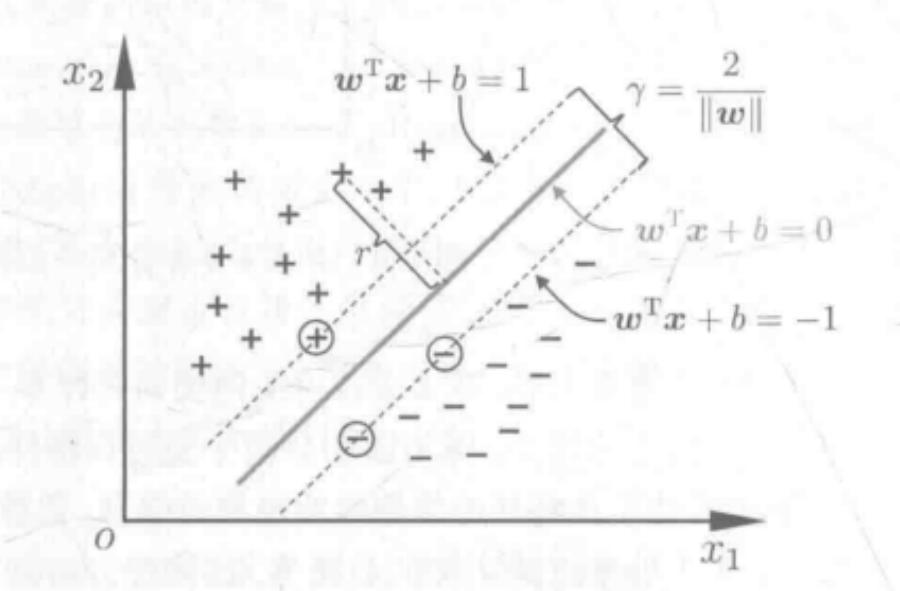
假设超平面 (\mathbf{w}, b) 能将训练样本正确分类，即对于 $(\mathbf{x}_i, y_i) \in D$ ，若 $y_i = +1$ ，则有 $\mathbf{w}^T \mathbf{x}_i + b > 0$ ；若 $y_i = -1$ ，则有 $\mathbf{w}^T \mathbf{x}_i + b < 0$ 。令

$$\begin{cases} \mathbf{w}^T \mathbf{x}_i + b \geqslant +1, & y_i = +1 \\ \mathbf{w}^T \mathbf{x}_i + b \leqslant -1, & y_i = -1 \end{cases} \quad (19)$$

如图所示，距离超平面最近的这几个训练样本点使式的等号成立，它们被称为支持向量 (support vector)，两个异类支持向量到超平面的距离之和为

$$\gamma = \frac{2}{\|\mathbf{w}\|} \quad (20)$$

它被称为间隔(margin)。



欲找到具有“最大间隔”(maximum margin)的划分超平面，也就是要找到能满足式中约束的参数 \mathbf{w} 和 b ，使得 γ 最大，即

$$\begin{aligned} & \max_{\mathbf{w}, b} \frac{2}{\|\mathbf{w}\|} \\ \text{s.t. } & y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \dots, m. \end{aligned} \tag{21}$$

此即支持向量机的基本型。

在决定分离超平面时只有支持向量起作用，而其他实例点并不起作用。如果移动支持向量将改变所求的解；但是如果在间隔边界以外移动其他实例点，甚至去掉这些点，则解是不会改变的。由于支持向量在确定分离超平面中起着决定性作用，所以将这种分类模型称为支持向量机。支持向量的个数一般很少，所以支持向量机由很少的重要的训练样本确定。

对偶方法

为了求解线性可分支持向量机的最优化问题，将它作为原始最优化问题，应用拉格朗日对偶性(参阅附录 C)，通过求解对偶问题(dual problem)得到原始问题(primal problem)的最优解，这就是线性可分支持向量机的对偶算法(dual algorithm)。这样做的优点，一是对偶问题往往更容易求解；二是自然引入核函数，进而推广到非线性分类问题。

首先构建拉格朗日函数 (Lagrange function)。为此，对每一个不等式约束引进拉格朗日乘子 (Lagrange multiplier) $\alpha_i \geq 0, i = 1, 2, \dots, N$ 定义拉格朗日函数：

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^N \alpha_i y_i (\mathbf{w} \cdot \mathbf{x}_i + b) + \sum_{i=1}^N \alpha_i \tag{22}$$

其中， $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_N)^T$ 为拉格朗日乘子向量。

根据拉格朗日对偶性，原始问题的对偶问题是极大极小问题：

$$\max_{\alpha} \min_{\mathbf{w}, b} L(\mathbf{w}, b, \alpha) \tag{23}$$

所以，为了得到对偶问题的解，需要先求 $L(w, b, \alpha)$ 对 w, b 的极小，再求对 α 的极大。

由偏导为 0 可以得到如下条件

$$\max_{\alpha} L(w, b, \alpha) = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \langle x_i \cdot x_j \rangle + \sum_{i=1}^N \alpha_i \quad (24)$$

再求对 α 的极大，有：

$$\begin{aligned} \min_{\alpha} & \quad \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^N \alpha_i \\ \text{s.t.} & \quad \sum_{i=1}^N \alpha_i y_i = 0 \\ & \quad \alpha_i \geq 0, \quad i = 1, 2, \dots, N \end{aligned} \quad (25)$$

重要定理：

设 $\alpha^* = (\alpha_1^*, \alpha_2^*, \dots, \alpha_l^*)^T$ 是对偶最优化问题的解，则存在下标 j ，使得 $\alpha_j^* > 0$ ，并可按下式求得原始最优化问题的解 w^*, b^* ：

$$\begin{aligned} w^* &= \sum_{i=1}^N \alpha_i^* y_i x_i \\ b^* &= y_j - \sum_{i=1}^N \alpha_i^* y_i (x_i \cdot x_j) \end{aligned} \quad (26)$$

线性可分支持向量机——算法

输入：线性可分训练集 $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ ，其中 $x_i \in \mathcal{X} = \mathbf{R}^n, y_i \in \mathcal{Y} = \{-1, +1\}, i = 1, 2, \dots, N$

输出：分离超平面和分类决策函数。

算法分析

(1) 列出对偶问题，构造并求解约束最优化问题

$$\begin{aligned} \min_{\alpha} & \quad \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^N \alpha_i \\ \text{s.t.} & \quad \sum_{i=1}^N \alpha_i y_i = 0 \quad \alpha_i \geq 0, \quad i = 1, 2, \dots, N \end{aligned} \quad (27)$$

求得最优解 $\alpha^* = (\alpha_1^*, \alpha_2^*, \dots, \alpha_N^*)^T$ 。

(2) 计算原始问题解

$$w^* = \sum_{i=1}^N \alpha_i^* y_i x_i \quad (28)$$

并选择 α^* 的一个正分量 $\alpha_j^* > 0$ ，计算

$$b^* = y_j - \sum_{i=1}^N \alpha_i^* y_i (x_i \cdot x_j) \quad (29)$$

(3) 求得分离超平面

$$w^* \cdot x + b^* = 0 \quad (30)$$

分类决策函数:

$$f(x) = \text{sign}(w^* \cdot x + b^*) \quad (31)$$

考试的话就是考这种题，务必熟悉公式和流程，手算对偶算法

例题

正例: $\mathbf{x}_1 = (3, 3)^T$, $\mathbf{x}_2 = (4, 3)^T$, 负例: $x_3 = (1, 1)^T$

1. 支持向量机算法会给每个点一个非负的标量系数 α , 支持向量的系数为正, 非支持向量的系数为 0。因此在求解过程中, 依靠偏微分方法解出的极值点可能不符合题设, 需要舍弃, 考虑边界。
2. 形式上是最小化

$$\begin{aligned} \frac{1}{2} < w_i, w_j > - \sum_{i=1}^N \alpha_i \\ \text{where } w_i &= \sum_{i=1}^N \alpha_i^* y_i x_i \\ w_j &= \sum_{j=1}^N \alpha_j^* y_j x_j \\ \text{hence } b &= y_j - w_i x_j \end{aligned} \quad (32)$$

3. 注意勿忘 $\frac{1}{2}$, 我错过无数次...
4. 我个人认为, 课件上的解答有问题而且讨论片面, 以下是原解答

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^N \alpha_i \\ = \quad & \min_{\alpha} \frac{1}{2} (18\alpha_1^2 + 25\alpha_2^2 + 2\alpha_3^2 + 42\alpha_1\alpha_2 - 12\alpha_1\alpha_3 \\ & - 14\alpha_2\alpha_3) - \alpha_1 - \alpha_2 - \alpha_3 \\ \text{s.t.} \quad & \alpha_1 + \alpha_2 - \alpha_3 = 0 \\ & \alpha_i \geq 0, i = 1, 2, 3 \end{aligned} \quad (33)$$

将 $\alpha_3 = \alpha_1 + \alpha_2$ 代入, 并记为:

$$s(\alpha_1, \alpha_2) = 4\alpha_1^2 + \frac{13}{2}\alpha_2^2 + 10\alpha_1\alpha_2 - 2\alpha_1 - 2\alpha_2 \quad (34)$$

通过求偏导并令其为 0, 易知 $s(\alpha_1, \alpha_2)$ 在点 $(\frac{3}{2}, -1)^T$ 取极值, 但该点不满足约束 $\alpha_2 \geq 0$ 所以最小值应该在边界上。

当 $\alpha_1 = 0$ 时，最小值 $s(0, \frac{1}{13}) = -\frac{1}{13}$ ，当 $\alpha_2 = 0$ 时，最小值 $s(\frac{1}{4}, 0) = -\frac{1}{4}$ ，于是 $s(\alpha_1, \alpha_2)$ 在 $\alpha_1 = \frac{1}{4}, \alpha_2 = 0$ 时达到最小，此时 $\alpha_3 = \alpha_1 + \alpha_2 = \frac{1}{4}$ 这样 α_1, α_3 对应的实例点 x_1, x_3 是支持向量

根据前面的公式得到：

$$w_1^* = w_2^* = \frac{1}{2}$$

$$b^* = -2$$

分离超平面为： $\frac{1}{2}x^{(1)} + \frac{1}{2}x^{(2)} - 2 = 0$

分类决策函数为： $f(x) = \text{sign}(\frac{1}{2}x^{(1)} + \frac{1}{2}x^{(2)} - 2)$

```
In[18]:= Minimize[{9 x^2 + 12.5 y^2 + z^2 + 21 x*y - 6 x*z - 7 y*z - x - y - z,
| 最小点值
  x + y - z == 0 && x >= 0 && y >= 0 && z >= 0}, {x, y, z}]
```

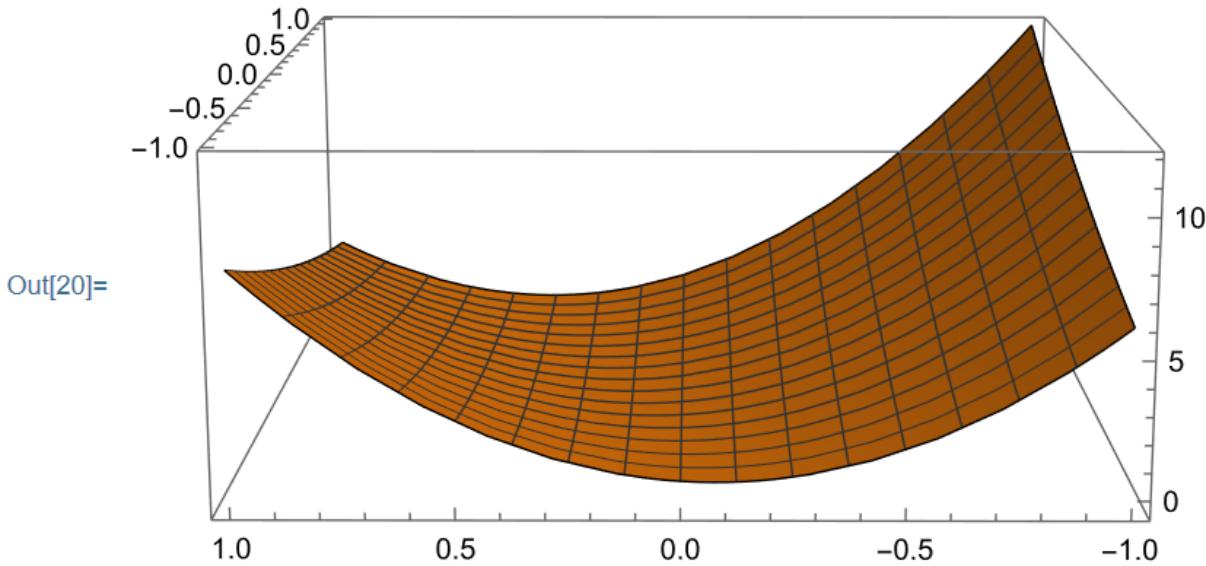
```
Out[18]= {-0.25, {x -> 0.25, y -> 5.47952*10^-9, z -> 0.25}}
```

答案当然是对的，我用 mathematica 验证了，但是过程有问题。多元可微函数最值在极值点或者边界取得，此处他否决了极值点求到的最值，认为最值点在边界取得，但是边界没讨论完全，至少应该对着 $\alpha_3 = 0$ 再讨论一波。 $\alpha_3 = 0$ 时，最小值在 $s(0, 0, 0) = 0$ 取得，虽然没被选择，但是实际上不能被遗忘。而且课件给我的感觉是，他带入 $\alpha_3 = \alpha_1 + \alpha_2$ 之后，就忽视了讨论 $\alpha_3 = 0$ ，而不是因为一眼就能看出 $\alpha_3 = 0$ 会被否决而没写。

实际上按照课件的做法，一开始带入 $\alpha_2 = \alpha_3 - \alpha_1$ ，得到了 $0.5\alpha_1^2 + 3\alpha_1\alpha_3 + 6.5\alpha_3^2 - 2\alpha_3$ ，那只验证 $\alpha_1 = 0$ 或者 $\alpha_3 = 0$ 一定得不出正确答案。此处验证 $\alpha_2 = 0$ 直接在化简的式子里带入 $\alpha_3 = \alpha_1$ 即可。

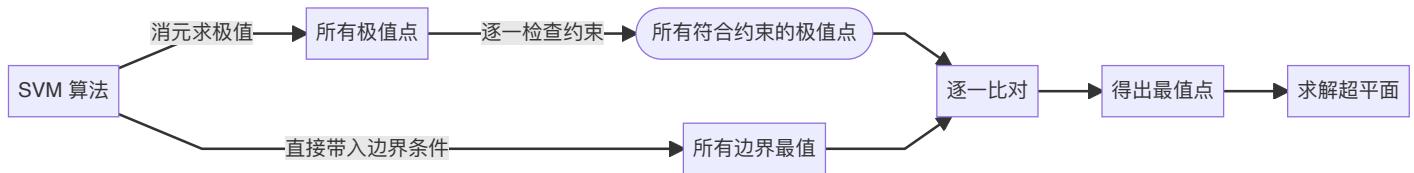
而且，这个地方企图通过极值点来默认充当极小值点也有问题，实际上，还应该判断 Hessian 矩阵或者计算高阶导数，此处课件忽视了这件事。可能他默认了一个结论，SVM 一定是个凸优化问题，得到的目标函数一定是凸函数（下凸函数），求出的极值点都是极小值。

In[20]:= Plot3D[0.5*x^2 - 3*x*y - 2*y + 6.5*y^2, {x, -1, 1}, {y, -1, 1}]
绘制三维图形



5. 这实际上完全是个多元函数求最值的问题，回顾下大家在微积分 A2 里打下的扎实数理基础，老师绝对有说过，求最值是最复杂的。正确的做法是：

- 首先经过化简，能够得到极值点集合
- 其次，无论如何都要检验边界，因为最值在边界或者极值点取得，极值点符合约束也为不一定就是最值点。所以直接把边界条件带回最原始的式子，对每个 $\alpha_i = 0$ 逐一讨论
- 检验极值点是否符合约束（一般而言就是 $\alpha_i \geq 0$ 的约束条件），符合约束则需要与边界点一一比对。不符合则最值在边界条件取得，一一比对每个边界最值点。



线性支持向量机与软间隔最大化

线性可分问题的支持向量机学习方法，对线性不可分训练数据是不适用的，因为这时上述方法中的不等式约束并不能都成立。怎么才能将它扩展到线性不可分问题呢？这就需要修改硬间隔最大化，使其成为软间隔最大化。

线性不可分意味着某些样本点 (x_i, y_i) 不能满足函数间隔大于等于 1 的约束条件。为了解决这个问题，可以对每个样本点 (x_i, y_i) 引进一个松弛变量 $\xi_i \geq 0$ ，使函数间隔加上松弛变量大于等于 1。这样，约束条件变为

$$w \cdot (w \cdot x_i + b) \geq 1 - \xi_i \quad (35)$$

同时, 对每个松弛变量 ξ_i , 支付一个代价 ξ_i 。目标函数由原来的 $\frac{1}{2}\|w\|^2$ 变成

$$\frac{1}{2}\|w\|^2 + C \sum_{i=1}^N \xi_i \quad (36)$$

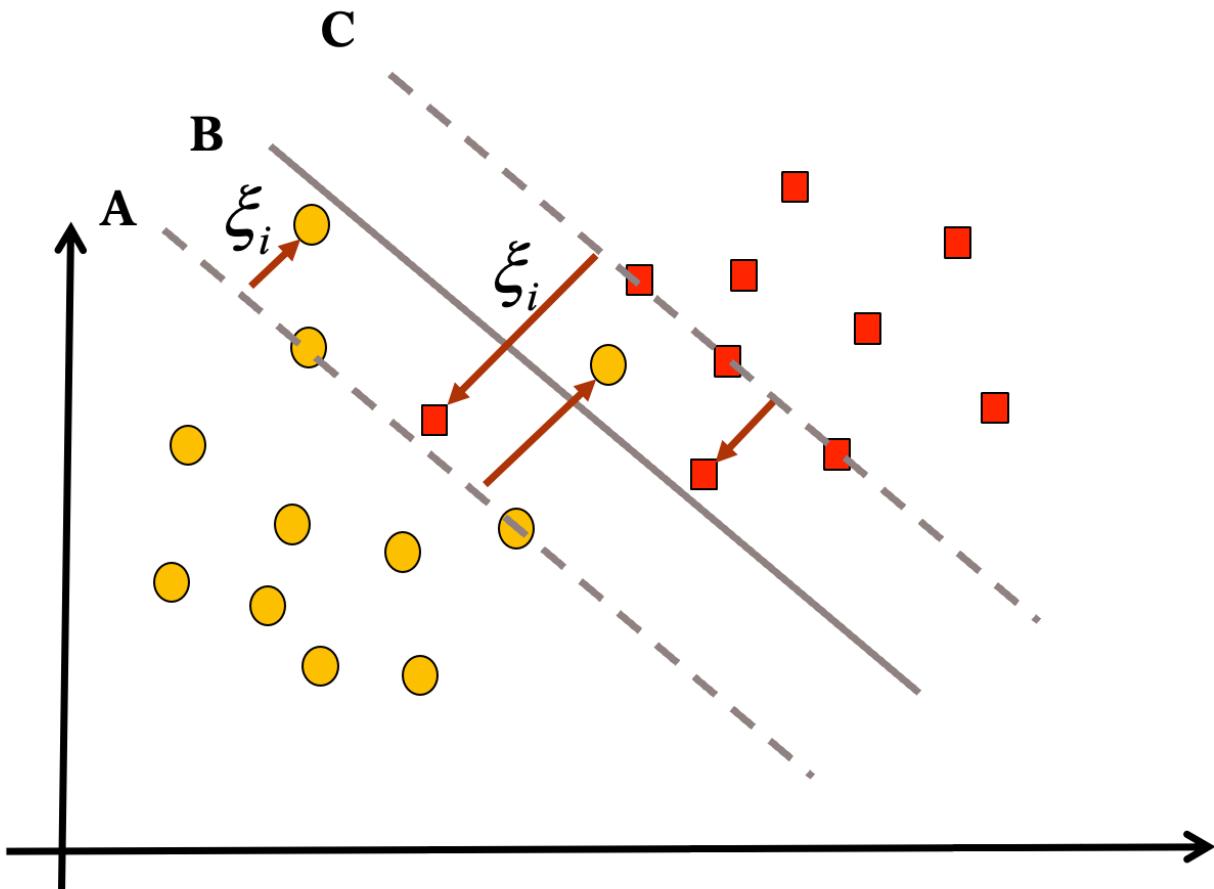
这里, $C > 0$ 称为惩罚参数, 一般由应用问题决定, C 值大时对误分类的惩罚增大, C 值小时对误分类的惩罚减小。最小化目标函数包含两层含义: 使 $\frac{1}{2}\|w\|^2$ 尽量小即间隔尽量大, 同时使误分类点的个数尽量小, C 是调和二者的系数。

因此, 问题转化为

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + C \sum_{i=1}^N \xi_i \\ \text{s.t.} \quad & y_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad i = 1, 2, \dots, N \\ & \xi_i \geq 0, \quad i = 1, 2, \dots, N \end{aligned} \quad (37)$$

理解

线性支持向量机



- ξ_i 实际上是不可线性分割的向量 (软支持向量) 的松弛距离, 到达支持向量线的欧氏距离
- C 取作 $+\infty$ 时, $\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C \sum_{i=1}^N \xi_i$ 的方法就是让所有 $\xi = 0$, 此即硬支持向量机
- 若 $\xi^* < C$ 则 $\xi^* = 0$ 表示该点为硬支持向量

- 3. 若 $\alpha_i < 0$, 则 $\xi_i = 0$, 又因为 x_i 在间隔边界与分离超平面之间, 既又在间隔上,
- 4. 若 $\alpha_i^* = C, 0 < \xi_i < 1$, 则分类正确, x_i 在间隔边界与分离超平面之间
- 5. 若 $\alpha_i^* = C, \xi_i = 1$, 则 x_i 在分离超平面上; 若 $\alpha_i^* = C, \xi_i > 1$, 则 x_i 位于分离超平面误分一侧

算法

输入: 训练数据集 $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$, 其中,
 $x_i \in \mathcal{X} = \mathbf{R}^n, y_i \in \mathcal{Y} = \{-1, +1\}, i = 1, 2, \dots, N$;

输出: 分离超平面和分类决策函数。

算法分析

(1) 列出对偶问题, 选择惩罚参数 $C > 0$, 构造并求解凸二次规划问题 (虽然原始问题形式不同, 但是对偶问题形式相同)

$$\begin{aligned} \min_{\alpha} \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^N \alpha_i \\ s.t. \sum_{i=1}^N \alpha_i y_i = 0 \end{aligned} \tag{38}$$

$$0 \leq \alpha_i \leq C, \quad i = 1, 2, \dots, N$$

求得最优解 $\alpha^* = (\alpha_1^*, \alpha_2^*, \dots, \alpha_N^*)^T$.

(2) 计算原始问题解, 计算 $w^* = \sum_{i=1}^N \alpha_i^* y_i x_i$

选择 α^* 的一个分量 α_j^* 适合条件 $0 < \alpha_j^* < C$, 计算

$$b^* = y_j - \sum_{i=1}^N y_i \alpha_i^* (x_i \cdot x_j) \tag{39}$$

(3) 获得答案, 求得分离超平面

$$w^* \cdot x + b^* = 0 \tag{40}$$

分类决策函数:

$$f(x) = \text{sign}(w^* \cdot x + b^*) \tag{41}$$

步骤(2)中, 对任一适合条件 $0 < \alpha_j^* < C$ 的 α_j^* , 按式都可求出 b^* , 但是由于原始问题对 b 的解并不唯一, 所以实际计算时可以取在所有符合条件的样本点上的平均值。

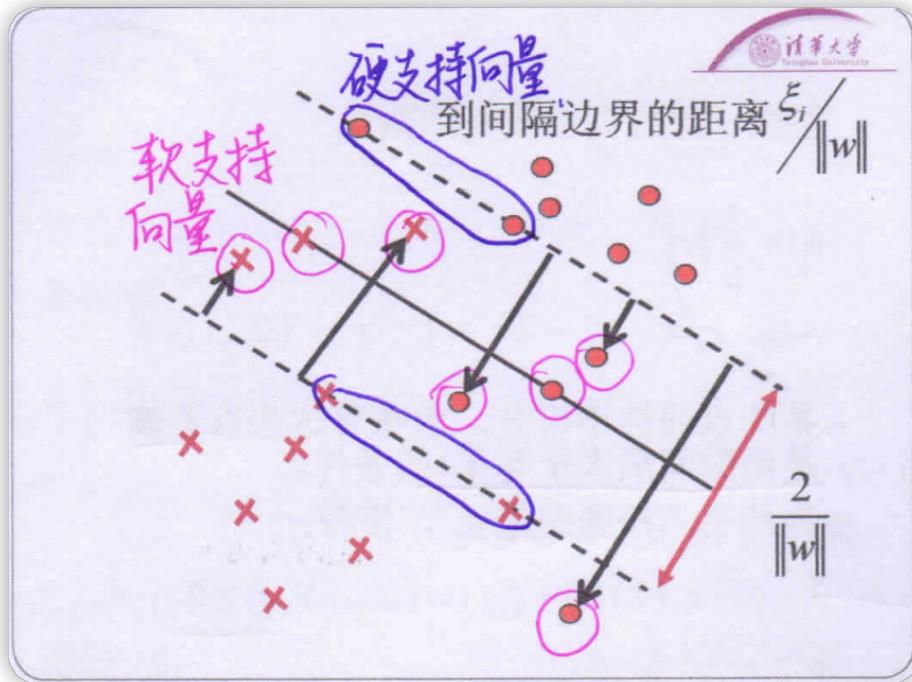
注意与前者的区别, 计算过程大体相同

又待加重

在线性不可分的情况下, 将对偶问题的解 $\alpha^* = (\alpha_1^*, \alpha_2^*, \dots, \alpha_N^*)^T$ 中对应于 $\alpha_i^* > 0$ 的样本点 (x_i, y_i) 的实例 x_i 称为支持向量(软间隔的支持向量). 如图 7.5 所示, 这时的支持向量要比线性可分时的情况复杂一些。图中, 分离超平面由实线表示, 间隔边界由虚线表示, 正例点由“.”表示, 负例点由“ \times ”表示。图中还标出了实例 x_i 到间隔边界的距离 $\frac{\xi_i}{\|w\|}$ 。

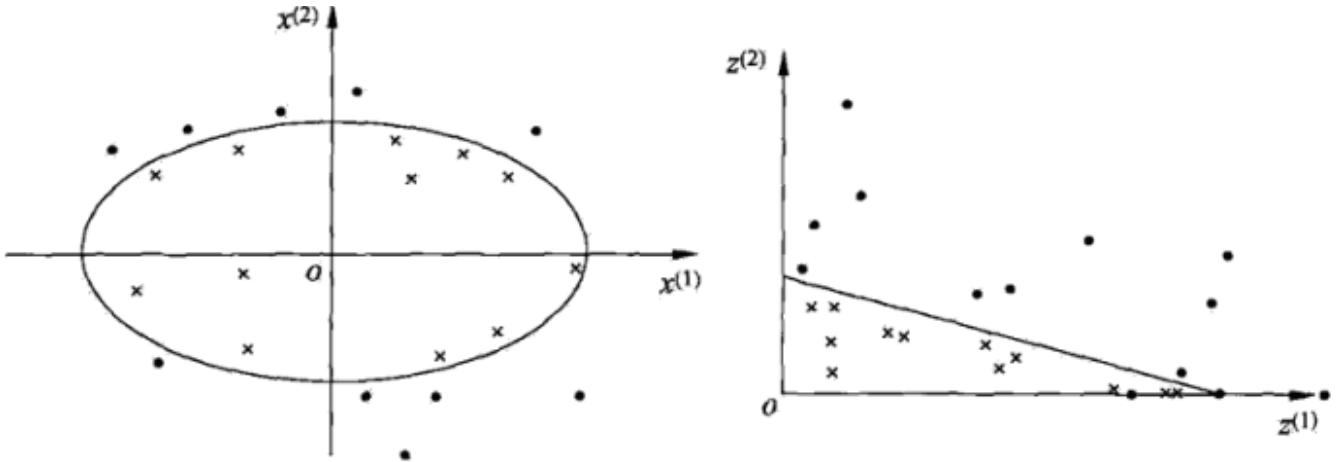
软间隔的支持向量 x_i 或者在间隔边界上, 或者在间隔边界与分离超平面之间, 或者在分离超平面误分一侧。

- 若 $\alpha_i^* < C$, 则 $\xi_i = 0$, 支持向量 x_i 恰好落在间隔边界上;
- 若 $\alpha_i^* = C$, $0 < \xi_i < 1$, 则分类正确, x_i 在间隔边界与分离超平面之间;
- 若 $\alpha_i^* = C$, $\xi_i = 1$, 则 x_i 在分离超平面上; 若 $\alpha_i^* = C$, $\xi_i > 1$, 则 x_i 位于分离超平面误分一侧。



非线性可分支持向量机与核函数

非线性问题往往不好求解, 所以希望能用解线性分类问题的方法解决这个问题. 所采取的方法是进行一个非线性变换, 将非线性问题变换为线性问题, 通过解变换后的线性问题的方法求解原来的非线性问题. 例如: 通过变换, 将左图中椭圆变成右图中的直线, 将非线性分类问题变换为线性分类问题.



核函数

设 \mathcal{X} 是输入空间 (欧氏空间 \mathbf{R}^n 的子集或离散集合), 设 \mathcal{H} 为特征空间 (希尔伯特空间) , 如果存在一个从 \mathcal{X} 到 \mathcal{H} 的映射

$$\phi(x) : \mathcal{X} \rightarrow \mathcal{H} \quad (42)$$

使得对所有 $x, z \in \mathcal{X}$, 函数 $K(x, z)$ 满足条件

$$K(x, z) = \phi(x) \cdot \phi(z) \quad (43)$$

则称 $K(x, z)$ 为核函数, $\phi(x)$ 为映射函数, 式中 $\phi(x) \cdot \phi(z)$ 为 $\phi(x)$ 和 $\phi(z)$ 的内积。

应用于支持向量机中:

在线性支持向量机的对偶问题中, 无论是目标函数还是决策函数 (分离超平面) 都只涉及输入实例与实例之间的内积。在对偶问题的目标函数中的内积 $x_i \cdot x_j$ 可以用核函数 $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$ 来代替. 此时对偶问题的目标函数成

$$W(\alpha) = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^N \alpha_i \quad (44)$$

同样, 分类决策函数中的内积也可以用核函数代替, 而分类决策函数式成

$$f(x) = \text{sign} \left(\sum_{i=1}^{N_s} a_i^* y_i \phi(x_i) \cdot \phi(x) + b^* \right) = \text{sign} \left(\sum_{i=1}^{N_1} a_i^* y_i K(x_i, x) + b^* \right) \quad (45)$$

这等价于经过映射函数 ϕ 将原来的输入空间变换到一个新的特征空间, 将输入空间中的内积 $x_i \cdot x_j$ 变换为特征空间中的内积 $\phi(x_i) \cdot \phi(x_j)$, 在新的特征空间里从训练样本中学习线性支持向量机。当映射函数是非线性函数时, 学习到的含有核函数的支持向量机是非线性分类模型。

- 多项式核函数

$$K(x, z) = (x \cdot z + 1)^p \quad (46)$$

- 高斯核函数

$$K(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right) \quad (47)$$

理解

实际上就是把之前的内积运算改成了核函数的内积运算

算法

输入: 训练数据集 $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$, 其中 $x_i \in \mathcal{X} = \mathbf{R}^n, y_i \in \mathcal{Y} = \{-1, +1\}, i = 1, 2, \dots, N$;

输出: 分类决策函数。

算法分析

(1) 列出对偶问题, 选取适当的核函数 $K(x, z)$ 和适当的参数 C , 构造并求解最优化问题

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^N \alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^N \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, \quad i = 1, 2, \dots, N \end{aligned} \quad (48)$$

求得最优解 $\alpha^* = (\alpha_1^*, \alpha_2^*, \dots, \alpha_N^*)^T$.

(2) 计算原始问题解, 选择 α^* 的一个正分量 $0 < \alpha_j^* < C$, 计算

$$b^* = y_j - \sum_{i=l}^N \alpha_i^* y_i K(x_i \cdot x_j) \quad (49)$$

(3) 获得答案, 构造决策函数

$$f(x) = \text{sign} \left(\sum_{i=1}^N \alpha_i^* y_i K(x \cdot x_i) + b^* \right) \quad (50)$$

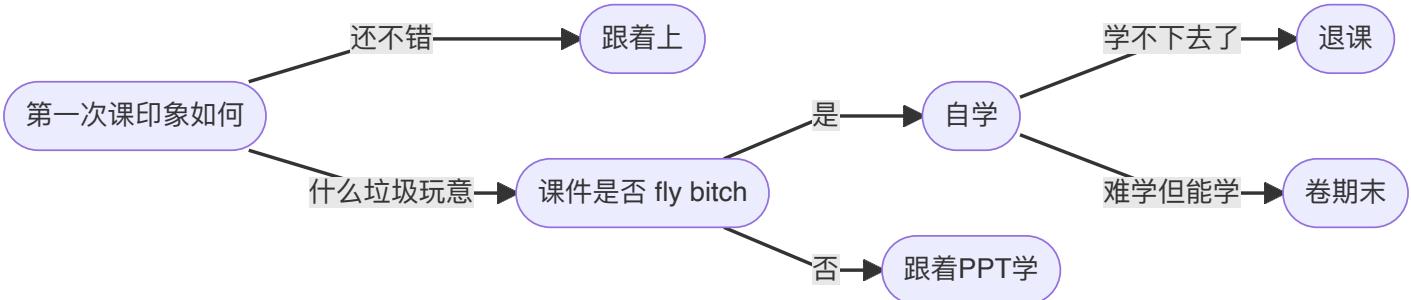
注意决策函数的区别

决策树 (Decision-Tree)

决策树(decision tree)是一种基本的分类与回归方法。

举个通俗易懂的例子, 如下图所示的上课流程图就是一个决策树。长方形代表判断模块(**decision block**), 椭圆形

成代表终止模块(**terminating block**)，表示已经得出结论，可以终止运行。从判断模块引出的左右箭头称作为**分支**(**branch**)，它可以达到另一个判断模块或者终止模块。我们还可以这样理解，分类决策树模型是一种描述对实例进行分类的树形结构。决策树由结点(node)和有向边(directed edge)组成。结点有两种类型：内部结点(internal node)和叶结点(leaf node)。内部结点表示一个特征或属性，叶结点表示一个类。



这只是一个简单的分类流程图，真实情况根据 feature 数量而复杂度大大升高。

原理

我们可以把决策树看成一个 if-then 规则的集合，将决策树转换成 if-then 规则的过程是这样的：

由决策树的根结点(root node)到叶结点(leaf node)的每一条路径构建一条规则；路径上内部结点的特征对应着规则的条件，而叶结点的类对应着规则的结论。

决策树的路径或其对应的 if-then 规则集合具有一个重要的性质：互斥并且完备。这就是说，每一个实例都被一条路径或一条规则所覆盖，而且只被一条路径或一条规则所覆盖。这里所覆盖是指实例的特征与路径上的特征一致或实例满足规则的条件。

决策树的构建

一般分为三步：特征选择、决策树的生成和决策树的修剪。

特征选择

特征选择在于选取对训练数据具有分类能力的特征。这样可以提高决策树学习的效率，如果利用一个特征进行分类的结果与随机分类的结果没有很大差别，则称这个特征是没有分类能力的。经验上扔掉这样的特征对决策树学习的精度影响不大。通常特征选择的标准是信息增益(**information gain**)或信息增益比，为了简单，使用信息增益作为选择特征的标准。

什么是信息增益？考虑确定选择特征的准则。直观上，如果一个特征具有更好的分类能力，或者说，按照这一特征将训练数据集分割成子集，使得各个子集在当前条件下有最好的分类，那么就更应该选择这个特征。信息增益就能够很好地表示这一直观的准则。

在划分数据集前后信息发生的变化称为信息增益，知道如何计算信息增益，我们就可以计算每个特征值划分数据集获得的信息增益，获得信息增益最高的特征就是最好的选择。

在可以评测哪个数据划分方式是最好的数据划分之前，我们必须学习如何计算信息增益。集合信息的度量方式称为香农熵或者简称为熵(**entropy**)，这个名字来源于信息论之父克劳德·香农（没错，就是提出香农定理的那位祖师爷）。

熵定义为信息的期望值。在信息论与概率统计中，熵是表示随机变量不确定性的度量。如果待分类的事物可能划分在多个分类之中，则 x 的信息定义为：

$$l(x_i) = -\log_2 p(x_i) \quad (51)$$

其中 $p(x_i)$ 是选择该分类的概率。

通过上式，我们可以得到所有类别的信息。为了计算熵，我们需要计算所有类别所有可能值包含的信息期望值(数学期望)，通过下面的公式得到：

$$H = - \sum_{i=1}^n p(x_i) \log_2 p(x_i) \quad (52)$$

其中 n 是分类的数目。熵越大，随机变量的不确定性就越大。

当熵中的概率由数据估计(特别是最大似然估计)得到时，所对应的熵称为经验熵(**empirical entropy**)。

我们定义样本数据为训练数据集 D ，则训练数据集 D 的经验熵为 $H(D)$ ， $|D|$ 表示其样本容量，也即样本个数。设有 k 个类 $C_k = 1, 2, 3 \dots k$ ， $|C_k|$ 为属于类 k 的样本个数，因此经验熵公式就可以写为：

$$H(D) = - \sum_{k=1}^K \frac{|C_k|}{|D|} \log_2 \frac{|C_k|}{|D|} \quad (53)$$

信息增益

如何选择特征，需要看信息增益。也就是说，信息增益是相对于特征而言的，信息增益越大，特征对最终的分类结果影响也就越大，我们就应该选择对最终分类结果影响最大的那个特征作为我们的分类特征。

条件熵

随机变量 Y 的条件熵(**conditional entropy**) $H(Y|X)$ ，定义为 X 给定条件下 Y 的条件概率分布的熵对 X 的数学期望(下为全概率公式)：

$$H(Y | X) = \sum_{i=1}^n p_i H(Y | X = x_i) \quad (54)$$

此处

$$p_i = P(X = x_i), i = 1, 2, \dots, n \quad (55)$$

紧接着，我们有特征 A 对训练数据集 D 的信息增益 $g(D, A)$ ，定义为集合 D 的经验熵 $H(D)$ 与特征 A 给定条件下 D 的经验条件熵 $H(D | A)$ 之差，即：

$$g(D, A) = H(D) - H(D | A) \quad (56)$$

设训练数据集为 D , $|D|$ 表示其样本容量, 即样本个数. 设有 K 个类 $C_k, k = 1, 2, \dots, K, |C_k|$ 为属于类 k 的样本

个数, $\sum_{k=1}^K |C_k| = |D|$. 设特征 A 有 n 个不同的取值 $\{a_1, a_2, \dots, a_n\}$, 根据特征 A 的取值将 D 划分为 n 个子集 D_1, D_2, \dots, D_n , $|D_i|$ 为 D_i 的样本个数, $\sum_{i=1}^n |D_i| = |D|$. 记子集 D_i 中属于类 C_k 的样本的集合为 D_{ik} , 即 $D_{ik} = D_i \cap C_k$, $|D_{ik}|$ 为 D_{ik} 的样本个数。

则条件熵为:

$$H(D | A) = \sum_{i=1}^n \frac{|D_i|}{|D|} H(D_i) = - \sum_{i=1}^n \frac{|D_i|}{|D|} \sum_{k=1}^K \frac{|D_{ik}|}{|D_i|} \log_2 \frac{|D_{ik}|}{|D_i|} \quad (57)$$

先积类别, 再积特征!

信息增益算法

输入: 训练数据集 D 和特征 A;

输出: 特征 A 对训练数据集 D 的信息增益 $g(D, A)$.

(1) 计算数据集 D 的经验熵 $H(D)$

$$H(D) = - \sum_{k=1}^K \frac{|D_k|}{|D|} \log_2 \frac{|D_k|}{|D|} \quad (58)$$

(2) 计算特征 A 对数据集 D 的经验条件熵 $H(D | A)$

$$H(D | A) = \sum_{i=1}^n \frac{|D_i|}{|D|} H(D_i) = - \sum_{i=1}^n \frac{|D_i|}{|D|} \sum_{k=1}^K \frac{|D_{ik}|}{|D_i|} \log_2 \frac{|D_{ik}|}{|D_i|} \quad (59)$$

(3) 计算信息增益:

$$g(D, A) = H(D) - H(D | A) \quad (60)$$

信息增益比

信息增益值的大小是相对于训练数据集而言的, 并没有绝对意义。在分类问题困难时, 也就是说在训练数据集的经验熵大的时候, 信息增益值会偏大。反之, 信息增益值会偏小。使用信息增益比 (information gain ratio) 可以对这一问题进行校正, 这是特征选择的另一准则。

特征 A 对训练数据集 D 的信息增益比 $g_R(D, A)$ 定义为其信息增益 $g(D, A)$ 与其对训练数据集 D 的经验熵 $H_A(D)$ 之比:

$$g_R(D, A) = \frac{g(D, A)}{H_A(D)} = \frac{H(D) - H(D | A)}{H_A(D)} \quad (61)$$

其实这里定义了三种熵, 如下图所示:

ID	年龄 A1	有工作 A2	有房子 A3	信用情况 A4	类别
1	青年	否	否	一般	否
2	青年	否	否	好	否
3	青年	是	否	好	是
4	青年	是	是	一般	是
5	青年	否	否	一般	否
6	中年	否	否	一般	否
7	中年	否	否	好	否
8	中年	是	是	好	是
9	中年	否	是	非常好	是
10	中年	否	是	非常好	是
11	老年	否	是	非常好	是
12	老年	否	是	好	是
13	老年	是	否	好	是
14	老年	是	否	非常好	是
15	老年	否	否	一般	否

以下用 A 来表示属性（年龄，有无工作，有无房产这些是属性），用 D 表示训练集，训练集被类别分为了若干个类，比如上图的是与否。

$H(D)$ 是经验熵，也即对训练集被类别分成的若干个类做熵。

$$H(D) = - \sum_{i=1}^K \frac{|D_i|}{|D|} \log_2 \frac{|D_i|}{|D|} \quad (62)$$

$H_A(D)$ 是 A 属性对训练集 D 的经验熵，也即对训练集被 A 属性分成的若干个类做熵。

$$H_A(D) = - \sum_{i=1}^K \frac{|D_{A_i}|}{|D|} \log_2 \frac{|D_{A_i}|}{|D|} \quad (63)$$

$H(D | A)$ 是特征 A 对数据集 D 的经验条件熵，也即先积属性，再积特征。

$$H(D | A) = \sum_{i=1}^n \frac{|D_i|}{|D|} H(D_i) = - \sum_{i=1}^n \frac{|D_i|}{|D|} \sum_{k=1}^K \frac{|D_{ik}|}{|D_i|} \log_2 \frac{|D_{ik}|}{|D_i|} \quad (64)$$

决策树的生成

我们已经学习了从数据集构造决策树算法所需要的子功能模块，包括经验熵的计算和最优特征的选择，其工

作原理如下：得到原始数据集，然后基于最好的属性值划分数据集，由于特征值可能多于两个，因此可能存在大于两个分支的数据集划分。第一次划分之后，数据集被向下传递到树的分支的下一个结点。在这个结点上，我们可以再次划分数据。因此我们可以采用递归的原则处理数据集。

ID3算法

ID3算法的核心是在决策树各个结点上应用信息增益准则选择特征，递归地构建决策树。具体方法是：从根结点(root node)开始，对结点计算所有可能的特征的信息增益，选择信息增益最大的特征作为结点的特征，由该特征的不同取值建立子结点；再对子结点递归地调用以上方法，构建决策树：直到所有特征的信息增益均很小或没有特征可以选择为止。最后得到一个决策树，ID3相当于用极大似然法进行概率模型的选择。

输入：训练数据集 D，特征集 A，阈值 ε ；

输出：决策树 T。

1. 若 D 中所有实例属于同一类 C_k ，则 T 为单结点树，并将类 C_k 作为该结点的类标记，返回 T；
2. 若 $A = \emptyset$ ，则 T 为单结点树，并将 D 中实例数最大的类 C_k 作为该结点的类标记，返回 T；
3. 否则，计算 A 中各特征对 D 的信息增益，选择信息增益最大的特征 A_g ；
4. 如果 A_g 的信息增益小于阈值 ε ，则置 T 为单结点树，并将 D 中实例数最大的类 C_k 作为该结点的类标记，返回 T；
5. 否则，对 A_g 的每一可能值 a_i ，依 $A_g = a_i$ 将 D 分割为若干非空子集 D_i ，将 D_i 中实例数最大的类作为类标记，构建子结点，由结点及其子结点构成树 T，返回 T；
6. 对第 i 个子结点，以 D_i 为训练集，以 $A - \{A_g\}$ 为特征集，递归地调用步(1)~步(5)，得到子树 T_i ，返回 T_i 。

C4.5算法

与上面完全相同，不过使用了信息增益比作为判断依据。

ID	年龄 A1	有工作 A2	有房子 A3	信贷情况 A4	类别
1	青年	否	否	一般	否
2	青年	否	否	好	否
3	青年	是	否	好	是
4	青年	是	是	一般	是
5	青年	否	否	一般	否
6	中年	否	否	一般	否
7	中年	否	否	好	否
8	中年	是	是	好	是
9	中年	否	是	非常好	是
10	中年	否	是	非常好	是
11	老年	否	是	非常好	是
12	老年	否	是	好	是
13	老年	是	否	好	是
14	老年	是	否	非常好	是
15	老年	否	否	一般	否

对此图，采用 ID3 算法，直接先对 A3 分类，再对 A2 分类熵就归零了。

采用 C4.5 算法：

$$H(D) = -\frac{9}{15} \log_2 \frac{9}{15} - \frac{6}{15} \log_2 \frac{6}{15} = 0.971 \quad (65)$$

$$H(D|A_1) = 0.8879, H(D|A_2) = 0.647, H(D|A_3) = 0.5509, H(D|A_4) = 0.60796$$

$$H_{A_1}(D) = 1.585, H_{A_2}(D) = 0.918, H_{A_3}(D) = 0.971, H_{A_4}(D) = 0.363$$

决策树剪枝

决策树生成算法递归地产生决策树，直到不能继续下去为止，这样产生的树往往对训练数据的分类很准确，但对未知的测试数据的分类却没有那么准确，即出现过拟合现象。过拟合的原因在于学习时过多地考虑如何提高对训练数据的正确分类，从而构建出过于复杂的决策树。解决这个问题的办法是考虑降低决策树的复杂度，对已生成的决策树进行简化。

决策树的剪枝往往通过极小化决策树整体的损失函数 (loss function) 来实现。设树 T 的叶结点个数为 $|T|$, t 是树 T 的叶结点, 该叶结点有 N_t 个样本点, 其中 k 类的样本点有 N_{tk} 个, $k = 1, 2, \dots, K$, $H_t(T)$ 为叶节点 t 上的经验熵, $\alpha \geq 0$ 为参数, 则决策树学习的损失函数可以定义为

$$C_\alpha(T) = \sum_{t=1}^{|T|} N_t H_t(T) + \alpha |T| \quad (66)$$

其中经验熵为

$$H_t(T) = - \sum_k \frac{N_{tk}}{N_t} \log \frac{N_{tk}}{N_t} \quad (67)$$

在损失函数中：

$$C(T) := \sum_{t=1}^{|T|} N_t H_t(T) = - \sum_{t=1}^{|T|} \sum_{k=1}^K N_{tk} \log \frac{N_{tk}}{N_t} \quad (68)$$

这时有

$$C_\alpha(T) = C(T) + \alpha |T| \quad (69)$$

剪枝算法

输入: 生成算法产生的整个树 T , 参数 α ;

输出: 修剪后的子树 T_α .

(1) 计算每个结点的经验熵.

(2) 递归地从树的叶结点向上回缩. 设一组叶结点回缩到其父结点之前与之后的整体树分别为 T_B 与 T_A , 其对应的损失函数值分别是 $C_\alpha(T_B)$ 与 $C_\alpha(T_A)$, 如果

$$C_\alpha(T_A) \leq C_\alpha(T_B) \quad (70)$$

则进行剪枝, 即将父结点变为新的叶结点。

(3) 返回 (2), 直至不能继续为止, 得到损失函数最小的子树 T_α 。

不太相信能考剪枝, 毕竟计算量太大了, 不过还是掌握原理和公式吧 (还是看看远处的雪山吧, 嘉人们)