

第三章 线性模型

线性模型 (Linear Model) 是机器学习中应用最广泛的模型, 指通过样本特征的线性组合来进行预测的模型。上一章中介绍的线性回归就是典型的线性模型。给定一个 d 维样本 $[x_1, \dots, x_d]^T$, 其线性组合函数为

$$f(\mathbf{x}, \mathbf{w}) = w_1x_1 + w_2x_2 + \dots + w_dx_d + b \quad (3.1)$$

$$= \mathbf{w}^T \mathbf{x}, \quad (3.2)$$

其中 $\mathbf{x} = [1, x_1, \dots, x_d]^T$ 和 $\mathbf{w} = [b, w_1, \dots, w_d]^T$ 分别为 $d + 1$ 维的增广特征向量和增广权重向量。

在线性回归问题中, 可以直接用 $f(\mathbf{x}, \mathbf{w})$ 来预测输出目标。但在分类问题中, 由于输出目标是一些离散的标签或者是这些标签的后验概率 (在 $(0, 1)$ 之间), 而 $f(\mathbf{x}, \mathbf{w})$ 的值域为实数, 因此无法直接用 $f(\mathbf{x}, \mathbf{w})$ 来进行预测, 需要引入一个非线性的**激活函数** (Activation Function) $g(\cdot)$ 来预测输出目标。

在统计文献中, $g^{-1}(\cdot)$ 也称为联系函数 (Link Function)。

$$g(\mathbf{x}, \mathbf{w}) = g(f(\mathbf{x}, \mathbf{w})) \quad (3.3)$$

$$= g(\mathbf{w}^T \mathbf{x}). \quad (3.4)$$

对于简单的两类分类问题, $g(\cdot)$ 可以为阶跃函数:

$$g(\mathbf{x}, \mathbf{w}) = \begin{cases} +1 & \text{当 } \mathbf{w}^T \mathbf{x} > 0 \\ -1 & \text{当 } \mathbf{w}^T \mathbf{x} < 0 \end{cases} \quad (3.5)$$

公式(3.5)定义了一个典型的两类分类问题的线性决策函数, 其结构如图3.1所示。在高维的特征空间中, 所有满足 $f(\mathbf{x}, \mathbf{w}) = 0$ 的点组成用一个分割**超平面** (Hyperplane)。这个超平面将特征空间一分为二, 划分成两个区域, 每个区域

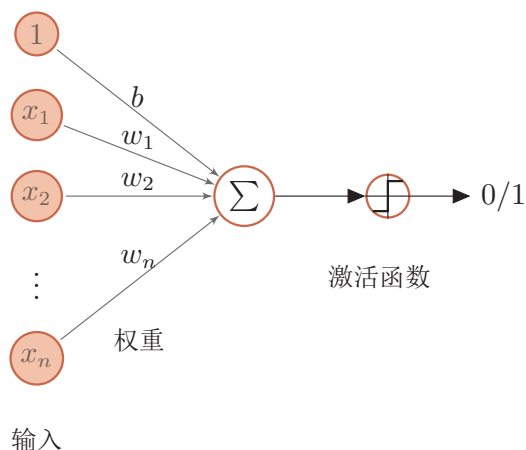


图 3.1: 两类分类的线性模型。

对应一个类别。因此，这个分割超平面也称为**决策边界**（Decision Boundary）或**决策平面**（Decision Surface）。在二维空间中，决策边界为一个直线。在三维空间中，分类界面为一个平面。在高维空间中，分类界面为一个超平面。

对于分类问题，如果一个分类模型的决策边界是由样本特征 \mathbf{x} 的线性函数 $f(\mathbf{x}, \mathbf{w})$ 构成，我们就称这个模型为**线性分类模型**，也叫**线性分类器**（Linear Classifier）。线性函数 $f(\mathbf{x}, \mathbf{w})$ 也称为**线性判别函数**（Linear Discriminant Function）。图3.2中给了一个二维数据的线性判别函数以及对应的决策边界。对于线性判别函数来说，其权重向量与决策平面正交。

对于线性分类模型来说，一个关键的问题是如何学习参数 \mathbf{w} ，也就是如何定义损失函数以及优化方法。在本章，我们主要介绍几种不同线性分类模型，包括 Logistic 回归，Softmax 回归，感知器和支持向量机。这些模型的参数学习方式也都不尽相同。

超平面就是三维空间中的平面在更高维空间的推广。在 d 维空间中的超平面是 $d-1$ 维的。

3.1 Logistic 回归

我们先来介绍一个两类分类问题的线性模型：**Logistic 回归**（Logistic Regression）。两类分类问题中，预测的目标标签 y 只有两种取值，一般为 $\{0, 1\}$ 或者 $\{-1, +1\}$ 。为了符合 logistic 回归的描述习惯，在本节中我们采用 $y \in \{0, 1\}$ 。

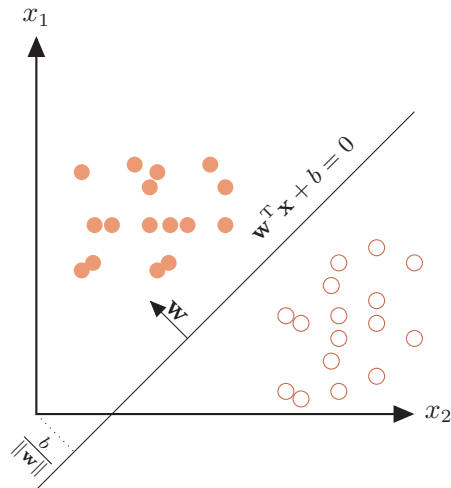


图 3.2: 两类分类线性判别函数。样本特征向量 $\mathbf{x} = [x_1, x_2]$, 权重向量 $\mathbf{w} = [w_1, w_2]$ 。

对于分类问题, 使用线性回归算法来求解是不合适的。一是线性函数的输出值域和目标标签的值域不相同, 二是损失函数很难定义。如果使用平方损失会导致比较大的误差。图3.3a给出了使用线性回归算法来解决一维的两类分类问题示例。

为了解决连续的线性函数不适合进行分类的问题, 我们引入非线性的 logistic 函数作为激活函数, 来预测目标标签 $y = 1$ 的后验概率。

$$p(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x}) \triangleq \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}, \quad (3.6)$$

Logistic 函数相当于把线性函数的值域从实数区间“挤压”到了 $(0, 1)$ 之间, 可以用来表示概率。

其中 $\sigma(\cdot)$ 为 logistic sigmoid 函数, 其值域为 $(0, 1)$ 。

这样, 标签 $y = 0$ 的后验概率为

$$p(y = 0|\mathbf{x}) = 1 - p(y = 1|\mathbf{x}) \quad (3.7)$$

$$= \frac{\exp(-\mathbf{w}^T \mathbf{x})}{1 + \exp(-\mathbf{w}^T \mathbf{x})}. \quad (3.8)$$

将公式 (3.6) 进行变换后得到

$$\mathbf{w}^T \mathbf{x} = \log \frac{P(y = 1|\mathbf{x})}{1 - P(y = 1|\mathbf{x})} \quad (3.9)$$

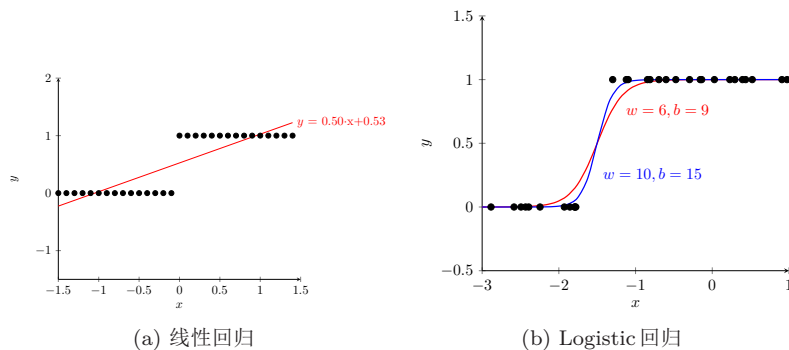


图 3.3: 一维两类问题示例

$$= \log \frac{P(y=1|\mathbf{x})}{P(y=0|\mathbf{x})}, \quad (3.10)$$

其中 $\frac{P(y=1|\mathbf{x})}{P(y=0|\mathbf{x})}$ 为样本 \mathbf{x} 为正反例后验概率的比值，称为**几率**（odds），几率的对数称为**对数几率**（log odds，或 logit）。而公式 (3.9) 的左边是线性函数，logistic 回归可以看作是预测值为“标签的对数几率”的线性回归模型。因此，logistic 回归也称为**对数几率回归**（Logit Regression）。

3.1.1 参数学习

Logistic 回归采用交叉熵作为损失函数，并使用使用梯度下降法来对参数进行优化。

给定 N 个训练样本 $\{(\mathbf{x}^{(i)}, y^{(i)})\}, 1 \leq i \leq N$ ，用 Logistic 回归模型对每个样本 $\mathbf{x}^{(i)}$ 进行预测，输出 $\mathbf{x}^{(i)}$ 的标签为 1 的后验概率，记为 $\hat{y}^{(i)}$ ，

$$\hat{y}^{(i)} = \sigma(\mathbf{w}^T \mathbf{x}^{(i)}), 1 \leq i \leq N. \quad (3.11)$$

使用交叉熵损失函数，其风险函数为：

$$\mathcal{R}(\mathbf{w}) = -\frac{1}{N} \sum_{i=1}^N \left(p(y^{(i)} = 1) \log \hat{y}^{(i)} + p(y^{(i)} = 0) \log(1 - \hat{y}^{(i)}) \right) \quad (3.12)$$

$$= -\frac{1}{N} \sum_{i=1}^N \left(y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right). \quad (3.13)$$

简单起见，这里忽略了正则化项。

因为 $y^{(i)} \in \{0, 1\}$ ，所以
 $p(y^{(i)} = 1) = y^{(i)}$ ，
 $p(y^{(i)} = 0) = 1 - y^{(i)}$ 。

风险函数 $\mathcal{R}(\mathbf{w})$ 关于参数 \mathbf{w} 的梯度为:

$$\frac{\partial \mathcal{R}(\mathbf{w})}{\partial \mathbf{w}} = -\frac{1}{N} \sum_{i=1}^N \left(y^{(i)} \frac{\hat{y}^{(i)}(1-\hat{y}^{(i)})}{\hat{y}^{(i)}} \mathbf{x}^{(i)} - (1-y^{(i)}) \frac{\hat{y}^{(i)}(1-\hat{y}^{(i)})}{1-\hat{y}^{(i)}} \mathbf{x}^{(i)} \right) \quad (3.14)$$

$$= -\frac{1}{N} \sum_{i=1}^N \left(y^{(i)}(1-\hat{y}^{(i)}) \mathbf{x}^{(i)} - (1-y^{(i)}) \hat{y}^{(i)} \mathbf{x}^{(i)} \right) \quad (3.15)$$

$$= -\frac{1}{N} \sum_{i=1}^N \mathbf{x}^{(i)} (y^{(i)} - \hat{y}^{(i)}). \quad (3.16)$$

因此, Logistic 回归的训练过程为: 初始化 $\mathbf{w}_0 \leftarrow \mathbf{0}$, 然后用梯度下降法进行更新

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha \frac{1}{N} \sum_{i=1}^N \mathbf{x}^{(i)} (y^{(i)} - \hat{y}_{\mathbf{w}_t}^{(i)}). \quad (3.17)$$

其中 α 是学习率, $\hat{y}_{\mathbf{w}_t}^{(i)}$ 是当参数为 \mathbf{w}_t 时, logistic 回归模型的输出。

从公式(3.13)可知, 风险函数 $\mathcal{R}(\mathbf{w})$ 是关于参数 \mathbf{w} 的连续可导的凸函数。因此除了梯度下降法之外, logistic 回归还可以用高阶的优化方法, 比如牛顿法, 来进行优化。

3.1.2 扩展到多类分类

多类分类 (Multi-class Classification) 问题是指分类的类别数 K 大于 2。由于 logistic 回归是一个两类分类模型, 需要通过一些策略来推广到多类分类问题, 比如 softmax 回归。除此之外, 还可以使用一些通用的策略, 将多类分类问题转换为多个两类分类问题, 然后基于两类分类的结果, 通过投票方法来进一步确定多类的分类结果。

假设一个多类分类问题的类别为 $\{1, 2, \dots, C\}$, 共 C 个类别, 将多类问题转换为两类问题的方式一般有以下两种:

1. “一对其余”方式 (One VS. Rest) 把多类分类问题转换为 C 个“一对其余”的两类分类问题, 其中第 i 个分类器 f_i 是将类 i 的样本作为正例, 将其它类的样本作为负例训练出来的两类分类器。

softmax 回归参见第3.2节, 第47页。

$i < j \leq K$

2. “一对一”方式 (One VS. One) 把多类分类问题转换为 $C(C-1)/2$ 个“一对一”的两类分类问题, 其中第 (i, j) 个分类器是把类 i 和类 j 的样本分别作为正负例来训练的分类器。

这样, 当对一个样本进行分类时, 首先用多个两类分类器进行分类, 然后进行投票, 选择一个得分最高的类别。

但是上面两种转换方法都存在一个缺陷: 空间中的存在一些区域, 这些区域中点的类别是不能区分确定的。在“一对其余”方式中, 如果所有的分类器都将一个样本分为负例, 或者有两个以上的分类器将一个样本分为正例。在“一对一”方式中, 有两个以上类别的得分是一样的。

3.2 Softmax 回归

Softmax 回归 (Softmax Regression), 也称为多项 (Multinomial) 或多类 (Multi-class) 的 Logistic 回归, 是 logistic 回归在多类分类问题上的推广。

对于多类问题, 类别标签 $y \in \{1, 2, \dots, C\}$ 可以有 C 个取值。给定一个样本 \mathbf{x} , 其属于类别 c 后验概率为:

$$p(y = c|\mathbf{x}) = \text{softmax}(\mathbf{w}_c^T \mathbf{x}) \quad (3.18)$$

$$= \frac{\exp(\mathbf{w}_c^T \mathbf{x})}{\sum_{i=1}^C \exp(\mathbf{w}_i^T \mathbf{x})}, \quad (3.19)$$

其中 \mathbf{w}_c 是第 c 类的权重向量。

softmax 函数参见第 A.5.5 节, 第 256 页。

在测试时, 对于样本 \mathbf{x} , 如果存在类别 c , 对于所有的其他类别 $\tilde{c} (\tilde{c} \neq c)$ 都满足 $p(y = c|\mathbf{x}) > p(y = \tilde{c}|\mathbf{x})$, 那么 \mathbf{x} 属于类别 c 。softmax 的决策函数可以表示为:

$$\hat{y} = \arg \max_{c=1}^C p(y = c|\mathbf{x}) \quad (3.20)$$

$$= \arg \max_{c=1}^C \mathbf{w}_c^T \mathbf{x}. \quad (3.21)$$

与 logistic 回归的关系 当类别数 $C = 2$ 时, softmax 回归的决策函数为

$$\hat{y} = \arg \max_{y \in \{0,1\}} \mathbf{w}_y^T \mathbf{x} \quad (3.22)$$

$$= I(\mathbf{w}_1^T \mathbf{x} - \mathbf{w}_0^T \mathbf{x} > 0) \quad (3.23)$$

$$= I((\mathbf{w}_1 - \mathbf{w}_0)^T \mathbf{x} > 0) \quad (3.24)$$

这里, $I()$ 是指示函数。对比公式 (3.5) 中的两类分类决策函数, 可以发现两类分类中的权重向量 $\mathbf{w} = \mathbf{w}_1 - \mathbf{w}_0$ 。

向量表示 公式 (3.19) 用向量形式可以写为:

$$\hat{\mathbf{y}} = \text{softmax}(W^T \mathbf{x}) \quad (3.25)$$

$$= \frac{\exp(W^T \mathbf{x})}{\mathbf{1}^T \exp(W^T \mathbf{x})}, \quad (3.26)$$

其中 $W = [\mathbf{w}_1, \dots, \mathbf{w}_C]$ 是由 C 个类的权重向量组成的矩阵; $\hat{\mathbf{y}} \in \mathbb{R}^C$ 的第 c 维的值是第 c 类的预测后验概率。

3.2.1 参数学习

给定 N 个训练样本 $\{(\mathbf{x}^{(i)}, y^{(i)})\}, 1 \leq i \leq N$, softmax 回归使用交叉熵损失函数来学习最优的参数矩阵 W 。

为了方便起见, 我们用 C 维的 one-hot 向量 $\mathbf{y} \in \{0, 1\}^C$ 来表示类别标签。对于类别 c , 其向量表示为

$$\mathbf{y} = [I(1 = c), I(2 = c), \dots, I(C = c)]^T. \quad (3.27)$$

采样交叉熵损失函数, softmax 回归模型的风险函数为:

$$\mathcal{R}(W) = -\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C \mathbf{y}_c^{(i)} \log \hat{\mathbf{y}}_c^{(i)} + \frac{\lambda}{2} \|W\|^2 \quad (3.28)$$

$$= -\frac{1}{N} \sum_{i=1}^N (\mathbf{y}^{(i)})^T \log \hat{\mathbf{y}}^{(i)} + \frac{\lambda}{2} \|W\|^2, \quad (3.29)$$

其中 $\mathbf{y}^{(i)} = \text{softmax}(W^T \mathbf{x}^{(i)})$ 为样本 $\mathbf{x}^{(i)}$ 在每个类别的后验概率。

采样梯度下降法, 风险函数 $\mathcal{R}(W)$ 关于 W 的梯度为

$$\frac{\partial \mathcal{R}(W)}{\partial W} = -\frac{1}{N} \sum_{i=1}^N \mathbf{x}^{(i)} (\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)})^T + \lambda W. \quad (3.30)$$

计算公式 (3.30) 中的梯度，关键在于计算每个样本的损失函数 $\mathcal{L}^{(i)}(W) = -(\mathbf{y}^{(i)})^\top \log \hat{\mathbf{y}}^{(i)}$ 关于参数 W 的梯度，其中需要用到的两个导数公式为：

1. 若 $\mathbf{y} = \text{softmax}(\mathbf{z})$ ，则 $\frac{\partial \mathbf{y}}{\partial \mathbf{z}} = \mathbf{diag}(\mathbf{y}) - \mathbf{y}\mathbf{y}^\top$ 。
2. 若 $\mathbf{z} = W^\top \mathbf{x} = [\mathbf{w}_1^\top \mathbf{x}, \mathbf{w}_2^\top \mathbf{x}, \dots, \mathbf{w}_C^\top \mathbf{x}]^\top$ ，则 $\frac{\partial \mathbf{z}}{\partial \mathbf{w}_c}$ 为第 c 列为 \mathbf{x} ，其余为 0 的矩阵。

$$\frac{\partial \mathbf{z}}{\partial \mathbf{w}_c} = \left[\frac{\partial \mathbf{w}_1^\top \mathbf{x}}{\partial \mathbf{w}_c}, \frac{\partial \mathbf{w}_2^\top \mathbf{x}}{\partial \mathbf{w}_c}, \dots, \frac{\partial \mathbf{w}_C^\top \mathbf{x}}{\partial \mathbf{w}_c} \right] \quad (3.31)$$

$$= [0, 0, \dots, \mathbf{x}, \dots, 0] \quad (3.32)$$

$$\triangleq M(\mathbf{x}, c). \quad (3.33)$$

根据链式法则， $\mathcal{L}^{(i)}(W) = -(\mathbf{y}^{(i)})^\top \log \hat{\mathbf{y}}^{(i)}$ 关于 \mathbf{w}_c 的梯度为

$$\frac{\partial \mathcal{L}^{(i)}(W)}{\partial \mathbf{w}_c} = - \frac{\partial ((\mathbf{y}^{(i)})^\top \log \hat{\mathbf{y}}^{(i)})}{\partial \mathbf{w}_c} \quad (3.34)$$

$$= - \frac{\partial \mathbf{z}^{(i)}}{\partial \mathbf{w}_c} \frac{\partial \hat{\mathbf{y}}^{(i)}}{\partial \mathbf{z}^{(i)}} \frac{\partial \log \hat{\mathbf{y}}^{(i)}}{\partial \hat{\mathbf{y}}^{(i)}} \mathbf{y}^{(i)} \quad (3.35)$$

$$= -M(\mathbf{x}^{(i)}, c) \left(\mathbf{diag}(\hat{\mathbf{y}}^{(i)}) - \hat{\mathbf{y}}^{(i)}(\hat{\mathbf{y}}^{(i)})^\top \right) \left(\mathbf{diag}(\hat{\mathbf{y}}^{(i)}) \right)^{-1} \mathbf{y}^{(i)} \quad (3.36)$$

$\mathbf{y}^\top \mathbf{diag}(\mathbf{y})^{-1} = \mathbf{1}_C^\top$
为全 1 的横向量。

$$= -M(\mathbf{x}^{(i)}, c) \left(I - \hat{\mathbf{y}}^{(i)} \mathbf{1}_C^\top \right) \mathbf{y}^{(i)} \quad (3.37)$$

$$= -M(\mathbf{x}^{(i)}, c) \left((\mathbf{y}^{(i)}) - \hat{\mathbf{y}}^{(i)} \mathbf{1}_C^\top \mathbf{y}^{(i)} \right) \quad (3.38)$$

因为 \mathbf{y} 为 onehot 向量，
所以 $\mathbf{1}_C^\top \mathbf{y} = 1$ 。

$$= -M(\mathbf{x}^{(i)}, c) \left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)} \right) \quad (3.39)$$

$$= -\mathbf{x}^{(i)} \left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)} \right)_c. \quad (3.40)$$

如果采用 $y \in \{1, \dots, C\}$ 的离散表示形式，梯度公式 (3.40) 也可以写为：

$$\frac{\partial \mathcal{L}^{(i)}(W)}{\partial \mathbf{w}_c} = -\mathbf{x}^{(i)} \left(I(y^{(i)} = c) - p(y^{(i)} = c | \mathbf{x}^{(i)}; W) \right). \quad (3.41)$$

$I(\cdot)$ 是指示函数。

根据公式 (3.40) 可以得到

$$\frac{\partial \mathcal{L}^{(i)}(W)}{\partial W} = -\mathbf{x}^{(i)} \left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)} \right)^\top. \quad (3.42)$$

因此，softmax 回归的训练过程为：初始化 $W_0 = 0$ ，然后用梯度下降法进行更新

$$W_{t+1} \leftarrow W_t + \alpha \left(\frac{1}{N} \sum_{i=1}^N \mathbf{x}^{(i)} \left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}_{W_t}^{(i)} \right)^\top - \lambda W_t \right), \quad (3.43)$$

其中 α 是学习率， λ 为正则化项系数， $\hat{y}_{W_t}^{(i)}$ 是当参数为 W_t 时，softmax回归模型的输出。

3.3 感知器

Frank Rosenblatt, 1928年-1971年，美国心理学家，人工智能领域开拓者。

最早发明的感知器是一台机器而不是一种算法，后来才被实现为IBM 704机器上可运行的程序。

感知器（Perceptron）由Roseblatt于1957年提出，是一种广泛使用的线性分类器。感知器可谓是最简单的人工神经网络，只有一个神经元。

感知器是对生物神经元的简单数学模拟，有与生物神经元相对应的部件，如权重（突触）、偏置（阈值）及激活函数（细胞体），输出为0或1。

感知器是一种简单的两类线性分类模型，其分类准则与公式(3.5)相同。

$$\hat{y} = \begin{cases} +1 & \text{当 } \mathbf{w}^T \mathbf{x} > 0 \\ -1 & \text{当 } \mathbf{w}^T \mathbf{x} < 0 \end{cases} \quad (3.44)$$

3.3.1 参数学习

感知器的参数学习也是线性分类器的一个经典学习算法。

给定 N 个样本的训练集： $\{(\mathbf{x}^{(i)}, y^{(i)})\}, 1 \leq i \leq N$ ，其中 $y^{(i)} \in \{+1, -1\}$ ，感知器试图学习到参数 \mathbf{w}^* ，使得对于每个样本 $(\mathbf{x}^{(i)}, y^{(i)})$ 有

$$\begin{aligned} \mathbf{w}^{*T} \mathbf{x}^{(i)} &> 0 & \text{当 } y^{(i)} = 1, \\ \mathbf{w}^{*T} \mathbf{x}^{(i)} &< 0 & \text{当 } y^{(i)} = -1. \end{aligned} \quad (3.45)$$

即

$$y^{(i)} \mathbf{w}^{*T} \mathbf{x}^{(i)} > 0. \quad (3.46)$$

感知器的学习算法是一种错误驱动的在线学习算法[Rosenblatt, 1958]。先初始化一个权重向量 \mathbf{w}_0 （通常是全零向量），然后每次分错一个样本时，就用这个样本来更新权重。具体的感知器参数学习策略如算法3.1所示。

根据感知器的学习策略，可以反推出感知器的损失函数为：

$$\mathcal{L}(\mathbf{w}; \mathbf{x}, y) = \max(0, -y\mathbf{w}^T \mathbf{x}). \quad (3.47)$$

算法 3.1: 两类感知器算法

```

输入: 训练集:  $(\mathbf{x}^{(i)}, y^{(i)}), i = 1, \dots, N$ 
      迭代次数:  $T$ 

1 初始化:  $\mathbf{w}_0 \leftarrow 0$ ;
2  $k \leftarrow 0$ ;
3 for  $t = 1 \dots T$  do
4   for  $i = 1 \dots N$  do
5     选取一个样本  $(\mathbf{x}^{(i)}, y^{(i)})$ ;
6     if  $\mathbf{w}_k^T (y^{(i)} \mathbf{x}^{(i)}) \leq 0$  then
7        $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k + y^{(i)} \mathbf{x}^{(i)}$ ;
8        $k \leftarrow k + 1$ ;
9     end
10  end
11 end

输出:  $\mathbf{w}_k$ 

```

如果采用随机梯度下降，其每次更新的梯度为：

$$\frac{\partial \mathcal{L}(\mathbf{w}; \mathbf{x}, y)}{\partial \mathbf{w}} = \begin{cases} 0 & \text{当 } y\mathbf{w}^T \mathbf{x} > 0 \\ -y\mathbf{x} & \text{当 } y\mathbf{w}^T \mathbf{x} < 0 \end{cases} \quad (3.48)$$

图3.4给出了感知器参数学习的更新过程。

3.3.2 收敛性证明

Novikoff [1963] 证明对于两类问题，如果训练集是线性可分的，那么感知器算法可以在有限次迭代后收敛。然而，如果训练集不是线性分隔的，那么这个算法则不能确保会收敛。

定义 3.1 – 两类线性可分： 对于训练集 $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^N$ (其中 $y^{(i)} \in \{-1, 1\}$)，如果存在一个正的常数 $\gamma (\gamma > 0)$ 和权重向量 \mathbf{w}^* ，

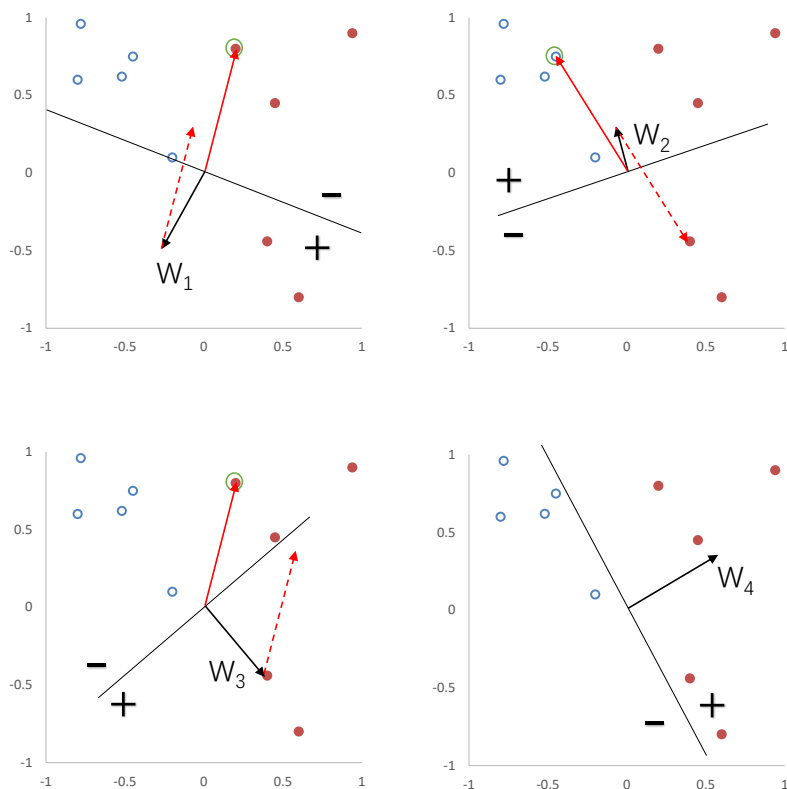


图 3.4: 感知器参数学习的更新过程。红色实心点为正例，蓝色空心点为负例。红色虚线表示权重的更新方向。

并且 $\|\mathbf{w}^*\| = 1$ ，对所有 i 都满足 $(\mathbf{w}^*)^\top (y^{(i)} \mathbf{x}^{(i)}) \geq \gamma$ ($\mathbf{x}^{(i)} \in \mathbb{R}^d$ 为样本 $x^{(i)}$ 的增广特征向量)，那么训练集 \mathcal{D} 是线性可分的。

在数据集是两类线性可分的条件下，我们可以证明如下定理。

定理 3.1 – 感知器收敛性： 给定一个训练集 $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^N$ ，假设 R 是训练集中最大的特征向量的模，

$$R = \max_i \|\mathbf{x}^{(i)}\|.$$

如果训练集 \mathcal{D} 线性可分，感知器学习算法3.1的权重更新次数不超过 $\frac{R^2}{\gamma^2}$ 。

证明. 感知器的权重向量的更新方式为

$$\mathbf{w}_k = \mathbf{w}_{k-1} + y^{(k)} \mathbf{x}^{(k)}, \quad (3.49)$$

其中 $\mathbf{x}^{(k)}, y^{(k)}$ 表示第 k 个错误分类的样本。

因为初始权重向量为0，在第 K 次更新时感知器的权重向量为

$$\mathbf{w}_K = \sum_{k=1}^K y^{(k)} \mathbf{x}^{(k)}. \quad (3.50)$$

分别计算 $\|\mathbf{w}_K\|^2$ 的上下界。

(1) $\|\mathbf{w}_K\|^2$ 的上界为:

$$\|\mathbf{w}_K\|^2 = \|\mathbf{w}_{K-1} + y^{(K)} \mathbf{x}^{(K)}\|^2 \quad (3.51)$$

$$= \|\mathbf{w}_{K-1}\|^2 + \|y^{(K)} \mathbf{x}^{(K)}\|^2 + 2y^{(K)} \mathbf{w}_{K-1}^T \mathbf{x}^{(K)} \quad (3.52) \quad y_k \mathbf{w}_{K-1}^T \mathbf{x}^{(K)} \leq 0.$$

$$\leq \|\mathbf{w}_{K-1}\|^2 + R^2 \quad (3.53)$$

$$\leq \|\mathbf{w}_{K-2}\|^2 + 2R^2 \quad (3.54)$$

$$\leq KR^2 \quad (3.55)$$

(2) $\|\mathbf{w}_K\|^2$ 的下界为:

$$\|\mathbf{w}_K\|^2 = \|\mathbf{w}^*\|^2 \cdot \|\mathbf{w}_K\|^2 \quad (3.56) \quad \|\mathbf{w}^*\| = 1.$$

$$\geq \|\mathbf{w}^{*T} \mathbf{w}_K\|^2 \quad (3.57) \quad \text{两个向量内积的平方一定小于等于这两个向量的模的乘积}$$

$$= \|\mathbf{w}^{*T} \sum_{k=1}^K (y^{(k)} \mathbf{x}^{(k)})\|^2 \quad (3.58)$$

$$= \left\| \sum_{k=1}^K \mathbf{w}^{*T} (y^{(k)} \mathbf{x}^{(k)}) \right\|^2 \quad (3.59) \quad \mathbf{w}^{*T} (y^{(i)} \mathbf{x}^{(i)}) \geq \gamma, \forall i.$$

$$\geq K^2 \gamma^2. \quad (3.60)$$

由公式 (3.55) 和 (3.60)，得到

$$K^2 \gamma^2 \leq \|\mathbf{w}_K\|^2 \leq KR^2. \quad (3.61)$$

取最左和最右的两项，进一步得到， $K^2\gamma^2 \leq KR^2$ 。然后两边都除 K ，最终得到

$$K \leq \frac{R^2}{\gamma^2}. \quad (3.62)$$

因此，在线性可分的条件下，算法3.1会在 $\frac{R^2}{\gamma^2}$ 步内收敛。□

虽然感知器在线性可分的数据上可以保证收敛，找到一个超平面把两类数据分开，但并不能保证能其泛化能力。每次迭代的顺序不一致时，找到的分割超平面也往往不一致。如果训练集不是线性可分的，就永远不会收敛 [Freund and Schapire, 1999]。

3.3.3 参数平均感知器

待整理

从定理3.1可以看出，如果训练数据是线性可分的，那么感知器可以找到一个判别函数来分割不同类的数据。并且如果边际距离越大，收敛越快。但是，感知器并不能保证找到的判别函数是最优的（比如泛化能力高），这样可能导致过拟合。

此外，从感知器的迭代算法可以看出：在迭代次序上排在后面的错误点，比前面的错误点对最终的权重向量影响更大。比如有 1,000 个训练样本，在迭代 100 个样本后，感知器已经学习到一个很好的权重向量。在接下来的 899 个样本上都预测正确，也没有更新权重向量。但是在最后第 1,000 个样本时预测错误，并更新了权重。这次更新可能反而使得权重向量变差了。

为了这种情况，可以使用“参数平均”的策略来提高感知器的鲁棒性，也叫**投票感知器**（Voted Perceptron）[Freund and Schapire, 1999]。

投票感知器记录第 k 次更新后得到的权重 \mathbf{w}_k 在其后分类中正确分类样本的次数 c_k 。这样最后的分类器形式为（假设输出为 0 或 1）：

$$\hat{y} = \mathbf{sign}\left(\sum_{k=1}^K c_k \mathbf{sign}(\mathbf{w}_k^T x)\right). \quad (3.63)$$

投票感知器虽然提高了感知器的泛化能力，但是需要保存 K 个权重向量。在实际操作中会带来额外的开销。因此，人们经常会使用一个简化的版本，也叫做**平均感知器**（Averaged Perceptron）[Collins, 2002]。

$$\hat{y} = \mathbf{sign}\left(\sum_{k=1}^K c_k (\mathbf{w}_k^T x)\right)$$

$$\begin{aligned}
&= \mathbf{sign}\left(\left(\sum_{k=1}^K c_k \mathbf{w}_k\right)^T x\right) \\
&= \mathbf{sign}(\bar{\mathbf{w}}^T x),
\end{aligned} \tag{3.64}$$

其中， $\bar{\mathbf{w}}$ 为平均的权重向量。

假设 $\mathbf{w}^{t,i}$ 是在第 t 轮更新到第 i 个样本时权重向量的值，平均的权重向量 $\bar{\mathbf{w}}$ 也可以写为

$$\bar{\mathbf{w}} = \frac{\sum_{t=1}^T \sum_{i=1}^n \mathbf{w}^{t,i}}{nT} \tag{3.65}$$

这个方法非常简单，只需要在算法3.3中增加一个 $\bar{\mathbf{w}}$ ，并且在处理每一个样本后，更新

$$\bar{\mathbf{w}} = \bar{\mathbf{w}} + \mathbf{w}^{t,i} \tag{3.66}$$

这里要注意的是， $\bar{\mathbf{w}}$ 需要在处理每一个样本时都需要更新，并且 $\bar{\mathbf{w}}$ 和 $\mathbf{w}^{t,i}$ 都是稠密向量。因此，这个操作比较费时。为了提高迭代速度，有很多改进的方法，让这个更新只需要在错误预测发生时才进行更新。算法3.2给出了一个改进的平均感知器算法的训练过程 [Daumé III]。

3.3.4 扩展到多类分类

普通的感知器是一种两类分类模型，但也可以很容易地扩展到多类分类问题，甚至是更一般的结构化学习问题 [Collins, 2002]。

之前介绍的分类模型中，分类函数都是在输入 \mathbf{x} 的特征空间上。为了使得感知器可以处理更复杂的输出，我们引入一个构建输入输出联合空间上的特征函数 $\phi(\mathbf{x}, \mathbf{y})$ ，将样本 (\mathbf{x}, \mathbf{y}) 对映射到一个特征向量空间。

在联合特征空间中，我们可以建立一个广义的感知器模型，

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y} \in \text{Gen}(\mathbf{x})} \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}), \tag{3.67}$$

其中 \mathbf{w} 为权重向量， $\text{Gen}(\mathbf{x})$ 表示输入 \mathbf{x} 所有的输出目标集合。当处理 C 类分类问题时， $\text{Gen}(\mathbf{x}) = \{1, \dots, C\}$ 。

在 C 类分类中，一种常用的特征函数 $\phi(\mathbf{x}, \mathbf{y})$ 是 \mathbf{y} 和 \mathbf{x} 的外积，其中 \mathbf{y} 为类别的 one-hot 向量表示。

$$\phi(\mathbf{x}, \mathbf{y}) = \mathbf{vec}(\mathbf{y}\mathbf{x}^T), \tag{3.68}$$

算法 3.2: 平均感知器算法

输入: 训练集: $(\mathbf{x}^{(i)}, y^{(i)}), i = 1, \dots, N$, 最大迭代次数: T

```

1  $\mathbf{w} = 0$  ;
2  $\mathbf{u} = 0$  ;
3  $c = 0$  ;
4 for  $t = 1 \dots T$  do
5   for  $i = 1 \dots N$  do
6     选取一个样本  $(\mathbf{x}^{(i)}, y^{(i)})$ ;
7     用公式 (3.67) 计算预测类别  $\hat{y}_t$ ;
8     if  $\hat{y}_t \neq y_t$  then
9        $\mathbf{w} = \mathbf{w} + y^{(i)} \mathbf{x}^{(i)}$ ;
10       $\mathbf{u} = \mathbf{u} + c y^{(i)} \mathbf{x}^{(i)}$ ;
11    end
12     $c = c + 1$  ;
13  end
14 end
15  $\bar{\mathbf{w}} = \mathbf{w}_T - \frac{1}{c} \mathbf{u}$  ;
输出:  $\bar{\mathbf{w}}$ 

```

其中 vec 是向量化算子。

给定样本 (\mathbf{x}, \mathbf{y}) , 若 $\mathbf{x} \in \mathbb{R}^d$, \mathbf{y} 为第 c 维为 1 的 one-hot 向量, 则

$$\phi(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \vdots \\ 0 \\ x_1 \\ \vdots \\ x_d \\ 0 \\ \vdots \end{bmatrix} \begin{array}{l} \leftarrow \text{第 } c * d + 1 \text{ 行} \\ \in \mathbb{R}^{(d \times C)} \\ \leftarrow \text{第 } c * d + d \text{ 行} \end{array} \quad (3.69)$$

广义感知器算法的训练过程如算法3.3所示。

算法 3.3: 广义感知器参数学习算法

输入: 训练集: $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}), i = 1, \dots, N$, 最大迭代次数: T

```

1  $\mathbf{w}_0 = \mathbf{0}$ ;
2  $k = 0$ ;
3 for  $t = 1 \dots T$  do
4   for  $i = 1 \dots N$  do
5     选取一个样本  $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ ;
6     用公式 (3.67) 计算预测类别  $\hat{\mathbf{y}}^{(i)}$ ;
7     if  $\hat{\mathbf{y}}^{(i)} \neq \mathbf{y}^{(i)}$  then
8        $\mathbf{w}_{k+1} = \mathbf{w}_k + (\phi(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) - \phi(\mathbf{x}^{(i)}, \hat{\mathbf{y}}^{(i)}))$ ;
9        $k = k + 1$ ;
10    end
11  end
12 end
```

输出: \mathbf{w}_k

收敛性证明 广义感知器在满足广义线性可分（定义3.2）条件时，也能够保证在有限步骤内收敛（定理3.2）。Collins [2002] 给出了多类感知器在多类线性可分的收敛性证明，具体推导过程和两类分类器比较类似。

定义 3.2—广义线性可分: 对于训练集 $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^N$ ，如果存在一个正的常数 $\gamma (\gamma > 0)$ 和权重向量 \mathbf{w}^* ，并且 $\|\mathbf{w}^*\| = 1$ ，对所有 i 都满足 $\langle \mathbf{w}^*, \phi(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \rangle - \langle \mathbf{w}^*, \phi(\mathbf{x}^{(i)}, \mathbf{y}) \rangle \geq \gamma, \mathbf{y} \neq \mathbf{y}^{(i)}$ （ $\phi(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \in \mathbb{R}^d$ 为样本 $\mathbf{x}^{(i)}, \mathbf{y}^{(i)}$ 的联合特征向量），那么训练集 \mathcal{D} 是线性可分的。

定理 3.2—广义感知器收敛性: 对于满足广义线性可分的训练集 $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^N$ ，假设 R 是所有样本中真实标签和错误标签在特征空间 $\phi(\mathbf{x}, \mathbf{y})$ 最远的距离。

$$R = \max_i \max_{\mathbf{z} \neq \mathbf{y}^{(i)}} \|\phi(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) - \phi(\mathbf{x}^{(i)}, \mathbf{z})\|.$$

那么广义感知器参数学习算法3.3的权重更新次数不超过 $\frac{R^2}{\gamma^2}$ 。

3.4 支持向量机

支持向量机 (Support Vector Machine, SVM) 是一个经典两类分类算法, 其找到的分割超平面具有更好的鲁棒性, 因此广泛使用在很多任务上, 并表现出了很强优势。

给定一个两类分类器数据集 $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^N, y_i \in \{+1, -1\}$, 如果两类样本是线性可分的, 即存在一个超平面

$$\mathbf{w}^T \mathbf{x} + b = 0 \quad (3.70)$$

将两类样本分开, 那么对于每个样本都有 $y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + b) > 0$ 。

数据集 \mathcal{D} 中每个样本 $\mathbf{x}^{(i)}$ 到分割超平面的距离为:

$$\gamma^{(i)} = \frac{\|\mathbf{w}^T \mathbf{x}^{(i)} + b\|}{\|\mathbf{w}\|} = \frac{y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + b)}{\|\mathbf{w}\|}. \quad (3.71)$$

我们定义整个数据集 \mathcal{D} 中所有样本到分割超平面的最短距离为**间隔** (Margin) γ

$$\gamma = \min_i \gamma^{(i)}. \quad (3.72)$$

如果间隔 γ 越大, 其分割超平面对两个数据集的划分越稳定, 不容易受噪声等因素影响。

支持向量机的目标是寻找一个超平面 (\mathbf{w}^*, b^*) 使得 γ 最大, 即

$$\begin{aligned} \max_{\mathbf{w}, b} \quad & \gamma \\ \text{s.t.} \quad & \frac{y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + b)}{\|\mathbf{w}\|} \geq \gamma, \forall i \end{aligned} \quad (3.73)$$

我们定义 $\|\mathbf{w}\| \cdot \gamma = 1$, 则公式(3.73)等价于

$$\begin{aligned} \max_{\mathbf{w}, b} \quad & \frac{1}{\|\mathbf{w}\|^2} \\ \text{s.t.} \quad & y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + b) \geq 1, \forall i \end{aligned} \quad (3.74)$$

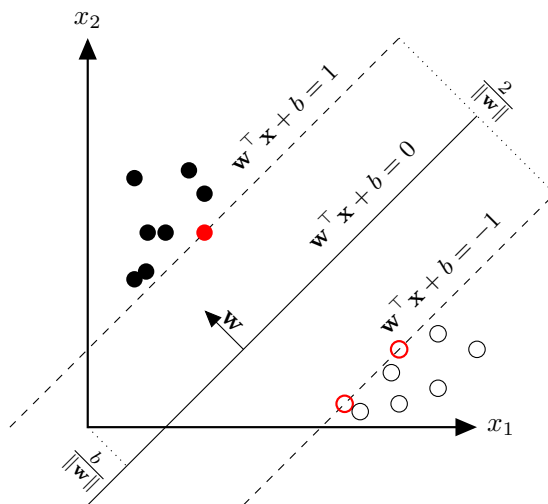


图 3.5: 支持向量机。红色样本点为支持向量。

数据集中所有满足 $y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + b) = 1$ 的样本点, 都称为支持向量 (Support Vector)。

对于一个线性可分的数据集, 其分割超平面有很多个, 但是间隔最大的超平面是唯一的。图3.5给出了支持向量机的最大间隔分割超平面的示例。

3.4.1 参数学习

为了找到最大间隔分割超平面, 将目标函数 (公式 (3.74)) 写为凸优化问题

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{1}{2} \|\mathbf{w}\|^2. \\ \text{s.t.} \quad & 1 - y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + b) \leq 0, \forall i \end{aligned} \quad (3.75)$$

使用拉格朗日乘法, 公式 (3.75) 的拉格朗日函数为

参 见 第 A.2.1 节,
第 238 页。

$$\Lambda(\mathbf{w}, b, \lambda) = \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^N \lambda_i (1 - y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + b)), \quad (3.76)$$

其中 $\lambda_1, \dots, \lambda_N$ 为拉格朗日乘数。计算 $\Lambda(\mathbf{w}, b, \lambda)$ 关于 \mathbf{w} 和 b 的导数, 并令其等

于0得到

$$\mathbf{w} = \sum_{i=1}^N \lambda_i y^{(i)} \mathbf{x}^{(i)}, \quad (3.77)$$

$$0 = \sum_{i=1}^N \lambda_i y^{(i)}. \quad (3.78)$$

将公式(3.77)代入公式(3.76), 并利用公式(3.78)得到

$$\Lambda(\mathbf{w}, b, \lambda) = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j y^{(i)} y^{(j)} (\mathbf{x}^{(i)})^T \mathbf{x}^{(j)} + \sum_{i=1}^N \lambda_i. \quad (3.79)$$

通过最大化其对偶问题 $\max_{\lambda} \Lambda(\mathbf{w}, b, \lambda)$, 得到拉格朗日乘数 λ^* , 根据公式(3.77)和(3.78)计算出 \mathbf{w}^* 和 b^* . 得到支持向量机的决策函数为

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + b) \quad (3.80)$$

$$= \text{sign} \left(\sum_{i=1}^N \lambda_i^* y^{(i)} (\mathbf{x}^{(i)})^T \mathbf{x} + b^* \right). \quad (3.81)$$

从公式(3.81)看出, 支持向量机的决策函数只依赖 $\lambda_i^* > 0$ 的样本点, 这些样本点也就是支持向量, 即离决策平面距离最近的点。

KKT 条件

支持向量机的求解可以通过二次优化方法得到全局最优解, 这使它有着其他统计学习技术难以比拟的优越性。同时, 还使用核函数将原始的样本空间向高维空间进行变换, 能够解决原始样本线性不可分的问题。支持向量机的优点在于通用性较好, 且分类精度高、分类速度快、分类速度与训练样本个数无关, 是最常用的分类器之一。

3.4.2 软间隔

公式3.74的约束条件比较强, 为了能够容忍部分不满足约束的样本, 可以引入松弛变量 ξ :

3.5 总结和深入阅读

和回归问题不同, 分类问题中的目标标签 y 是离散类别标签, 因此分类问题中的决策函数需要输出离散值或是标签的后验概率。线性分类模型一般是一

	激活函数	损失函数	优化方法
线性回归	-	$(y - \mathbf{w}^T \mathbf{x})^2$	最小二乘、梯度下降
Logistic 回归	logistic 函数	$\mathbf{y} \log \sigma(\mathbf{w}^T \mathbf{x})$	梯度下降
Softmax 回归	softmax 函数	$\mathbf{y} \log \text{softmax}(\mathbf{W}^T \mathbf{x})$	梯度下降
感知器	阶跃函数	$\max(0, -y\mathbf{w}^T \mathbf{x})$	随机梯度下降
支持向量机	阶跃函数	$\max(0, 1 - y\mathbf{w}^T \mathbf{x})$	二次规划、SMO 等

表 3.1: 几种不同的线性模型对比。在 Logistic 回归和 softmax 回归中, \mathbf{y} 为类别的 one-hot 向量表示; 在感知器和支持向量机中, y 为 $\{+1, -1\}$ 。

个广义线性函数, 即一个线性判别函数 $f(\mathbf{x}, \mathbf{w})$ 加上一个非线性激活函数 $g(\cdot)$ 。

表3.1给出了几种常见的线性模型比较。

Rosenblatt [1958] 最早提出了两类感知器算法, 并随后给出了感知机收敛定理。但是感知器的输出是离散的以及学习算法比较简单, 不能解决线性不可分问题, 限制了其应用范围。1969年, Marvin Minsky 和他的学生 Seymour Papert 出版了《感知器》[Minsky and Seymour, 1969], 分析了感知器的局限性, 证明感知器不能解决非常简单的异或 (XOR) 问题。从现在的角度看, 《感知器》一书仅仅给出了一个显而易见的证明: 线性模型不能解决非线性问题, 但依然给感知器以及人工智能领域的研究造成了很大的负面影响。虽然书中也认为多层的网络可以解决非线性问题, 但是遗憾的是, 在当时这个问题还不可解。直到1980年以后, Geoffrey Hinton、Yann LeCun 等人用连续输出代替离散输出, 并将反向传播算法 (Backpropagation, BP) [Werbos, 1974] 引入到多层感知器 [Williams and Hinton, 1986], 人工神经网络才又重新引起人们的注意。Minsky and Papert [1987] 也修正之前的看法。另外一方面, 人们对感知器本身的认识也在不断发展。Freund and Schapire [1999] 提出了使用核技巧改进感知器学习算法, 并用投票感知器来提高泛化能力。Collins [2002] 将感知器算法扩展到结构化学习, 给出了相应的收敛性证明, 并且提出一种更加有效并且实用的参数平均化策略。[McDonald et al., 2010] 又扩展了平均感知器算法, 使得感知器可以在分布式计算环境中并行计算, 这样感知器可以用在大规模机器学习问题上。

参考文献

- Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, 2002.
- Hal Daumé III. A course in machine learning. <http://ciml.info/>. [Online].
- Yoav Freund and Robert E Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296, 1999.
- Ryan McDonald, Keith Hall, and Gideon Mann. Distributed training strategies for the structured perceptron. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 456–464. Association for Computational Linguistics, 2010.
- Marvin Minsky and Papert Seymour. *Perceptrons*. 1969.
- Marvin L Minsky and Seymour A Papert. *Perceptrons - Expanded Edition: An Introduction to Computational Geometry*. MIT press Boston, MA:, 1987.
- Albert BJ Novikoff. On convergence proofs for perceptrons. Technical report, DTIC Document, 1963.
- F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386–408, 1958.
- Paul Werbos. Beyond regression: New tools for prediction and analysis in the behavioral sciences. 1974.
- DE Rumelhart GE Hinton RJ Williams and GE Hinton. Learning representations by back-propagating errors. *Nature*, pages 323–533, 1986.

习题 3-1 证明线性分类模型中，权重向量 \mathbf{w} 与决策平面正交。

习题 3-2 在 logistic 回归中，是否可以用 $\hat{y} = \sigma(\mathbf{w}^T \mathbf{x})$ 去逼近正确的标签 y ，并用平方损失 $(y - \hat{y})^2$ 最小化来优化参数 \mathbf{w} ？

习题 3-3 在 softmax 回归的风险函数（公式 (3.29)）中，如果去掉正则化项会有什么影响？

习题 3-4 证明定理 3.2。

习题 3-5 若数据集线性可分，证明支持向量机中将两类样本正确分开的最大间隔分割超平面存在且唯一。