第 3 章　线性模型

正确的判断来自于经验，而经验来自于错误的判断。

— Frederick P. Brooks

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。给定一个 d 维样本 $[x_1, \cdots, x_d]^{\mathrm{T}}$，其线性组合函数为

$$f(\mathbf{x}, \mathbf{w}) = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d + b \tag{3.1}$$

$$= \mathbf{w}^{\mathrm{T}} \mathbf{x} + b, \tag{3.2}$$

简单起见，这里用 $f(\mathbf{x}, \mathbf{w})$ 来表示 $f(\mathbf{x}, \mathbf{w}, b)$。

其中 $\mathbf{w} = [w_1, \cdots, w_d]^{\mathrm{T}}$ 为 d 维的权重向量，b 为偏置。上一章中介绍的线性回归就是典型的线性模型，直接用 $f(\mathbf{x}, \mathbf{w})$ 来预测输出目标 $y = f(\mathbf{x}, \mathbf{w})$。

在分类问题中，由于输出目标 y 是一些离散的标签，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决策函数（decision function）$g(\cdot)$ 来预测输出目标

$$y = g\big(f(\mathbf{x}, \mathbf{w})\big). \tag{3.3}$$

对于两类分类问题，$g(\cdot)$ 可以是符号函数（sign function）

$$g\big(f(\mathbf{x}, \mathbf{w})\big) = \mathrm{sgn}\big(f(\mathbf{x}, \mathbf{w})\big) \tag{3.4}$$

$$\triangleq \begin{cases} +1 & \text{if} \quad f(\mathbf{x}, \mathbf{w}) > 0, \\ -1 & \text{if} \quad f(\mathbf{x}, \mathbf{w}) < 0, \end{cases} \tag{3.5}$$

其中当 $f(\mathbf{x}, \mathbf{w}) = 0$ 时不进行预测。公式 (3.5) 定义了一个典型的两类分类问题的决策函数，其结构如图3.1所示。

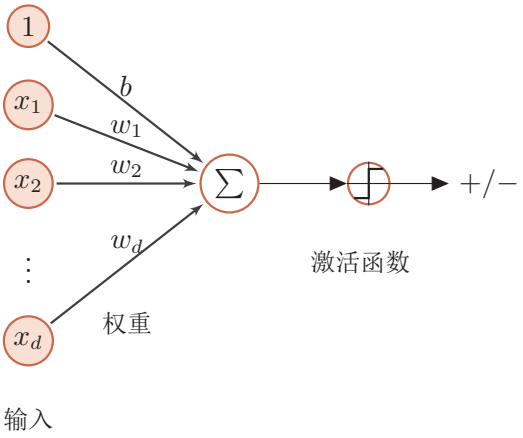


图 3.1 两类分类的线性模型

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

在特征空间中，所有满足 $f(\mathbf{x}, \mathbf{w}) = 0$ 的点组成用一个分割*超平面*（hyperplane），称为*决策边界*（decision boundary）或*决策平面*（decision surface）。决策边界将特征空间一分为二，划分成两个区域，每个区域对应一个类别。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。

图3.2给出了一个两维数据的线性决策边界示例，其中样本特征向量 $\mathbf{x} = [x_1, x_2]$，权重向量 $\mathbf{w} = [w_1, w_2]$。对于线性模型来说，其权重向量 $\mathbf{w}$ 与决策平面正交。

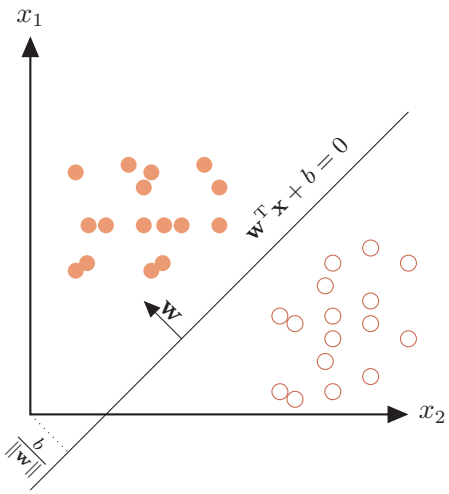


图 3.2 两类分类的决策边界示例

线性分类模型 对于分类问题，函数 $f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b$ 也称为线性判别函数（Linear Discriminant Function），其决策边界为一个超平面。这种决策边界为超平面的模型也称为线性分类模型，或线性分类器（Linear Classifier）。

以两类分类为例，给定 N 个样本的训练集 $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$，其中 $y^{(n)} \in \{+1, -1\}$，线性模型试图学习到参数 $\mathbf{w}^*$ 和 b^*，使得对于每个样本 $(\mathbf{x}^{(n)}, y^{(n)})$ 尽量满足

$$\begin{aligned} \mathbf{w}^{*\mathrm{T}}\mathbf{x}^{(n)} + b^* > 0 \quad & \text{if} \quad y^{(n)} = 1, \\ \mathbf{w}^{*\mathrm{T}}\mathbf{x}^{(n)} + b^* < 0 \quad & \text{if} \quad y^{(n)} = -1. \end{aligned} \tag{3.6}$$

上面两个公式也可以合并，即参数 $\mathbf{w}^*$ 和 b^* 尽量满足

$$y^{(n)}(\mathbf{w}^{*\mathrm{T}}\mathbf{x}^{(n)} + b^*) > 0, \qquad \forall n \in [1, N]. \tag{3.7}$$

为了学习参数 $\mathbf{w}$ 和 b，我们需要定义合适的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。这些模型区别主要在于使用了不同的损失函数。

3.1 Logistic 回归

Logistic 回归（Logistic Regression，LR）是一种最常用的处理两类分类问题的线性模型。两类分类问题中，预测的目标标签 y 只有两种取值，通常可以设为 $\{0, 1\}$ 或者 $\{-1, +1\}$。在本节中我们采用 $y \in \{0, 1\}$ 以符合 logistic 回归的描述习惯。

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，来预测类别标签 $y = 1$ 的后验概率。

$$p(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}) \triangleq \frac{1}{1 + \exp(-\mathbf{w}^{\mathrm{T}}\mathbf{x})}, \tag{3.8}$$

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

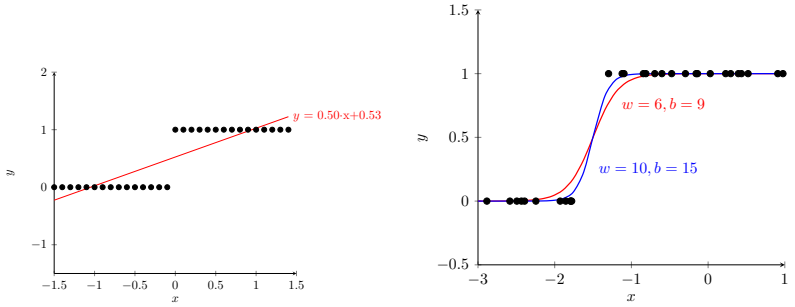
简单起见，这里 $\mathbf{x} = [x_1, \cdots, x_d, 1]^{\mathrm{T}}$ 和 $\mathbf{w} = [w_1, \cdots, w_d, b]^{\mathrm{T}}$ 分别为 $d+1$ 维的增广特征向量和增广权重向量。

标签 $y = 0$ 的后验概率为

$$p(y = 0|\mathbf{x}) = 1 - p(y = 1|\mathbf{x}) \tag{3.9}$$

$$= \frac{\exp(-\mathbf{w}^{\mathrm{T}}\mathbf{x})}{1+\exp(-\mathbf{w}^{\mathrm{T}}\mathbf{x})}. \tag{3.10}$$

图3.3b给出了使用两种不同参数的logistic函数来解决一维的两类分类问题示例。



(a) 线性回归 (b) Logistic 回归

图 3.3 一维数据的两类问题示例

将公式(3.8)进行变换后得到

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} = \log \frac{P(y=1|\mathbf{x})}{1-P(y=1|\mathbf{x})} \tag{3.11}$$

$$= \log \frac{P(y=1|\mathbf{x})}{P(y=0|\mathbf{x})}, \tag{3.12}$$

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）。公式(3.11)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，并使用梯度下降法来对参数进行优化。

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

$$\hat{y}^{(n)} = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(n)}), 1 \leq n \leq N. \tag{3.13}$$

由于 $y^{(n)} \in \{0,1\}$，样本 $(\mathbf{x}^{(n)}, y^{(n)})$ 的真实条件概率可以表示为

$$p_r(y^{(n)}=1|\mathbf{x}^{(n)}) = y^{(n)}, \tag{3.14}$$

$$p_r(y^{(n)}=0|\mathbf{x}^{(n)})=1-y^{(n)}. \tag{3.15}$$

使用交叉熵损失函数，其风险函数为：

简单起见，这里忽略了正则化项。

$$\mathcal{R}(\mathbf{w})=-\frac{1}{N}\sum_{n=1}^{N}\Big(p_r(y^{(n)}=1|\mathbf{x}^{(n)})\log\hat{y}^{(n)}+p_r(y^{(n)}=0|\mathbf{x}^{(n)})\log(1-\hat{y}^{(n)})\Big) \tag{3.16}$$

$$=-\frac{1}{N}\sum_{n=1}^{N}\Big(y^{(n)}\log\hat{y}^{(n)}+(1-y^{(n)})\log(1-\hat{y}^{(n)})\Big). \tag{3.17}$$

风险函数$\mathcal{R}(\mathbf{w})$关于参数$\mathbf{w}$的偏导数为：

$$\frac{\partial\mathcal{R}(\mathbf{w})}{\partial\mathbf{w}}=-\frac{1}{N}\sum_{n=1}^{N}\left(y^{(n)}\frac{\hat{y}^{(n)}(1-\hat{y}^{(n)})}{\hat{y}^{(n)}}\mathbf{x}^{(n)}-(1-y^{(n)})\frac{\hat{y}^{(n)}(1-\hat{y}^{(n)})}{1-\hat{y}^{(n)}}\mathbf{x}^{(n)}\right) \tag{3.18}$$

$$=-\frac{1}{N}\sum_{n=1}^{N}\left(y^{(n)}(1-\hat{y}^{(n)})\mathbf{x}^{(n)}-(1-y^{(n)})\hat{y}^{(n)}\mathbf{x}^{(n)}\right) \tag{3.19}$$

$$=-\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}^{(n)}\big(y^{(n)}-\hat{y}^{(n)}\big). \tag{3.20}$$

采用梯度下降法，logistic回归的训练过程为：初始化$\mathbf{w}_0\leftarrow 0$，然后通过下式来迭代更新参数。

$$\mathbf{w}_{t+1}\leftarrow\mathbf{w}_t+\alpha\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}^{(n)}\left(y^{(n)}-\hat{y}^{(n)}_{\mathbf{w}_t}\right), \tag{3.21}$$

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

从公式(3.17)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C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，比如下一节介绍的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$\{1,2,\cdots,C\}$，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 c 个分类器 f_c 是将类 c 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。
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。

$1 \leq i < j \leq C$

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：在特征空间中，一些区域的类别难以确定的。比如在“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，就很难确定这个样本的类别。而在“一对一”方式中，如果一个样本在两个类别上的得分是一样的，那么这个样本的类别也很难确定。

增加图例。

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。

Softmax 回归也可以看作是一种条件最大熵模型，参见第11.1.5.1节。

对于多类问题，类别标签 $y \in \{1, 2, \cdots, C\}$ 可以有 C 个取值。给定一个样本 $\mathbf{x}$，softmax 回归预测的属于类别 c 的条件概率为：

$$p(y = c|\mathbf{x}) = \text{softmax}(\mathbf{w}_c^{\mathrm{T}}\mathbf{x}) \tag{3.22}$$

$$= \frac{\exp(\mathbf{w}_c^{\top}\mathbf{x})}{\sum_{c=1}^{C}\exp(\mathbf{w}_c^{\top}\mathbf{x})}, \tag{3.23}$$

其中 $\mathbf{w}_c$ 是第 c 类的权重向量。

softmax 函数参见第B.2.5节。

在预测样本 $\mathbf{x}$ 的类别时，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 回归的决策函数可以表示为：

$$\hat{y} = \arg\max_{c=1}^{C} p(y = c|\mathbf{x}) \tag{3.24}$$

$$= \arg\max_{c=1}^{C} \mathbf{w}_c^{\mathrm{T}}\mathbf{x}. \tag{3.25}$$

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

$$\hat{y} = \arg\max_{y\in\{0,1\}} \mathbf{w}_y^{\mathrm{T}}\mathbf{x} \tag{3.26}$$

$$= I\Big(\mathbf{w}_1^{\mathrm{T}}\mathbf{x} - \mathbf{w}_0^{\mathrm{T}}\mathbf{x} > 0\Big) \tag{3.27}$$

$$= I\Big((\mathbf{w}_1 - \mathbf{w}_0)^{\mathrm{T}}\mathbf{x} > 0\Big), \tag{3.28}$$

其中 $I(\cdot)$ 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 (3.23) 用向量形式可以写为：

$$\hat{\mathbf{y}} = \mathrm{softmax}(W^{\mathrm{T}}\mathbf{x}) \tag{3.29}$$

$$= \frac{\exp(W^{\mathrm{T}}\mathbf{x})}{\mathbf{1}^{\mathrm{T}}\exp(W^{\mathrm{T}}\mathbf{x})}, \tag{3.30}$$

其中 $W = [\mathbf{w}_1, \cdots, \mathbf{w}_C]$ 是由 C 个类的权重向量组成的矩阵，$\mathbf{1}$ 为全 1 向量，$\hat{\mathbf{y}} \in \mathbb{R}^C$ 为所有类别的预测条件概率组成的向量，第 c 维的值是第 c 类的预测条件概率。

3.2.1 参数学习

给定 N 个训练样本 $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N$，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

$$\mathbf{y} = [I(1=c), I(2=c), \cdots, I(C=c)]^{\mathrm{T}}, \tag{3.31}$$

其中 $I(\cdot)$ 是指示函数。

采用交叉熵损失函数，softmax 回归模型的风险函数为：

简单起见，这里忽略了正则化项。

$$\mathcal{R}(W) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C}\mathbf{y}_c^{(n)}\log\hat{\mathbf{y}}_c^{(n)} \tag{3.32}$$

$$= -\frac{1}{N}\sum_{n=1}^{N}(\mathbf{y}^{(n)})^{\mathrm{T}}\log\hat{\mathbf{y}}^{(n)}, \tag{3.33}$$

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

风险函数 $\mathcal{R}(W)$ 关于 W 的梯度为

$$\frac{\partial\mathcal{R}(W)}{\partial W} = -\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}^{(n)}\left(\mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)}\right)^{\mathrm{T}}. \tag{3.34}$$

证明. 计算公式 (3.34) 中的梯度，关键在于计算每个样本的损失函数 $\mathcal{L}^{(n)}(W) = -(\mathbf{y}^{(n)})^{\mathrm{T}}\log\hat{\mathbf{y}}^{(n)}$ 关于参数 W 的梯度，其中需要用到的两个导数公式为：

1. 若 $\mathbf{y} = \mathrm{softmax}(\mathbf{z})$，则 $\frac{\partial\mathbf{y}}{\partial\mathbf{z}} = \mathrm{diag}(\mathbf{y}) - \mathbf{y}\mathbf{y}^{\top}$。

softmax 函数的导数参见第 B.2.5 节。

2. 若 $\mathbf{z} = W^{\mathrm{T}}\mathbf{x} = [\mathbf{w}_1^{\mathrm{T}}\mathbf{x}, \mathbf{w}_2^{\mathrm{T}}\mathbf{x}, \cdots, \mathbf{w}_C^{\mathrm{T}}\mathbf{x}]^{\mathrm{T}}$，则 $\frac{\partial \mathbf{z}}{\partial \mathbf{w}_c}$ 为第 c 列为 $\mathbf{x}$，其余为 0 的矩阵。

$$\frac{\partial \mathbf{z}}{\partial \mathbf{w}_c} = [\frac{\partial \mathbf{w}_1^{\mathrm{T}}\mathbf{x}}{\partial \mathbf{w}_c}, \frac{\partial \mathbf{w}_2^{\mathrm{T}}\mathbf{x}}{\partial \mathbf{w}_c}, \cdots, \frac{\partial \mathbf{w}_C^{\mathrm{T}}\mathbf{x}}{\partial \mathbf{w}_c}] \tag{3.35}$$

$$= [\mathbf{0}, \mathbf{0}, \cdots, \mathbf{x}, \cdots, \mathbf{0}] \tag{3.36}$$

$$\triangleq \mathbb{M}_c(\mathbf{x}). \tag{3.37}$$

根据链式法则，$\mathcal{L}^{(n)}(W) = -(\mathbf{y}^{(n)})^{\mathrm{T}} \log \hat{\mathbf{y}}^{(n)}$ 关于 $\mathbf{w}_c$ 的偏导数为

$$\frac{\partial \mathcal{L}^{(n)}(W)}{\partial \mathbf{w}_c} = -\frac{\partial \left((\mathbf{y}^{(n)})^{\mathrm{T}} \log \hat{\mathbf{y}}^{(n)}\right)}{\partial \mathbf{w}_c} \tag{3.38}$$

$$= -\frac{\partial \mathbf{z}^{(n)}}{\partial \mathbf{w}_c} \frac{\partial \hat{\mathbf{y}}^{(n)}}{\partial \mathbf{z}^{(n)}} \frac{\partial \log \hat{\mathbf{y}}^{(n)}}{\partial \hat{\mathbf{y}}^{(n)}} \mathbf{y}^{(n)} \tag{3.39}$$

$\mathbf{y}^{\mathrm{T}} \operatorname{diag}(\mathbf{y})^{-1} = \mathbf{1}^{\mathrm{T}}$ 为全 $\mathbf{1}$ 的行向量。

$$= -\mathbb{M}_c(\mathbf{x}^{(n)}) \left(\operatorname{diag}\left(\hat{\mathbf{y}}^{(n)}\right) - \hat{\mathbf{y}}^{(n)}(\hat{\mathbf{y}}^{(n)})^{\mathrm{T}}\right) \left(\operatorname{diag}(\hat{\mathbf{y}}^{(n)})\right)^{-1} \mathbf{y}^{(n)} \tag{3.40}$$

$$= -\mathbb{M}_c(\mathbf{x}^{(n)}) \left(\mathbf{I} - \hat{\mathbf{y}}^{(n)}\mathbf{1}^{\mathrm{T}}\right) \mathbf{y}^{(n)} \tag{3.41}$$

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

$$= -\mathbb{M}_c(\mathbf{x}^{(n)}) \left(\mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)}\mathbf{1}^{\mathrm{T}}\mathbf{y}^{(n)}\right) \tag{3.42}$$

$$= -\mathbb{M}_c(\mathbf{x}^{(n)}) \left(\mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)}\right) \tag{3.43}$$

$$= -\mathbf{x}^{(n)} \left[\mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)}\right]_c. \tag{3.44}$$

公式 (3.44) 也可以表示为非向量形式，

$$\frac{\partial \mathcal{L}^{(n)}(W)}{\partial \mathbf{w}_c} = -\mathbf{x}^{(n)} \left(I(y^{(n)} = c) - \hat{\mathbf{y}}_c^{(n)}\right), \tag{3.45}$$

其中 $I(\cdot)$ 是指示函数。

根据公式 (3.44) 可以得到

$$\frac{\partial \mathcal{L}^{(n)}(W)}{\partial W} = -\mathbf{x}^{(n)} \left(\mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)}\right)^{\mathrm{T}}. \tag{3.46}$$

□

采用梯度下降法，softmax 回归的训练过程为：初始化 $W_0 \leftarrow 0$，然后通过下式进行迭代更新。

$$W_{t+1} \leftarrow W_t + \alpha \left(\frac{1}{N} \sum_{n=1}^{N} \mathbf{x}^{(n)} \left(\mathbf{y}^{(n)} - \hat{\mathbf{y}}_{W_t}^{(n)}\right)^{\mathrm{T}}\right), \tag{3.47}$$

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

3.3 感知器

感知器（Perceptron）由Frank 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。

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

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+1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。

$$\hat{y} = \text{sgn}(\mathbf{w}^{\text{T}}\mathbf{x}). \tag{3.48}$$

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。

3.3.1 参数学习

感知器学习算法也是一个经典的线性分类器的参数学习算法。

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

$$y^{(n)}\mathbf{w}^{*\text{T}}\mathbf{x}^{(n)} > 0, \qquad \forall n \in [1, N]. \tag{3.49}$$

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$\mathbf{w} \leftarrow 0$（通常是全零向量），然后每次分错一个样本$(\mathbf{x}, y)$时，即$y\mathbf{w}^{\text{T}}\mathbf{x} < 0$，就用这个样本来更新权重。

$$\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}. \tag{3.50}$$

具体的感知器参数学习策略如算法3.1所示。

算法 3.1: 两类感知器算法

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

初始化：$\mathbf{w}_0 \leftarrow 0$, $k \leftarrow 0$;
for $t = 1 \cdots T$ **do**
  随机对训练样本进行随机排序;
  for $n = 1 \cdots N$ **do**
    选取一个样本$(\mathbf{x}^{(n)}, y^{(n)})$;
    if $\mathbf{w}_k^{\text{T}}(y^{(n)}\mathbf{x}^{(n)}) \leq 0$ **then**
      $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k + y^{(n)}\mathbf{x}^{(n)}$;
      $k \leftarrow k + 1$;
    end
  end
end

输出: $\mathbf{w}_k$

根据感知器的学习策略，可以反推出感知器的损失函数为：

$$\mathcal{L}(\mathbf{w}; \mathbf{x}, y) = \max(0, -y\mathbf{w}^{\mathrm{T}}\mathbf{x}). \tag{3.51}$$

采用随机梯度下降，其每次更新的梯度为

$$\frac{\partial \mathcal{L}(\mathbf{w}; \mathbf{x}, y)}{\partial \mathbf{w}} = \begin{cases} 0 & \text{当} \quad y\mathbf{w}^{\mathrm{T}}\mathbf{x} > 0, \\ -y\mathbf{x} & \text{当} \quad y\mathbf{w}^{\mathrm{T}}\mathbf{x} < 0. \end{cases} \tag{3.52}$$

图3.4给出了感知器参数学习的更新过程，其中红色实心点为正例，蓝色空心点为负例。黑色箭头表示权重向量，红色虚线箭头表示权重的更新方向。

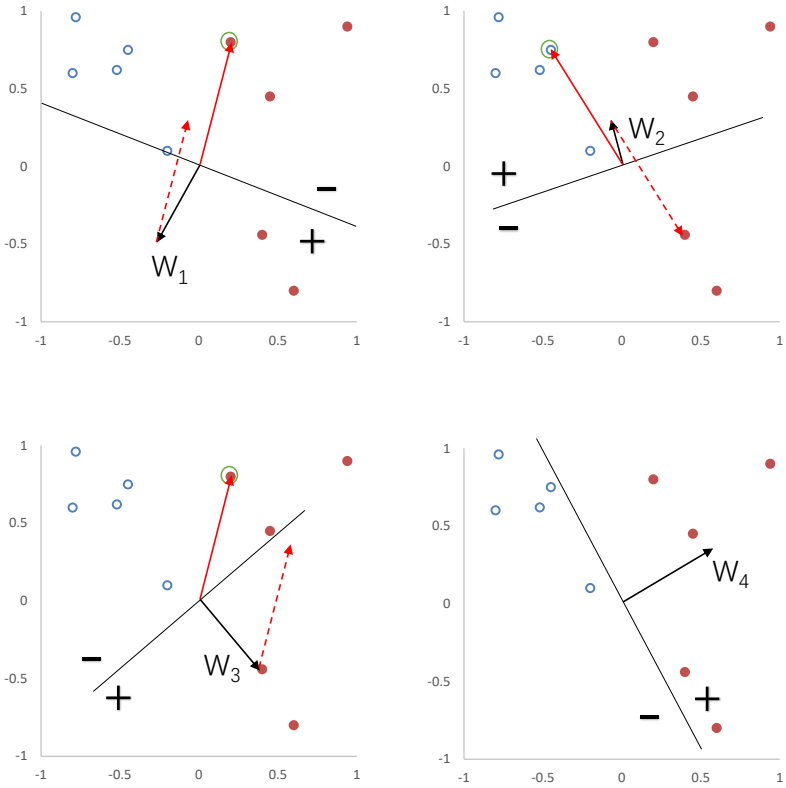


图 3.4 感知器参数学习的更新过程

3.3.2 感知器的收敛性

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 $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$，其中 $\mathbf{x}^{(n)}$ 为样本的增广特征向量，$y^{(n)} \in \{-1, 1\}$，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 n 都满足 $(\mathbf{w}^*)^{\mathrm{T}}(y^{(n)}\mathbf{x}^{(n)}) \geq \gamma$，那么训练集 $\mathcal{D}$ 是线性可分的。

当数据集是两类线性可分时，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 $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$，假设 R 是训练集中最大的特征向量的模，

$$R = \max_n \|x^{(n)}\|.$$

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

$$\mathbf{w}_k = \mathbf{w}_{k-1} + y^{(k)}\mathbf{x}^{(k)}, \tag{3.53}$$

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

$$\mathbf{w}_K = \sum_{k=1}^{K} y^{(k)}\mathbf{x}^{(k)}. \tag{3.54}$$

分别计算 $\|\mathbf{w}_K\|^2$ 的上下界：

（1）$\|\mathbf{w}_K\|^2$ 的上界为：

$$\begin{aligned}
\|\mathbf{w}_K\|^2 &= \|\mathbf{w}_{K-1} + y^{(K)}\mathbf{x}^{(K)}\|^2 && (3.55)\\
&= \|\mathbf{w}_{K-1}\|^2 + \|y^{(K)}\mathbf{x}^{(K)}\|^2 + 2y^{(K)}\mathbf{w}_{K-1}^{\mathrm{T}}\mathbf{x}^{(K)} && (3.56) \quad y_k\mathbf{w}_{K-1}^{\mathrm{T}}\mathbf{x}^{(K)} \leq 0.\\
&\leq \|\mathbf{w}_{K-1}\|^2 + R^2 && (3.57)\\
&\leq \|\mathbf{w}_{K-2}\|^2 + 2R^2 && (3.58)\\
&\leq KR^2 && (3.59)
\end{aligned}$$

（2）$\|\mathbf{w}_K\|^2$ 的下界为：

$\|\mathbf{w}^*\| = 1.$

$$\|\mathbf{w}_K\|^2 = \|\mathbf{w}^*\|^2 \cdot \|\mathbf{w}_K\|^2 \tag{3.60}$$

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

$$\geq \|\mathbf{w}^{*\mathrm{T}}\mathbf{w}_K\|^2 \tag{3.61}$$

$$= \|\mathbf{w}^{*\mathrm{T}} \sum_{k=1}^{K} (y^{(k)}\mathbf{x}^{(k)})\|^2 \tag{3.62}$$

$\mathbf{w}^{*\mathrm{T}}(y^{(n)}\mathbf{x}^{(n)}) \geq \gamma, \forall n.$

$$= \| \sum_{k=1}^{K} \mathbf{w}^{*\mathrm{T}}(y^{(k)}\mathbf{x}^{(k)})\|^2 \tag{3.63}$$

$$\geq K^2\gamma^2. \tag{3.64}$$

由公式 (3.59) 和 (3.64)，得到

$$K^2\gamma^2 \leq ||\mathbf{w}_K||^2 \leq KR^2. \tag{3.65}$$

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

$$K \leq \frac{R^2}{\gamma^2}. \tag{3.66}$$

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，但其存在以下不足之处：

1. 在数据集线性可分时，感知器虽然可以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。
2. 感知器对样本顺序比较敏感。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。
3. 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

根据定理3.1，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。如果间隔 γ 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。

感知器的学习到的权重向量和训练样本的顺序相关。在迭代次序上排在后面的错误样本，比前面的错误样本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。

为了改善这种情况，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]。

投票感知器是一种集成模型，参见第10.1节。

投票感知器记录第 k 次更新后得到的权重 $\mathbf{w}_k$ 在之后的训练过程中正确分类样本的次数 c_k。这样最后的分类器形式为：

$$\hat{y} = \text{sgn}\Big(\sum_{k=1}^{K} c_k \,\text{sgn}(\mathbf{w}_k^{\mathrm{T}}\mathbf{x})\Big) \tag{3.67}$$

其中 $\text{sgn}(\cdot)$ 为符号函数。

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]。

$$\hat{y} = \text{sgn}\Big(\sum_{k=1}^{K} c_k(\mathbf{w}_k^{\mathrm{T}}\mathbf{x})\Big) \tag{3.68}$$

$$= \text{sgn}\Big((\sum_{k=1}^{K} c_k\mathbf{w}_k)^{\mathrm{T}}\mathbf{x}\Big) \tag{3.69}$$

$$= \text{sgn}(\bar{\mathbf{w}}^{\mathrm{T}}\mathbf{x}), \tag{3.70}$$

其中 $\bar{\mathbf{w}}$ 为平均的权重向量。

假设 $\mathbf{w}_{t,n}$ 是在第 t 轮更新到第 n 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

$$\bar{\mathbf{w}} = \frac{\sum_{t=1}^{\mathrm{T}}\sum_{n=1}^{n}\mathbf{w}_{t,n}}{nT} \tag{3.71}$$

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

$$\bar{\mathbf{w}} \leftarrow \bar{\mathbf{w}} + \mathbf{w}_{t,n}. \tag{3.72}$$

但这个方法需要在处理每一个样本时都要更新 $\bar{\mathbf{w}}$。因为 $\bar{\mathbf{w}}$ 和 $\mathbf{w}_{t,n}$ 都是稠密向量，因此更新操作比较费时。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]。

参见习题3-4。

3.3.4 扩展到多类分类

原始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]。

算法 3.2: 平均感知器算法

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

初始化：$\mathbf{w} \leftarrow 0$, $\mathbf{u} \leftarrow 0$, $c \leftarrow 0$;
for $t = 1 \cdots T$ **do**
  随机对训练样本进行随机排序;
  for $n = 1 \cdots N$ **do**
    选取一个样本 $(\mathbf{x}^{(n)}, y^{(n)})$;
    计算预测类别 $\hat{y}_t$;
    if $\hat{y}_t \neq y_t$ **then**
      $\mathbf{w} \leftarrow \mathbf{w} + y^{(n)}\mathbf{x}^{(n)}$;
      $\mathbf{u} \leftarrow \mathbf{u} + cy^{(n)}\mathbf{x}^{(n)}$;
    end
    $c \leftarrow c + 1$;
  end
end
$\bar{\mathbf{w}} = \mathbf{w}_T - \frac{1}{c}\mathbf{u}$;

输出: $\bar{\mathbf{w}}$

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 $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$，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，

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \texttt{Gen}(\mathbf{x})}{\arg\max}\, \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}), \tag{3.73}$$

其中 $\mathbf{w}$ 为权重向量，$\texttt{Gen}(\mathbf{x})$ 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，$\texttt{Gen}(\mathbf{x}) = \{1, \cdots, C\}$。

通过引入特征函数 $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$，感知器不但可以用于多类分类问题，也可以用于结构化学习问题，比如输出是序列形式。

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

$$\boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) = \mathbf{vec}(\mathbf{y}\mathbf{x}^{\mathrm{T}}) \in \mathbb{R}^{(d \times C)}, \tag{3.74}$$

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

$$\boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \vdots \\ 0 \\ x_1 \\ \vdots \\ x_d \\ 0 \\ \vdots \end{bmatrix} \begin{array}{l} \\ \\ \leftarrow \text{第 } (c-1) \times d + 1 \text{ 行} \\ \\ \leftarrow \text{第 } (c-1) \times d + d \text{ 行} \\ \\ \end{array} \tag{3.75}$$

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:$\{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^{N}$，最大迭代次数 T

初始化：$\mathbf{w}_0 \leftarrow 0$, $k \leftarrow 0$;
for $t = 1 \cdots T$ **do**
  随机对训练样本进行随机排序;
  for $n = 1 \cdots N$ **do**
    选取一个样本 $(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})$;
    用公式 (3.73) 计算预测类别 $\hat{\mathbf{y}}^{(n)}$;
    if $\hat{\mathbf{y}}^{(n)} \neq \mathbf{y}^{(n)}$ **then**
      $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k + \big(\boldsymbol{\phi}(\mathbf{x}^{(n)}, \mathbf{y}^{(n)}) - \boldsymbol{\phi}(\mathbf{x}^{(n)}, \hat{\mathbf{y}}^{(n)})\big)$;
      $k = k + 1$;
    end
  end
end

输出: $\mathbf{w}_k$

3.3.4.1 广义感知器的收敛性

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）。

定义 3.2–广义线性可分： 对于训练集 $\mathcal{D} = \big\{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\big\}_{n=1}^{N}$，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 n 都满足 $\langle \mathbf{w}^*, \boldsymbol{\phi}(\mathbf{x}^{(n)}, \mathbf{y}^{(n)}) \rangle - \langle \mathbf{w}^*, \boldsymbol{\phi}(\mathbf{x}^{(n)}, \mathbf{y}) \rangle \geq \gamma, \mathbf{y} \neq \mathbf{y}^{(n)}$

（$\phi(\mathbf{x}^{(n)}, \mathbf{y}^{(n)}) \in \mathbb{R}^d$ 为样本 $\mathbf{x}^{(n)}, \mathbf{y}^{(n)}$ 的联合特征向量），那么训练集 $\mathcal{D}$ 在联合特征向量空间中是线性可分的。

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感知器比较类似。

参见习题3-5。

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 $\mathcal{D} = \left\{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\right\}_{n=1}^{N}$，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。

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

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 $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$，其中 $y_n \in \{+1, -1\}$，如果两类样本是线性可分的，即存在一个超平面

本节中不使用增广的特征向量和特征权重。

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = 0 \tag{3.76}$$

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

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

$$\gamma^{(n)} = \frac{\|\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(n)} + b\|}{\|\mathbf{w}\|} = \frac{y^{(n)}(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(n)} + b)}{\|\mathbf{w}\|}. \tag{3.77}$$

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

$$\gamma = \min_{n} \gamma^{(n)}. \tag{3.78}$$

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

$$\begin{aligned} \max_{\mathbf{w}, b} \quad & \gamma \\ \text{s.t.} \quad & \frac{y^{(n)}(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(n)} + b)}{\|\mathbf{w}\|} \geq \gamma, \forall n \end{aligned} \tag{3.79}$$

令 $\|\mathbf{w}\| \cdot \gamma = 1$，则公式 (3.79) 等价于

$$\begin{aligned} \max_{\mathbf{w},b} \quad & \frac{1}{\|\mathbf{w}\|^2} \\ \text{s.t.} \quad & y^{(n)}(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(n)} + b) \geq 1, \forall n \end{aligned} \tag{3.80}$$

数据集中所有满足 $y^{(n)}(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(n)} + b) = 1$ 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给定了支持向量机的最大间隔分割超平面的示例，其红色样本点为支持向量。

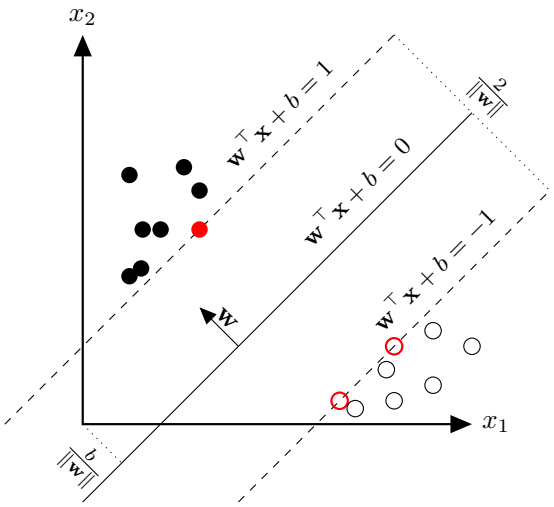


图 3.5 支持向量机示例

3.4.1 参数学习

为了找到最大间隔分割超平面，将公式 (3.80) 的目标函数写为凸优化问题

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \quad & 1 - y^{(n)}(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(n)} + b) \leq 0, \qquad \forall n \end{aligned} \tag{3.81}$$

使用拉格朗日乘数法，公式 (3.81) 的拉格朗日函数为

参见第??节。

$$\Lambda(\mathbf{w}, b, \lambda) = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{n=1}^{N} \lambda_n \Big(1 - y^{(n)}(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(n)} + b)\Big), \tag{3.82}$$

其中 $\lambda_1, \cdots, \lambda_N$ 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

$$\mathbf{w} = \sum_{n=1}^{N} \lambda_n y^{(n)} \mathbf{x}^{(n)}, \tag{3.83}$$

$$0 = \sum_{n=1}^{N} \lambda_n y^{(n)}. \tag{3.84}$$

将公式 (3.83) 代入公式 (3.82)，并利用公式 (3.84)，得到拉格朗日对偶函数

$$\Gamma(\lambda) = -\frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_m \lambda_n y^{(m)} y^{(n)} (\mathbf{x}^{(m)})^{\mathrm{T}} \mathbf{x}^{(n)} + \sum_{n=1}^{N} \lambda_n. \tag{3.85}$$

支持向量机的主优化问题为凸优化问题，满足强对偶性，即主优化问题可以通过最大化对偶函数 $\max_{\lambda \geq 0} \Gamma(\lambda)$ 来求解。对偶函数 $\Gamma(\lambda)$ 是一个凹函数，因此最大化对偶函数是一个凸优化问题，可以通过多种凸优化方法来进行求解，得到拉格朗日乘数的最优值 λ^*。但由于其约束条件的数量为训练样本数量，一般的优化方法代价比较高，因此在实践中通常采用比较高效的优化方法，比如 SMO（Sequential Minimal Optimization）算法 [Platt, 1998] 等。

参见公式 (C.26)。

根据 KKT 条件中的互补松弛条件，最优解满足 $\lambda_n^* \left(1 - y^{(n)} (\mathbf{w}^{*\mathrm{T}} \mathbf{x}^{(n)} + b^*)\right) = 0$。如果样本 $\mathbf{x}^{(n)}$ 不在约束边界上，$\lambda_n^* = 0$，其约束失效；如果样本 $\mathbf{x}^{(n)}$ 在约束边界上，$\lambda_n^* \geq 0$。这些在约束边界上样本点称为**支持向量**（support vector），即离决策平面距离最近的点。

再计算出 λ^* 后，根据公式 (3.83) 计算出最优权重 $\mathbf{w}^*$，最优偏置 b^* 可以通过任选一个支持向量 $(\tilde{\mathbf{x}}, \tilde{y})$ 计算得到。

$$b^* = \tilde{y} - \mathbf{w}^{*\mathrm{T}} \tilde{\mathbf{x}}. \tag{3.86}$$

最优参数的支持向量机的决策函数为

$$f(\mathbf{x}) = \mathrm{sgn}(\mathbf{w}^{*\mathrm{T}} \mathbf{x} + b^*) \tag{3.87}$$

$$= \mathrm{sgn}\left(\sum_{n=1}^{N} \lambda_n^* y^{(n)} (\mathbf{x}^{(n)})^{\mathrm{T}} \mathbf{x} + b^* \right). \tag{3.88}$$

支持向量机的决策函数只依赖 $\lambda_n^* > 0$ 的样本点，即支持向量。

支持向量机的目标函数可以通过 SMO 等优化方法得到全局最优解，因此比其它分类器的学习效率更高。此外，支持向量机的决策函数只依赖与支持向量，与训练样本总数无关，分类速度比较快。

3.4.2 核函数

支持向量机还有一个重要的优点是可以使用*核函数*（kernel function）隐式地将样本从原始特征空间映射到更高维的空间，并解决原始特征空间中的线性不可分问题。比如在一个变换后的特征空间 ϕ 中，支持向量机的决策函数为

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w}^{*\mathrm{T}}\phi(\mathbf{x}) + b^*) \tag{3.89}$$

$$= \text{sgn}\left(\sum_{n=1}^{N} \lambda_n^* y^{(n)} K(\mathbf{x}^{(n)}, \mathbf{x}) + b^*\right), \tag{3.90}$$

其中 $K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^{\mathrm{T}}\phi(\mathbf{z})$ 为核函数。通常我们不需要显式地给出 $\phi(\mathbf{x})$ 的具体形式，可以通过*核技巧*（kernel trick）来构造。比如以 $\mathbf{x}, \mathbf{z} \in \mathbb{R}^2$ 为例，我们可以构造一个核函数

参见习题3-7。

$$K(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}^{\mathrm{T}}\mathbf{z})^2 = \phi(\mathbf{x})^{\mathrm{T}}\phi(\mathbf{z}), \tag{3.91}$$

来隐式地计算 $\mathbf{x}, \mathbf{z}$ 在特征空间 ϕ 中的内积，其中 $\phi(\mathbf{x}) = [1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2]^{\mathrm{T}}$。

3.4.3 软间隔

在支持向量机的优化问题中，约束条件比较严格。如果训练集中的样本在特征空间中不是线性可分的，就无法找到最优解。为了能够容忍部分不满足约束的样本，我们可以引入松弛变量 ξ，将优化问题变为

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n \\ \text{s.t.} \quad & 1 - y^{(n)}(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(n)} + b) - \xi_n \leq 0, \qquad \forall n \\ & \xi_n \geq 0, \qquad \forall n \end{aligned} \tag{3.92}$$

其中参数 $C > 0$ 用来控制间隔和松弛变量惩罚的平衡。引入松弛变量的间隔称为*软间隔*（soft margin）。公式 (3.92) 也可以表示为经验风险 + 正则化项的形式。

$$\min_{\mathbf{w},b} \quad \sum_{n=1}^{N} \max\left(0, 1 - y^{(n)}(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(n)} + b)\right) + \frac{1}{C} \cdot \frac{1}{2}\|\mathbf{w}\|^2, \tag{3.93}$$

其中 $\max\left(0, 1 - y^{(n)}(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(n)} + b)\right)$ 称为 *hinge 损失函数*，$\frac{1}{C}$ 可以看作是正则化系数。

参见公式 (2.21)。

软间隔支持向量机的参数学习和原始支持向量机类似，其最终决策函数也只和支持向量有关，即满足 $1 - y^{(n)}(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(n)} + b) - \xi_n = 0$ 的样本。

参见习题3-8。

3.5 损失函数对比

本章介绍的三种两类分类模型：logistic 回归、感知器和支持向量机。虽然它们的决策函数相同，但由于使用了不同的损失函数以及相应的优化方法，导致它们之间在实际任务上的表现存在一定的差异。

为了比较这些损失函数，我们统一定义类别标签 $y \in \{+1, -1\}$，并定义 $f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b$。这样对于样本 $(\mathbf{x}, y)$，logistic 回归的损失函数可以改写为

$$\begin{aligned}\mathcal{L}_{LR} &= -\log p(y|\mathbf{x}) && (3.94)\\ &= -I(y=1)\log\sigma\Big(f(\mathbf{x},\mathbf{w})\Big) - I(y=-1)\log\sigma\Big(-f(\mathbf{x},\mathbf{w})\Big) && (3.95)\\ &= \log\Big(1+\exp\big(-yf(\mathbf{x},\mathbf{w})\big)\Big). && (3.96)\end{aligned}$$

$1-\sigma(x)=\sigma(-x).$

感知器的损失函数为

$$\mathcal{L}_p = \max\big(0, -yf(\mathbf{x},\mathbf{w})\big). \tag{3.97}$$

软间隔支持向量机的损失函数为 hinge 损失

$$\mathcal{L}_{hinge} = \max\big(0, 1-yf(\mathbf{x},\mathbf{w})\big). \tag{3.98}$$

0-1 损失可以重写为

$$\mathcal{L}_{01} = I\big(yf(\mathbf{x},\mathbf{w}) > 0\big), \tag{3.99}$$

其中 $I(\cdot)$ 为指示函数。

平方损失可以重写为

$$\begin{aligned}\mathcal{L}_{squared} &= \big(y - f(\mathbf{x},\mathbf{w})\big)^2 && (3.100)\\ &= 1 - 2yf(\mathbf{x},\mathbf{w}) + (yf(\mathbf{x},\mathbf{w}))^2 && (3.101)\\ &= \big(1 - yf(\mathbf{x},\mathbf{w})\big)^2. && (3.102)\end{aligned}$$

$y^2 = 1.$

图3.6给出了不同损失函数的对比。对于两类分类来说，当 $yf(\mathbf{x},\mathbf{w}) > 0$ 时，分类器预测正确，并且 $yf(\mathbf{x},\mathbf{w})$ 越大，模型的预测越正确；当 $yf(\mathbf{x},\mathbf{w}) < 0$ 时，分类器预测错误，并且 $yf(\mathbf{x},\mathbf{w})$ 越小，模型的预测越错误。因此，一个好的损失函数应该随着 $yf(\mathbf{x},\mathbf{w})$ 的增大而减少。从图3.6中看出，除了平方损失，其它损失函数都比较适合于两类分类问题。

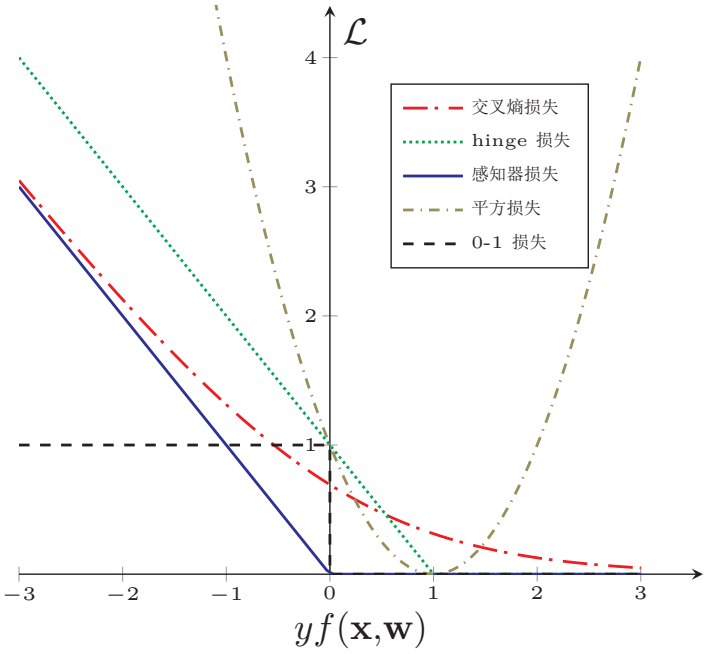


图 3.6 不同损失函数的对比

3.6 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 是离散的类别标签，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个广义线性函数，即一个线性判别函数 $f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$ 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的比较。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 $\{+1, -1\}$。

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。Minsky and Seymour [1969] 在《感知器》一书中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。从现在的视角看，Minsky and Seymour [1969] 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的输出，并将*反向传播算法* [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

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

表 3.1 几种不同的线性模型对比

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。

要深入了解支持向量机以及核方法，可以参考《Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond》[Scholkopf and Smola, 2001]。

习题

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 $\hat{y} = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x})$ 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 (3.33)）中，如果去掉正则化项会有什么影响？

习题 3-4 验证平均感知器训练算法3.2中给出的平均权重向量的计算方式和公式 (3.72) 等价。

习题 3-5 证明定理3.2。

习题 3-6 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。

习题 3-7 验证公式(3.91)。

习题 3-8 在软间隔支持向量机中，试给出原始优化问题的对偶问题，并列出其KKT条件。

参考文献

Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the conference on Empirical methods in natural language processing*, pages 1–8. Association for Computational Linguistics, 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.

John Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, April 1998.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

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.