《机器学习》课程系列

判别函数的线性分类基础*

武汉纺织大学数学与计算机学院

杜小勤

2019/02/26

Contents

*本系列文档属于讲义性质，仅用于学习目的。Last updated on: November 18, 2020。

1 概述

分类问题指的是，将输入变量 $\boldsymbol{x}$ 划分到 K 个离散类别 c_k 中的某一个，其中 $k = 1, \cdots, K$。一般情况下，这些类别互不相交，每个输入变量 $\boldsymbol{x}$ 被划分到唯一的一个类别中。

从分类的观点看，在输入空间，这些输入变量 $\boldsymbol{x}$ 位于不同的决策区域（Decision Region），它们的边界被称为决策边界（Decision Boundary）或者决策面（Decision Surface）。

在本章，我们将讨论线性分类的判别函数（Discriminant Function）。所谓线性分类的判别函数，指的是利用判别函数直接将样本点 $\boldsymbol{x}$ 映射到某个类别 $c_k (k = 1, \cdots, K)$，判别函数也被称为决策函数。其决策面是关于输入变量 $\boldsymbol{x}$ 的线性函数。例如，在 D 维输入空间中，决策面是 $D-1$ 维的超平面（Hyperplane）[1]。分类判别函数有别于分类的概率生成式模型与概率判别式模型这 2 种方法，稍后会作一个简单的对比介绍。

如果数据集可以被线性决策面精确地分类，那么称该数据集是线性可分的（Linearly Separable），否则称数据集是线性不可分的（Linearly Inseparable）。

在监督的分类问题中，目标变量 y 是一个实值向量或标量，用来表示类别标签（Class Label）。例如，在二分类问题中，y 是一个标量，其取值为 $\{c_1, c_2\}$。但是，为了便于数学或算法表达，一般将取值限定为 $\{0, 1\}$ 或 $\{-1, 1\}$。而在多分类问题中，为简化数学公式，一般使用 2 种常用的表示方法：标量形式，y 的取值为 $\{c_1, c_2, \ldots, c_K\}$，其中 K 表示类别个数；One-Hot 向量形式，例如，在手写体数字识别问题中，$K = 10$，数字 3 和 4 被分别编码成 $y = (0, 0, 0, 1, 0, 0, 0, 0, 0, 0)$ 和 $y = (0, 0, 0, 0, 1, 0, 0, 0, 0, 0)$，其余数字的编码，依此类推。在后文，为了描述方便，在不引起歧义的情况下，公式中将混用这两种形式：使用“$y = c_k$”描述“y

[1]例如，在具有 2 个特征分量的二维平面中，线性决策面是一维直线或线段。

属于类别 c_k”这样的事实，虽然 y 同时也是一个 One-Hot 向量。因此，在多分类问题中，不会使用粗体字表示 One-Hot 向量 y。

针对分类问题，有三种不同的解决方案，依照复杂度从高到低，它们分别是：

- 概率生成式方法

  首先，对类的条件概率分布 $p(\boldsymbol{x}|c_k)$ 及类的先验概率分布 $p(c_k)$ 建模；然后使用贝叶斯定理计算后验概率分布：

  $$p\left(c_k|\boldsymbol{x}\right) = \frac{p(\boldsymbol{x}|c_k)p\left(c_k\right)}{p(\boldsymbol{x})} \tag{1}$$

  最后，使用训练集最大化求解该后验概率分布 (即最大后验估计)，得到最优的模型参数；

- 概率判别式方法

  直接对后验概率分布 $p(c_k|\boldsymbol{x})$ 建模，例如把该分布表示为参数模型，然后使用训练集最大化求解该参数模型，得到最优的模型参数；

- 判别函数或决策函数

  该方法直接把样本点 $\boldsymbol{x}$ 映射到具体的类别中；

在线性回归模型中，“线性”指的是，模型预测函数 $y(\boldsymbol{x}, \boldsymbol{w})$ 是关于模型参数 $\boldsymbol{w}$ 的线性函数[2]。在最简单的情况下，线性模型 $y(\boldsymbol{x}, \boldsymbol{w})$ 也是关于输入变量 $\boldsymbol{x}$ 的线性函数，即 $y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$（为了简化符号，省略了模型参数 $\boldsymbol{w}$）。

而在分类问题中，输入变量 $\boldsymbol{x}$ 的预测值是离散类别标签的后验概率分布。此时，为了完成分类任务，可以考虑将最简单的线性模型 $y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$ 进行扩展——在线性变换的基础上，添加一个非线性变换 f：

$$y(\boldsymbol{x}) = f\left(\boldsymbol{w}^T\boldsymbol{x} + w_0\right) \tag{2}$$

其中，函数 f 在机器学习领域被称为激活函数（Activation Function），其反函数在统计学领域被称为连接函数（Link Function）。

如果使用公式（2）进行分类决策，那么 $y(\boldsymbol{x})$ 的决策面是 $\boldsymbol{w}^T\boldsymbol{x} + w_0$，这意味着决策面仍然是线性决策面。因此，不论 f 是否线性函数，决策面始终是 $\boldsymbol{x}$ 的线性函数，公式（2）被称为广义线性模型（Generalized Linear Model, GLM）[3]。

[2]在多数情况下，线性模型 $y(\boldsymbol{x}, \boldsymbol{w})$ 与 $\boldsymbol{x}$ 之间的关系可能是非线性关系。

[3]关于广义线性模型更复杂的定义及其应用，请阅读相关文献。

但是，需要注意的是，与线性回归模型中的情况不一样，$y(\boldsymbol{x})$ 不再是模型参数 $\boldsymbol{w}$ 的线性模型，因为 f 是一个非线性函数。这将导致线性分类模型的计算比线性回归模型的计算要复杂些。

与线性基函数的回归模型类似，在后面的内容中，我们也将引入基函数到分类模型（公式2）中，即使用基函数向量 $\boldsymbol{\phi}(\boldsymbol{x})$ 替换输入向量 $\boldsymbol{x}$：

$$y(\boldsymbol{x}) = f\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}) + w_0\right) \tag{3}$$

在上面的公式中，先使用非线性基函数 $\boldsymbol{\phi}$ 对输入向量 $\boldsymbol{x}$ 进行变换，然后再执行常规线性分类。这意味着，分类模型不再是 $\boldsymbol{x}$ 的线性分类模型，但其功能更加灵活。这是引入基函数之后带来的好处。需要注意的是，下面讨论的算法同样适用于基于线性基函数的分类模型。

2 分类

2.1 二分类基础

首先，考虑二分类模型，然后将它扩展到 $K > 2$ 的分类模型。

最简单的二分类判别函数具有如下形式：

$$y(\boldsymbol{x}) = \text{sign}\left(\boldsymbol{w}^T\boldsymbol{x} + w_0\right) \tag{4}$$

其中，$\boldsymbol{w}$ 被称为权值向量（Weight Vector），w_0 被称为偏置（Bias），sign 表示符号函数[4]：

$$\text{sign}(z) = \begin{cases} +1, & z \geqslant 0 \\ -1, & z < 0 \end{cases} \tag{5}$$

因此，当 $\boldsymbol{w}^T\boldsymbol{x} + w_0 \geqslant 0$ 时，$y(\boldsymbol{x}) = +1$，表示 $\boldsymbol{x}$ 被划分到类别 $+1$ 中；否则，$y(\boldsymbol{x}) = -1$，表示 $\boldsymbol{x}$ 被划分到类别 -1 中。有时候，偏置 w_0 的相反数被称为阈值（Threshold），因为：当 $\boldsymbol{w}^T\boldsymbol{x} \geqslant -w_0$ 时，$y(\boldsymbol{x}) = +1$；否则，$y(\boldsymbol{x}) = -1$。

在公式（4）中，（线性）决策边界或决策面由 $\boldsymbol{w}^T\boldsymbol{x} + w_0 = 0$ 确定，它是一个超平面。其中，$\boldsymbol{w}$ 是该超平面的“法向量”，即向量 $\boldsymbol{w}$ 与决策面上的任何向量都

[4]可以把符号函数 sign 看作是一种阈值函数，当输入变量值大于等于某个阈值（例如，本例中阈值为 0）时，函数的输出值为 $+1$；否则，函数的输出值为 -1。在一些情况下，为处理方便，也可以允许相应的函数输出值为 1 和 0。

正交：假设 $\boldsymbol{x}_1$ 和 $\boldsymbol{x}_2$ 都位于决策面上，那么有 $\boldsymbol{w}^T\boldsymbol{x}_1+w_0=0$ 和 $\boldsymbol{w}^T\boldsymbol{x}_2+w_0=0$，从而有 $\boldsymbol{w}^T(\boldsymbol{x}_1-\boldsymbol{x}_2)=0$。因此，$\boldsymbol{w}$ 决定了决策面的方向。

在明确了向量 $\boldsymbol{w}$ 的意义后，我们来看一下，如何计算原点到决策面的带符号垂直距离。假设 $\boldsymbol{x}$ 是决策面上的点，那么有 $\boldsymbol{w}^T\boldsymbol{x}+w_0=0$，则：

$$\begin{aligned}\boldsymbol{w}^T\boldsymbol{x}&=-w_0\\ \frac{\boldsymbol{w}^T\boldsymbol{x}}{\|\boldsymbol{w}\|}&=-\frac{w_0}{\|\boldsymbol{w}\|}\\ \frac{\boldsymbol{w}^T}{\|\boldsymbol{w}\|}\boldsymbol{x}&=-\frac{w_0}{\|\boldsymbol{w}\|}\end{aligned}\tag{6}$$

在上面的第 3 行公式中，$\frac{\boldsymbol{w}^T}{\|\boldsymbol{w}\|}$ 表示 $\boldsymbol{w}$ 的单位向量，$\frac{\boldsymbol{w}^T}{\|\boldsymbol{w}\|}\boldsymbol{x}$ 表示 $\boldsymbol{w}$ 的单位向量与向量 $\boldsymbol{x}$ 的点积（内积），其含义就是向量 $\boldsymbol{x}$ 在决策面的单位“法向量”方向上的带符号长度。由于向量 $\boldsymbol{x}$ 与原点之间的特殊关系，则原点到决策面的带符号距离就是 $\frac{w_0}{\|\boldsymbol{w}\|}$[5]。因此，偏置 w_0 决定了决策面的位置。

实际上，对于任意向量 $\boldsymbol{x}$ 而言，公式 $\frac{\boldsymbol{w}^T\boldsymbol{x}+w_0}{\|\boldsymbol{w}\|}$ 给出了该向量到决策面的带符号距离。为了说明这一点，考虑任意一点 $\boldsymbol{x}$ 在决策面上的投影 $\boldsymbol{x}_\perp$，如图2-1所示，可得：

$$\begin{aligned}\boldsymbol{x}&=\boldsymbol{x}_\perp+r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\\ \boldsymbol{w}^T\boldsymbol{x}&=\boldsymbol{w}^T\boldsymbol{x}_\perp+r\frac{\boldsymbol{w}^T\boldsymbol{w}}{\|\boldsymbol{w}\|}\\ \boldsymbol{w}^T\boldsymbol{x}+w_0&=\boldsymbol{w}^T\boldsymbol{x}_\perp+w_0+r\frac{\boldsymbol{w}^T\boldsymbol{w}}{\|\boldsymbol{w}\|}\\ r&=\frac{\boldsymbol{w}^T\boldsymbol{x}+w_0}{\|\boldsymbol{w}\|}\end{aligned}\tag{7}$$

其中，上述第 3 行推导到第 4 行，利用了公式 $\boldsymbol{w}^T\boldsymbol{x}_\perp+w_0=0$ 和 $\boldsymbol{w}^T\boldsymbol{w}=\|\boldsymbol{w}\|^2$。

有时候，为了简化公式，将偏置 w_0 并入到权值向量 $\boldsymbol{w}$ 中，然后在输入向量 $\boldsymbol{x}$ 中引入一个对应的输入 1，仍然使用 $\boldsymbol{w}$ 和 $\boldsymbol{x}$ 分别表示扩展之后的权值向量与输入向量，可得：

$$y(\boldsymbol{x})=\operatorname{sign}\left(\boldsymbol{w}^T\boldsymbol{x}\right)\tag{8}$$

本节中，将会混用这 2 种表示方法。

[5]注意，向量 $\boldsymbol{x}$ 在 $\boldsymbol{w}$ 的单位“法向量”上的带符号长度，与原点到决策面的带符号距离，刚好是相反的。更一般地，设决策面为 $\boldsymbol{w}^T\boldsymbol{x}+w_0$，对该决策面按法向量 $\boldsymbol{w}$ 进行归一化处理：$\frac{\boldsymbol{w}^T\boldsymbol{x}}{\|\boldsymbol{w}\|}+\frac{w_0}{\|\boldsymbol{w}\|}$，这就是任意点 $\boldsymbol{x}$ 到该决策面的带符号距离。例如，将原点代入，得到 $\frac{w_0}{\|\boldsymbol{w}\|}$，这与前面的推导结果一致。

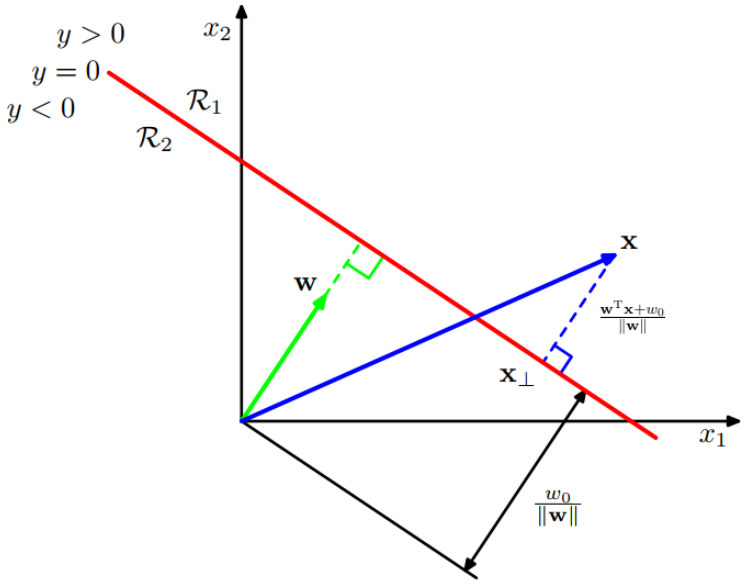


图 2-1: 二维线性判别函数中的一维决策面（一条线）

2.2 多分类基础

对于多分类问题，可以尝试组合前面的二分类判别函数，即使用多个二分类判别函数来构造一个 K 类判别函数。

首先，可以想到 2 种较为直接的从二分类判别函数构造 K 类判别函数的方法。第 1 种方法是，使用 $K-1$ 个二分类器，每个二分类器解决一个二分类问题：把属于类别 c_k 和不属于类别 c_k 的点分开，这种分类器被称为一对多（One-Versus-The-Rest）分类器。然而，这种方法会产生无法分类的区域[6]，如图2-2所示，绿色区域表示无法分类的区域。第 2 种方法是，使用 $\frac{K(K-1)}{2}$ 个二分类器，即为每对类别设置一个二分类器，而每个点的类别将依据多数表决方式来确定：考虑所有判别函数的分类结果，选择表决最多的分类作为最后的分类结果，这种分类器被称为一对一（One-Versus-One）分类器。但是，这也会造成无法分类的区域，如图2-3所示。

实际上，存在一种简单的解决方案：引入 K 个最简形式的决策面来解决 K

[6]这种区域被称为歧义性区域（Ambiguous Region）。

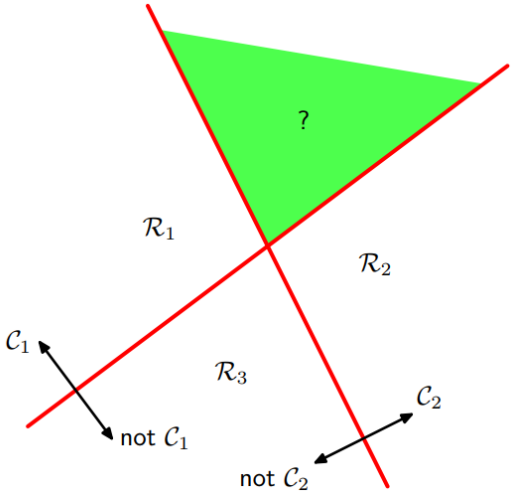


图 2-2: 一对多分类器存在的问题

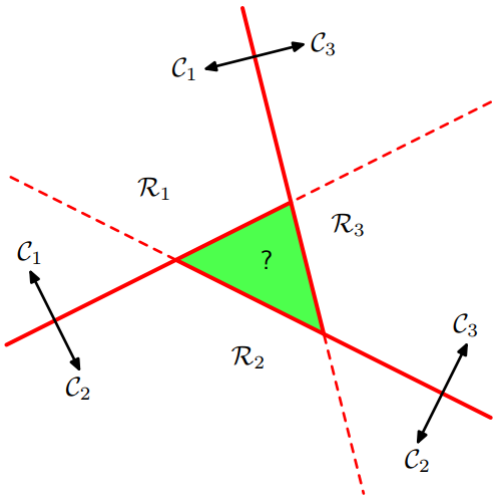


图 2-3: 一对一分类器存在的问题

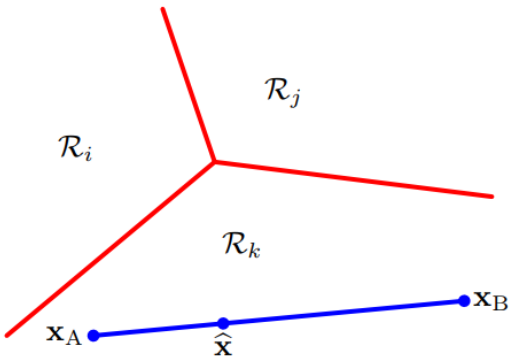


图 2-4: K 类判别函数形成的凸单连通决策区域

类分类问题，从而形成一个 K 类别判别函数[7]：

$$\begin{cases} y_1(\boldsymbol{x}) = \boldsymbol{w}_1^T\boldsymbol{x} + w_{10}, & c_1 \\ y_2(\boldsymbol{x}) = \boldsymbol{w}_2^T\boldsymbol{x} + w_{20}, & c_2 \\ \cdots & \\ y_K(\boldsymbol{x}) = \boldsymbol{w}_K^T\boldsymbol{x} + w_{K0}, & c_K \end{cases} \tag{9}$$

其分类规则如下：对每个待分类的点 $\boldsymbol{x}$，如果 $y_k(\boldsymbol{x}) > y_j(\boldsymbol{x})$ for all $j \neq k$ 成立，那么 $\boldsymbol{x}$ 将被划分为 c_k 类别。

显然，任意 2 个类别 c_k 和 c_j 之间的决策面存在如下的关系：

$$\begin{aligned} y_k(\boldsymbol{x}) = y_j(\boldsymbol{x}) = 0 \\ \left(\boldsymbol{w}_k - \boldsymbol{w}_j\right)^T\boldsymbol{x} + \left(w_{k0} - w_{j0}\right) = 0 \end{aligned} \tag{10}$$

这种 K 类判别函数在输入空间上划分形成的决策区域具有如下特点：

- 决策区域之间的决策面是超平面；
- 决策区域总是单连通的凸区域；

  为了表明这个结论是正确的，假设在决策区域 $\mathcal{R}_k$ 中有 2 个点 $\boldsymbol{x}_A$ 和 $\boldsymbol{x}_B$，如图2-4所示。那么，任何位于 $\boldsymbol{x}_A$ 和 $\boldsymbol{x}_B$ 连线上的点都可以表示成下面的形式：

$$\widehat{\boldsymbol{x}} = \lambda\boldsymbol{x}_A + (1-\lambda)\boldsymbol{x}_B \tag{11}$$

[7]例如，用于分类的神经网络也采取类似的解决方案，它为每个类别设计一个输出。训练时，为每个输入 $\boldsymbol{x}$ 配置一个理想输出 y，整体输出则采取 One-Hot 编码形式。分类预测时，神经网络在所有的 K 个输出中，选取输出值最高的输出作为整体的分类结果。

其中 $0 \leqslant \lambda \leqslant 1$。根据 K 类判别函数的线性性质，下式成立：

$$y_k(\widehat{\boldsymbol{x}}) = \lambda y_k(\boldsymbol{x}_A) + (1-\lambda) y_k(\boldsymbol{x}_B) \tag{12}$$

由于 $\boldsymbol{x}_A$ 和 $\boldsymbol{x}_B$ 位于 $\mathcal{R}_k$ 的内部,则 $y_k(\boldsymbol{x}_A) > y_j(\boldsymbol{x}_A)$ 和 $y_k(\boldsymbol{x}_B) > y_j(\boldsymbol{x}_B)$,$\forall j \neq k$ 成立，则可得 $y_k(\widehat{\boldsymbol{x}}) > y_j(\widehat{\boldsymbol{x}})$。这表明 $\widehat{\boldsymbol{x}}$ 也在区域 $\mathcal{R}_k$ 的内部，即 $\mathcal{R}_k$ 是凸单连通区域。

有了判别函数，下面将具体讨论使用训练数据对参数进行调整的方法：最小平方法、Fisher 线性判别函数与感知机算法。

3 基本判别函数

3.1 最小平方分类方法

对于简单的二分类情形，既可以使用 K 类别判别函数（其中 $K=2$），也可以使用第2.1节中介绍的单判别函数。因此，下面以通用的 K 类别判别函数进行讨论。

对于 K 类判别函数，每个类 c_k 有自己的线性模型：

$$y_k(\boldsymbol{x}) = \boldsymbol{w}_k^T \boldsymbol{x} + w_{k0} \tag{13}$$

为了简化公式，可以将 w_{k0} 并入向量 $\boldsymbol{w}_k$ 中，并在输入变量 $\boldsymbol{x}$ 中增加一个虚输入 $x_0 = 1$，仍然使用原向量符号 $\boldsymbol{w}$ 和 $\boldsymbol{x}$ 来表示扩展后的向量，即：

$$y_k(\boldsymbol{x}) = \boldsymbol{w}_k^T \boldsymbol{x} \tag{14}$$

其中，$\boldsymbol{w}_k = (w_{k0}, w_{k1}, w_{k2}, \ldots)^T$ 和 $\boldsymbol{x} = (1, x_1, x_2, \ldots)^T$。

下面将 K 个线性判别函数使用矩阵来表示：

$$y(\mathbf{x}) = \boldsymbol{W}^T \boldsymbol{x} \tag{15}$$

其中，$\boldsymbol{W}$ 是一个矩阵，第 k 列是向量 $\boldsymbol{w}_k$。在这种表示方式下，新输入 $\boldsymbol{x}$ 将被划分到 $y_k = \boldsymbol{w}_k^T \boldsymbol{x}$ 取得最大值的类别 c_k 中[8]。

[8]注意，虽然 y 表示多类别，但是仍然使用非粗体字来表示。原因在于，为公式表达的简便性，$y = c_k$ 及其 One-Hot 向量形式会混用，前文已经讨论过。

考虑训练数据集 $\{\boldsymbol{x}_n, y_n\}$, $n = 1, \ldots, N$，定义矩阵 $\boldsymbol{Y}$，它的第 n 行是 y_n 的 One-Hot 向量；定义矩阵 $\boldsymbol{X}$，它的第 n 行是向量 $\boldsymbol{x}_n^T$。因此，该训练数据集的平方误差函数可以表示成：

$$E_D(\boldsymbol{W}) = \frac{1}{2}\mathrm{Tr}\left\{(\boldsymbol{X}\boldsymbol{W} - \boldsymbol{Y})^T(\boldsymbol{X}\boldsymbol{W} - \boldsymbol{Y})\right\} \tag{16}$$

然后，令平方误差函数关于 $\boldsymbol{W}$ 的导数等于 0，可以得到 $\boldsymbol{W}$ 的解析解：

$$\boldsymbol{W} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{Y} = \boldsymbol{X}^\dagger\boldsymbol{Y} \tag{17}$$

其中，$\boldsymbol{X}^\dagger$ 被称为 $\boldsymbol{X}$ 的 Moore-Penrose 伪逆矩阵（Pseudo-Inverse Matrix），形式为：

$$\boldsymbol{X}^\dagger \equiv \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T \tag{18}$$

因此，得到了如下形式的 K 类判别函数：

$$y(\boldsymbol{x}) = \boldsymbol{W}^T\boldsymbol{x} = \boldsymbol{Y}^T\left(\boldsymbol{X}^\dagger\right)^T\boldsymbol{x} \tag{19}$$

值得注意的是，如果训练集里每个目标向量 y_n 都满足某个线性限制：

$$\boldsymbol{a}^T y_n + b = 0 \tag{20}$$

其中 $\boldsymbol{a}$ 和 b 分别为常向量和常量，那么对于任何输入变量 $\boldsymbol{x}$，模型的预测值也满足同样的限制：

$$\boldsymbol{a}^T y(\boldsymbol{x}) + b = 0 \tag{21}$$

因此，在目标向量 y_n 使用“1-of-K”或 One-Hot 形式时，相应的模型预测值将具有如下性质：对于任意的输入变量 $\boldsymbol{x}$，$y(\boldsymbol{x})$ 的所有分量之和等于 1。但是，需要注意，这并不意味着模型的预测输出表现为概率形式，因为预测值并没有限制在 $0-1$ 范围内。

虽然最小平方法给出了 K 类判别函数的精确解析解，但是，同回归中的问题一样，最小平方法对于离群点缺少鲁棒性，如图3-5所示。图中，洋红色直线表示最小平方法找到的决策边界，绿色直线表示逻辑回归模型找到的决策边界。可以看出，最小平方法对离群点非常敏感，而逻辑回归模型却具有很好的鲁棒性。这是由最小平方法的性质造成的，它会惩罚那些远离决策边界 (即过于正确) 的预测。后面将会考虑另外几种误差函数，它们不会出现这种问题。

然而，最小平方法还存在更加严重的问题，如图3-6所示。该训练数据集由三

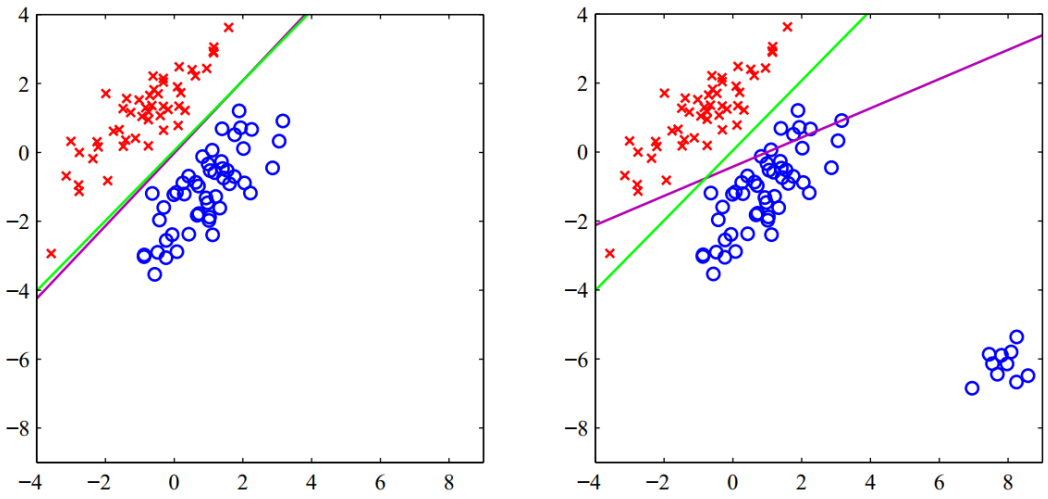


图 3-5: 最小平方法对离群点很敏感

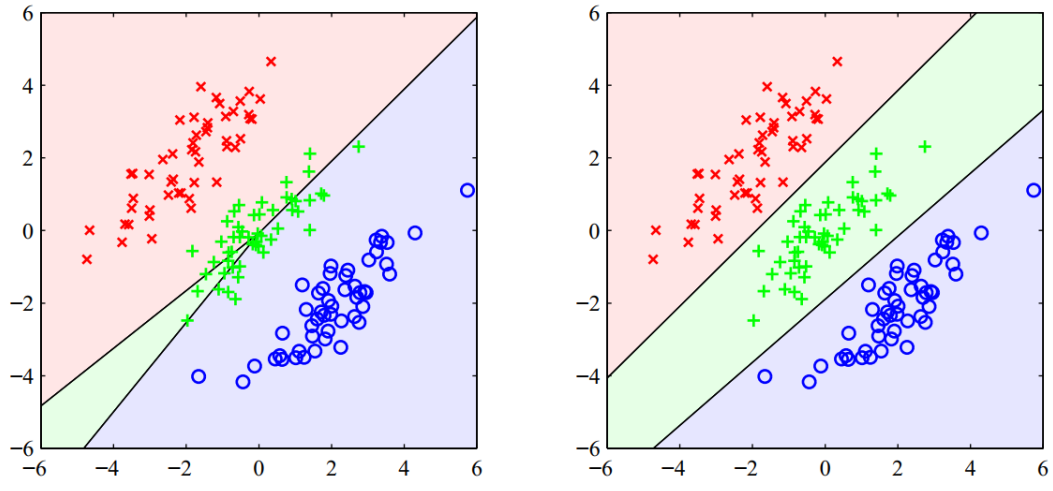


图 3-6: 最小平方法不能正确地进行多分类的例子

个类别组成，本身是线性可分的。左图是最小平方法分类的结果，右图是逻辑回归分类的结果。显然，最小平方法没有产生正确的分类，而逻辑回归给出了正确的分类。

最小平方法无法产生正确的分类，其根本原因在于，这种方法对应于高斯条件分布假设下的最大似然法，而 y_n 中的每个分量是二值 $\{0,1\}$ 形式，它显然不服从高斯分布。因此，可以使用更加恰当的概率分布并转换为相应的损失函数来解决这个问题。

3.2 Fisher 二分类方法

在前面的线性分类判别方法中，无论是二分类问题，还是多分类问题，都需要定义线性变换函数 $\boldsymbol{w}^T\boldsymbol{x}$，它的作用是形成一个决策边界，从而对输入空间产生一个划分。

从维度降低的角度来看，它的作用是将一个 D 维输入向量 $\boldsymbol{x}$ 经过权值向量 $\boldsymbol{w}$ 投影到 1 维上。对于二分类问题，如果投影的结果（标量）大于等于 0（阈值），那么该输入变量 $\boldsymbol{x}$ 将被划分到类别 c_1，否则将被划分到类别 c_2。对于多分类问题，其分类规则为：对每个待划分的点 $\boldsymbol{x}$，如果 $y_k(\boldsymbol{x}) > y_j(\boldsymbol{x}),\forall j \neq k$ 成立，那么 $\boldsymbol{x}$ 将被划分到 c_k 类别，其效果相当于确定了多个阈值，并依据阈值对输入空间进行划分。

通常来说，向 1 维进行投影会造成相当多的信息损失，并使得原本能够在 D 维输入空间线性可分的训练数据集，在投影到 1 维空间后造成重叠，从而变得线性不可分。

但是，假设依然要将 D 维输入空间投影到 1 维空间上，我们可以选取类别之间分离效果最好的投影，即寻求最佳的投影权值向量，这就是 Fisher 线性分类器的基本思想。

为了推导出 Fisher 线性分类器，首先考虑二分类情形，然后推广到 K 分类情形。假设 c_1 类中存在 N_1 个点，c_2 类中存在 N_2 个点。那么，这 2 个类别的均值向量分别为：

$$\boldsymbol{m}_1 = \frac{1}{N_1}\sum_{n\in c_1}\boldsymbol{x}_n, \quad \boldsymbol{m}_2 = \frac{1}{N_2}\sum_{n\in c_2}\boldsymbol{x}_n \tag{22}$$

它们在原输入空间中的类别距离是 $\boldsymbol{m}_2 - \boldsymbol{m}_1$，经过 $\boldsymbol{w}$ 投影后，在 1 维空间中的

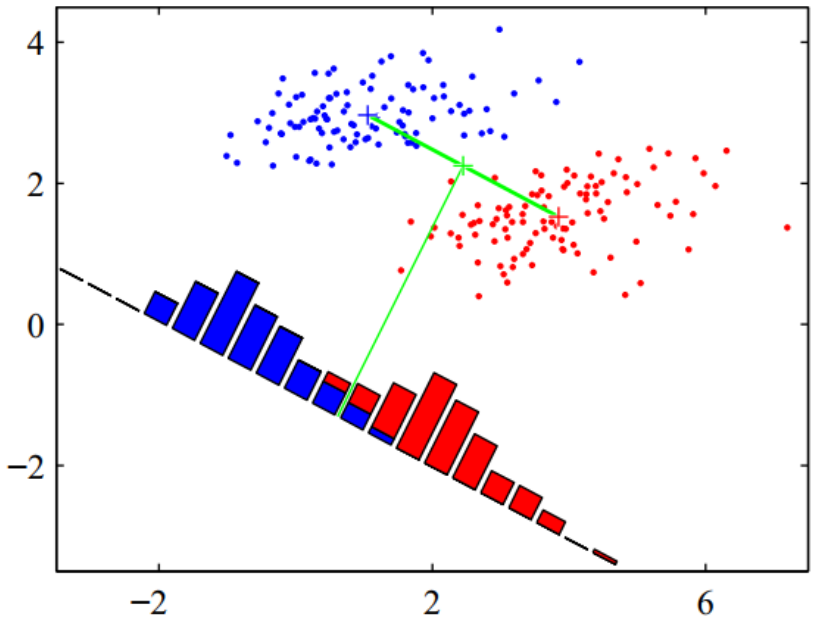


图 3-7: 类别均值投影的距离最大化方法存在的问题

类别距离是：

$$m_2 - m_1 = \boldsymbol{w}^T (\boldsymbol{m}_2 - \boldsymbol{m}_1) \tag{23}$$

其中：

$$m_k = \boldsymbol{w}^T \boldsymbol{m}_k \tag{24}$$

m_k 是来自类别 c_k 的投影均值。公式（23）表明，原输入空间中的类别距离是类别均值之间的距离，该距离的投影等于两个类别分别投影后的均值之差。显然，这是一种非常简单的类别距离度量方式。为了获得最佳的类别间分离效果，可以关于权值向量 $\boldsymbol{w}$ 最大化求解公式（23）。

但是，需要注意的是，必须对权值向量 $\boldsymbol{w}$ 施加限制，否则，权值向量 $\boldsymbol{w}$ 可以无限制地增大。因此，可以将权值向量 $\boldsymbol{w}$ 限制为单位向量，即 $\sum_i w_i^2 = 1$——问题转化为带约束的拉格朗日优化问题，可以使用拉格朗日乘数法来求解最优权值向量 $\boldsymbol{w}$，并得到结果：$\boldsymbol{w} \propto (\boldsymbol{m}_2 - \boldsymbol{m}_1)$。

这个结论很有趣，它表明，最佳权值向量 $\boldsymbol{w}$ 与均值向量 $\boldsymbol{m}_2 - \boldsymbol{m}_1$ 的方向一致，那么分离超平面将垂直于均值向量，分类情形如图3-7所示。可以看出，稍长的绿色直线表示分离超平面，稍短的绿色直线表示均值向量，直方图形式可以直观地表示类别均值投影之后的距离 $m_2 - m_1$ 的最大化——要最大化投影之后的均值距离，就要选择这样的分离超平面，它垂直于均值向量，且经过 2 个均值向量

的中点——这的确是一种非常简单的类别距离度量方式。

然而，该方法存在一个问题：原本 2 个类别在原始空间中是线性可分的，但是在投影之后，2 个类别之间就存在一定程度的重叠！显然，问题出在投影向量 $\boldsymbol{w}$ 上，需要对它进行调整。

如何调整投影向量 $\boldsymbol{w}$ 呢？注意到，如果类别分布的非对角协方差较大，那么这种问题就会出现。这给了我们一种改进算法的思路——寻找的投影向量 $\mathbf{w}$，既要最大化类别间的投影距离，又要最小化类内的方差，这样可以确保找到较为完美的分离超平面，它既能够最大限度地分离输入点，同时又使得类别重叠的部分最小化。这便是 Fisher 线性分类器的设计思路。

首先，定义 c_k 的类内方差：

$$s_k^2 = \sum_{n \in c_k} (y_n - m_k)^2 \tag{25}$$

其中，$y_n = \boldsymbol{w}^T \boldsymbol{x}_n$，$m_k$ 是来自类别 c_k 的投影均值。

然后，定义总的类内方差（Within-Class Variance）为 $s_1^2 + s_2^2$，定义 Fisher 准则为：

$$J(\boldsymbol{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} \tag{26}$$

它表示类间方差（Between-Class Variance）与类内方差的比值。使用前面的相关公式替换其中的 m_1、m_2、s_1 和 s_2，可以得到关于 $\boldsymbol{w}$ 的公式：

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}} \tag{27}$$

其中，$\boldsymbol{S}_B$ 表示类间协方差矩阵：

$$\boldsymbol{S}_B = (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T \tag{28}$$

$\boldsymbol{S}_W$ 表示类内协方差矩阵：

$$\boldsymbol{S}_W = \sum_{n \in c_1} (\boldsymbol{x}_n - \boldsymbol{m}_1)(\boldsymbol{x}_n - \boldsymbol{m}_1)^T + \sum_{n \in c_2} (\boldsymbol{x}_n - \boldsymbol{m}_2)(\boldsymbol{x}_n - \boldsymbol{m}_2)^T \tag{29}$$

对公式（27）关于 $\boldsymbol{w}$ 最大化，得到：

$$\left(\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}\right) \boldsymbol{S}_W \boldsymbol{w} = \left(\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}\right) \boldsymbol{S}_B \boldsymbol{w} \tag{30}$$

从公式（28）可以看出：

$$\begin{aligned} \boldsymbol{S}_B \boldsymbol{w} &= (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T \boldsymbol{w} \\ &= c\,(\boldsymbol{m}_2 - \boldsymbol{m}_1) \end{aligned} \tag{31}$$

其中 $c = (\boldsymbol{m}_2 - \boldsymbol{m}_1)^T \boldsymbol{w}$ 表示某个常量。上式表明，$\boldsymbol{S}_B\boldsymbol{w}$ 总是在向量 $(\boldsymbol{m}_2 - \boldsymbol{m}_1)$ 的方向上。此外，我们仅仅需要 $\boldsymbol{w}$ 的方向，而不关心它的长度，因此，可以将标量因子 $\left(\boldsymbol{w}^T\boldsymbol{S}_B\boldsymbol{w}\right)$ 和 $\left(\boldsymbol{w}^T\boldsymbol{S}_W\boldsymbol{w}\right)$ 从公式（30）中去掉，再使用公式（31）进行相应的替换（也去掉常量因子），最后公式（30）两端同时乘以 $\boldsymbol{S}_W^{-1}$，从而得到：

$$\boldsymbol{w} \propto \boldsymbol{S}_W^{-1}(\boldsymbol{m}_2 - \boldsymbol{m}_1) \tag{32}$$

注意，改进方法比标准方法多了一个矩阵变换 $\boldsymbol{S}_W^{-1}$，即类内协方差矩阵的逆变换，它的意义也非常明确——除了考虑类间距离最大化之外，还显式地考虑了类内协方差的最小化[9]，这是改进算法的优势。

值得注意的是，如果类内协方差矩阵 $\boldsymbol{S}_W$ 是各向同性的（Isotropic），那么它与单位矩阵成正比，此时 $\boldsymbol{w}$ 就与类间均值向量 $(\boldsymbol{m}_2 - \boldsymbol{m}_1)$ 成正比，这与标准方法得到的结果一样。

公式（32）被称为 Fisher 线性判别式。但是，严格来说，它并不是一个判别式，而是样本数据点投影方向的一个特定选择。然而，将数据向 $\boldsymbol{w}$ 投影之后，确实可以构造出一个判别式用来将输入 $\boldsymbol{x}$ 进行分类：如果 $y(\boldsymbol{x}) \geq y_0$，那么 $\boldsymbol{x}$ 属于类别 c_1；否则，$\boldsymbol{x}$ 属于类别 c_2。

如何确定最优阈值 y_0 呢？我们可以假设类条件概率密度 $p(y|c_k)$ 服从高斯分布[10]，然后通过最大似然法求解出高斯分布的参数。在对每个类别求解了高斯分布参数之后，可以通过最小化误分类率的方式获得最优阈值[11]。

3.3 最小平方分类方法与 Fisher 分类方法

最小平方法的训练准则是，尽可能地让模型预测与理想值接近。而 Fisher 方法的训练准则是，尽可能地拉开类别间的距离，而缩小类别内的距离。

可以看出，它们之间存在相似性——对于最小平方法而言，它需要确定出最佳的决策面，以确保误分类率最小。而从分类的投影解释来看，选取最佳决策面，也是为了尽可能地拉开类别间的距离。因此，直觉上来看，这 2 种方法之间必然存在着某种联系。

下面，我们将表明，对于二分类问题，如果选取适当的目标向量编码方式，那么这 2 种方法实际上是等价的。

[9] 类内协方差逆的最大化，相当于类内协方差的最小化。

[10] 注意到，$y = \boldsymbol{w}^T\boldsymbol{x}$ 是一组随机变量的和，根据中心极限定理，我们可以做出高斯分布的假设。

[11] 详细方法，请参考文献 [2]，P39，1.5.1 节。

在前面的讨论中，我们已经知道，对于二分类目标值，可以使用 $\{c_1, c_2\}$ 这样的编码方式，例如 $\{0, 1\}$ 或 $\{-1, 1\}$；而对于多分类目标值，可以使用 1-of-K 或 One-Hot 编码方式。

现在，我们使用另一种编码方式。令类别 $c_1 = \frac{N}{N_1}$，类别 $c_2 = -\frac{N}{N_2}$，其中 N_1 和 N_2 分别表示训练数据集中类别 c_1 和 c_2 的个数。

最小平方法使用如下的误差函数：

$$E = \frac{1}{2}\sum_{n=1}^{N}\left(\boldsymbol{w}^T\boldsymbol{x}_n + w_0 - y_n\right)^2 \tag{33}$$

然后，求解误差函数 E 关于参数 w_0 和 $\boldsymbol{w}$ 的偏导数，并令其等于 0，可得：

$$\begin{aligned}\frac{\partial E}{\partial w_0} &= \sum_{n=1}^{N}\left(\boldsymbol{w}^T\boldsymbol{x}_n + w_0 - y_n\right) = 0\\ \frac{\partial E}{\partial \boldsymbol{w}} &= \sum_{n=1}^{N}\left(\boldsymbol{w}^T\boldsymbol{x}_n + w_0 - y_n\right)\boldsymbol{x}_n = 0\end{aligned} \tag{34}$$

求解 w_0，得到：

$$w_0 = -\boldsymbol{w}^T\boldsymbol{m} \tag{35}$$

其中，我们使用了如下公式：

$$\sum_{n=1}^{N} y_n = N_1\frac{N}{N_1} - N_2\frac{N}{N_2} = 0 \tag{36}$$

$\boldsymbol{m}$ 表示整个数据集的均值向量：

$$\boldsymbol{m} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n = \frac{1}{N}\left(N_1\boldsymbol{m}_1 + N_2\boldsymbol{m}_2\right) \tag{37}$$

求解 $\boldsymbol{w}$，可得：

$$\left(\boldsymbol{S}_W + \frac{N_1N_2}{N}\boldsymbol{S}_B\right)\boldsymbol{w} = N\left(\boldsymbol{m}_1 - \boldsymbol{m}_2\right) \tag{38}$$

其中，$\boldsymbol{S}_W$ 和 $\boldsymbol{S}_B$ 分别表示类内协方差矩阵和类间协方差矩阵，与 Fisher 方法中的定义一样。注意到，$\boldsymbol{S}_B\boldsymbol{w}$ 总是在向量 $(\boldsymbol{m}_2 - \boldsymbol{m}_1)$ 的方向上，于是：

$$\boldsymbol{w} \propto \boldsymbol{S}_W^{-1}\left(\boldsymbol{m}_2 - \boldsymbol{m}_1\right) \tag{39}$$

该公式与 Fisher 方法中得到的结果一致。

考虑到 $w_0 = -\boldsymbol{w}^T\boldsymbol{m}$，因此，对于待预测输入向量 $\boldsymbol{x}$，当 $y(\boldsymbol{x}) = \boldsymbol{w}^T(\boldsymbol{x} - \boldsymbol{m}) > 0$ 时，$\boldsymbol{x}$ 属于类别 c_1，否则属于类别 c_2。

3.4 Fisher 多分类方法

下面将 Fisher 二分类线性判别函数泛化到多分类线性判别函数，此处的多分类指的是类别个数 $K > 2$。设输入向量 $\boldsymbol{x}$ 的维度为 D，并引入输出向量 y，[12]其维度为 D'，它的每个元素 $y_k = \boldsymbol{w}_k^T \boldsymbol{x}$，$k = 1, \cdots, D'$。为了方便起见，将使用矩阵形式表示：

$$y = \boldsymbol{W}^T \boldsymbol{x} \tag{40}$$

其中，权值矩阵 $\boldsymbol{W}$ 的列是权值向量 $\boldsymbol{w}_k$。

为了得到 Fisher 多分类判别函数，需要仿照二分类情形，定义相应的投影前后的协方差矩阵：输入空间中的类内协方差矩阵 $\boldsymbol{S}_W$ 和类间协方差矩阵 $\boldsymbol{S}_B$、输出空间（投影空间）中的类内协方差矩阵 $\boldsymbol{s}_W$ 和类间协方差矩阵 $\boldsymbol{s}_B$，并基于上述协方差矩阵，定义多分类情形的 Fisher 优化准则 $J(\boldsymbol{W})$——类间方差与类内方差的比值，它是一个标量。$J(\boldsymbol{W})$ 的最大化，既确保了最大化类间方差，又确保了最小化类内方差，因而确保了能够找到较为完美的分离超平面。

首先，定义输入空间中的 K 类别类内协方差矩阵 $\boldsymbol{S}_W$，定义相当直接：

$$\boldsymbol{S}_W = \sum_{k=1}^{K} \boldsymbol{S}_k \tag{41}$$

其中，$\boldsymbol{S}_k$ 定义为：

$$\begin{aligned} \boldsymbol{S}_k &= \sum_{n \in c_k} (\boldsymbol{x}_n - \boldsymbol{m}_k)(\boldsymbol{x}_n - \boldsymbol{m}_k)^T \\ \boldsymbol{m}_k &= \frac{1}{N_k} \sum_{n \in c_k} \boldsymbol{x}_n \end{aligned} \tag{42}$$

其中，N_k 是类别 c_k 的样本点个数。

输入空间中的 K 类别类间协方差矩阵 $\boldsymbol{S}_B$ 的定义不是很直接，需要借助于全协方差矩阵：

$$\boldsymbol{S}_T = \sum_{n=1}^{N} (\boldsymbol{x}_n - \boldsymbol{m})(\boldsymbol{x}_n - \boldsymbol{m})^T \tag{43}$$

其中，$\boldsymbol{m}$ 是整个数据集的均值向量：

$$\boldsymbol{m} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n = \frac{1}{N} \sum_{k=1}^{K} N_k \boldsymbol{m}_k \tag{44}$$

[12]由于前述原因，不使用粗体字表示它。

其中，$N=\sum_k N_k$ 是整个数据集的样本点个数。有了全协方差矩阵和类内协方差矩阵，就可以按如下方式定义类间协方差矩阵：

$$\boldsymbol{S}_T = \boldsymbol{S}_{\boldsymbol{W}} + \boldsymbol{S}_B \tag{45}$$

从而得到：

$$\boldsymbol{S}_B = \sum_{k=1}^{K} N_k \left(\boldsymbol{m}_k - \boldsymbol{m}\right)\left(\boldsymbol{m}_k - \boldsymbol{m}\right)^T \tag{46}$$

下面，在投影空间中定义相应的协方差矩阵。类似地，类内协方差矩阵定义为：

$$\boldsymbol{s}_W = \sum_{k=1}^{K}\sum_{n\in c_k} \left(y_n - \boldsymbol{\mu}_k\right)\left(y_n - \boldsymbol{\mu}_k\right)^T \tag{47}$$

类间协方差矩阵定义为：

$$\boldsymbol{s}_B = \sum_{k=1}^{K} N_k \left(\boldsymbol{\mu}_k - \boldsymbol{\mu}\right)\left(\boldsymbol{\mu}_k - \boldsymbol{\mu}\right)^T \tag{48}$$

其中：

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{n\in c_k} y_n, \quad \boldsymbol{\mu} = \frac{1}{N}\sum_{k=1}^{K} N_k \boldsymbol{\mu}_k \tag{49}$$

有了 K 类别投影空间中的类间和类内协方差矩阵，仿照二分类情形，定义如下的 Fisher 最优化准则：

$$J(\boldsymbol{W}) = \mathrm{Tr}\left\{\boldsymbol{s}_W^{-1}\boldsymbol{s}_B\right\} \tag{50}$$

最大化该公式将确保同时最大化类间方差与最小化类内方差。与二分类情形类似，使用上述的权值矩阵 $\boldsymbol{W}$、输入空间中的类内协方差矩阵 $\boldsymbol{S}_W$ 和类间协方差矩阵 $\boldsymbol{S}_B$ 进行替换，可以得到如下形式的公式：

$$J(\boldsymbol{W}) = \mathrm{Tr}\left\{\left(\boldsymbol{W}\boldsymbol{S}_W\boldsymbol{W}^T\right)^{-1}\left(\boldsymbol{W}\boldsymbol{S}_B\boldsymbol{W}^T\right)\right\} \tag{51}$$

最大化 $J(\boldsymbol{W})$ 将获得最优权值矩阵 $\boldsymbol{W}$，它是 $\boldsymbol{S}_W^{-1}\boldsymbol{S}_B$ 的前 D' 个（最大特征值对应的）特征向量。

值得注意的是，观察公式（46），可以发现，$\boldsymbol{S}_B$ 是由 K 个矩阵叠加而成的，而每个矩阵的秩为 1，[13]再由公式（44）可知，$\boldsymbol{S}_B$ 的 K 个外积（矩阵）只有 $K-1$

[13]注意到，每个矩阵都是由 2 个向量的外积得到的，这种矩阵的秩为 1。原因是，无论是列向量，还是行向量，都可以看作是特殊的矩阵，它们的秩为 1，设列向量为 A，行向量为 B，则外积 AB 是一个矩阵，其秩满足下面的关系：$R(AB) \le R(A)R(B) = 1$。

个是相互独立的，故矩阵 $\boldsymbol{S}_B$ 的秩至多是 $K-1$，其非零特征值至多只有 $K-1$ 个。这表明，向 $\boldsymbol{S}_B$ 的特征向量张成的 $K-1$ 维子空间进行投影，并不会改变 $J(\boldsymbol{W})$ 的值，因而投影空间的维度 D' 不会超过 $K-1$。

3.5 感知机

感知机 (Perceptron) 于 1957 年由 Rosenblatt 提出，虽然其形式与学习算法都非常简单，也不是第 1 个神经元模型[14]，但是它在模式识别与机器学习的发展历史上占有重要的地位。

感知机的出现，标志着研究者们开始真正地对学习过程进行数学理论上的研究。1962 年，Novikoff 证明了关于感知机的第 1 个定理，这标志着学习理论研究的开始。该定理在创建学习理论的过程中起了十分重要的作用。从某种意义上说，它建立了学习算法的泛化能力与训练集误差的最小化之间的联系。

感知机的决策函数为：

$$f(\boldsymbol{x}) = \text{sign}(\boldsymbol{w}^T\boldsymbol{x} + b) \tag{52}$$

其中，$\boldsymbol{w}$ 被称为权值向量 (Weight Vector)；b 被称为偏置 (Bias)；sign 为符号函数：

$$\text{sign}(z) = \begin{cases} +1 & z \geqslant 0 \\ -1 & z < 0 \end{cases} \tag{53}$$

可以看出，感知机是一种二分类判别模型，其输入为样本点的特征向量 $\boldsymbol{x}$；输出 y 分为 2 类：为算法实现上的便利，取值分别为 $+1$ 和 -1。通过第2.1部分的讨论，可知，权值向量 $\boldsymbol{w}$ 用于控制决策面 $\boldsymbol{w}^T\boldsymbol{x}+b=0$ 的方向，而偏置 b 用于控制决策面的位置。感知机模型的假设空间是输入特征空间 $\boldsymbol{x}$ 上的所有线性分类模型，即 $\{f|f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b\}$。

与 Fisher 分类方法不同，感知机直接以决策面 $\boldsymbol{w}^T\boldsymbol{x}+b=0$ 为界，将输入空间划分为 2 部分：$\boldsymbol{w}$ 所指向的一边，类别为 $+1$，称为正类；反之，类别为 -1，称为负类。而 Fisher 分类方法首先需要将数据样本点进行降维，并在确定最佳投影

[14] 第 1 个神经元模型是 McCulloch-Pitts 模型 (M-P 模型，1943 年提出)，一般被认为是神经网络的开端之作，它使用 Sigmoid 或 Tanh 作为激活函数。实际上，感知机是一种基于 M-P 结构的模型，但它使用了不连续函数 Sign 作为激活函数。虽然感知机的思想并不新颖，但是 Rosenblatt 使用计算机程序展示了感知机的功能，并通过简单的实验验证了感知机模型具有一定的泛化能力。

方向之后，还需要找到最佳分界点，最后完成对样本点的分类。另外，无论从历史发展的角度看，还是从技术视角上看，神经网络和 SVM 都是感知机模型的一种延伸。

3.5.1 学习策略

首先，给出数据集的线性可分性定义。

定义 **3.1**（数据集的线性可分性）给定一个数据集

$$T = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\},$$

其中 $\boldsymbol{x}_i \in \mathbf{R}^n$，$y_i = \{+1, -1\}$，$i = 1, 2, \ldots, N$，如果存在某个决策面 $\boldsymbol{w}^T\boldsymbol{x} + b = 0$，能够将数据集的正负样本点完全正确地划分到决策面的两侧，即对所有的 $y_i = +1$ 样本点，有 $\boldsymbol{w}^T\boldsymbol{x}_i + b > 0$，对所有的 $y_j = -1$ 样本点，有 $\boldsymbol{w}^T\boldsymbol{x}_j + b < 0$，则称数据集 T 是线性可分的 *(Linearly Separable)*；否则，称数据集 T 是线性不可分的 *(Linearly Inseparable)*。

如何从数据集 T 中学习权值向量 $\boldsymbol{w}$ 和偏置 b？对于学习算法而言，一般要定义一个目标函数。常用的目标函数，包括效用函数 (Agent 或智能体)、值函数 (强化学习)、损失函数或负对数似然函数 (监督学习) 等。对于监督学习，常常选取损失函数或负对数似然函数作为目标函数，并执行最优化过程，用于从数据集中学习权值参数 $\boldsymbol{w}$ 和偏置 b。

对于损失函数的选取，需要考虑以下几个重要因素：

- 能够准确地反映误分类及其代价；
- 便于最优化地求取参数：凸函数；连续可导函数；

一种很自然的想法是，对于单个样本点而言，正确分类时，损失为 0；错误分类时，损失为 1，即得到如下的 (单样本点)0-1 损失函数：

$$l(y_i, f(\boldsymbol{x}_i)) = \begin{cases} 1 & y_i \neq f(\boldsymbol{x}_i) \\ 0 & y_i = f(\boldsymbol{x}_i) \end{cases} \tag{54}$$

其中，$f(\boldsymbol{x}_i) = \text{sign}(\boldsymbol{w}^T\boldsymbol{x}_i + b)$。利用 y 和 $f(\boldsymbol{x}_i)$ 取值为 $\{+1, -1\}$ 的特点，将上述损失函数改写为：

$$l(y_i, f(\boldsymbol{x}_i)) = \begin{cases} 1 & y_i f(\boldsymbol{x}_i) < 0 \\ 0 & y_i f(\boldsymbol{x}_i) > 0 \end{cases} \tag{55}$$

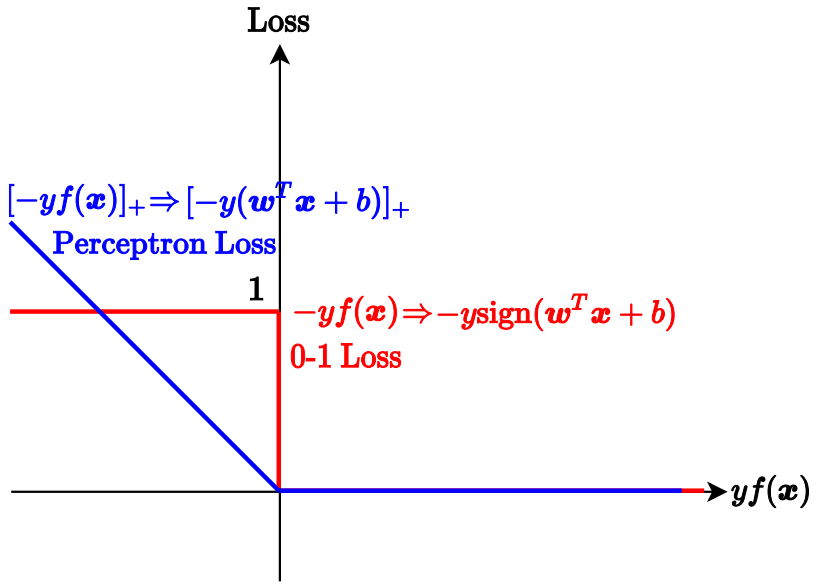


图 3-8: 0-1 损失函数与感知机损失函数

进一步改写为：

$$l(y_i, f(\boldsymbol{x}_i)) = \begin{cases} -y_i f(\boldsymbol{x}_i)(\equiv 1) & y_i f(\boldsymbol{x}_i) < 0 \\ 0 & y_i f(\boldsymbol{x}_i) > 0 \end{cases} \tag{56}$$

然而，0-1 损失函数是一个非凸函数，且不是参数 $\boldsymbol{w}$ 和 b 的可导函数，不易进行优化，如图3-8中红线所示。其中的一个关键因素在于，使用 sign 对 $\boldsymbol{w}^T\boldsymbol{x}_i + b$ 进行了符号钳制 (即二值化为 $+1$ 或 -1)，使得 $-y_i\text{sign}(\boldsymbol{w}^T\boldsymbol{x}_i + b)$ 在 $(0, 0)$ 处，出现了阶跃。

因此，可以考虑将 sign 从损失函数中去掉[15]，于是便得到了感知机的 (单样本点) 损失函数 $-y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)$，如图3-8中蓝线所示[16]：

$$l(y_i, f(\boldsymbol{x}_i)) = \begin{cases} -y_i f(\boldsymbol{x}_i)(\equiv -y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)) & y_i f(\boldsymbol{x}_i) < 0 \\ 0 & y_i f(\boldsymbol{x}_i) > 0 \end{cases} \tag{58}$$

[15] 当然，仍然需要保留正确分类时函数值为 0 的情形。

[16] 实际上，感知机损失函数的核心函数可以表示为：

$$[z]_+ = \begin{cases} z & z > 0 \\ 0 & z \leqslant 0 \end{cases} \tag{57}$$

其中，下标 $+$ 表示当 $z > 0$ 时，函数值取 z；否则，函数值为 0。

可以看出，感知机损失函数在误分类 (即 $y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)<0$) 时，能够提供梯度信息。而 0-1 损失函数在误分类时，无法提供梯度信息，不能使学习算法进行有效的优化。

对于上述的感知机损失函数，也可以从几何的角度来理解：

- 对于分类正确的样本点，其损失或代价为 0；
- 对于分类错误的样本点，令 $\|\boldsymbol{w}\|=1$(单位向量)，则其损失或代价为该样本点到决策面 $\boldsymbol{w}^T\boldsymbol{x}+b=0$ 的 (无符号) 距离：

$$-y_i\frac{(\boldsymbol{w}^T\boldsymbol{x}_i+b)}{\|\boldsymbol{w}\|}\equiv -y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b) \tag{59}$$

其中，$\frac{(\boldsymbol{w}^T\boldsymbol{x}_i+b)}{\|\boldsymbol{w}\|}$ 表示样本点 $\boldsymbol{x}_i$ 到决策面 $\boldsymbol{w}^T\boldsymbol{x}+b=0$ 的有符号距离；对于误分类样本点 $\boldsymbol{x}_i$，$-y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)>0$ 成立，它表示 (无符号) 距离 ($\|\boldsymbol{w}\|=1$)。

于是，对于给定的数据集 T，假设误分类点的集合为 M，那么所有误分类点到决策面 $\boldsymbol{w}^T\boldsymbol{x}+b$ 的总距离为：

$$L(\boldsymbol{w},b)=-\sum_{\boldsymbol{x}_i\in M}y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b) \tag{60}$$

其中 $\|\boldsymbol{w}\|=1$。上式就是感知机模型的损失函数。

感知机的学习策略是从假设空间 $\{f|f(\boldsymbol{x})=\boldsymbol{w}^T\boldsymbol{x}+b\}$ 中选取使损失函数 (公式 (60)) 最小的模型参数 $\boldsymbol{w}$ 和 b。于是，将感知机模型的参数学习问题变成了一个最优化问题。

3.5.2 学习的原始算法

在损失函数表示为公式 (60) 时，感知机模型的参数学习问题可以表示为如下的最优化问题：

$$\min_{\boldsymbol{w},b}L(\boldsymbol{w},b)=\min_{\boldsymbol{w},b}-\sum_{\boldsymbol{x}_i\in M}y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b) \tag{61}$$

其中，M 为误分类样本点的集合。

对于误分类样本点的集合 M，损失函数 $L(\boldsymbol{w},b)$ 关于权值参数 $\boldsymbol{w}$ 和偏置参数 b 的梯度可分别表示为：

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{w}}&=-\sum_{\boldsymbol{x}_i\in M}y_i\boldsymbol{x}_i\\ \frac{\partial L}{\partial b}&=-\sum_{\boldsymbol{x}_i\in M}y_i\end{aligned} \tag{62}$$

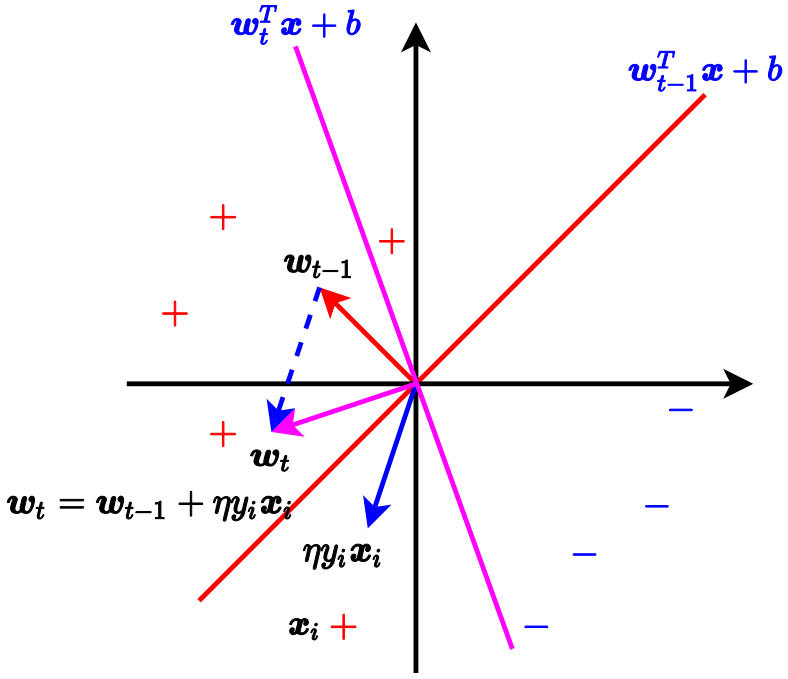


图 3-9: 权值向量更新公式的几何解释

对于这些误分类点，可以应用梯度下降方法，对初始权值向量 $\boldsymbol{w}_0$ 和偏置 b_0 进行迭代更新，直至其收敛为止，从而得到最优权值向量 $\boldsymbol{w}^*$ 和偏置 b^*。稍后将证明，在数据集线性可分的情况下，感知机学习算法将确保收敛。易知，感知机模型的学习是由误分类样本点驱动的。

在实际应用中，一般不使用误分类点集合 M 执行批梯度下降[17]，而是使用随机梯度下降对权值向量与偏置进行更新。具体而言，权值向量 $\boldsymbol{w}$ 和偏置 b 的更新公式如下：

$$\begin{aligned}\boldsymbol{w}_t &= \boldsymbol{w}_{t-1}+\eta y_i\boldsymbol{x}_i\\ b_t &= b_{t-1}+\eta y_i\end{aligned} \tag{63}$$

其中，η 为学习率，$(\boldsymbol{x}_i, y_i)$ 为误分类样本点对。上述权值向量与偏置的更新公式，具有很好的几何直观性与解释性，如图3-9所示。图中，红直线表示 $t-1$ 时刻的决策面，紫直线表示 t 时刻的决策面。当一个实例点 $\boldsymbol{x}_i$ 被误分类时，学习算法将调整 $\boldsymbol{w}$ 和 b(图中只展示了权值向量 $\boldsymbol{w}$ 的更新)，使决策面向误分类点 $\boldsymbol{x}_i$ 的一侧旋转和移动，以减少该误分类点与决策面的距离，或使得新决策面能够正确地识

[17]在批梯度下降法中，每次迭代时，需要执行一次遍历，以找出误分类点集合 M。因此，使用随机梯度法替代。

别误分类点[18]，从而降低了总体损失。上述模型参数的更新过程，迭代进行，一直进行到没有误分类点出现为止，算法停止。需要注意的是，感知机模型的最优参数 $\boldsymbol{w}^*$ 和 b^* 不具有唯一性，它们与初始值有很大的关系。

感知机学习算法有 2 种形式：原始形式与对偶形式。下面，首先给出感知机学习算法的原始形式。

算法 **3.1**（感知机学习算法的原始形式）

```
Input:
      Dataset: T
      Learning Rate: eta
Output:
      w, b
def train(T, eta):
    w, b = 0, 0
    finish = False
    while not finish:
        finish = True
        for (xi, yi) in T:
            if yi * (w * xi + b) <= 0:
                w = w + eta * yi * xi
                b = b + eta * yi
                finish = False
        if finish:
            return w, b
```

3.5.3 算法的收敛性

下面将证明，在数据集线性可分的情况下，感知机的学习算法经过有限次迭代后，参数收敛，算法停止，且能够正确地将数据集进行分类。

为了便于表达，将决策面 $\boldsymbol{w}^T\boldsymbol{x}+b$ 写成如下形式：

$$\hat{\boldsymbol{w}}^T\hat{\boldsymbol{x}} \equiv \boldsymbol{w}^T\boldsymbol{x}+b \tag{64}$$

即令 $\hat{\boldsymbol{w}}=\left(\boldsymbol{w}^T,b\right)^T$，$\hat{\boldsymbol{x}}=\left(\boldsymbol{x}^T,1\right)^T$。

定理 **3.1**（Novikoff收敛定理）设数据集 T：

$$\{(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)\},$$

是线性可分的，其中 $\boldsymbol{x}_i\in\mathbf{R}^n$，$y_i=\{+1,-1\}$，$i=1,2,\ldots,N$，则：

[18]当然，在调整后，可能会使得原本正确分类的样本点被误分类。但是，整体损失或代价在不断地降低。

1. 一定存在决策面 $\hat{\boldsymbol{w}}_*^T\hat{\boldsymbol{x}}=0$(其中 $\|\hat{\boldsymbol{w}}_*\|=1$)，它能够完全正确地将数据集分开，且存在 $\gamma>0$，下式成立：

$$y_i\left(\hat{\boldsymbol{w}}_*^T\hat{\boldsymbol{x}}_i\right)\geqslant\gamma\quad\forall i,i=1,2,\ldots,N \tag{65}$$

2. 令 $R=\max\limits_{1\leqslant i\leqslant N}\|\hat{\boldsymbol{x}}_i\|$，则感知机学习算法在训练数据集 T 上的误分类次数 k 满足不等式：

$$k\leqslant\left(\frac{R}{\gamma}\right)^2 \tag{66}$$

证明：

1. 由于数据集 T 是线性可分的，根据定义3.1可知，存在某个决策面 (令其为 $\hat{\boldsymbol{w}}_*^T\hat{\boldsymbol{x}}=0$，且 $\|\hat{\boldsymbol{w}}_*\|=1$)，能够完全正确地将数据集分类，且对于 $\forall(\hat{\boldsymbol{x}}_i,y_i)(i=1,2,\ldots,N)$，$y_i\hat{\boldsymbol{w}}_*^T\hat{\boldsymbol{x}}_i>0$ 成立，于是令 $\gamma=\min\limits_{(\hat{\boldsymbol{x}}_i,y_i)}y_i\hat{\boldsymbol{w}}_*^T\hat{\boldsymbol{x}}_i$，则 $y_i\hat{\boldsymbol{w}}_*^T\hat{\boldsymbol{x}}_i\geqslant\gamma(\forall i,i=1,2,\ldots,N)$ 成立。

2. 设误分类样本点为 $(\hat{\boldsymbol{x}}_i,y_i)$，根据感知机学习算法的原始形式 (算法3.1) 可知，此时的权值需要更新，其更新公式如下：

$$\begin{aligned}&\boldsymbol{w}_k=\boldsymbol{w}_{k-1}+\eta y_i\boldsymbol{x}_i\\&b_k=b_{k-1}+\eta y_i\end{aligned}\quad\Rightarrow\hat{\boldsymbol{w}}_k=\hat{\boldsymbol{w}}_{k-1}+\eta y_i\hat{\boldsymbol{x}}_i \tag{67}$$

其中，误分类点 $(\hat{\boldsymbol{x}}_i,y_i)$ 满足条件 $y_i\hat{\boldsymbol{w}}_{k-1}^T\hat{\boldsymbol{x}}_i\leqslant0$；且当 $k=0$ 时，有 $\hat{\boldsymbol{w}}_0=\boldsymbol{0}$。

对公式 (67) 两端乘以 $\hat{\boldsymbol{w}}_*^T$，可得：

$$\hat{\boldsymbol{w}}_*^T\hat{\boldsymbol{w}}_k=\hat{\boldsymbol{w}}_*^T\hat{\boldsymbol{w}}_{k-1}+\eta y_i\hat{\boldsymbol{w}}_*^T\hat{\boldsymbol{x}}_i \tag{68}$$

将 $y_i\hat{\boldsymbol{w}}_*^T\hat{\boldsymbol{x}}_i\geqslant\gamma$ 代入上式，得到不等式：

$$\hat{\boldsymbol{w}}_*^T\hat{\boldsymbol{w}}_k\geqslant\hat{\boldsymbol{w}}_*^T\hat{\boldsymbol{w}}_{k-1}+\eta\gamma \tag{69}$$

于是，对上述不等式中的 $\hat{\boldsymbol{w}}_*^T\hat{\boldsymbol{w}}_{k-1}$ 继续应用类似的过程，并一直持续到 $\hat{\boldsymbol{w}}_0=\boldsymbol{0}$ 为止，可得：

$$\hat{\boldsymbol{w}}_*^T\hat{\boldsymbol{w}}_k\geqslant\hat{\boldsymbol{w}}_*^T\hat{\boldsymbol{w}}_{k-1}+\eta\gamma\geqslant\hat{\boldsymbol{w}}_*^T\hat{\boldsymbol{w}}_{k-2}+2\eta\gamma\geqslant\ldots\geqslant k\eta\gamma \tag{70}$$

由此，得到不等式：

$$k\eta\gamma\leqslant\hat{\boldsymbol{w}}_*^T\hat{\boldsymbol{w}}_k \tag{71}$$

利用 $\|\hat{\boldsymbol{w}}_*\| = 1$，可得：

$$k\eta\gamma \leqslant \hat{\boldsymbol{w}}_*^T \hat{\boldsymbol{w}}_k = \|\hat{\boldsymbol{w}}_*\|\|\hat{\boldsymbol{w}}_k\| \cos\theta \leqslant \|\hat{\boldsymbol{w}}_*\|\|\hat{\boldsymbol{w}}_k\| = \|\hat{\boldsymbol{w}}_k\| \tag{72}$$

其中，θ 为 $\hat{\boldsymbol{w}}_*$ 和 $\hat{\boldsymbol{w}}_k$ 的夹角。继续对上式变形，可得：

$$k\eta\gamma \leqslant \|\hat{\boldsymbol{w}}_k\| = \sqrt{\hat{\boldsymbol{w}}_k^T \hat{\boldsymbol{w}}_k} \tag{73}$$

将 $\hat{\boldsymbol{w}}_k = \hat{\boldsymbol{w}}_{k-1} + \eta y_i \hat{\boldsymbol{x}}_i$ 代入，可得：

$$\begin{aligned} k\eta\gamma &\leqslant \sqrt{\left(\hat{\boldsymbol{w}}_{k-1} + \eta y_i \hat{\boldsymbol{x}}_i\right)^T \left(\hat{\boldsymbol{w}}_{k-1} + \eta y_i \hat{\boldsymbol{x}}_i\right)} \\ &= \sqrt{\hat{\boldsymbol{w}}_{k-1}^T \hat{\boldsymbol{w}}_{k-1} + 2\eta y_i \hat{\boldsymbol{w}}_{k-1}^T \hat{\boldsymbol{x}}_i + \eta^2 y_i^2 \hat{\boldsymbol{x}}_i^T \hat{\boldsymbol{x}}_i} \end{aligned} \tag{74}$$

利用 $y_i^2 = 1$，继续变形，可得：

$$\begin{aligned} k\eta\gamma &\leqslant \sqrt{\hat{\boldsymbol{w}}_{k-1}^T \hat{\boldsymbol{w}}_{k-1} + 2\eta y_i \hat{\boldsymbol{w}}_{k-1}^T \hat{\boldsymbol{x}}_i + \eta^2 \hat{\boldsymbol{x}}_i^T \hat{\boldsymbol{x}}_i} \\ &= \sqrt{\hat{\boldsymbol{w}}_{k-1}^T \hat{\boldsymbol{w}}_{k-1} + 2\eta y_i \hat{\boldsymbol{w}}_{k-1}^T \hat{\boldsymbol{x}}_i + \eta^2 \|\hat{\boldsymbol{x}}_i\|^2} \\ &\leqslant \sqrt{\hat{\boldsymbol{w}}_{k-1}^T \hat{\boldsymbol{w}}_{k-1} + \eta^2 \|\hat{\boldsymbol{x}}_i\|^2} \\ &\leqslant \sqrt{\hat{\boldsymbol{w}}_{k-1}^T \hat{\boldsymbol{w}}_{k-1} + \eta^2 R^2} \end{aligned} \tag{75}$$

其中，$R = \max\limits_{1 \leqslant i \leqslant N} \|\hat{\boldsymbol{x}}_i\|$。于是，对上述不等式中的 $\hat{\boldsymbol{w}}_{k-1}^T \hat{\boldsymbol{w}}_{k-1}$ 继续应用类似的过程，并一直持续到 $\hat{\boldsymbol{w}}_0 = \boldsymbol{0}$ 为止，可得：

$$k\eta\gamma \leqslant \sqrt{\hat{\boldsymbol{w}}_{k-1}^T \hat{\boldsymbol{w}}_{k-1} + \eta^2 R^2} \leqslant \sqrt{\hat{\boldsymbol{w}}_{k-2}^T \hat{\boldsymbol{w}}_{k-2} + 2\eta^2 R^2} \leqslant \ldots \leqslant \sqrt{k\eta^2 R^2} = \sqrt{k}\eta R \tag{76}$$

最后，得到：

$$k \leqslant \left(\frac{R}{\gamma}\right)^2 \tag{77}$$

可以看出，对于数据集线性可分情形，误分类的次数 k(即权值向量 $\hat{\boldsymbol{w}}$ 的调整次数) 是有上界的，感知机学习算法的原始形式 (算法3.1) 必定收敛。

□

3.5.4 学习的对偶算法

感知机的学习算法，除了原始形式之外，还存在另一种形式——对偶算法。在原始算法中，每次迭代直接更新权值 $\boldsymbol{w}$ 和偏置 b。而在对偶算法中，首先为每个

样本点配置一个标量 α_i，用来记录每个样本点被误分的“次数”[19]；然后，无论是 $\boldsymbol{w}$，还是 b，都可以使用 $\boldsymbol{\alpha}$ 进行 (间接) 更新。其中，向量 $\boldsymbol{\alpha}$ 的元素由 α_i 组成 $(i = 1, 2, \ldots, N)$。

对偶算法的优点在于：

- 将 $\boldsymbol{w}$ 和 b 表示成样本点对 $(\boldsymbol{x}_i, y_i)$ 的线性组合形式，为核函数的引入提供了基础；
- 在每次迭代的过程中，数据样本点之间仅以向量内积 $\boldsymbol{x}_i^T \boldsymbol{x}_j$ 的形式出现。因此，可以预先计算它们之间的内积，从而形成了 Gram 矩阵：

$$\boldsymbol{G} = \left[\boldsymbol{x}_i^T \boldsymbol{x}_j\right]_{N \times N} \tag{78}$$

下面将讨论，如何从原始算法得到等价的对偶算法。在原始算法3.1中，权值向量与偏置的更新公式分别如下：

$$\begin{aligned} \boldsymbol{w} &\leftarrow \boldsymbol{w} + \eta y_i \boldsymbol{x}_i \\ b &\leftarrow b + \eta y_i \end{aligned} \tag{79}$$

其中，$(\boldsymbol{x}_i, y_i)$ 为误分类点对。仔细观察，可以发现，每次 $(\boldsymbol{x}_i, y_i)$ 被误分类时，权值 $\boldsymbol{w}$ 和偏置 b 就分别执行一次更新 $\eta y_i \boldsymbol{x}_i$ 和 ηy_i。除去样本点对 $(\boldsymbol{x}_i, y_i)$ 之外，更新量只与 η 有关。因此，可令 n_i 表示样本点对 $(\boldsymbol{x}_i, y_i)$ 的更新次数[20]，并令 $\alpha_i = n_i \eta$，于是得到公式 (79) 的等价形式：

$$\begin{aligned} \boldsymbol{w} &= \sum_{i=1}^{N} n_i \eta y_i \boldsymbol{x}_i \\ b &= \sum_{i=1}^{N} n_i \eta y_i \end{aligned} \quad \Rightarrow \quad \begin{aligned} \boldsymbol{w} &= \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i \\ b &= \sum_{i=1}^{N} \alpha_i y_i \end{aligned} \tag{80}$$

于是，在迭代过程中，对偶算法只需要对 α_i 的值进行更新：

$$\alpha_i \leftarrow \alpha_i + \eta \tag{81}$$

[19]注意，“次数”与学习率 η 的取值有关：当 $\eta = 1$ 时，α_i 就是第 i 个样本点的误分次数；否则，如果 $0 < \eta < 1$，则 $\frac{\alpha_i}{\eta}$ 才是误分次数。综合两种情况，$\frac{\alpha_i}{\eta}$ 就表示误分次数。

[20]注意，在算法3.1中，$\boldsymbol{w}_0 = \boldsymbol{0}$ 和 $b_0 = 0$。

前提条件是，当样本点对 $(\boldsymbol{x}_i, y_i)$ 是误分类点，即满足如下条件时：

$$\begin{aligned} & y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + b\right) \leqslant 0 \quad \Rightarrow \\ & y_i\left(\boldsymbol{x}_i^T\boldsymbol{w} + b\right) \leqslant 0 \quad \Rightarrow \\ & y_i\left(\boldsymbol{x}_i^T\sum_{j=1}^{N}\alpha_j y_j \boldsymbol{x}_j + \sum_{j=1}^{N}\alpha_j y_j\right) \leqslant 0 \quad \Rightarrow \\ & y_i\left(\sum_{j=1}^{N}\alpha_j y_j \boldsymbol{x}_j^T\boldsymbol{x}_i + \sum_{j=1}^{N}\alpha_j y_j\right) \leqslant 0 \end{aligned} \tag{82}$$

α_i 的意义也是非常明确的：其值越大，表明样本点对 $(\boldsymbol{x}_i, y_i)$ 被误分类的次数越多，它离决策面也越近，对决策面的形成有较大的影响[21]；相反，那些 $\alpha_i = 0$ 的点，离决策面较远，对决策面的形成没有任何影响。

下面给出感知机学习的对偶算法。

算法 **3.2**（感知机学习算法的对偶形式）

```
Input:
      Dataset: T=(x, y), Vector: x, Vector: y
      Learning Rate: eta
Output:
      Vector: alpha
def train(T, eta):
    alpha = 0
    finish = False
    while not finish:
        finish = True
        for (xi, yi) in T:
            wxi = sum(alpha[j] * y[j] * dot(x[j], xi)) for all j
            b = sum(alpha[j] * y[j]) for all j
            if yi * (wxi + b) <= 0:
                alpha[i] = alpha[i] + eta
                finish = False
        if finish:
            return alpha
```

4 附录

4.1 标量函数对矩阵的求导

标量函数 $f(\boldsymbol{X})$ 对矩阵 $\boldsymbol{X}$ 的导数，定义为：

$$\frac{\partial f}{\partial \boldsymbol{X}} = \left[\frac{\partial f}{\partial X_{ij}}\right] \tag{83}$$

[21]在支持向量机中，这样的样本点被称为支持向量。

即 $f(\boldsymbol{X})$ 对 $\boldsymbol{X}$ 逐元素求导，且排列成与矩阵 $\boldsymbol{X}$ 各维度相同的矩阵。在实际应用中，可以利用一些基本的运算规则，进行有效的推导，下面将逐一进行介绍。

为此，首先需要定义一些基本的操作与运算规则。在推导的过程中，需要利用微分与导数或梯度的关系。在一元微积分中，标量函数 $f(x)$ 对标量变量 x 的导数为 $f'(x)$，其与微分 df 的关系如下：

$$df = f'(x)dx \tag{84}$$

在多元微积分中，梯度[22]与微分的关系如下：

$$df = \sum_i \frac{\partial f}{\partial x_i} dx_i = \frac{\partial f}{\partial \boldsymbol{x}}^T d\boldsymbol{x} \tag{85}$$

其中，df 表示全微分，$\frac{\partial f}{\partial \boldsymbol{x}}$ 为偏导数向量，$d\boldsymbol{x}$ 为微分向量，$df = \frac{\partial f}{\partial \boldsymbol{x}}^T d\boldsymbol{x}$ 表示全微分等于两个向量的内积。

类似地，也可以在全微分 df 与标量函数 $f(x)$ 对矩阵 $\boldsymbol{X}$ 的导数之间建立联系：

$$df = \sum_{i=1}^{m}\sum_{j=1}^{n} \frac{\partial f}{\partial X_{ij}} dX_{ij} = \mathrm{tr}\left(\frac{\partial f}{\partial \boldsymbol{X}}^T d\boldsymbol{X}\right) \tag{86}$$

其中，tr 表示迹运算[23]。全微分 df 等于 $\frac{\partial f}{\partial \boldsymbol{X}}$ 和 $d\boldsymbol{X}$ 这 2 个矩阵的标准内积，上式表明，它可以方便地使用迹来表示。

下面是关于全微分的一些基本运算：

- $d(\boldsymbol{X} \pm \boldsymbol{Y}) = d\boldsymbol{X} \pm d\boldsymbol{Y}$；
- $d(\boldsymbol{XY}) = d(\boldsymbol{X})\boldsymbol{Y} + \boldsymbol{X}d\boldsymbol{Y}$；
- $d\left(\boldsymbol{X}^T\right) = (d\boldsymbol{X})^T$；
- $d\mathrm{tr}(\boldsymbol{X}) = \mathrm{tr}(d\boldsymbol{X})$；

[22]即标量函数对向量的导数或标量函数对向量各分量的偏导数。

[23]迹运算表示方阵主对角线上的元素之和，这是一种应用广泛的运算。其应用之一是，定义矩阵 $\boldsymbol{X}, \boldsymbol{Y} \in \mathbf{R}^{m\times n}$ 的标准内积为：

$$\langle \boldsymbol{X}, \boldsymbol{Y} \rangle = \mathrm{tr}\left(\boldsymbol{X}^T\boldsymbol{Y}\right) = \sum_{i=1}^{m}\sum_{j=1}^{n} X_{ij}Y_{ij} \tag{87}$$

注意，如果 $\boldsymbol{A}$ 与 $\boldsymbol{B}$ 符合矩阵乘法规则，则 $\mathrm{tr}(\boldsymbol{AB}) = \mathrm{tr}(\boldsymbol{BA})$ 成立。如果 $\boldsymbol{A}$、$\boldsymbol{B}$ 与 $\boldsymbol{C}$ 均为方阵，则 $\mathrm{tr}(\mathbf{ABC}) = \mathrm{tr}(\mathbf{BCA}) = \mathrm{tr}(\mathbf{CAB})$ 成立。请参考《机器学习》课程系列之《基础知识》，Chapter1-CN.pdf。

- $d\boldsymbol{X}^{-1} = -\boldsymbol{X}^{-1}d\boldsymbol{X}\boldsymbol{X}^{-1}$；[24]
- $d|\boldsymbol{X}| = \mathrm{tr}(\boldsymbol{X}^* d\boldsymbol{X})$，其中 $|\boldsymbol{X}|$ 表示 n 阶矩阵 $\boldsymbol{X}$ 的行列式；$\boldsymbol{X}^*$ 表示 $\boldsymbol{X}$ 的伴随矩阵，当 $\boldsymbol{X}$ 可逆时，$\boldsymbol{X}^* = |\boldsymbol{X}|\boldsymbol{X}^{-1}$，则 $d|\boldsymbol{X}| = |\boldsymbol{X}|\mathrm{tr}(\boldsymbol{X}^{-1}d\boldsymbol{X})$；[25]
- $d(\boldsymbol{X}\odot\boldsymbol{Y}) = d\boldsymbol{X}\odot\boldsymbol{Y} + \boldsymbol{X}\odot d\boldsymbol{Y}$，$\odot$ 表示 Hadamard 积 (Hadamard Product)，即两个矩阵对应元素的积，这两个矩阵的维度必须相同，结果矩阵的维度与原矩阵保持一致；
- $d\sigma(\boldsymbol{X}) = \sigma'(\boldsymbol{X})\odot d\boldsymbol{X}$，其中 $\sigma(\boldsymbol{X}) = [\sigma(X_{ij})]$ 是对矩阵 $\boldsymbol{X}$ 的每个元素进行逐元素运算的标量函数，$\sigma'(\boldsymbol{X}) = [\sigma'(X_{ij})]$ 是与 $\sigma(\boldsymbol{X})$ 对应的逐元素求导的标量函数。例如：

$$\boldsymbol{X} = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}, d\sin(\boldsymbol{X}) = \begin{bmatrix} \cos X_{11}dX_{11} & \cos X_{12}dX_{12} \\ \cos X_{21}dX_{21} & \cos X_{22}dX_{22} \end{bmatrix} = \cos(\boldsymbol{X})\odot d\boldsymbol{X}$$

现在，可以利用全微分的迹运算公式 (86) 来获取标量函数 $f(\boldsymbol{X})$ 对矩阵 $\boldsymbol{X}$ 的导数：通过迹运算，在导数 $\frac{\partial f}{\partial \boldsymbol{X}}$、微分 $d\boldsymbol{X}$ 及全微分 df 之间建立联系，将全微分变换成该公式的右侧形式，这样就直接获得了导数 $\frac{\partial f}{\partial \boldsymbol{X}}$ 的整体形式，而不需要依据其定义 (公式 (83)) 进行低效率的逐元素求解。

为了应用上述策略，需要使用下面的迹技巧 (Trace Trick)：

- $a = \mathrm{tr}(a)$；
- $\mathrm{tr}(\boldsymbol{X}^T) = \mathrm{tr}(\boldsymbol{X})$；
- $\mathrm{tr}(\boldsymbol{X}\pm\boldsymbol{Y}) = \mathrm{tr}(\boldsymbol{X})\pm\mathrm{tr}(\boldsymbol{Y})$；
- $\mathrm{tr}(\boldsymbol{X}\boldsymbol{Y}) = \mathrm{tr}(\boldsymbol{Y}\boldsymbol{X})$，其中 $\boldsymbol{X}$ 与 $\boldsymbol{Y}$ 符合矩阵乘法规则，运算的结果是 $\sum_{i,j} X_{ij}Y_{ji}$；
- $\mathrm{tr}\left(\boldsymbol{X}^T(\boldsymbol{Y}\odot\boldsymbol{Z})\right) = \mathrm{tr}\left((\boldsymbol{X}\odot\boldsymbol{Y})^T\boldsymbol{Z}\right)$，其中，$\boldsymbol{X}$、$\boldsymbol{Y}$、$\boldsymbol{Z}$ 的维度都相同，运算的结果是 $\sum_{i,j} X_{ij}Y_{ij}Z_{ij}$；

[24] $\boldsymbol{X}\boldsymbol{X}^{-1} = \boldsymbol{I} \Rightarrow d(\boldsymbol{X}\boldsymbol{X}^{-1}) = d\boldsymbol{I} \Rightarrow (d\boldsymbol{X})\boldsymbol{X}^{-1} + \boldsymbol{X}d\boldsymbol{X}^{-1} = \boldsymbol{0} \Rightarrow d\boldsymbol{X}^{-1} = -\boldsymbol{X}^{-1}d\boldsymbol{X}\boldsymbol{X}^{-1}$。

[25] 可以使用 Laplace 展开证明，详见张贤达《矩阵分析与应用》第 279 页。

综上所述，可以直接利用公式 (85) 和公式 (86)，通过适当的变换，就可以直接求解出标量函数关于向量的偏导数以及标量函数关于矩阵的偏导数。这些适当的变换包括上述关于微分与迹的加减乘法、逆、行列式、逐元素运算函数等，也包括将迹运算应用于全微分 df 并利用迹中矩阵乘积的可交换性，将 $d\boldsymbol{X}$ 右侧项交换至左侧等技巧，最后比照全微分公式，就可以直接获得所需的偏导数。

需要注意的是，在求解偏导数的过程中，经常遇到的一种求导问题是所谓的复合求导。例如，设 $f(\boldsymbol{Y})$ 是矩阵 $\boldsymbol{Y}$ 的函数，而矩阵 $\boldsymbol{Y}$ 又是矩阵 $\boldsymbol{X}$ 的函数。假设已经求解出 $\frac{\partial f}{\partial \boldsymbol{Y}}$，那么如何求解出 $\frac{\partial f}{\partial \boldsymbol{X}}$？

求解 $\frac{\partial f}{\partial \boldsymbol{X}}$，不能简单地套用标量函数关于标量变量求导的链式法则。但是，通过函数嵌套的方式来使用全微分公式 ((85) 和 (86))，经过适当的变换与变形，最终也能够得到偏导数。例如，在公式 $df = \mathrm{tr}\left(\frac{\partial f}{\partial \boldsymbol{Y}}^T d\boldsymbol{Y}\right)$ 中，对其中的 $d\boldsymbol{Y}$ 再次应用全微分公式 $d\boldsymbol{Y} = \mathrm{tr}\left(\frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{X}}^T d\boldsymbol{X}\right)$，并将后者代入前者，最后使用迹技巧将 $d\boldsymbol{X}$ 右侧所有项交换到左侧，即可得到 $\frac{\partial f}{\partial \boldsymbol{X}}$。

例如，已知 $\frac{\partial f}{\partial \boldsymbol{Y}}$，$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}\boldsymbol{B}$，求解 $\frac{\partial f}{\partial \boldsymbol{X}}$ 的过程如下：

$$\begin{aligned} df &= \mathrm{tr}\left(\frac{\partial f}{\partial \boldsymbol{Y}}^T d\boldsymbol{Y}\right) = \mathrm{tr}\left(\frac{\partial f}{\partial \boldsymbol{Y}}^T \boldsymbol{A} d\boldsymbol{X} \boldsymbol{B}\right) = \mathrm{tr}\left(\boldsymbol{B}\frac{\partial f}{\partial \boldsymbol{Y}}^T \boldsymbol{A} d\boldsymbol{X}\right) \\ &= \mathrm{tr}\left(\left(\boldsymbol{A}^T \frac{\partial f}{\partial \boldsymbol{Y}} \boldsymbol{B}^T\right)^T d\boldsymbol{X}\right) \end{aligned} \tag{88}$$

在上面的推导过程中，使用了如下推导：

$$d\boldsymbol{Y} = (d\boldsymbol{A})\boldsymbol{X}\boldsymbol{B} + \boldsymbol{A}d\boldsymbol{X}\boldsymbol{B} + \boldsymbol{A}\boldsymbol{X}d\boldsymbol{B} = \boldsymbol{A}d\boldsymbol{X}\boldsymbol{B} \tag{89}$$

其中，$\boldsymbol{A}$ 和 $\boldsymbol{B}$ 均为常量。最后，得到 $\frac{\partial f}{\partial \boldsymbol{X}} = \boldsymbol{A}^T \frac{\partial f}{\partial \boldsymbol{Y}} \boldsymbol{B}^T$。

下面，将列举若干实例，应用上述原则求解所需偏导数：

- 已知 $f(\boldsymbol{X}) = \boldsymbol{a}^T\boldsymbol{X}\boldsymbol{b}$，其中 $\boldsymbol{a}$ 和 $\boldsymbol{b}$ 分别为 m 维和 n 维列向量，$\boldsymbol{X}$ 为 $m \times n$ 维矩阵，f 为标量函数，求解 $\frac{\partial f}{\partial \boldsymbol{X}}$：

$$\begin{aligned} &f(\boldsymbol{X}) = \boldsymbol{a}^T\boldsymbol{X}\boldsymbol{b} \quad \Rightarrow \quad df = d(\boldsymbol{a}^T\boldsymbol{X}\boldsymbol{b}) \quad \Rightarrow \\ &df = d\boldsymbol{a}^T\boldsymbol{X}\boldsymbol{b} + \boldsymbol{a}^T d\boldsymbol{X}\boldsymbol{b} + \boldsymbol{a}^T\boldsymbol{X}d\boldsymbol{b} = \boldsymbol{a}^T d\boldsymbol{X}\boldsymbol{b} \quad \Rightarrow \\ &df = \mathrm{tr}(df) = \mathrm{tr}(\boldsymbol{a}^T d\boldsymbol{X}\boldsymbol{b}) = \mathrm{tr}(\boldsymbol{b}\boldsymbol{a}^T d\boldsymbol{X}) = \mathrm{tr}\left(\left(\boldsymbol{a}\boldsymbol{b}^T\right)^T d\boldsymbol{X}\right) \end{aligned} \tag{90}$$

可得 $\frac{\partial f}{\partial \boldsymbol{X}} = \boldsymbol{a}\boldsymbol{b}^T$。

- 已知 $f(\boldsymbol{X})=\boldsymbol{a}^T e^{\boldsymbol{Xb}}$，其中 $\boldsymbol{a}$ 和 $\boldsymbol{b}$ 分别为 m 维和 n 维列向量，$\boldsymbol{X}$ 为 $m\times n$ 维矩阵，f 为标量函数，此处的指数函数 e 执行逐元素运算，求解 $\frac{\partial f}{\partial \boldsymbol{X}}$：

$$\begin{aligned}
&f(\boldsymbol{X})=\boldsymbol{a}^T e^{\boldsymbol{Xb}} \quad \Rightarrow \quad df=d(\boldsymbol{a}^T e^{\boldsymbol{Xb}}) \quad \Rightarrow \\
&df=d\boldsymbol{a}^T e^{\boldsymbol{Xb}}+\boldsymbol{a}^T d\left(e^{\boldsymbol{Xb}}\right)=\boldsymbol{a}^T\left(e^{\boldsymbol{Xb}}\odot d(\boldsymbol{Xb})\right) \quad \Rightarrow \\
&df=\mathrm{tr}(df)=\mathrm{tr}\left(\boldsymbol{a}^T\left(e^{\boldsymbol{Xb}}\odot d(\boldsymbol{Xb})\right)\right)=\mathrm{tr}\left(\left(\boldsymbol{a}\odot e^{\boldsymbol{Xb}}\right)^T d(\boldsymbol{Xb})\right) \\
&\quad =\mathrm{tr}\left(\boldsymbol{b}\left(\boldsymbol{a}\odot e^{\boldsymbol{Xb}}\right)^T d\boldsymbol{X}\right)=\mathrm{tr}\left(\left(\left(\boldsymbol{a}\odot e^{\boldsymbol{Xb}}\right)\boldsymbol{b}^T\right)^T d\boldsymbol{X}\right)
\end{aligned} \tag{91}$$

  可得 $\frac{\partial f}{\partial \boldsymbol{X}}=\left(\boldsymbol{a}\odot e^{\boldsymbol{Xb}}\right)\boldsymbol{b}^T$

- 已知 $f(\boldsymbol{Y})=\mathrm{tr}(\boldsymbol{Y}^T\boldsymbol{AY})$，$\boldsymbol{Y}=\sigma(\boldsymbol{BX})$，其中 $\boldsymbol{Y}$ 为 $l\times n$ 矩阵，$\boldsymbol{A}$ 为 $l\times l$ 对称矩阵，$\boldsymbol{B}$ 为 $l\times m$ 矩阵，$\boldsymbol{X}$ 为 $m\times n$ 矩阵，σ 为逐元素运算函数，求解 $\frac{\partial f}{\partial \boldsymbol{X}}$：

$$\begin{aligned}
&f(\boldsymbol{Y})=\mathrm{tr}(\boldsymbol{Y}^T\boldsymbol{AY}) \quad \Rightarrow \quad df=d\left(\mathrm{tr}(\boldsymbol{Y}^T\boldsymbol{AY})\right) \quad \Rightarrow \\
&df=\mathrm{tr}\left(d\left(\boldsymbol{Y}^T\boldsymbol{AY}\right)\right)=\mathrm{tr}\left(d\left(\boldsymbol{Y}^T\right)\boldsymbol{AY}+\boldsymbol{Y}^T d\left(\boldsymbol{AY}\right)\right) \\
&\quad =\mathrm{tr}\left((d\boldsymbol{Y})^T\boldsymbol{AY}+\boldsymbol{Y}^T\boldsymbol{A}d\boldsymbol{Y}\right)=\mathrm{tr}\left((d\boldsymbol{Y})^T\boldsymbol{AY}\right)+\mathrm{tr}\left(\boldsymbol{Y}^T\boldsymbol{A}d\boldsymbol{Y}\right) \\
&\quad =\mathrm{tr}\left((\boldsymbol{AY})^T d\boldsymbol{Y}\right)+\mathrm{tr}\left(\boldsymbol{Y}^T\boldsymbol{A}d\boldsymbol{Y}\right)=\mathrm{tr}\left(\boldsymbol{Y}^T\boldsymbol{A}^T d\boldsymbol{Y}\right)+\mathrm{tr}\left(\boldsymbol{Y}^T\boldsymbol{A}d\boldsymbol{Y}\right) \\
&\quad =2\mathrm{tr}\left(\boldsymbol{Y}^T\boldsymbol{A}d\boldsymbol{Y}\right)=2\mathrm{tr}\left((\sigma(\boldsymbol{BX}))^T\boldsymbol{A}d\left(\sigma(\boldsymbol{BX})\right)\right) \\
&\quad =2\mathrm{tr}\left((\sigma(\boldsymbol{BX}))^T\boldsymbol{A}\left(\sigma'(\boldsymbol{BX})\odot\boldsymbol{B}d\boldsymbol{X}\right)\right) \\
&\quad =2\mathrm{tr}\left(\left(\boldsymbol{A}^T\sigma(\boldsymbol{BX})\odot\sigma'(\boldsymbol{BX})\right)^T\boldsymbol{B}d\boldsymbol{X}\right) \\
&\quad =2\mathrm{tr}\left(\left(\boldsymbol{B}^T\boldsymbol{A}^T\sigma(\boldsymbol{BX})\odot\sigma'(\boldsymbol{BX})\right)^T d\boldsymbol{X}\right)
\end{aligned} \tag{92}$$

  可得 $\frac{\partial f}{\partial \boldsymbol{X}}=2\boldsymbol{B}^T\boldsymbol{A}^T\sigma(\boldsymbol{BX})\odot\sigma'(\boldsymbol{BX})=2\boldsymbol{B}^T\boldsymbol{A}\sigma(\boldsymbol{BX})\odot\sigma'(\boldsymbol{BX})$。

- 线性回归模型的损失函数为 $l=\|\boldsymbol{Xw}-\boldsymbol{y}\|^2$，其中 $\boldsymbol{y}$ 为 m 维列向量，$\boldsymbol{X}$ 为 $m\times n$ 矩阵，$\boldsymbol{w}$ 为 n 维列向量，模型参数 $\boldsymbol{w}$ 的解析解：

$$\begin{aligned}
&l=\|\boldsymbol{Xw}-\boldsymbol{y}\|^2 \quad \Rightarrow \quad dl=d\left(\|\boldsymbol{Xw}-\boldsymbol{y}\|^2\right) \quad \Rightarrow \\
&dl=d\left((\boldsymbol{Xw}-\boldsymbol{y})^T(\boldsymbol{Xw}-\boldsymbol{y})\right) \quad \Rightarrow \\
&dl=(d\boldsymbol{Xw})^T(\boldsymbol{Xw}-\boldsymbol{y})+(\boldsymbol{Xw}-\boldsymbol{y})^T d\boldsymbol{Xw} \quad \Rightarrow \\
&dl=2(\boldsymbol{Xw}-\boldsymbol{y})^T d\boldsymbol{Xw} \quad \Rightarrow \quad dl=2(\boldsymbol{Xw}-\boldsymbol{y})^T\boldsymbol{X}d\boldsymbol{w}
\end{aligned} \tag{93}$$

其中，倒数第 2 步的推导利用了向量内积等式：$\boldsymbol{u}^T\boldsymbol{v} = \boldsymbol{v}^T\boldsymbol{u}$。

根据公式 (85)，可得 $\frac{\partial f}{\partial \boldsymbol{x}} = 2\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$，令 $\frac{\partial f}{\partial \boldsymbol{x}} = 0$，则：

$$2\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) = 0 \quad \Rightarrow \quad \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y} \quad \Rightarrow \quad \boldsymbol{w} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{94}$$

其中，令 $\boldsymbol{X}^\dagger = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T$，它被称为矩阵 $\boldsymbol{X}$ 的 Moore-Penrose 伪逆矩阵（Pseudo-Inverse Matrix）。实际上，它可以看作逆矩阵概念的泛化——将逆矩阵推广到非方阵上。如果 $\boldsymbol{X}$ 是方阵且可逆，那么 Moore-Penrose 伪逆矩阵就退化为逆矩阵 $\boldsymbol{X}^\dagger \equiv \boldsymbol{X}^{-1}$。

- 多元高斯分布的定义为：

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{D}{2}}}\frac{1}{|\boldsymbol{\Sigma}|^{\frac{1}{2}}}\exp\left\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\right\} \tag{95}$$

其中 D 维向量 $\boldsymbol{\mu}$ 被称为均值向量，$D \times D$ 矩阵 $\boldsymbol{\Sigma}$ 被称为协方差矩阵，$|\boldsymbol{\Sigma}|$ 表示 $\boldsymbol{\Sigma}$ 的行列式。设样本 $T = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$，其中 $\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}, \boldsymbol{\Sigma})$，试利用最大似然方法估计均值 $\boldsymbol{\mu}$ 和协方差矩阵 $\boldsymbol{\Sigma}$。

样本数据集的对数似然函数为：

$$\begin{aligned}\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}; \boldsymbol{X}) &= \log\prod_{i=1}^{N}\frac{1}{(2\pi)^{\frac{D}{2}}}\frac{1}{|\boldsymbol{\Sigma}|^{\frac{1}{2}}}\exp\left\{-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu})\right\} \\ &= -\frac{DN}{2}\log 2\pi - \frac{N}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{x}_i - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu})\end{aligned} \tag{96}$$

损失函数表示为负对数似然函数：

$$\begin{aligned}L(\boldsymbol{\mu}, \boldsymbol{\Sigma}; \boldsymbol{X}) &= -\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}; \boldsymbol{X}) \\ &= \frac{DN}{2}\log 2\pi + \frac{N}{2}\log|\boldsymbol{\Sigma}| + \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{x}_i - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu})\end{aligned} \tag{97}$$

观察上式，可知，$\boldsymbol{\mu}$ 与 $\boldsymbol{\Sigma}$ 相互独立，可以单独优化。首先，优化 $\boldsymbol{\mu}$，固定 $\boldsymbol{\Sigma}$：

$$
\begin{aligned}
&dL = d\left(\frac{DN}{2}\log 2\pi + \frac{N}{2}\log|\boldsymbol{\Sigma}| + \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{x}_i-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu})\right) \quad \Rightarrow \\
&dL = \frac{1}{2}\sum_{i=1}^{N}\left[d(\boldsymbol{x}_i-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu}) + (\boldsymbol{x}_i-\boldsymbol{\mu})^T d\left(\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu})\right)\right] \quad \Rightarrow \\
&dL = \frac{1}{2}\sum_{i=1}^{N}\left[-d\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu}) - (\boldsymbol{x}_i-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}d\boldsymbol{\mu}\right] \quad \Rightarrow \\
&dL = -\sum_{i=1}^{N}(\boldsymbol{x}_i-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}d\boldsymbol{\mu}
\end{aligned}
\tag{98}
$$

其中，最后一步的推导，利用了向量内积等式 $\boldsymbol{u}^T\boldsymbol{v} = \boldsymbol{v}^T\boldsymbol{u}$ 以及 $\boldsymbol{\Sigma}$ 为对称矩阵的事实。

根据公式 (85)，得到：$\frac{\partial f}{\partial \boldsymbol{\mu}} = -\sum\limits_{i=1}^{N}\left(\boldsymbol{\Sigma}^{-1}\right)^T(\boldsymbol{x}_i-\boldsymbol{\mu})$。令 $\frac{\partial f}{\partial \boldsymbol{\mu}} = \boldsymbol{0}$，可得：

$$
\begin{aligned}
&-\sum_{i=1}^{N}\left(\boldsymbol{\Sigma}^{-1}\right)^T(\boldsymbol{x}_i-\boldsymbol{\mu}) = \boldsymbol{0} \quad \Rightarrow \quad \sum_{i=1}^{N}\left(\boldsymbol{\Sigma}^{-1}\right)^T\boldsymbol{x}_i = N\left(\boldsymbol{\Sigma}^{-1}\right)^T\boldsymbol{\mu} \quad \Rightarrow \\
&\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i
\end{aligned}
\tag{99}
$$

然后，优化 $\boldsymbol{\Sigma}$，固定 $\boldsymbol{\mu}$：

$$
\begin{aligned}
&dL = d\left(\frac{DN}{2}\log 2\pi + \frac{N}{2}\log|\boldsymbol{\Sigma}| + \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{x}_i-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu})\right) \quad \Rightarrow \\
&dL = \frac{N}{2}\frac{1}{|\boldsymbol{\Sigma}|}d|\boldsymbol{\Sigma}| + \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{x}_i-\boldsymbol{\mu})^T d\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu}) \quad \Rightarrow \\
&dL = \frac{N}{2}\frac{1}{|\boldsymbol{\Sigma}|}|\boldsymbol{\Sigma}|\mathrm{tr}(\boldsymbol{\Sigma}^{-1}d\boldsymbol{\Sigma}) - \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{x}_i-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}d\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu}) \quad \Rightarrow \\
&dL = \mathrm{tr}\left(dL\right) = \frac{N}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}d\boldsymbol{\Sigma}) - \frac{1}{2}\sum_{i=1}^{N}\mathrm{tr}\left((\boldsymbol{x}_i-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}d\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu})\right) \quad \Rightarrow \\
&dL = \frac{N}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}d\boldsymbol{\Sigma}) - \frac{1}{2}\sum_{i=1}^{N}\mathrm{tr}\left(\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu})(\boldsymbol{x}_i-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}d\boldsymbol{\Sigma}\right) \quad \Rightarrow \\
&dL = \mathrm{tr}\left(\frac{N}{2}\boldsymbol{\Sigma}^{-1}d\boldsymbol{\Sigma} - \frac{1}{2}\sum_{i=1}^{N}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu})(\boldsymbol{x}_i-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}d\boldsymbol{\Sigma}\right)
\end{aligned}
\tag{100}
$$

继续对上面的公式变形：

$$dL = \mathrm{tr}\left(\left[\frac{N}{2}\boldsymbol{\Sigma}^{-1} - \frac{1}{2}\sum_{i=1}^{N}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu})(\boldsymbol{x}_i-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}\right]d\boldsymbol{\Sigma}\right) \tag{101}$$

于是，根据公式 (86)，可得 $\frac{\partial L}{\partial \boldsymbol{\Sigma}} = \frac{N}{2}\boldsymbol{\Sigma}^{-1} - \frac{1}{2}\sum_{i=1}^{N}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu})(\boldsymbol{x}_i-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}$，其中利用了 $\boldsymbol{\Sigma}$ 为对称矩阵的事实。令 $\frac{\partial L}{\partial \boldsymbol{\Sigma}} = \mathbf{0}$，则：

$$\begin{aligned}
\frac{N}{2}\boldsymbol{\Sigma}^{-1} &= \frac{1}{2}\sum_{i=1}^{N}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu})(\boldsymbol{x}_i-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1} \\
\frac{N}{2}\boldsymbol{I} &= \frac{1}{2}\sum_{i=1}^{N}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu})(\boldsymbol{x}_i-\boldsymbol{\mu})^T \\
\boldsymbol{\Sigma} &= \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i-\boldsymbol{\mu})(\boldsymbol{x}_i-\boldsymbol{\mu})^T
\end{aligned} \tag{102}$$

其中 $\boldsymbol{I}$ 表示单位矩阵。

- 在多分类逻辑回归模型中，使用负对数似然函数表示交叉熵损失函数[26]：

$$L(\boldsymbol{W}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_i^{(k)}\log\sigma(\boldsymbol{w_k}^T\boldsymbol{x_i}) \tag{103}$$

其中 K 表示类别个数，N 表示样本点个数，$\sigma(\cdot)$ 表示 Softmax 函数。下面，将使用本节介绍的方法求解其梯度。为此，首先将多分类逻辑回归模型的损失函数表示为矩阵向量方式：

$$L(\boldsymbol{W}) = -\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{y}_i^T\log\sigma(\boldsymbol{W}\boldsymbol{x}_i) \tag{104}$$

其中，$\boldsymbol{W}$ 表示 $K\times M$ 权值矩阵，每行表示每个分类的权值向量；$\boldsymbol{x}_i$ 表示 M 维列向量 (第 i 个样本点)；$\boldsymbol{y}_i$ 表示对应的 K 维 One-Hot 列向量 (第 i 个样本点)；$\sigma(\boldsymbol{W}\boldsymbol{x}_i) = \frac{\exp(\boldsymbol{W}\boldsymbol{x}_i)}{\mathbf{1}^T\exp(\boldsymbol{W}\boldsymbol{x}_i)}$ 表示 Softmax 函数逐元素运算后得到的列向量，其中 $\mathbf{1}$ 表示全 1 向量，$\exp(\cdot)$ 为逐元素运算指数函数。于是：

$$\begin{aligned}
&L(\boldsymbol{W}) = -\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{y}_i^T\log\sigma(\boldsymbol{W}\boldsymbol{x}_i) \quad \Rightarrow \\
&L(\boldsymbol{W}) = -\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{y}_i^T\left(\boldsymbol{W}\boldsymbol{x}_i - \mathbf{1}\log\left(\mathbf{1}^T\exp(\boldsymbol{W}\boldsymbol{x}_i)\right)\right)
\end{aligned} \tag{105}$$

[26]在《机器学习》课程系列之《逻辑回归》中，已经做了详细的介绍，并使用类似一元微积分的方法求解了梯度。

对 $L(\boldsymbol{W})$ 两端应用全微分算子，可得：

$$\begin{aligned}
&dL = d\left(-\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{y}_i^T\left(\boldsymbol{W}\boldsymbol{x}_i - \mathbf{1}\log\left(\mathbf{1}^T\exp(\boldsymbol{W}\boldsymbol{x}_i)\right)\right)\right) \quad \Rightarrow \\
&dL = -\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{y}_i^T\left(d\left(\boldsymbol{W}\boldsymbol{x}_i\right) - d\left(\mathbf{1}\log\left(\mathbf{1}^T\exp(\boldsymbol{W}\boldsymbol{x}_i)\right)\right)\right) \quad \Rightarrow \\
&dL = -\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{y}_i^T\left(d\boldsymbol{W}\boldsymbol{x}_i - \mathbf{1}\frac{1}{\mathbf{1}^T\exp(\boldsymbol{W}\boldsymbol{x}_i)}d\left(\mathbf{1}^T\exp(\boldsymbol{W}\boldsymbol{x}_i)\right)\right) \quad \Rightarrow \\
&dL = -\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{y}_i^T\left(d\boldsymbol{W}\boldsymbol{x}_i - \mathbf{1}\frac{1}{\mathbf{1}^T\exp(\boldsymbol{W}\boldsymbol{x}_i)}\mathbf{1}^T\left(\exp(\boldsymbol{W}\boldsymbol{x}_i)\odot d\boldsymbol{W}\boldsymbol{x}_i\right)\right)
\end{aligned} \tag{106}$$

利用公式 $\boldsymbol{y}_i^T\mathbf{1} = 1$ 和 $\mathbf{1}^T(\boldsymbol{u}\odot\boldsymbol{v}) = \boldsymbol{u}^T\boldsymbol{v}$，继续变形：

$$\begin{aligned}
&dL = -\frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{y}_i^T d\boldsymbol{W}\boldsymbol{x}_i - \frac{1}{\mathbf{1}^T\exp(\boldsymbol{W}\boldsymbol{x}_i)}\exp(\boldsymbol{W}\boldsymbol{x}_i)^T d\boldsymbol{W}\boldsymbol{x}_i\right) \quad \Rightarrow \\
&dL = -\frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{y}_i^T d\boldsymbol{W}\boldsymbol{x}_i - \sigma^T(\boldsymbol{W}\boldsymbol{x}_i)d\boldsymbol{W}\boldsymbol{x}_i\right) \quad \Rightarrow \\
&dL = \mathrm{tr}(dL) = -\frac{1}{N}\sum_{i=1}^{N}\left[\mathrm{tr}\left(\boldsymbol{y}_i^T d\boldsymbol{W}\boldsymbol{x}_i\right) - \mathrm{tr}\left(\sigma^T(\boldsymbol{W}\boldsymbol{x}_i)d\boldsymbol{W}\boldsymbol{x}_i\right)\right] \quad \Rightarrow \\
&dL = -\frac{1}{N}\sum_{i=1}^{N}\left[\mathrm{tr}\left(\boldsymbol{x}_i\boldsymbol{y}_i^T d\boldsymbol{W}\right) - \mathrm{tr}\left(\boldsymbol{x}_i\sigma^T(\boldsymbol{W}\boldsymbol{x}_i)d\boldsymbol{W}\right)\right] \quad \Rightarrow \\
&dL = \mathrm{tr}\left(-\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i\left(\boldsymbol{y}_i^T - \sigma^T(\boldsymbol{W}\boldsymbol{x}_i)\right)d\boldsymbol{W}\right)
\end{aligned} \tag{107}$$

于是，根据公式 (86)，可得：

$$\frac{\partial L}{\partial \boldsymbol{W}} = -\frac{1}{N}\sum_{i=1}^{N}\left[\boldsymbol{x}_i\left(\boldsymbol{y}_i^T - \sigma^T(\boldsymbol{W}\boldsymbol{x}_i)\right)\right]^T = -\frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{y}_i - \sigma(\boldsymbol{W}\boldsymbol{x}_i)\right)\boldsymbol{x}_i^T \tag{108}$$

- 神经网络：函数的多层嵌套视角

  神经网络对输入样本 $\boldsymbol{x}$ 连续地应用非线性函数——第 $i-1$ 层的输出是第 i 层的输入，最后在终端获得相应的输出，这是一种典型的数据流水线。从函数的角度看，神经网络相当于多层函数嵌套。

  现假设有一个 2 层神经网络，输出层使用 Softmax 函数 ($\sigma_2(\cdot)$) 作为激活函数，它的前一级网络使用 Sigmoid 函数 ($\sigma_1(\cdot)$) 作为激活函数。对于给定的

数据集，该神经网络的损失函数可表示为[27]：

$$L(\boldsymbol{W}_2,\boldsymbol{W}_1,\boldsymbol{b}_2,\boldsymbol{b}_1)=-\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{y}_i^T\log\sigma_2\left(\boldsymbol{W}_2\sigma_1\left(\boldsymbol{W}_1\boldsymbol{x}_i+\boldsymbol{b}_1\right)+\boldsymbol{b}_2\right)\tag{109}$$

其中，$\boldsymbol{x}_i$ 表示 S 维列向量 (第 i 个样本点)；$\boldsymbol{W}_1$ 表示 $M\times S$ 权值矩阵；$\boldsymbol{b}_1$ 为 M 维列向量；$\sigma_1(\boldsymbol{z}_1)=\frac{1}{1+\exp(-\boldsymbol{z}_1)}$ 为逐元素运算函数，$\boldsymbol{z}_1=\boldsymbol{W}_1\boldsymbol{x}+\boldsymbol{b}_1$；$\boldsymbol{W}_2$ 表示 $K\times M$ 权值矩阵，每行表示每个分类的权值向量；$\boldsymbol{b}_2$ 为 K 维列向量；$\sigma_2(\boldsymbol{z}_2)=\frac{\exp(\boldsymbol{z}_2)}{\mathbf{1}^T\exp(\boldsymbol{z}_2)}$ 表示 Softmax 函数逐元素运算后得到的列向量，其中 $\mathbf{1}$ 表示全 1 向量，$\exp(\cdot)$ 为逐元素运算指数函数，$\boldsymbol{z}_2=\boldsymbol{W}_2\sigma_1(\cdot)+\boldsymbol{b}_2$；$\boldsymbol{y}_i$ 表示对应的 K 维 One-Hot 列向量 (第 i 个样本点)。试计算各权值矩阵与偏置向量的梯度[28]。

为便于表达，令 $\boldsymbol{z}_i^{(1)}=\boldsymbol{W}_1\boldsymbol{x}_i+\boldsymbol{b}_1$，$\boldsymbol{z}_i^{(2)}=\boldsymbol{W}_2\sigma_1\left(\boldsymbol{z}_i^{(1)}\right)+\boldsymbol{b}_2$。首先，将损失函数 (公式 (109)) 表示为如下形式：

$$\begin{aligned}
&L=-\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{y}_i^T\log\sigma_2\left(\boldsymbol{z}_i^{(2)}\right)\quad\Rightarrow\\
&L=-\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{y}_i^T\left(\boldsymbol{z}_i^{(2)}-\mathbf{1}\log\left(\mathbf{1}^T\exp(\boldsymbol{z}_i^{(2)})\right)\right)\quad\Rightarrow\\
&L=-\frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{y}_i^T\boldsymbol{z}_i^{(2)}-\log\left(\mathbf{1}^T\exp(\boldsymbol{z}_i^{(2)})\right)\right)
\end{aligned}\tag{110}$$

于是，对上式两端应用全微分算子：

$$\begin{aligned}
&dL=-\frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{y}_i^T d\boldsymbol{z}_i^{(2)}-d\log\left(\mathbf{1}^T\exp(\boldsymbol{z}_i^{(2)})\right)\right)\quad\Rightarrow\\
&dL=-\frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{y}_i^T d\boldsymbol{z}_i^{(2)}-\frac{1}{\mathbf{1}^T\exp(\boldsymbol{z}_i^{(2)})}\mathbf{1}^T d\exp(\boldsymbol{z}_i^{(2)})\right)\quad\Rightarrow\\
&dL=-\frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{y}_i^T d\boldsymbol{z}_i^{(2)}-\frac{1}{\mathbf{1}^T\exp(\boldsymbol{z}_i^{(2)})}\mathbf{1}^T\left(\exp(\boldsymbol{z}_i^{(2)})\odot d\boldsymbol{z}_i^{(2)}\right)\right)\quad\Rightarrow\\
&dL=-\frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{y}_i^T d\boldsymbol{z}_i^{(2)}-\frac{1}{\mathbf{1}^T\exp(\boldsymbol{z}_i^{(2)})}\exp^T(\boldsymbol{z}_i^{(2)})d\boldsymbol{z}_i^{(2)}\right)
\end{aligned}\tag{111}$$

[27]相当于在上例 (多分类逻辑回归模型) 的基础上添加了函数嵌套功能。

[28]实际上，利用所谓的中间量概念，可以建立层与层之间的梯度递推公式，详见《人工智能》课程系列之《人工神经网络基础》，Chapter8-CN.pdf。在下文中，也为矩阵与向量的梯度引入中间梯度量，从而建立了相邻层间的梯度递推公式。

继续变形，得到：

$$\begin{aligned} dL &= -\frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{y}_i^T d\boldsymbol{z}_i^{(2)} - \sigma_2^T\left(\boldsymbol{z}_i^{(2)}\right) d\boldsymbol{z}_i^{(2)}\right) \quad \Rightarrow \\ dL &= -\frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{y}_i^T - \sigma_2^T\left(\boldsymbol{z}_i^{(2)}\right)\right) d\boldsymbol{z}_i^{(2)} \end{aligned} \tag{112}$$

对上式两端应用迹运算，得到：

$$\begin{aligned} dL &= \mathrm{tr}(dL) = \mathrm{tr}\left(-\frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{y}_i^T - \sigma_2^T\left(\boldsymbol{z}_i^{(2)}\right)\right) d\boldsymbol{z}_i^{(2)}\right) \quad \Rightarrow \\ dL &= -\frac{1}{N}\sum_{i=1}^{N}\mathrm{tr}\left(\left(\boldsymbol{y}_i - \sigma_2\left(\boldsymbol{z}_i^{(2)}\right)\right)^T d\boldsymbol{z}_i^{(2)}\right) \end{aligned} \tag{113}$$

于是，得到 (梯度) 中间量[29]：

$$\frac{\partial L}{\partial \boldsymbol{z}_i^{(2)}} = \boldsymbol{y}_i - \sigma_2\left(\boldsymbol{z}_i^{(2)}\right) \tag{114}$$

可以将 dL 简化为：

$$dL = -\frac{1}{N}\sum_{i=1}^{N}\mathrm{tr}\left(\left(\frac{\partial L}{\partial \boldsymbol{z}_i^{(2)}}\right)^T d\boldsymbol{z}_i^{(2)}\right) \tag{115}$$

将 $\boldsymbol{z}_i^{(2)} = \boldsymbol{W}_2\sigma_1\left(\boldsymbol{z}_i^{(1)}\right) + \boldsymbol{b}_2$ 代入上式，可得：

$$\begin{aligned} dL &= -\frac{1}{N}\sum_{i=1}^{N}\mathrm{tr}\left(\left(\frac{\partial L}{\partial \boldsymbol{z}_i^{(2)}}\right)^T d\left(\boldsymbol{W}_2\sigma_1\left(\boldsymbol{z}_i^{(1)}\right) + \boldsymbol{b}_2\right)\right) \quad \Rightarrow \\ dL &= -\frac{1}{N}\sum_{i=1}^{N}\mathrm{tr}\left(\left(\frac{\partial L}{\partial \boldsymbol{z}_i^{(2)}}\right)^T d\boldsymbol{W}_2\sigma_1\left(\boldsymbol{z}_i^{(1)}\right)\right) \\ &\quad -\frac{1}{N}\sum_{i=1}^{N}\mathrm{tr}\left(\left(\frac{\partial L}{\partial \boldsymbol{z}_i^{(2)}}\right)^T \boldsymbol{W}_2 d\sigma_1\left(\boldsymbol{z}_i^{(1)}\right)\right) \\ &\quad -\frac{1}{N}\sum_{i=1}^{N}\mathrm{tr}\left(\left(\frac{\partial L}{\partial \boldsymbol{z}_i^{(2)}}\right)^T d\boldsymbol{b}_2\right) \end{aligned} \tag{116}$$

从上式的第 1 项可以得到权值 $\boldsymbol{W}_2$ 的梯度为：

$$\frac{\partial L}{\partial \boldsymbol{W}_2} = -\frac{1}{N}\sum_{i=1}^{N}\frac{\partial L}{\partial \boldsymbol{z}_i^{(2)}}\sigma_1^T\left(\boldsymbol{z}_i^{(1)}\right) \tag{117}$$

[29]需要注意的是，在本小节中，中间量的定义与常规有所不同，它们之间相差一个“-”号。

从第 3 项可以得到偏置 $\boldsymbol{b}_2$ 的梯度为：

$$\frac{\partial L}{\partial \boldsymbol{b}_2} = -\frac{1}{N}\sum_{i=1}^{N}\frac{\partial L}{\partial \boldsymbol{z}_i^{(2)}} \tag{118}$$

对第 2 项继续变换，得到：

$$\begin{aligned}
&-\frac{1}{N}\sum_{i=1}^{N}\mathrm{tr}\left(\left(\frac{\partial L}{\partial \boldsymbol{z}_i^{(2)}}\right)^T \boldsymbol{W}_2 d\sigma_1\left(\boldsymbol{z}_i^{(1)}\right)\right) \quad \Rightarrow \\
&-\frac{1}{N}\sum_{i=1}^{N}\mathrm{tr}\left(\left(\frac{\partial L}{\partial \boldsymbol{z}_i^{(2)}}\right)^T \boldsymbol{W}_2\left(\sigma_1'\left(\boldsymbol{z}_i^{(1)}\right)\odot d\boldsymbol{z}_i^{(1)}\right)\right) \quad \Rightarrow \\
&-\frac{1}{N}\sum_{i=1}^{N}\mathrm{tr}\left(\left(\boldsymbol{W}_2^T\frac{\partial L}{\partial \boldsymbol{z}_i^{(2)}}\odot\sigma_1'\left(\boldsymbol{z}_i^{(1)}\right)\right)^T d\boldsymbol{z}_i^{(1)}\right) \quad \Rightarrow
\end{aligned} \tag{119}$$

于是，得到 (梯度) 中间量：

$$\frac{\partial L}{\partial \boldsymbol{z}_i^{(1)}} = \boldsymbol{W}_2^T\frac{\partial L}{\partial \boldsymbol{z}_i^{(2)}}\odot\sigma_1'\left(\boldsymbol{z}_i^{(1)}\right) \tag{120}$$

对公式 (119) 进行简化，可得：

$$-\frac{1}{N}\sum_{i=1}^{N}\mathrm{tr}\left(\left(\frac{\partial L}{\partial \boldsymbol{z}_i^{(1)}}\right)^T d\boldsymbol{z}_i^{(1)}\right) \tag{121}$$

为得到权值 $\boldsymbol{W}_1$ 与偏置 $\boldsymbol{b}_1$ 的梯度，将 $\boldsymbol{z}_i^{(1)} = \boldsymbol{W}_1\boldsymbol{x}_i + \boldsymbol{b}_1$ 代入上式，可得：

$$\begin{aligned}
&-\frac{1}{N}\sum_{i=1}^{N}\mathrm{tr}\left(\left(\frac{\partial L}{\partial \boldsymbol{z}_i^{(1)}}\right)^T d\left(\boldsymbol{W}_1\boldsymbol{x}_i + \boldsymbol{b}_1\right)\right) \quad \Rightarrow \\
&-\frac{1}{N}\sum_{i=1}^{N}\mathrm{tr}\left(\left(\frac{\partial L}{\partial \boldsymbol{z}_i^{(1)}}\right)^T d\boldsymbol{W}_1\boldsymbol{x}_i\right) - \frac{1}{N}\sum_{i=1}^{N}\mathrm{tr}\left(\left(\frac{\partial L}{\partial \boldsymbol{z}_i^{(1)}}\right)^T d\boldsymbol{b}_1\right)
\end{aligned} \tag{122}$$

由此，得到权值 $\boldsymbol{W}_1$ 和偏置 $\boldsymbol{b}_1$ 的梯度分别为：

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{W}_1} &= -\frac{1}{N}\sum_{i=1}^{N}\frac{\partial L}{\partial \boldsymbol{z}_i^{(1)}}\boldsymbol{x}_i^T \\
\frac{\partial L}{\partial \boldsymbol{b}_1} &= -\frac{1}{N}\sum_{i=1}^{N}\frac{\partial L}{\partial \boldsymbol{z}_i^{(1)}}
\end{aligned} \tag{123}$$

实际上，使用 $i-1$ 替换公式中的层索引 1，使用 i 替换公式中的层索引 2，上述公式均适用于任意相邻层。

4.2 感知机实验

IRIS(鸢尾花) 数据集

由 Fisher 在 1936 年整理，包含 4 个特征: Sepal.Length（花萼长度）、Sepal.Width（花萼宽度）、Petal.Length（花瓣长度）、Petal.Width（花瓣宽度），它的特征值都为正浮点数，单位为厘米。

目标值为鸢尾花的分类:

- Iris Setosa（山鸢尾）：0；
- Iris Versicolour（杂色鸢尾）：1；
- Iris Virginica（维吉尼亚鸢尾））：2；

```
In [1]: import pandas as pd
        import numpy as np
        from sklearn.datasets import load_iris
        import matplotlib.pyplot as plt
        %matplotlib inline
        %config InlineBackend.figure_format = 'svg'
```

```
In [2]: from IPython.display import Image
        Image(filename="iris.jpg",width=200, height=100)
```

Out[2]:

载入 sklearn 鸢尾花数据集

```
In [3]: iris = load_iris()
        print(iris.keys()) # 数据集对象的属性
        print(iris.data.shape) # 维度信息
        print(iris.feature_names) # 特征名称
        print(iris['target_names']) # 分类名称
        print(iris['target']) # 分类值
```

```
dict_keys(['feature_names', 'data', 'target_names', 'DESCR', 'target'])
(150, 4)
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
['setosa' 'versicolor' 'virginica']
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

将数据存入 DataFrame 中

```
In [4]: iris_df = pd.DataFrame(iris.data, columns = iris.feature_names)
        iris_df[:5]
```

Out[4]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

添加分类属性

```
In [5]: iris_df['label'] = iris.target
        iris_df[:5]
```

Out[5]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	\
0	5.1	3.5	1.4	0.2	
1	4.9	3.0	1.4	0.2	
2	4.7	3.2	1.3	0.2	
3	4.6	3.1	1.5	0.2	
4	5.0	3.6	1.4	0.2	

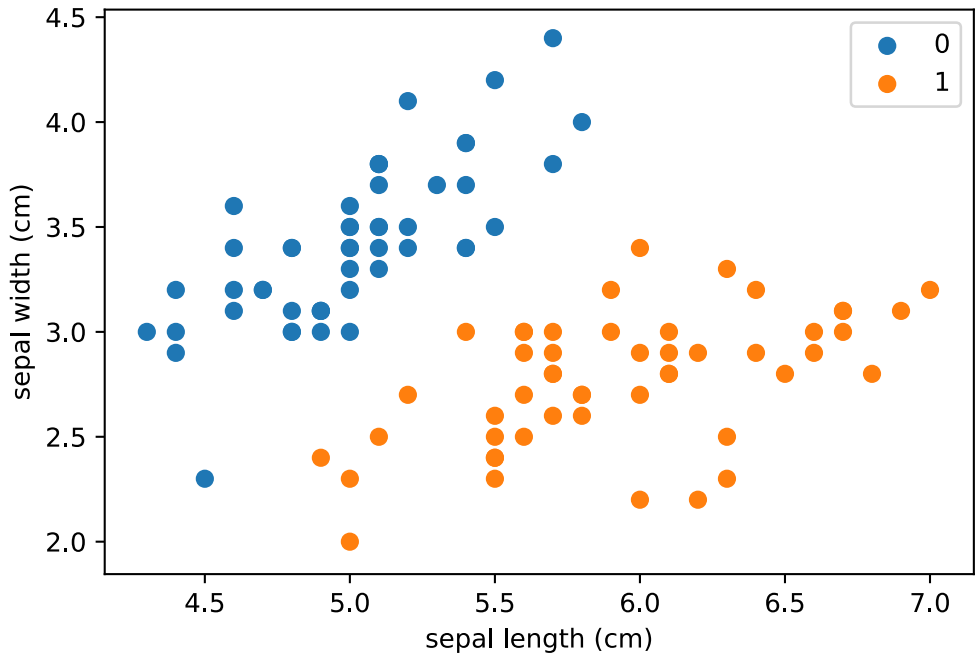
	label
0	0
1	0
2	0
3	0
4	0

```
In [6]: print(iris_df['label'].value_counts()) # 显示不同分类的详细数值
```

```
2    50
1    50
0    50
Name: label, dtype: int64
```

显示 2 分类数据集（为便于可视化，只选取前 2 列数据，即花萼长度与宽度属性数据，并且只选取前 100 条记录，即 0 与 1 分类）

```
In [7]: plt.scatter(iris_df[:50]['sepal length (cm)'], iris_df[:50]
         ['sepal width (cm)'], label='0')
        plt.scatter(iris_df[50:100]
         ['sepal length (cm)'], iris_df[50:100]['sepal width (cm)'], label='1')
        plt.xlabel('sepal length (cm)')
        plt.ylabel('sepal width (cm)')
        plt.legend()
        plt.savefig('PERCEPTRON_OUTPUT1.pdf', bbox_inches='tight')
```



提取训练数据

```
        # 取出第 0 列、第 1 列、最后一列，前 100 条记录
In [8]: data = np.array(iris_df.iloc[:100, [0, 1, -1]])
        X, Y = data[:, :-1], data[:, -1]
        # 转换成适合感知机训练的分类：1、-1
        Y = np.array([1 if y == 0 else -1 for y in Y])
        print(data.shape)
```

```
(100, 3)
```

定义感知机模型

```
In [9]: class Perceptron:
            def __init__(self):
                self.alpha = 0.05
                self.paras =[]

            def w_x_b(self, x):
                return np.dot(self.W, x) + self.b

            # 随机梯度下降
            def fit(self, X, Y):
                self.W = np.ones(X.shape[1], dtype = np.float32)
                self.b = -5

                x0_mean = np.mean(X[0])
                x1_mean = np.mean(X[1])
                self.W[0] = x1_mean/x0_mean
                print(self.W, self.b)
                self.paras = [(self.W, self.b)]

                finish = False
                while not finish:
                    finish = True
                    for index in range(len(X)):
                        x = X[index]
                        y = Y[index]
                        if y * self.w_x_b(x) <= 0:
                            # 注意，不能使用 += 运算符；否则，计算错误
                            self.W = self.W + self.alpha * y * x
                            self.b = self.b + self.alpha * y
                            finish = False

                            self.paras.append((self.W, self.b))

                    if finish:
                        return 'Perceptron training completed!'
```

使用训练数据集训练感知机

```
In [10]: np.random.seed()
         model = Perceptron()
         model.fit(X, Y)
```

```
[ 0.91860467  1.        ] -5
```

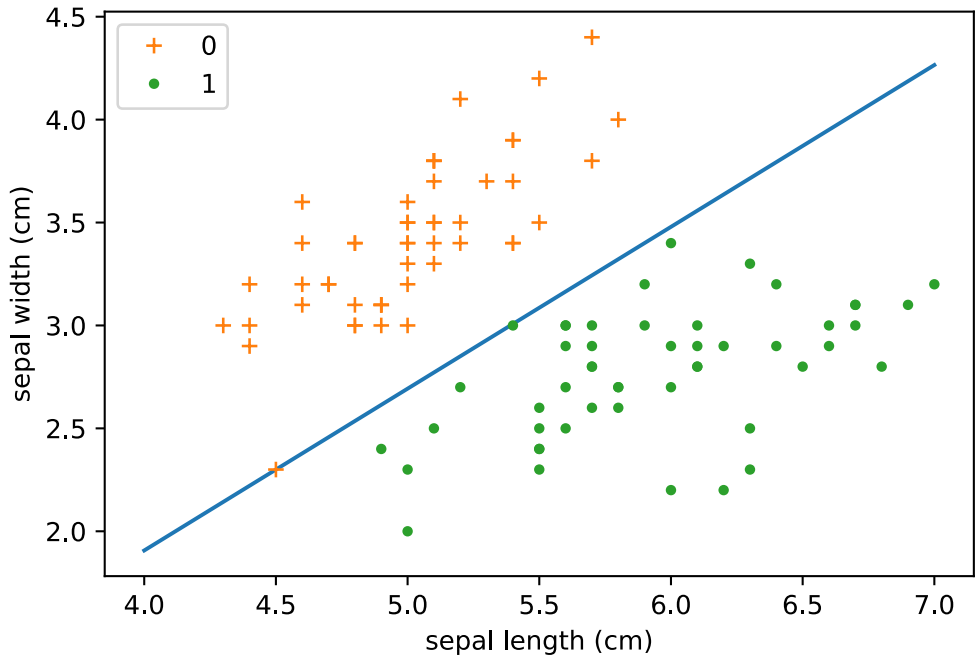
```
Out[10]: 'Perceptron training completed!'
```

绘制训练动画

```
In [11]: import matplotlib.animation as animation
         from IPython.display import HTML
         # 动画播放，如果是%matplotlib inline，则会嵌入图片
         %matplotlib notebook

         fig = plt.figure()
         ax = plt.axes(xlim=(3, 7), ylim=(2, 5))

         ax.plot(data[:50, 0], data[:50, 1], '+', label='0')
         ax.plot(data[50:100, 0], data[50:100, 1], '.', label='1')
         ax.set_xlabel('sepal length (cm)')
         ax.set_ylabel('sepal width (cm)')
         ax.set_title('Perceptron')
         ax.legend(loc = 'upper left')

         line, = ax.plot([], [], lw = 2)
         def update(para):
             W = para[0]
             b = para[1]

             X_points = np.linspace(4, 7,10)
             Y_points = -(W[0] * X_points + b)/W[1]
             line.set_data(X_points, Y_points)
         ani = animation.FuncAnimation(fig, update, model.paras)
         # 视频播放
         # 安装 FFMPEG 库
         # 在 anaconda 环境下，需要执行“conda install -c conda-forge ffmpeg”
         # 需要 FFMPEG.exe，直接下载 BUILD，解压到目录，例如 E:\ffmpeg\bin
         # 然后在环境变量 PATH 中添加“E:\ffmpeg\bin”，
         # 路径名称需要根据实际情况做相应的调整
         HTML(ani.to_html5_video())

In [12]: ani.save('Fitting-duxiaoqin.mp4')
```

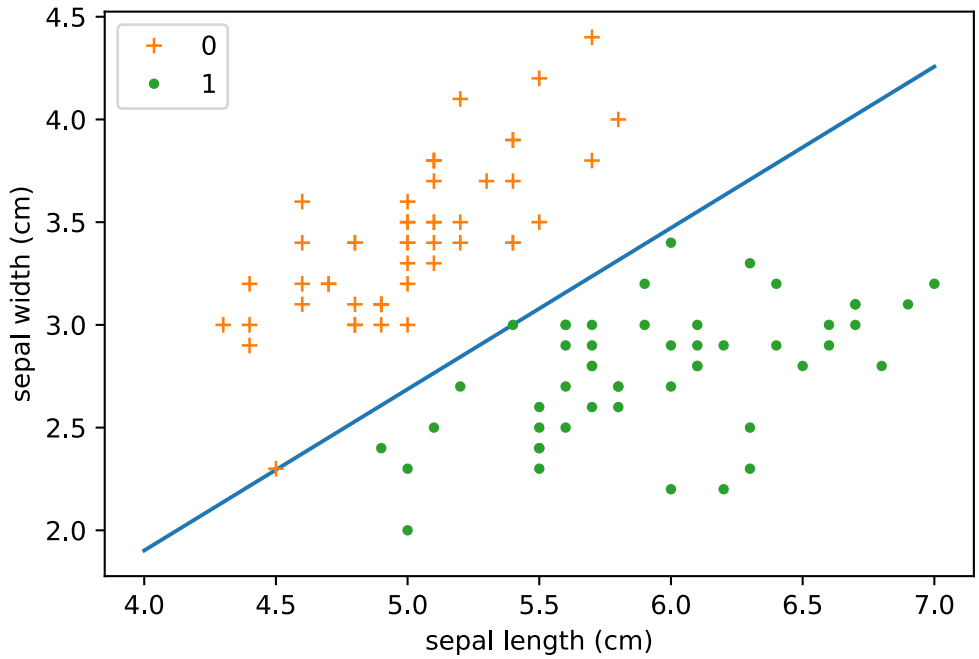
绘制训练结果

```
In [13]: %matplotlib inline
         X_points = np.linspace(4, 7,10)
         Y_points = -(model.W[0] * X_points + model.b)/model.W[1]
         plt.plot(X_points, Y_points)

         plt.plot(data[:50, 0], data[:50, 1], '+', label='0')
         plt.plot(data[50:100, 0], data[50:100, 1], '.', label='1')
         plt.xlabel('sepal length (cm)')
         plt.ylabel('sepal width (cm)')
         plt.legend()
         plt.savefig('PERCEPTRON_OUTPUT2.pdf', bbox_inches='tight')
```



Perceptron（sklearn）实验

```
In [14]: from sklearn.linear_model import Perceptron

In [15]: model = Perceptron(max_iter = 1000)
         model.fit(X, Y)

Out[15]: Perceptron(alpha=0.0001, class_weight=None, eta0=1.0, fit_intercept=True,
               max_iter=1000, n_iter=None, n_jobs=1, penalty=None, random_state=0,
               shuffle=True, tol=None, verbose=0, warm_start=False)

In [16]: X_points = np.linspace(4, 7,10)
         Y_points = -(model.coef_[0][0] * X_points + model.intercept_)/
          model.coef_[0][1]
         plt.plot(X_points, Y_points)

         plt.plot(data[:50, 0], data[:50, 1], '+', label='0')
         plt.plot(data[50:100, 0], data[50:100, 1], '.', label='1')
         plt.xlabel('sepal length (cm)')
         plt.ylabel('sepal width (cm)')
         plt.legend()
         plt.savefig('PERCEPTRON_OUTPUT3.pdf', bbox_inches='tight')
```


0
1
4.5
4.0
3.5
3.0
2.5
2.0
sepal width (cm)
4.0
4.5
5.0
5.5
6.0
6.5
7.0
sepal length (cm)

5 参考文献

1. 周志华。《机器学习》，清华大学出版社，2016 年 1 月第 1 版。
2. Christopher M. Bishop. Pattern Recognition and Machine Learning. Springer, 2006.
3. Kevin P. Murphy. Machine Learning: A Probabilistic Perspective, The MIT Press, 2012.
4. 李航。《统计学习方法》，清华大学出版社，2019 年 5 月第 2 版。
5. Vladimir N. Vapnik。《统计学习理论的本质》，张学工译，2000 年，清华大学出版社。
6. Kaare Brandt Petersen, Michael Syskind Pedersen. The Matrix Cookbook, Nov. 15, 2012. http://matrixcookbook.com.
7. 矩阵求导 I. https://zhuanlan.zhihu.com/p/24709748.
8. 矩阵求导 II. https://zhuanlan.zhihu.com/p/24863977.
9. Fisher 线性判别与线性感知机。https://zhuanlan.zhihu.com/p/88657842.
10. https://github.com/wzyonggege/statistical-learning-method;