常见面试问题

1.SVM用什么损失函数？为什么用Hinge loss?

https://www.cnblogs.com/guoyaohua/p/9436237.html

按照LibSVM的习惯，SVM的目标函数是这样的：

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, i = 1, \ldots, l, \end{aligned}$$

这里的参数C代表的是在线性不可分的情况下，对分类错误的惩罚程度。

C值越大，分类器就越不愿意允许分类错误（“离群点”）。如果C值太大，分类器就会竭尽全力地在训练数据上少犯错误，而实际上这是不可能/没有意义的，于是就造成过拟合。

而C值过小时，分类器就会过于“不在乎”分类错误，于是分类性能就会较差。

2.核函数的选取

.一般选取线性核和高斯核，也就是线性核与RBF核。

线性核：

主要用于线性可分的情况，参数少，速度快，对于一般数据，分类效果已经很理想。

RBF核：

主要用于线性不可分的情况，参数多，分类结果很依赖参数。很多人通过训练数据的交叉验证来寻找合适的参数，但是过程耗时。这个核会将原始空间映射为无穷维空间。如果 σ 选得很大的话，高次特征上的权重实际上衰减得非常快，所以实际上（数值上近似一下）相当于一个低维的子空间；反过来，如果 σ 选得很小，则可以将任意的数据映射为线性可分——当然，这并不一定是好事，因为随之而来的可能是非常严重的过拟合问题。不过，总的来说，通过调控参数σ，高斯核实际上具有相当高的灵活性，也是使用最广泛的核函数之一。

如果feature数量很大，跟样本数量差不多，选线性核。

如果feature数量比较小，样本数量一般，选用高斯核。

作为当今最为流行的分类算法之一，SVM 已经拥有了不少优秀的实现库，如 libsvm 等，因此，我们不再需要自己手动实现 SVM（要知道，一个能用于生产环境的 SVM 模型并非课程中介绍的那么简单）。

在使用这些库时，我们通常需要声明 SVM 需要的两个关键部分：

1. 参数 C
2. 核函数（Kernel）

由于 C 可以看做与正规化参数 λ 作用相反，则对于 C 的调节：

- **低偏差，高方差**，即遇到了**过拟合**时：**减小** C 值。
- **高偏差，低方差**，即遇到了**欠拟合**时：**增大** C 值。

- 当特征维度 n 较高，而样本规模 m 较小时，**不宜使用核函数**，否则容易引起过拟合。
- 当特征维度 n 较低，而样本规模 m 足够大时，考虑使用**高斯核函数**。不过在使用高斯核函数前，需要进行特征缩放（feature scaling）。另外，当核函数的参数 δ **较大**时，特征 f_i 较为平缓，即各个样本的特征差异变小，此时会造成**欠拟合（高偏差，低方差）**：

1.了解SVM

支持向量机，英文support vector machine，简称SVM，是一种二类分类模型，基本模型是定义在特征空间上的间隔最大的线性分类器，但SVM还包含了核技巧，所以实质上可成为非线性分类器，其学习策略是间隔最大化，最终可转化为凸二次规划的最优解问题。

1.1 分类标准的起源：Logistic回归

线性分类器的概念：给定一些属于两类的数据点，找到一个线性分类器分开他们。如果用x表示数据点，y表示类别（+1、-1），一个线性分类器的学习目标便是要在n维的数据空间中找到一个超平面，这个超平面的方程如下：

wTx+b=0（T代表转置）

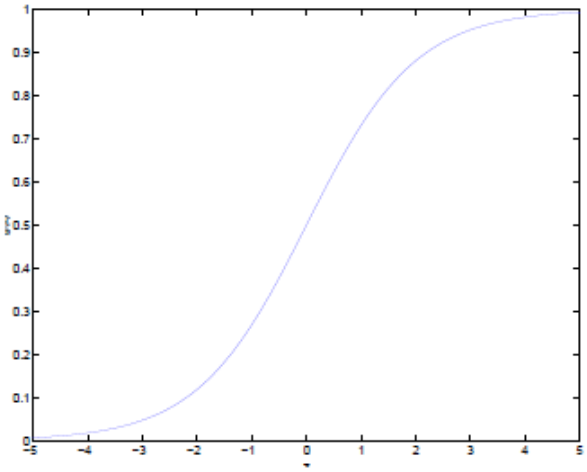
logistic回归的目的是从特征学习出一个0/1分类模型，将特征的线性组合作为自变量，自变量的取值范围是负无穷到正无穷，因此使用logistic函数（sigmoid)将自变量映射到0~1上，映射后的值被认为是属于y=1的概率。

假设函数：

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}},$$

其中x是n维特征向量，函数g就是logistic函数，

而 $g(z) = \frac{1}{1 + e^{-z}}$ 的图像是



可以看到，将无穷映射到了（0,1）。

而假设函数就是特征属于y=1的概率，

$$P(y=1 \mid x;\theta) = h_\theta(x)$$
$$P(y=0 \mid x;\theta) = 1 - h_\theta(x)$$

从而，当我们要判别一个新来的特征属于哪个类时，只需求$h_\theta(x)$即可，若$h_\theta(x)$大于0.5就是y=1的类，反之属于y=0类。

此外，$h_\theta(x)$只和$\theta^T x$有关，$\theta^T x>0$，那么$h_\theta(x) > 0.5$，而g(z)只是用来映射，真实的类别决定权还是在于$\theta^T x$。再者，当$\theta^T x \gg 0$时，$h_\theta(x)=1$，反之$h_\theta(x)=0$。如果我们只从$\theta^T x$出发，希望模型达到的目标就是让训练数据中y=1的特征$\theta^T x \gg 0$，而是y=0的特征$\theta^T x \ll 0$。Logistic回归就是要学习得到θ，使得正例的特征远大于0，负例的特征远小于0，而且要在全部训练实例上达到这个目标。

接下来，尝试把logistic回归做个变形。首先，将使用的结果标签y = 0和y = 1替换为y = -1,y = 1，然后将（$\theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$）中的$\theta_0$替换为b，最后将后面的$\theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$替换为$\theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$（即$w^T x$）。如此，则有了$\theta^T x = w^T x + b$。也就是说除了y由y=0变为y=-1外，线性分类函数跟logistic回归的形式化表示$h_\theta(x) = g(\theta^T x) = g(w^T x + b)$没区别。

进一步，可以将假设函数$h_{w,b}(x) = g(w^T x + b)$中的g(z)做一个简化，将其简单映射到y=-1和y=1上。映射关系如下：

$$g(z) = \begin{cases} 1, & z \geq 0 \\ -1, & z < 0 \end{cases}$$

以上，我们知道了要找一个分离超平面wTx+b=0，要找到最优的超平面，也就是找正确划分且间隔最大的直线。

1.2 函数间隔与几何间隔

在超平面w·x+b=0确定的情况下，|w·x+b|能够相对地表示点x距离超平面的远近，而w·x+b的符号与类y的符号是否一致能够表示分类是否正确，所以可以用y(w·x+b)来表示分类的正确性及确信度，这就是函数间隔的概念。

定义函数间隔（用 $\hat{\gamma}$ 表示）为：

$$\hat{\gamma} = y(w^T x + b) = yf(x)$$

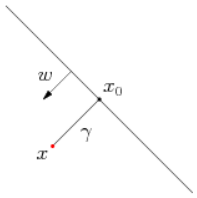
而超平面(w，b)关于T中所有样本点(xi，yi)的函数间隔最小值（其中，x是特征，y是结果标签，i表示第i个样本），便为超平面(w, b)关于训练数据集T的函数间隔：

$\hat{\gamma}$ = min $\hat{\gamma}$ i (i=1，...n)

但这样定义的函数间隔有问题，如果成比例的改变w和b（比如2w和2b)，则函数间隔值变为原来2倍，但超平面是一定的。

因此，我们对法向量w加些约束，如规范化||w||=1，使得间隔是确定的，这就是几何间隔的概念。

假定对于一个点 x，令其垂直投影到超平面上的对应点为 x0，w 是垂直于超平面的一个向量，γ 为样本x到超平面的距离，如下图所示：



点x到超平面的距离公式：

$$\gamma = \frac{w^T x + b}{\|w\|} = \frac{f(x)}{\|w\|}$$

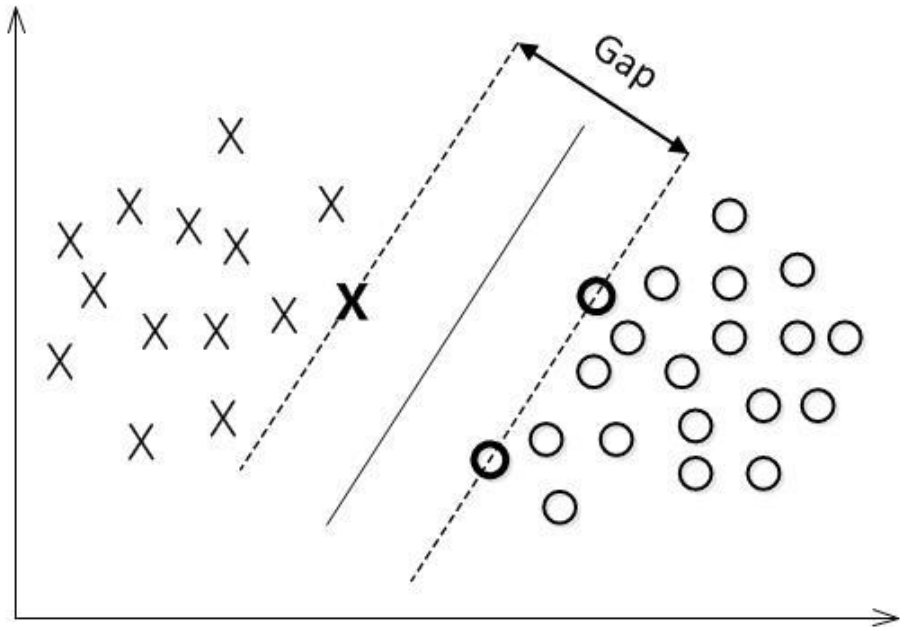
为了得到 γ 的绝对值，令 γ 乘上对应的类别 y，即可得出几何间隔（用 $\tilde{\gamma}$ 表示）的定义：

$$\tilde{\gamma} = y\gamma = \frac{\hat{\gamma}}{\|w\|}$$

综上，几何间隔就是函数间隔除以||w||（二范数），而且函数间隔y(wx+b)实际上就是|f(x)|，只是人为定义的一个间隔度量，而几何间隔|f(x)|/||w||才是直观上的点到超平面的距离。

1.3 最大间隔分类器的定义

要让选择的超平面能最大化数据点与平面的间隔，这个间隔就是下图Gap的一半。



函数间隔不合适，因为它可以任意变大，但几何间隔只随着超平面的改变而改变，因此我们要找的最大间隔是几何间隔。

这个问题用如下公式表达：

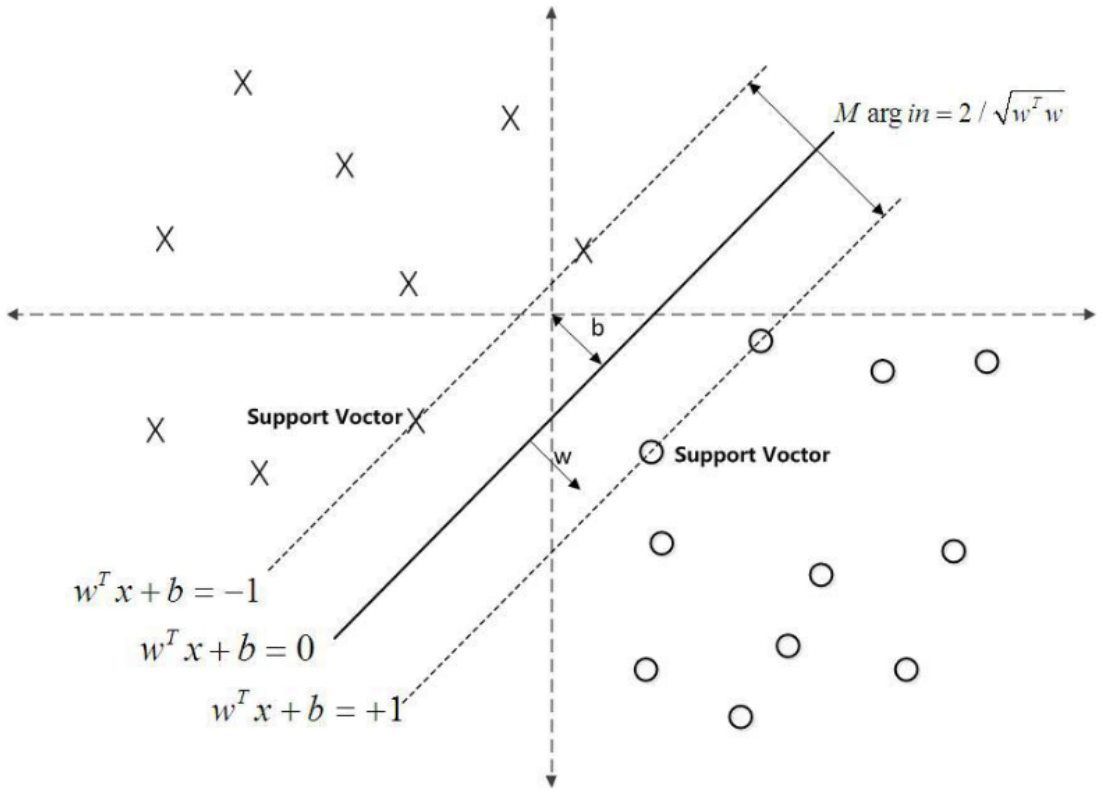
$$\max_{w,b} \frac{\hat{\gamma}}{\|w\|}$$

$$s.t \quad y_i(w \cdot x_i + b) \geq \hat{\gamma} \qquad i = 1, 2, \ldots, N.$$

函数间隔的取值并不影响最优化问题的解，我们取函数间隔=1，以及最大化1/||w|| 和最小化 ||w||^2/2是等价的，所以问题转化为：

$$\min \frac{1}{2}\|w\|^2$$

$$s.t \quad y_i(w \cdot x_i + b) - 1 \geq 0 \qquad i = 1, 2, \ldots, N.$$



2. SVM深入、公式推导（线性可分、线性不可分）

2.1 从线性可分到线性不可分

2.1.1 从原始问题到对偶问题的求解

原始问题如第一节最后所示，目标函数是二次的，约束条件是线性的，是一个凸二次规划问题，可以通过拉格朗日对偶性，求解对偶问题得到原始问题的最优解。这样做的优点：一是对偶问题往往更容易求解，二是自然引入核函数，进而推广到非线性分类器。

首先构建拉格朗日函数，

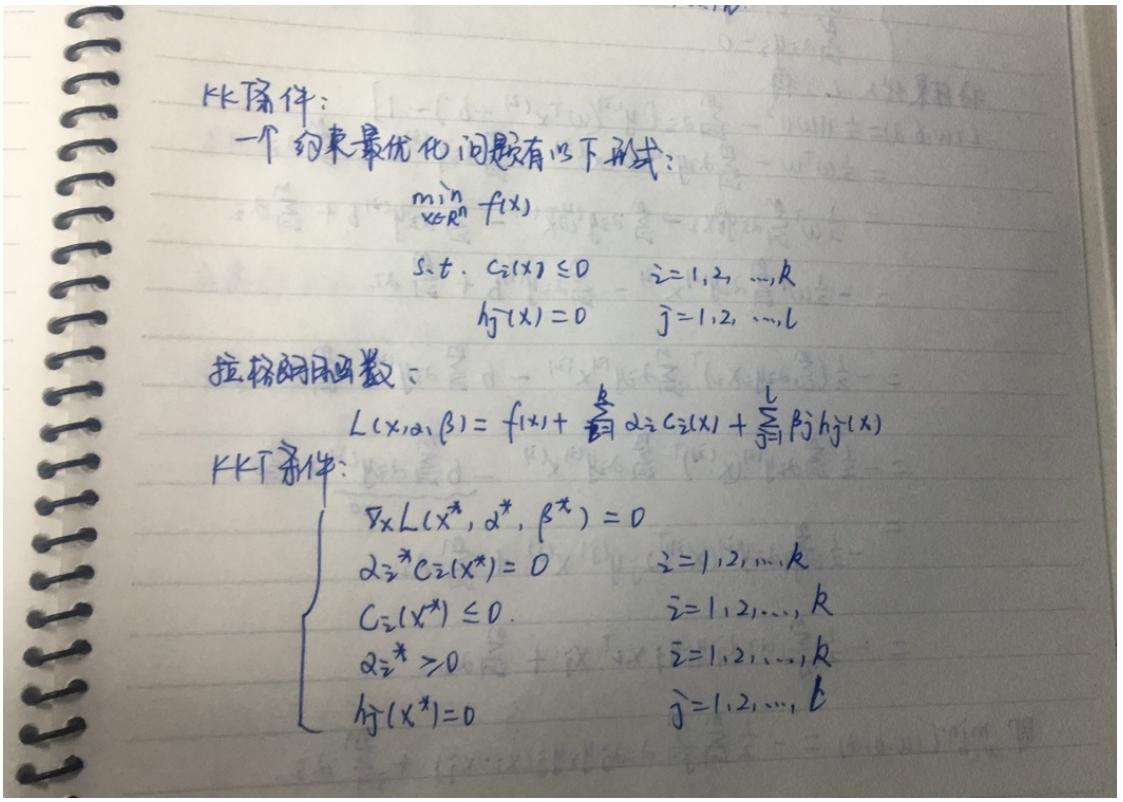
$$\mathcal{L}(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n}\alpha_i\left(y_i(w^T x_i + b) - 1\right)$$

根据拉格朗日对偶性，原始问题的对偶问题是极大极小问题

$$\max_{\alpha_i \geq 0} \min_{w,b} \mathcal{L}(w,b,\alpha) = d^*$$

2.1.2 KKT条件

KKT条件的意义：是一个非线性规划问题能有最优化解法的充要条件


KK条件：
一个约束最优化问题有以下形式：
$\min_{x \in R^n} f(x)$
$s.t.\ c_i(x) \le 0 \quad i=1,2,\dots,k$
$h_j(x)=0 \quad j=1,2,\dots,l$
拉格朗日函数：
$L(x,\alpha,\beta)=f(x)+\sum_{i=1}^{k}\alpha_i c_i(x)+\sum_{j=1}^{l}\beta_j h_j(x)$
KKT条件：
$\nabla_x L(x^*,\alpha^*,\beta^*)=0$
$\alpha_i^* c_i(x^*)=0 \quad i=1,2,\dots,k$
$c_i(x^*) \le 0 \quad i=1,2,\dots,k$
$\alpha_i^* \ge 0 \quad i=1,2,\dots,k$
$h_j(x^*)=0 \quad j=1,2,\dots,l$

2.1.3 对偶问题的求解

对偶问题的求解：

(1) 求 $\min_{w,b} L(w,b,\alpha)$，固定 α.

求拉格朗日函数 $L(w,b,\alpha)$ 分别对 w,b 求偏导数，并令其等于 0

$$\nabla_w L(w,b,\alpha) = w - \sum_{i=1}^{N} \alpha_i y_i x_i = 0$$

$$\nabla_b L(w,b,\alpha) = -\sum_{i=1}^{N} \alpha_i y_i = 0$$

得：

$$\begin{cases} w = \sum_{i=1}^{N} \alpha_i y_i x_i \\ \sum_{i=1}^{N} \alpha_i y_i = 0 \end{cases}$$

将结果代入 L，得：

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i [y^{(i)}(w^T x^{(i)} + b) - 1]$$

$$= \frac{1}{2} w^T w - \sum_{i=1}^{m} \alpha_i y^{(i)} w^T x^{(i)} - \sum_{i=1}^{m} \alpha_i y^{(i)} b + \sum_{i=1}^{m} \alpha_i$$

$$= \frac{1}{2} w^T \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} - \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)T} w - \sum_{i=1}^{m} \alpha_i y^{(i)} b + \sum_{i=1}^{m} \alpha_i$$

$$= -\frac{1}{2} w^T \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} - \sum_{i=1}^{m} \alpha_i y^{(i)} b + \sum_{i=1}^{m} \alpha_i$$

$$= -\frac{1}{2} \left(\sum_{i=1}^{m} \alpha_i y_i x_i\right)^T \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} - b \sum_{i=1}^{m} \alpha_i y^{(i)} + \sum_{i=1}^{m} \alpha_i$$

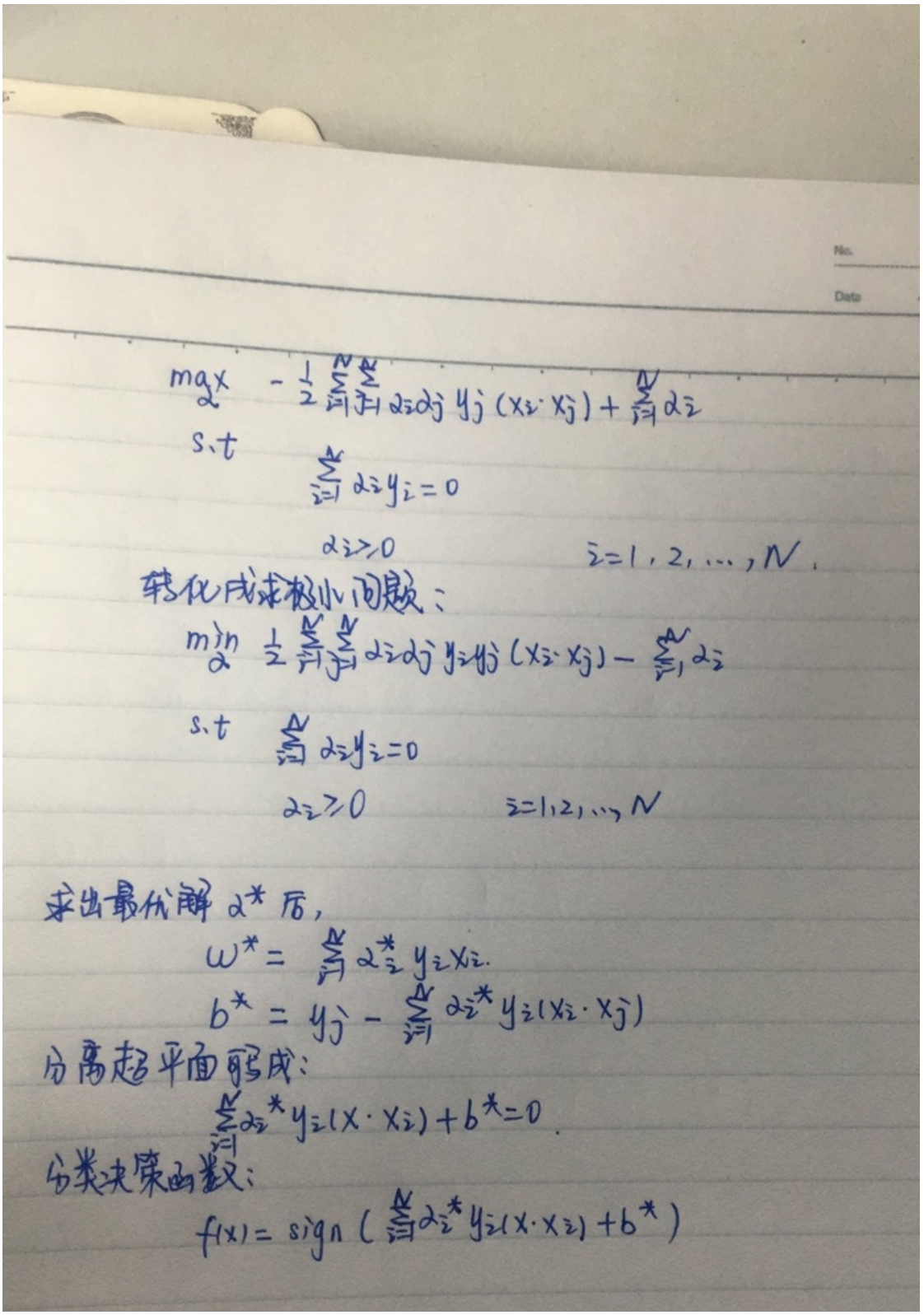
$$= -\frac{1}{2} \sum_{i=1}^{m} \alpha_i y^{(i)} (x^{(i)})^T \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \underbrace{- b \sum_{i=1}^{m} \alpha_i y^{(i)}}_{=0} + \sum_{i=1}^{m} \alpha_i$$

$$= -\frac{1}{2} \sum_{i=1}^{m} \alpha_i y^{(i)} (x^{(i)})^T \alpha_j y^{(j)} x^{(j)} + \sum_{i=1}^{m} \alpha_i$$

$$= -\frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{m} \alpha_i$$

即 $\min_{w,b} L(w,b,\alpha) = -\frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) + \sum_{i=1}^{m} \alpha_i$.

(2) 求 $\min_{w,b} L(w,b,\alpha)$ 对 α 的极大，即是对偶问题：



2.1.4 线性不可分的情况

根据上文得到的分类函数，我们发现，对于新点x的预测，只需要计算它与训练数据点的内积，这一点至关重要，是之后使用Kernel进行非线性推广的基本前提。所有非支持向量所对应的系数阿尔法都是等于0的，因此对于新点的内积计算实际上只要针对少量的“支持向量”而不是所有的训练数据即可。

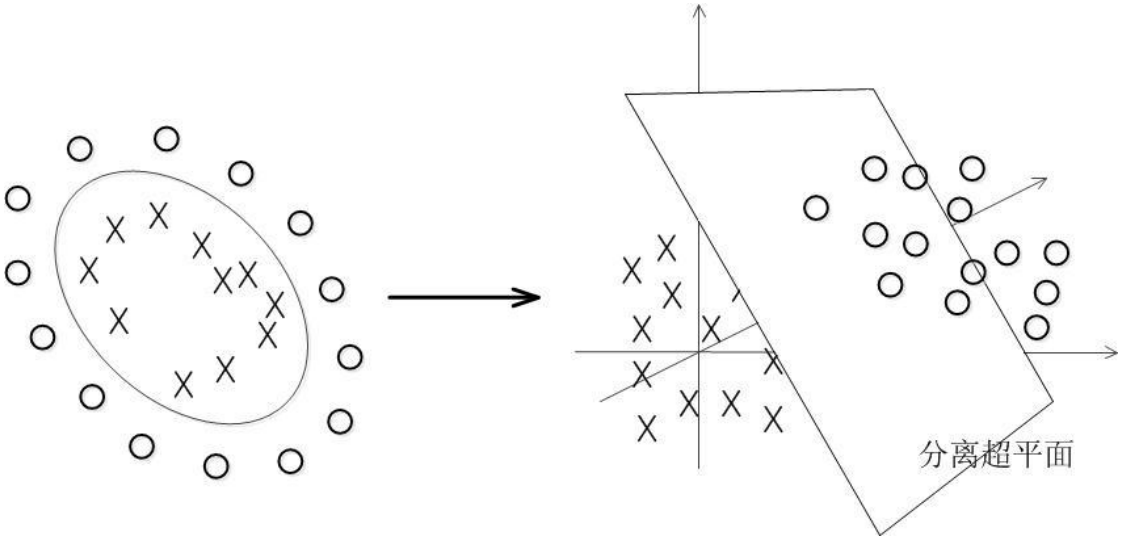
$$\max_{\alpha_i \geq 0} \mathcal{L}(w, b, \alpha) = \max_{\alpha_i \geq 0} \frac{1}{2} \|w\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i (w^T x_i + b) - 1 \right)$$

注意到如果 是支持向量的话，上式中红颜色的部分是等于 0 的（因为支持向量的 functional margin 等于 1 ），而对于非支持向量来说，functional margin 会大于 1 ，因此红颜色部分是大于零的，而α_i又是非负的，为了满足最大化，α_i必须等于 0 。这也就是这些非Supporting Vector的点的局限性。

2.2 核函数Kernel

2.2.1 特征空间的隐式映射：核函数

对于非线性的数据，SVM的处理方法是选择一个核函数，将数据映射到高维空间，解决在原始空间中线性不可分的问题。



核函数相当于把原来的分类函数：

$$f(x)=\sum_{i=1}^{n}\alpha_i y_i\langle x_i,x\rangle+b$$

映射成：

$$f(x)=\sum_{i=1}^{n}\alpha_i y_i\langle \phi(x_i),\phi(x)\rangle+b$$

而其中的α可以通过求解如下 dual 问题而得到的：

$$\begin{aligned}&\max_{\alpha}\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j\langle\phi(x_i),\phi(x_j)\rangle\\&s.t.,\ \alpha_i\geq 0,i=1,\ldots,n\\&\sum_{i=1}^{n}\alpha_i y_i=0\end{aligned}$$

核技巧的想法是，只定义核函数K(x,z)，而不显示地定义映射函数$(\phi$

此时对偶问题的目标函数为

$$\begin{aligned}&\max_{\alpha}\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j\kappa(x_i,x_j)\\&s.t.,\ \alpha_i\geq 0,i=1,\ldots,n\\&\sum_{i=1}^{n}\alpha_i y_i=0\end{aligned}$$

分类决策函数为

$$\sum_{i=1}^{n}\alpha_i y_i\kappa(x_i,x)+b$$

2.2.3 几个核函数

1. 多项式核

1. **多项式核函数**（polynomial kernel function）

$$K(x,z)=(x\cdot z+1)^p \tag{7.88}$$

对应的支持向量机是一个 p 次多项式分类器．在此情形下，分类决策函数成为

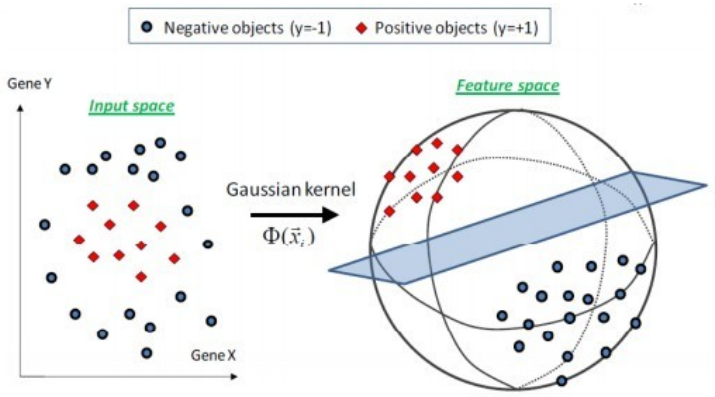
$$f(x)=\operatorname{sign}\left(\sum_{i=1}^{N_s}a_i^* y_i(x_i\cdot x+1)^p+b^*\right) \tag{7.89}$$

2. 高斯核

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

对应的支持向量机是高斯径向基函数分类器，可以将原始空间映射到无穷维，

*高斯核 $\kappa(x_1,x_2)=\exp\left(-\frac{\|x_1-x_2\|^2}{2\sigma^2}\right)$，这个核就是最开始提到过的会将原始空间映射为无穷维空间的那个家伙。不过，如果 σ 选得很大的话，高次特征上的权重实际上衰减得非常快，所以实际上（数值上近似一下）相当于一个低维的子空间；反过来，如果 σ 选得很小，则可以将任意的数据映射为线性可分——当然，这并不一定是好事，因为随之而来的可能是非常严重的过拟合问题。不过，总的来说，通过调控参数 σ，高斯核实际上具有相当高的灵活性，也是使用最广泛的核函数之一。下图所示的例子便是把低维线性不可分的数据通过高斯核函数映射到了高维空间：



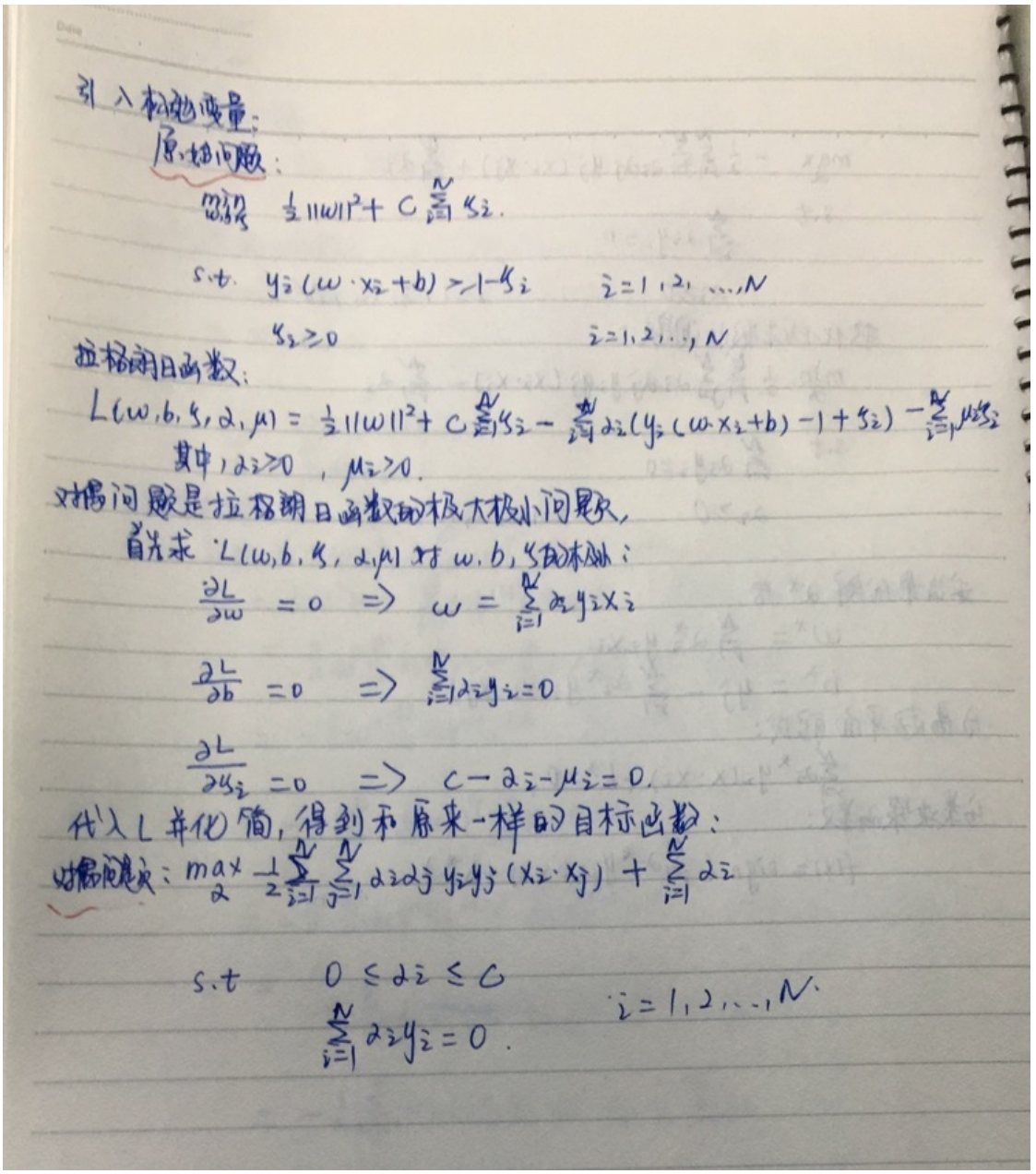
*线性核 $\kappa(x_1,x_2)=\langle x_1,x_2\rangle$，这实际上就是原始空间中的内积。这个核存在的主要目的是使得“映射后空间中的问题”和“映射前空间中的问题”两者在形式上统一起来了(意思是说，咱们有的时候，写代码，或写公式的时候，只要写个模板或通用表达式，然后再代入不同的核，便可以了，于此，便在形式上统一了起来，不用再分别写一个线性的，和一个非线性的)。

2.2.4 核函数的本质

1. 实际中我们经常遇到线性不可分的样例，常用做法是把样例特征映射到高维空间中，映射后相关特征被分开了，达到分类的目的
2. 如果遇到线性不可分就映射到高维空间，那么这个维度大小可能会非常非常高，计算非常复杂
3. 此时，要用到核函数，它的意义在于虽然也是讲特征从低维映射到高维，但是事先在低维上计算，而将实质上的分类效果表现在了高维上，避免了直接在高维空间中的复杂计算

2.3 使用松弛变量的SVM

训练数据中有一些特异点(outliner)，将这些特异点去除后，大部分数据是线性可分的。对于这种情况，允许数据点在一定程度上偏离超平面，引入一个松弛变量。



3.证明SVM

3.1 线性学习器-感知机算法

算法 2.1（感知机学习算法的原始形式）

输入：训练数据集 $T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$，其中 $x_i\in\mathcal{X}=\mathbf{R}^n$，$y_i\in\mathcal{Y}=\{-1,+1\}$，$i=1,2,\cdots,N$；学习率 $\eta\ (0<\eta\leqslant 1)$；

输出：w,b；感知机模型 $f(x)=\text{sign}(w\cdot x+b)$.

（1）选取初值 w_0,b_0

（2）在训练集中选取数据 (x_i,y_i)

（3）如果 $y_i(w\cdot x_i+b)\leqslant 0$

$$w\leftarrow w+\eta y_i x_i$$

$$b\leftarrow b+\eta y_i$$

（4）转至（2），直至训练集中没有误分类点. ■

这种学习算法直观上有如下解释：当一个实例点被误分类，即位于分离超平面的错误一侧时，则调整 w，b 的值，使分离超平面向该误分类点的一侧移动，以减少该误分类点与超平面间的距离，直至超平面越过该误分类点使其被正确分类.

那么，到底需要训练多少次呢？Novikoff定理告诉我们当间隔是正的时候感知机算法会在有限次数的迭代中收敛，也就是说Novikoff定理证明了感知机算法的收敛性，即能得到一个界，不至于无穷循环下去。

Novikoff定理：如果分类超平面存在，仅需在序列S上迭代几次，在界为

$\left(\frac{2R}{\gamma}\right)^2$

的错误次数下就可以找到分类超平面，算法停止。这里

$R=\max_{1\leq i\leq l}\|\mathbf{x}_i\|$

，

γ

为扩充间隔。根据误分次数公式可知，迭代次数与对应于扩充(包括偏置)权重的训练集的间隔有关。

3.2 非线性学习器

3.2.1 Mercer定理

Mercer定理 ：如果函数K是

$\mathbb{R}^n\times\mathbb{R}^n\to\mathbb{R}$

上的映射（也就是从两个n维向量映射到实数域）。那么如果K是一个有效核函数（也称为Mercer核函数），那么当且仅当对于训练样例

$\{x^{(1)},x^{(2)},\ldots,x^{(m)}\}$

，其相应的核函数矩阵是对称半正定的。

设A是n阶方阵，如果对任何非零向量X，都有X'AX≥0，其中X'表示X的转置，就称A为**半正定矩阵。**

3.3 损失函数

线性支持向量机学习还有另外一种解释，就是最小化以下目标函数：

$$\sum_{i=1}^{N}[1-y_i(w\cdot x_i+b)]_+ +\lambda\|w\|^2 \tag{7.57}$$

目标函数的第 1 项是经验损失或经验风险，函数

$$L(y(w\cdot x+b))=[1-y(w\cdot x+b)]_+ \tag{7.58}$$

称为合页损失函数（hinge loss function）. 下标“＋”表示以下取正值的函数.

$$[z]_+=\begin{cases}z, & z>0\\ 0, & z\leqslant 0\end{cases} \tag{7.59}$$

这就是说，当样本点(x_i,y_i)被正确分类且函数间隔（确信度）$y_i(w\cdot x_i+b)$大于 1 时，损失是 0，否则损失是$1-y_i(w\cdot x_i+b)$，注意到在图 7.5 中的实例点x_4被正确分类，但损失不是 0. 目标函数的第 2 项是系数为λ的w的L_2范数，是正则化项.

定理 7.4 线性支持向量机原始最优化问题：

$$\min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2+C\sum_{i=1}^{N}\xi_i \tag{7.60}$$

$$\text{s.t.} \quad y_i(w\cdot x_i+b)\geqslant 1-\xi_i, \quad i=1,2,\cdots,N \tag{7.61}$$

$$\xi_i\geqslant 0, \quad i=1,2,\cdots,N \tag{7.62}$$

等价于最优化问题

$$\min_{w,b} \quad \sum_{i=1}^{N}[1-y_i(w\cdot x_i+b)]_+ +\lambda\|w\|^2 \tag{7.63}$$

合页损失函数的图形如图 7.6 所示，横轴是函数间隔$y(w\cdot x+b)$，纵轴是损失. 由于函数形状像一个合页，故名合页损失函数.

图中还画出 0-1 损失函数，可以认为它是二类分类问题的真正的损失函数，而合页损失函数是 0-1 损失函数的上界. 由于 0-1 损失函数不是连续可导的，直接

优化由其构成的目标函数比较困难，可以认为线性支持向量机是优化由 0-1 损失函数的上界（合页损失函数）构成的目标函数．这时的上界损失函数又称为代理损失函数（surrogate loss function）．

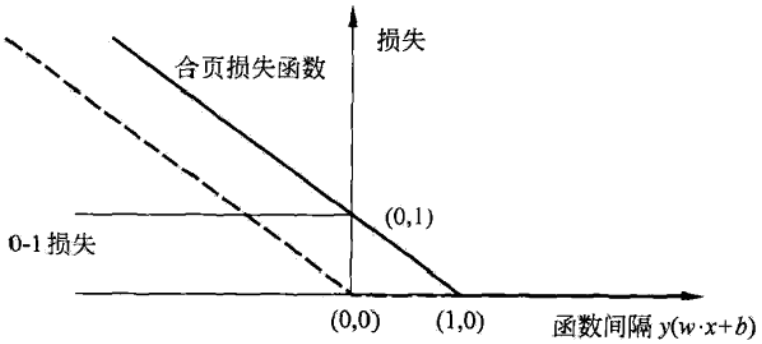


图 7.6　合页损失函数

图 7.6 中虚线显示的是感知机的损失函数 $[y_i(w\cdot x_i+b)]_+$．这时，当样本点 (x_i,y_i) 被正确分类时，损失是 0，否则损失是 $-y_i(w\cdot x_i+b)$．相比之下，合页损失函数不仅要分类正确，而且确信度足够高时损失才是 0．也就是说，合页损失函数对学习有更高的要求．

3.4 SMO算法

SMO 算法要解如下凸二次规划的对偶问题：

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(x_i,x_j)-\sum_{i=1}^{N}\alpha_i \tag{7.98}$$

$$\text{s.t.} \quad \sum_{i=1}^{N}\alpha_i y_i=0 \tag{7.99}$$

$$0\leqslant\alpha_i\leqslant C,\quad i=1,2,\cdots,N \tag{7.100}$$

SMO 算法是一种启发式算法，其基本思路是：如果所有变量的解都满足此最优化问题的 KKT 条件（Karush-Kuhn-Tucker conditions），那么这个最优化问题的解就得到了．因为 KKT 条件是该最优化问题的充分必要条件．否则，选择两个变量，固定其他变量，针对这两个变量构建一个二次规划问题．这个二次规划问题关于这两个变量的解应该更接近原始二次规划问题的解，因为这会使得原始二次规划问题的目标函数值变得更小．重要的是，这时子问题可以通过解析方法求解，这样就可以大大提高整个算法的计算速度．子问题有两个变量，一个是违反 KKT 条件最严重的那一个，另一个由约束条件自动确定．如此，SMO 算法将原问题不断分解为子问题并对子问题求解，进而达到求解原问题的目的．

注意，子问题的两个变量中只有一个是自由变量．假设 α_1，α_2 为两个变量，$\alpha_3,\alpha_4,\cdots,\alpha_N$ 固定，那么由等式约束 (7.99) 可知

$$\alpha_1=-y_1\sum_{i=2}^{N}\alpha_i y_i$$

如果 α_2 确定，那么 α_1 也随之确定．所以子问题中同时更新两个变量．

整个 SMO 算法包括两个部分：求解两个变量二次规划的解析方法和选择变量的启发式方法．

SMO的主要步骤：

第一步选取一对 α_i 和 α_j，选取方法使用启发式方法；

第二步，固定除 α_i 和 α_j 之外的其他参数，确定W极值条件下的 α_i，α_j 由 α_i 表示。

假定在某一次迭代中，需要更新 x_1，x_2 对应的拉格朗日乘子 α_1，α_2，那么这个小规模的二次规划问题写为：

$$L_s = \max_{\alpha}\{(\alpha_1+\alpha_2)+\sum_{i=3}^{n}\alpha_i-\frac{1}{2}\left\|\alpha_1 y_1\phi(\boldsymbol{x_1})+\alpha_1 y_1\phi(\boldsymbol{x_1})+\sum_{i=3}^{n}\alpha_i y_i\phi(\boldsymbol{x_i})\right\|^2\}$$

$$s.t.\,\alpha_1 y_1+\alpha_2 y_2=-\sum_{i=3}^{n}\alpha_i y_i, 0<\alpha_i<C, \forall i$$

那么在每次迭代中，如何更新乘子呢？引用这里的两张PPT说明下：

更新拉格朗日乘子α_1, α_2

- 步骤1：计算上下界L和H
  - $L = \max(0, \alpha_2^{old} - \alpha_1^{old}), H = \min(C, C + \alpha_2^{old} - \alpha_1^{old}), if\ y_1 \neq y_2$
  - $L = \max(0, \alpha_2^{old} + \alpha_1^{old} - C), H = \min(C, \alpha_2^{old} + \alpha_1^{old}), if\ y_1 = y_2$
- 步骤2：计算L_s的二阶导数
  - $\eta = 2\phi(\boldsymbol{x_1})^t\phi(\boldsymbol{x_2}) - \phi(\boldsymbol{x_1})^t\phi(\boldsymbol{x_1}) - \phi(\boldsymbol{x_2})^t\phi(\boldsymbol{x_2})$
- 步骤3：更新L_s

$$\alpha_2^{new} = \alpha_2^{old} - \frac{y_2(e_1 - e_2)}{\eta}$$

$$e_i = g^{old}(\boldsymbol{x_i}) - y_i$$

- 步骤4：计算变量α_2

$$\alpha^{temp} = \begin{cases} H, & \text{if}\alpha_2^{new} \geq H \\ \alpha_2^{new}, & \text{if } L \leq \alpha_2^{new} \leq H \\ L, & \text{if } \alpha_2^{new} \leq L \end{cases}$$

- 步骤5：更新α_1

$$\alpha_1^{new} = \alpha_1^{old} + y_1y_2(\alpha_2^{old} - \alpha^{temp})$$

知道了如何更新乘子，那么选取哪些乘子进行更新呢？具体选择方法有以下两个步骤：

步骤1：先“扫描”所有乘子，把第一个违反KKT条件的作为更新对象，令为a1；

步骤2：在所有不违反KKT条件的乘子中，选择使|E1 -E2|最大的a2进行更新，使得能最大限度增大目标函数的值（类似于梯度下降. 此外

$E_i = u_i - y_i$

，而

$u = \vec{w} \cdot \vec{x} - b$

，求出来的E代表函数ui对输入xi的预测值与真实输出类标记yi之差）。