Classification

方佳艳

1. 概述

分类问题的目标是将输入变量 x 分到 K 个离散的类别 C_k 中的某一类，也就是说，通过有标记的“训练集”学习一种划分的规则（我们希望规则越简单越好）。最常见的情况是，类别互相不相交，因此，每个输入被分到唯一的一个类别中。因此输入空间被划分为不同的决策区域（decision region），它的边界被称为决策边界（decision boundary）或者决策面（decision surface）。通常，我们考虑分类的线性模型。所谓分类线性模型，是指决策面是输入向量 x 的线性函数，因此被定义为 D 维输入空间中的（D-1）维超平面。如果数据集可以被线性决策面精确地分类，那么则称这个数据集是线性可分的（linearly separable）。经典的线性分类模型有感知器算法（Perceptron Algorithm），支持向量机（Support Vector Machine）。

2. 分类的线性模型

划分规则应当越简单越好，有以下几点原因：

（1）对于新的数据容易计算标记值；

（2）根据 VC-dimension 和“$\varepsilon - net$”：

“规则”越简单 $\Rightarrow VC-\dim$越低 $\Rightarrow$“$\varepsilon - net$”越小

反之，如果“规则”太复杂 $\Rightarrow VC-\dim$越高 $\Rightarrow$易出现数据过拟合问题

因此，我们以下均考虑分类的线性模型。

2.1. 判别函数

判别函数是一个以向量 x 为输入，把它分配到 K 个类别中的某一个类别（以二分类为例，正类别，负类别）的函数。很多时候我们会考虑线性判别函数（linear discriminant function），即那些决策面是超平面的判别函数。

线性判别函数的最简单形式是输入向量的线性函数，即

$$y(x) = w^T x + w_0$$

其中，w 被称为权向量，w_0 被称为偏置。偏置的相反数有时被称为阈值。对于一个输入向量 x，如果 $y(x) \geq 0$，那么它被分到正类别（positive class），否则被分到负类别（negative class）。对应的决策边界由 $y(x) = 0$ 确定，它对应于 D 维空间的一个（D-1）维的超平面。

考虑两个点 x_A 和 x_B，两个点都位于决策面上。由于 $y(x_A)=y(x_B)=0$，我们有 $w^T(x_A-x_B)=0$，因此向量 w 与决策面内的任何向量都正交，从而 w 确定了决策面的方向。类似的，如果 x 是决策面内的一个点，那么 $y(x)=0$，因此从原点到决策面的垂直距离为：

$$\frac{w^T x}{\|w\|}=-\frac{w_0}{\|x\|}$$

因此我们看到了偏置参数 w_0 确定了决策面的位置。图 2.1 给出了 D=2 的情况下的这些性质。

此外，我们注意到 $y(x)$ 的值给出了点 x 到决策面的垂直距离 r 的一个有符号的度量。考虑任意一点 x 和它在决策面上的投影 $x_\perp$，我们有

$$x=x_\perp+r\frac{w}{\|w\|}$$

将这个等式的两侧同时乘以 w^T，然后加上 w_0，并且使用 $y(x)=w^T+w_0$ 以及 $y(x_\perp)=w^T x_\perp+w_0=0$，我们有

$$r=\frac{y(x)}{\|w\|}$$

如图 2.1 所示。

此外，我们可以引入一个额外的虚“输入” $x_0=1$，这会使得记号更简洁。引入“虚”输入以后，我们定义 $\widetilde{w}=(w_0,w)$ 以及 $\widetilde{x}=(x_0,x)$，从而

$$y(x)=\widetilde{w}^T\widetilde{x}$$

在这种情况下，决策面是一个 D 维超平面，并且这个超平面会穿过 D+1 维扩展输入空间的原点。

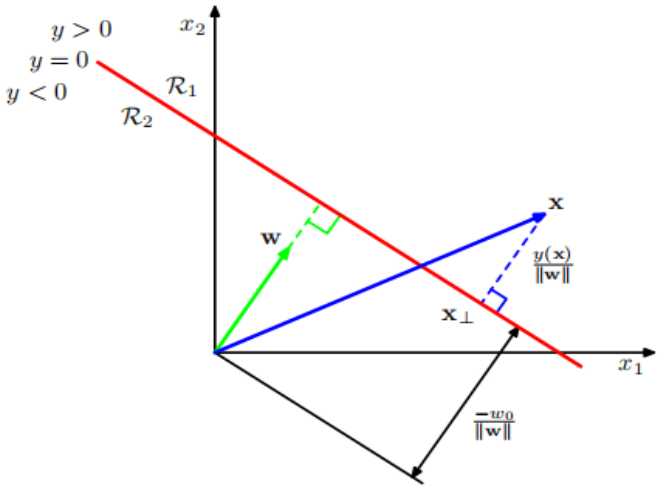


图 2.1

2.2. 感知器算法（Perceptron Algorithm）

线性判别模型的一个典型例子是 Rosenblatt(1962) 提出的感知器算法。它在模式识别算法的历史上占有重要的地位。它对应于一个二分类的模型，这个模型中，输入向量 x 首先使用一个固定的非线性变换得到一个特征向量 $\phi(x)$（将数据点上升到一个更高维度的空间中以寻求线性分类器，具体内容见 2.3 部分），这个特征向量然后被用于构造一个一般的线性模型，形式为

$$y(x)=f\left(w^T\phi(x)\right)$$

其中非线性激活函数 $f(\cdot)$ 是一个阶梯函数，形式为

$$f(a)=\begin{cases}+1\ , & a\geq 0\\ -1\ , & a<0\end{cases}$$

向量 $\phi(x)$ 通常包含一个偏置分量 $\phi_0(x)=1$。对于二分类问题，对于标记变量的表示方法为 $y\in\{0,1\}$，这对于概率模型来说很合适。然而，对于感知器而言，选择 $y\in\{+1,-1\}$，用 $y=+1$ 来表示正类别， $y=-1$ 来表示负类别更方便，这与激活函数的选择相匹配。

我们考虑一个误差函数，被称为感知器准则（perceptron criterion）。为了推导这个函数，我们寻找一个权向量 w 使得对于正类别中的数据 x_n 都 有 $w^T\phi(x_n)>0$，而对于负类别中的数据 x_n 都有 $w^T\phi(x_n)<0$。使用 $y\in\{+1,-1\}$ 这种标记变量的表示方法，我们要做的就是使得所有的模式都满足 $w^T\phi(x_n)y_n>0$。对于正确分类的模式，感知器准则赋予零误差，而对于误分类的模式 x_n，它试着最小化 $-w^T\phi(x_n)y_n$。因此，感知器准则为

$$E_P(w)=-\sum_{n\in M}w^T\phi_n y_n$$

其中， $\phi_n=\phi(x_n)$，M 表示所有误分类数据点的集合。某个特定的误分类数据点对于误差函数的贡献是空间中数据点被误分类的区域中 w 的线性函数，而在正确分类的区域，误差函数等于零。总的误差函数是分段线性的。

我们现在对这个误差函数使用随机梯度下降算法。这样，权向量 w 的变化为

$$w^{(\tau+1)}=w^{(\tau)}-\eta\nabla E_P(w)=w^{(\tau)}+\eta\phi_n y_n$$

其中 η 是学习率参数，τ 是一个整数，表示算法运行次数。如果我们将 w 乘以一个常数，那么感知器函数 $y(x,w)$ 不变，因此不失一般性我们可以令学习率参数 $\eta=1$。注意，随着训练过程中权向量的不断改变，误分类的数据也会改变。 感知器学习算法可以简单地表示如图 2.2。我们反复对于训练数据进行循环处理，对于每个数据 x_n，我们计算感知器函数。如果数据被正确分类，那么权向量保持不变，如果数据被分类错误，那么对于正类别，我们把向量 $\phi(x_n)$ 加到当前对于权向量 w 的估计值上，而对于负类别，我们从 w 中减掉向量 $\phi(x_n)$。图 2.2 说明了感知器算法。

如果我们考虑感知器算法中一次权值更新的效果，我们可以看到，一个误分类数据点对于误差函数的贡献会逐渐减小。因为

$$-w^{(t+1)T}\phi_n y_n = -w^{(t)T}\phi_n y_n - (\phi_n y_n)^T \phi_n y_n < -w^{(t)T}\phi_n y_n$$

其中，我们令$\eta = 1$，并且使用了不等式$\|\phi_n y_n\|^2 > 0$。然而，这并不表明其他的误分类的数据点对于误差函数的贡献会减小。此外，权向量的改变会使得某些之前正确分类的样本点变成误分类的点。因此，感知器学习规则并不保证在每个阶段都会减小整体的误差函数。

在引入以下定理之前，先定义 Margin 的概念：

γ_1为所有正类别数据点中距离超平面最近的点的距离值；

γ_2为所有负类别数据点中距离超平面最近的点的距离值。

超平面的 Margin γ 定义为$\gamma = \min\{\gamma_1, \gamma_2\}$，直觉上，$\gamma$ 越大越好。

感知器收敛定理：

感知器算法最多循环$\left(\frac{1}{\gamma^*}\right)^2$次，其中

$$\gamma^* = \max_{\gamma}\{\text{分开正负类别数据点的超平面的Margin}\}$$

证明：令w^*为取到最大 Margin 时对应的超平面的法向量。假设数据集在原始空间中是线性可分的，即$\phi_i = \phi(x_i) = x_i, i = 1, \cdots, N$。首先考虑$\langle w, w^* \rangle$

$$\begin{aligned}\langle w, w^* \rangle &\Rightarrow \langle w + y_i x_i, w^* \rangle \\ &= \langle w, w^* \rangle + \langle y_i x_i, w^* \rangle \\ &\geq \langle w, w^* \rangle + \gamma^*\end{aligned}$$

其中，$\langle y_i x_i, w^* \rangle$是$x_i$到$\langle x, w^* \rangle = 0$的距离

于是，经过 t 次循环后，$\langle w, w^* \rangle \geq t\gamma^*$。其次我们考虑$\|w\|^2$

$$\begin{aligned}\|w\|^2 &\Rightarrow \langle w + y_i x_i, w + y_i x_i \rangle \\ &= \langle w, w \rangle + 2y_i \langle w, x_i \rangle + \|x_i\|^2 \\ &\leq \|w\|^2 + 1 \\ &\Rightarrow \text{经过}t\text{次循环后}, \|w\|^2 \leq t\end{aligned}$$

于是，我们有

$$t\gamma^* \leq \langle w, w^* \rangle \leq \left\langle w, \frac{w}{\|w\|} \right\rangle = \|w\| \leq \sqrt{t}$$

$$\Rightarrow t\gamma^* \leq \sqrt{t} \Rightarrow t \leq \left(\frac{1}{\gamma^*}\right)^2$$

证毕。

感知器收敛定理表明，如果存在一个精确的解（即，训练数据集是线性可分的），那么感知器算法可以保证在有限步骤内找到一个精确解。但是需要注意的是，达到收敛状态所需的步骤数量可能非常大，并且在实际应用中，在达到收敛状态之前，我们不能够区分不可分问题与缓慢收敛问题。即使数据集是线性可分的，也可能有多个解，并且最终哪个解会被找到依赖于参数的初始化以及数据点出现的顺序。此外，对于线性不可分的数据集，感知器算法永远不会收敛。

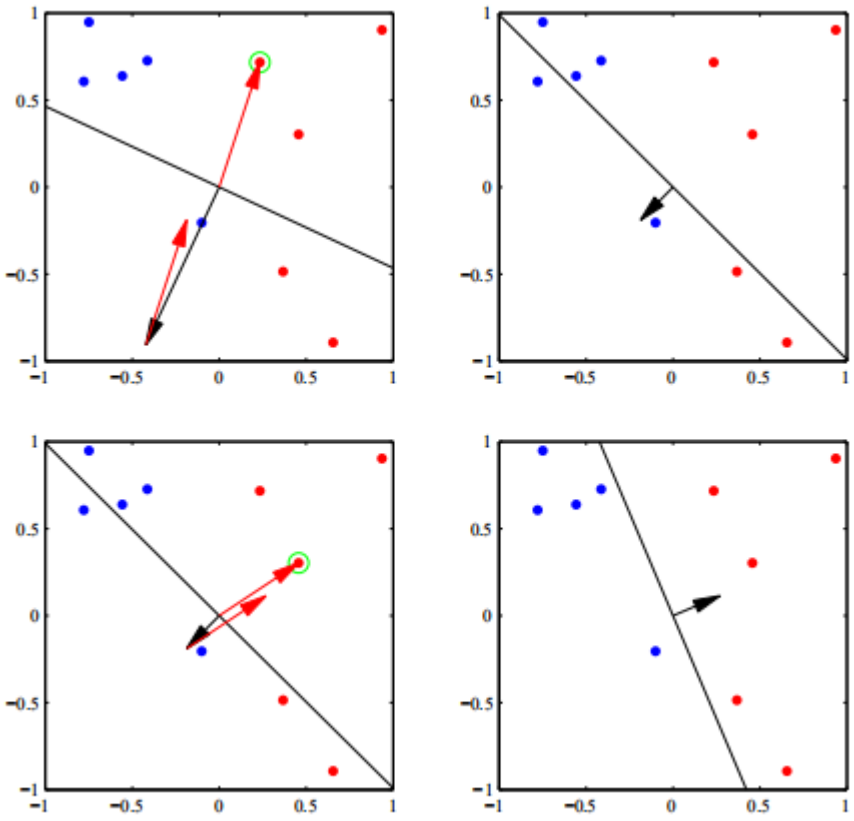


图 2.2

2.3. 支持向量机（Support Vector Machine）

2.3.1. 数据的特征转换

当数据集在原始的表示空间中不是线性可分时，就会引入 kernel 的工具。Kernel 的本质就是内积。和之前的 PCA,JL 变换的降维思想相反，kernel 的方法是反其道行之：在原有维度太低而不足以将数据集线性分开时，即，当在原始的 d 维空间中找不到超平面将数据进行分类时，就会考虑将原始的数据集上升到一个更高维度的空间中。

考虑一个二维空间的直角坐标系，以(x_0, y_0)为圆心，以r为半径的圆，分布在圆内的点记为正类别，分布在圆外的点记为负类别。显然，在二维空间中，这样分布的数据集无法通过超平面进行线性分类。

圆的方程：$(x-x_0)^2+(y-y_0)^2=r^2$，将方程展开得，

$x^2-2x_0x+y^2-2y_0y+x_0^2+y_0^2-r^2=0$，改写成内积的形式如下：

$$\left\langle (x^2, x, y^2, y, xy, 1), (1, -2x_0, 1, -2y_0, 0, x_0^2+y_0^2-r^2) \right\rangle = 0$$

于是，就从之前的二维向量表示上升到了 6 维向量表示，从 6 维空间的角度看，上述内积为 0 的等式代表了以$(1, -2x_0, 1, -2y_0, 0, x_0^2+y_0^2-r^2)$作为法向量$w$的超平面。

该例说明，在 2 维空间不足以将近似圆分布的数据集线性分开时，我们可以在一个 6 维的向量空间中寻找到一个超平面将原始数据集分开。该过程称为对数据集的核化/特征转换（feature transformation）。值得注意的是，在进行升维的过程中，一个很大的缺陷是它的计算复杂度上升了。在上述例子中，我们将坐标向量(x, y)替换成了$(x^2, x, y^2, y, xy, 1)$，改变 6 维向量每个维度前面的系数不影响结果，因为我们可以相应地来放缩向量每个维度的系数。于是我们可以通过改变 6 维向量前面的系数来降低高维空间中向量内积的计算量。

如何设置 6 维向量$(x^2, x, y^2, y, xy, 1)$每个维度变量前的系数：

$u_1=(x_1, y_1) \Rightarrow (\alpha_1 x_1^2, \alpha_2 x_1, \alpha_3 y_1^2, \alpha_4 y_1, \alpha_5 x_1 y_1, \alpha_6) = \phi(u_1)$

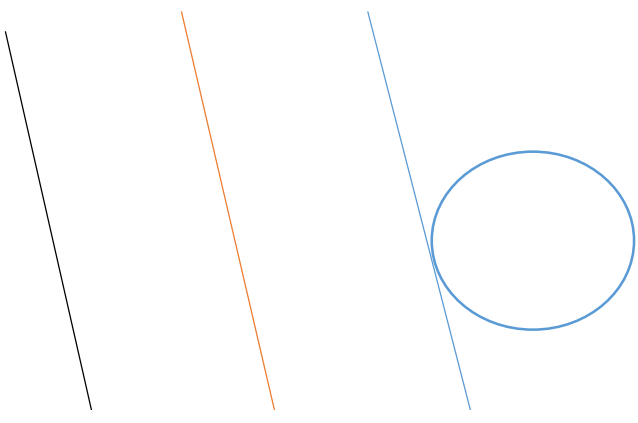
$u_2=(x_2, y_2) \Rightarrow (\alpha_1 x_2^2, \alpha_2 x_2, \alpha_3 y_2^2, \alpha_4 y_2, \alpha_5 x_2 y_2, \alpha_6) = \phi(u_2)$

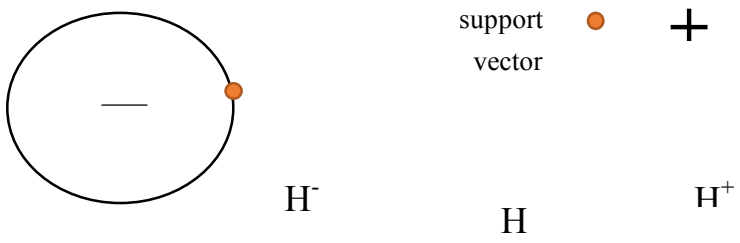
目标：通过设置$\alpha_1, \cdots, \alpha_6$的值来快速在 6 维空间中计算内积$\langle \phi(u_1), \phi(u_2) \rangle$。将$\langle \phi(u_1), \phi(u_2) \rangle$记为$K(u_1, u_2)$，称为 2 维空间向量$u_1$和$u_2$的核函数（kernel function）。

$$\begin{aligned}
\text{考虑}(\langle u_1, u_2 \rangle + 1)^2 &= (x_1x_2 + y_1y_2 + 1)^2 \\
&= x_1^2x_2^2 + 2x_1x_2 + y_1^2y_2^2 + 2y_1y_2 + 2x_1y_1x_2y_2 + 1 \\
&= \left\langle (x_1^2, \sqrt{2}x_1, y_1^2, \sqrt{2}y_1, \sqrt{2}x_1y_1, 1), (x_2^2, \sqrt{2}x_2, y_2^2, \sqrt{2}y_2, \sqrt{2}x_2y_2, 1) \right\rangle \\
&= K(\phi(u_1), \phi(u_2))
\end{aligned}$$

于是，已知 2 维空间内积$\langle u_1, u_2 \rangle$后，就能直接得到 6 维空间的内积，将数据的表示从 2 维空间上升到 6 维空间后，内积的计算量并没有变大。值得注意的是，这样的关于数据表示的特征转换技术同样适用于无限维空间。我们并不需要知道$\phi(u_1)$和$\phi(u_2)$的具体表达形式，可能是无限维空间中的向量，我们只需知道$\phi(u_1)$和$\phi(u_2)$的核函数$K(\phi(u_1), \phi(u_2))$的形式即可计算出$\phi(u_1)$和$\phi(u_2)$的内积。

2.3.2. Support Vector Machine（支持向量机）





确定了这些 support vector 之后也就确定了超平面 H。可以想象点集可以被凸包（convex hull）或者多面体（polytope）包围。

R^2：polygon （多边形） R^3：polyhedron R^d：polytope

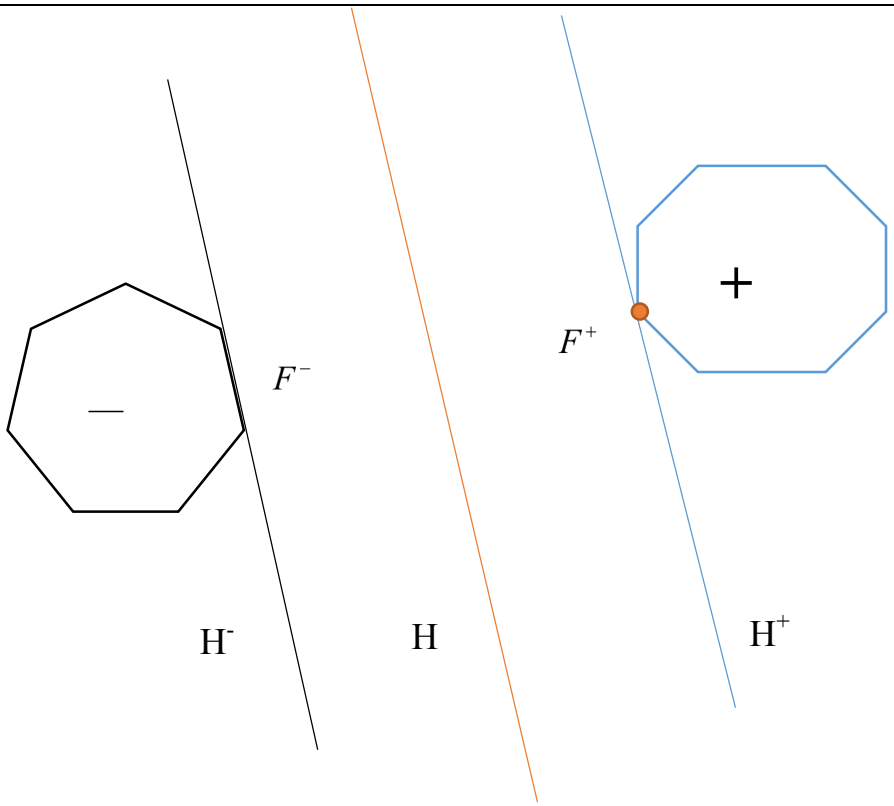
Polytope 的两个概念：

Facet：(d-1)-dim 的结构，可以看成是 d-1 维的凸包

Face：任何 k-dim 的结构，其中 $1 \le k \le d-1$，即任何 k 个顶点所包住的小区域。

设 margin 最大的超平面 H 对应的两个超平面 H^-和 H^+与两个凸包的接触区域分别为 F^-和 F^+。显然，接触区域为这两个 polytope 的 face。然后将 F^+投影到超平面 H^-上，记为 $\overline{F}^+$。

断言： $\overline{F}^+ \cap F^- \neq \varnothing$



证明思路：假设 $\overline{F}^+ \cap F^- = \varnothing$，则我们可以通过旋转超平面使得 $\overline{F}^+ \cap F^- \neq \varnothing$ 从而得到一个 margin 更大的超平面。

于是，我们可以在 F^+区域中找到一个点 q^+，使得该点刚好投影在 F^-上，记为 q^-。

设：正类别的点集为 $P^+ = \{u_1, u_2, \cdots, u_n\}$，负类别的点集为 $P^- = \{v_1, v_2, \cdots, v_m\}$

显然，q^+和q^-都在这两个凸包中，从而，

$$q^+ = \sum_{i=1}^{n} \alpha_i u_i, \quad \forall \alpha_i \geq 0, \quad \sum_{i=1}^{n} \alpha_i = 1$$

$$q^- = \sum_{j=1}^{m} \beta_j v_j, \quad \forall \beta_j \geq 0, \quad \sum_{j=1}^{m} \beta_j = 1$$

$$\begin{aligned} q^+ - q^- &= \sum_{i=1}^{n} \alpha_i u_i - \sum_{j=1}^{m} \beta_j v_j \\ &= \sum_{i=1}^{n} \alpha_i \left(\sum_{j=1}^{m} \beta_j \right) u_i - \sum_{j=1}^{m} \beta_j \left(\sum_{i=1}^{n} \alpha_i \right) v_j \\ &= \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j (u_i - v_j) \end{aligned}$$

其中，$\forall \alpha_i \beta_j \geq 0$并且$\sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j = 1$，因此，

$q^+ - q^-$是$u_i - v_j (i = 1, \cdots, n; j = 1, \cdots, m)$的一个凸组合，

即 $q^+ - q^-$ 在 $\{u_i - v_j | 1 \leq i \leq n; j \leq 1 \leq m\}$ 的凸包里，这个凸包称为 P^+和P^-的 Minkowski Difference，简记为 $MD(P^+, P^-)$。

易知，$\left\| q^+ - q^- \right\| = \min\{\|h\| h \in MD(P^+, P^-)\}$，$q^+ - q^-$是最佳超平面的法向量。

因此，我们在 $MD(P^+, P^-)$ 里面找到的长度最小的向量就是 SVM 模型中最佳超平面的法向量，也就找到了最佳超平面。于是我们得到了以下结论：

结论：SVM 的最佳超平面的法向量 $\Leftrightarrow$ $MD(P^+, P^-)$ 中长度最小的向量

如何找到 $MD(P^+, P^-)$ 中长度最小的向量是很重要的一个优化问题，称为 Polytope Distance. 是一个二次优化问题。

Formulation：给定一个由 $\{P_1, \cdots, P_t\}$ 组成的一个 polytope，找在这个 polytope 里的一个点，离原点最近，即：

$$\min \left\| \sum_{i=1}^{t} s_i P_i \right\|^2$$

$$s.t. \sum_{i=1}^{n} s_i = 1, s_i \geq 0, i = 1, \cdots, t$$

3. 相关文献

[1] Bishop C M. Pattern recognition and machine learning[M]. springer, 2006.