机器学习讲义(L11-B)：支持向量机

授课教师：王贝伦 / 助教：张嘉琦，黄旭，谈笑，徐浩卿

1 软间隔支持向量机

1.1 线性不可分情况

前面我们考虑的是数据点线性可分的情况，即指两类数据可以用一个超平面完全分开。然而在实际问题中，数据完全线性可分是非常罕见的。大多数情况下都会有一定程度的线性不可分现象，其来源于测量数据点时的微小误差，或者是由于问题本身的性质导致线性不可分。此外，线性不可分也有很多种情况：如仅有少数几个点互相混入，大体上仍可用超平面分隔，此种情况可称为近似线性可分；如果无论如何选取超平面，都有相当部分的数据点被误分类，又或者是数据点实际上是由曲线分隔开的，此种情况则是完全线性不可分。

我们先考虑简单情形，改进原型支持向量机，使它能够处理一定程度上线性不可分的数据集。

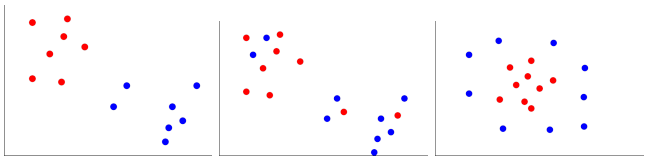

Figure 1: 线性可分示意图

如图所示，从左到右依次为线性可分情况、近似线性可分和完全线性不可分情况。

1.2 软间隔

在近似线性可分情况下，虽然存在超平面可以分隔开大部分的数据点，但总存在少量数据点被误分类。由于这些点的存在，这些分隔超平面无法满足原型 SVM 的约束条件，导致其无解。其主要原因是约束条件要求同一类的所有点都被分隔在同一侧，若我们可以容忍一定的误差，也许就可以求解出一个表现较好的分隔超平面。区别于原 SVM 的“硬”间隔条件，引入误差的 SVM 被称为“软”间隔 SVM。

基于上述推理，训练时需要在最大化间隔的同时最小化训练误差。而同时优化两个问题较为难以处理，超出了二次规划的范畴。我们参考机器学习常用的正则化方法，将训练误差用系数 C 缩放后与 $\theta^\top\theta$ 相加，得到优化问题如下：

$$\min \ \frac{1}{2}\theta^\top\theta + C\#\text{error}. \tag{1}$$

现在来考虑训练误差的形式。一种简单的想法是将误分类的点之数量作为训练误差，但这显然会破坏原目标函数的连续性与可导性。因此，我们考虑将误分类的点与其对应类的决策平面之间的距离作为误差。例如，假设类 A 在原 SVM 中的约束条件为 $\theta^\top x + b \geq 1$，若 x_k 属于类 A 且 $\theta^\top x_k + b < 1$，则此误分类点的误差 $\varepsilon_k = 1 - (\theta^\top x_k + b)$。注意到 $\varepsilon_k \geq 0$。此处的误差在数值上并非点到面的垂直距离，仅仅是一个松弛变量，但它的确与距离成正比，因而等价于点到面的距离。

注意到对于正确分类的点 x_m，$\theta^\top x_m + b \geq 1$，这意味着 $\varepsilon_m \leq 0$。为了能将误差适用于正确分类的数据点，人为定义它们的误差 $\varepsilon_m = 0$。最后写成凸优化的等价形式可得：

$$\begin{aligned} &\min_{\theta,\varepsilon_i} \ \frac{1}{2}\theta^\top\theta + C\sum_i \varepsilon_i, \\ &\text{s.t. } y_i(\theta^\top x_i + b) \geq 1 - \varepsilon_i, \\ &\quad \varepsilon_i \geq 0. \end{aligned} \tag{2}$$

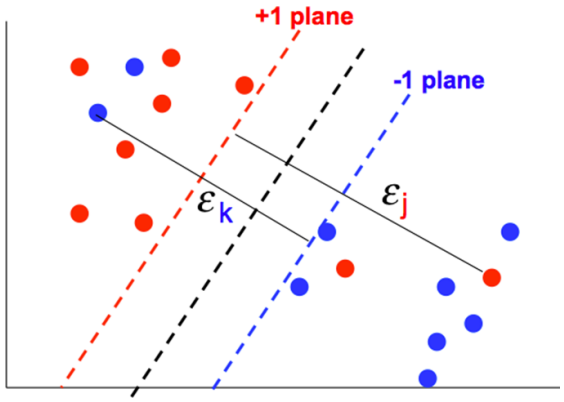

Figure 2: 软间隔误差示意图

如图，ε_j、ε_k 即为误分类点贡献的误差。

1.3 模型选择：权衡参数 C

优化问题 (2) 中，参数 C 并非优化得到，而是一个指定的超参数。与之前机器学习中的正则化方法一样，C 的选择也会影响模型的结果。

- C 数值较大：意味着错误分类的代价更加严重，因而最优解会偏向减少训练误差。但这意味着在测试集上的表现可能会更差，即有可能导致过拟合。
- C 数值较小：意味着错误分类的代价较小，而分类间隔更加重要。减少 C 的值来期望能得到更好的泛化效果，但训练误差会增加，有时会导致欠拟合。

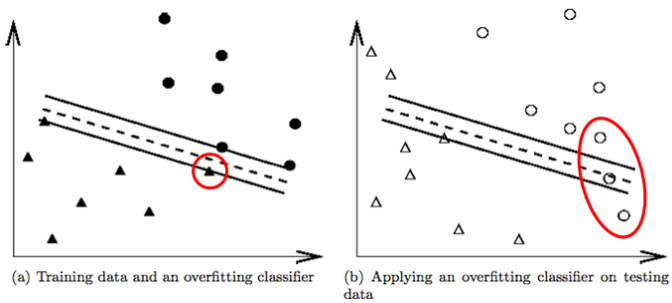

Figure 3: C 数值较大的决策边界

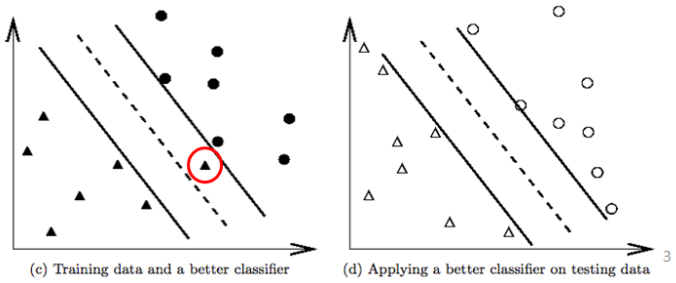

Figure 4: C 数值较小的决策边界

如图为不同的 C 对决策平面产生的影响。

2 支持向量机的对偶形式

在使用支持向量机的时候，为了加快求解过程，我们通常会使用支持向量机的对偶形式来求解。优化问题的对偶形式的解释详见附录 A。

2.1 硬间隔支持向量机的对偶形式

我们先考虑硬间隔支持向量机，即线性可分状态下的对偶形式。回顾硬间隔支持向量机，我们需要求解

$$\begin{aligned} \min_{\theta} \quad & \frac{1}{2}\theta^T\theta \\ \text{s.t.} \quad & y_i(\theta^T x_i) \geq 1, \text{ for } i = 1, 2, \cdots, N. \end{aligned} \tag{3}$$

我们可以简单的将公式（3）转化为拉格朗日乘子式

$$L(\theta, b, \alpha) = \frac{1}{2}\theta^T\theta - \sum_{i=1}^{N}\alpha_i[y_i(\theta^T x_i) - 1] \tag{4}$$

显然，硬间隔支持向量机的对偶形式为

$$\max_{\alpha \geq 0} \min_{\theta, b} \quad L(\theta, b, \alpha). \tag{5}$$

求解公式（5）时，我们从内层向外层求解。

- **第一步**：向将 $L(\theta, b, \alpha)$ 关于 θ, b 求导得到

$$\begin{aligned} \nabla_\theta L &= \theta - \sum_{i=1}^{N}\alpha_i y_i x_i = 0 \\ \nabla_b L &= -\sum_{i=1}^{N}\alpha_i y_i = 0 \end{aligned} \tag{6}$$

- **第二步**：将上面求导的结果代入 $L(\theta, b, \alpha)$ 可以得到

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{N}\alpha_2 = \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i^T x_j) \\ \text{s.t.} \quad & \alpha_i \geq 0 \\ & \sum_{i=1}^{N}\alpha_i y_i = 0 \text{ for } i = 1, 2, \cdots, N. \end{aligned} \tag{7}$$

  其中第一个约束来自于拉格朗日乘子本身的约束。而第二个约束来自于 $\nabla_b L = 0$。具体的推导过程可以参考引用资料。

- **第三步**：求解第二步得到的优化问题，得到最优解 α^*。利用 α^* 来计算 $\theta^* = \sum_i \alpha_i^* y_i x_i$ 与 b^*。比较传统的方法会使用 SMO 算法来快速求解 α^*。这里我们不详细解释 SMO 算法，有兴趣可以参考引用资料 [3,4]。

与原问题相比，对偶问题不再使用二次优化（quadratic programming）来求解，而是使用更加快速的 SMO 算法。因此通过将原问题转化为对偶形式，求解过程的时间成本大大减小了。

2.2 软间隔支持向量机的对偶形式

我们接着来考虑软间隔支持向量机，即线性不可分的情况。先回顾一下软间隔支持向量机的形式：

$$\begin{aligned} \min_{\theta} \quad & \frac{1}{2}\theta^T\theta + C\sum_{i=1}^{N}\varepsilon_i \\ \text{s.t.} \quad & y_i(\theta^T x_i) \geq 1 - \varepsilon_i \\ & \varepsilon_i \geq 0, \text{ for } i = 1, 2, \cdots, N. \end{aligned} \tag{8}$$

与硬间隔支持向量机类似，我们可以推导出软间隔支持向量机的对偶形式为

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{N}\alpha_2 = \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i^T x_j) \\ \text{s.t.} \quad & C \geq \alpha_i \geq 0 \\ & \sum_{i=1}^{N}\alpha_i y_i = 0, \text{ for } i = 1, 2, \cdots, N. \end{aligned} \tag{9}$$

公式（9）的求解也与间隔支持向量机一样，区别只是现在拉格朗日乘子 α 还拥有上界 C。在实际使用支持向量机时，我们需要通过交叉检验来选择最好的 C 的值。

3 使用核函数的支持向量机

虽然对于公式（9），仍然有求解 α^* 的算法。但是上界约束 $C \geq \alpha_i$ 终究会对优化过程造成困难。现在我们不使用松弛变量 ε_i，而是考虑从另一个方向来解决线性不可分的问题。

现实生活中大部分的数据其实都是线性不可分的，因此我们根本无法找到一个能用来分类的超平面。所以，通常我们会使用核方法将数据映射到高维空间中让数据变得稀疏，然后在高维空间寻找分类平面。核方法的详细解释可以参考附录 C。用 $K(x_i, x_j) =< \Phi(x_i), \Phi(x_j) >$ 表示将 x_i 与 x_j 映射到高维空间然后求内积。那么支持向量机的对偶形式就可以写成

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{N}\alpha_2 = \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j K(x_i, x_j) \\ \text{s.t.} \quad & \alpha_i \geq 0 \\ & \sum_{i=1}^{N}\alpha_i y_i = 0, \text{ for } i = 1, 2, \cdots, N. \end{aligned} \tag{10}$$

由于核函数 $K(x_i, x_j)$ 的存在，我们可以快速地计算出样本在高维空间中的内积。并且因为不存在松弛变量，我们求解对偶问题时不再需要考虑上界 C。不过在支持向量机中使用核函数时，我们需要在内存中存储一个 $N \times N$ 大小的核矩阵 $\mathcal{K}$。因此在大数据应用中，虽然支持向量机的时间成本很低，但是空间开销却会非常大。所以对于小规模数据支持向量机时很好的模型，但是对于大数据应用我们就需要考虑其他的分类方法了。

4 支持向量在对偶形式中的作用

我们在之前的讲义中解释了，在支持向量机中，那些分布在间隔平面上的点就叫做“支持向量”。在支持向量机的对偶形式中，支持向量还有更多的解释。

通过 KKT 条件（附录 B），我们可以得到

$$\alpha_i(y_i(\theta_i^T+b)-1)=0,\ \text{for } i=1,2,\cdots,N. \tag{11}$$

这个等式的成立有两种情况：

- 当 $\alpha_i=0$ 我们有 $y_i(\theta^T x_i+b)\neq 0$，即数据 x_i 不在分割平面 $\theta^T x_i+b=\pm 1$ 上。
- 当 $\alpha_i\neq 0$ 我们有 $y_i(\theta^T x_i+b)=0$，即数据 x_i 正好位于分割平面 $\theta^T x_i+b=\pm 1$ 上。

我们可以看到第二种情况正好符合我们对于支持向量的定义。图5展示了对于 10 个样本进行分类任务的结果。我们可以看到支持向量（标红的数据点）的对应 α 值不为 0。

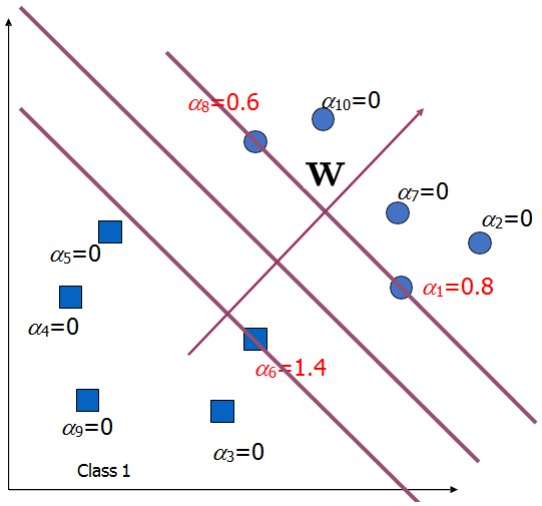

Figure 5: 支持向量。

我们现在回想之前讲过的利用对偶形式计算 θ。计算公式为 $\theta^*=\sum_i \alpha_i^* y_i x_i$。显然，只有支持向量对应不为 0 的 α_i^* 会对系数的计算造成影响。这正好符合我们对支持向量的定义。一般情况下，支持向量的数量之占样本总数量的一小部分，所以支持向量机的计算效率非常快。

附录 A：优化问题的对偶形式

我们考虑解这样一个优化问题

$$\begin{aligned}\min_x\ & f(x)\\ \text{s.t.}\ & h_i(x)\le 0,\ i=1,2,\cdots,m\\ & l_j(x)=0,\ \ j=1,2,\cdots,r\end{aligned} \tag{12}$$

我们通常称这个优化问题为**原问题（primal problem）**。在解这个问题 (12) 的时候，约束条件 $h_i(x)\le 0$ 和 $l_j(x)=0$ 会使得求解过程变得复杂。因此，为了简化优化问题，我们将约束条件整合到目标函数中：

$$\min_{x,u,v}\ f(x)+\sum_{i=1}^m u_i h_i(x)+\sum_{j=1}^r v_j l_j(x) \tag{13}$$

这个新的优化问题（13）被称为原问题的**拉格朗日乘子式**，其中 $u_i>0$ 和 $v_j>0$ 被称为**拉格朗日乘子**。一般来说，拉格朗日乘子都需要满足大于 0 这个条件。根据拉格朗日乘子式的形式，我们有

引理 4.1. 对于任意的可行解 x，我们有

$$f(x)=\sup_{u\ge 0,v} f(x)+\sum_{i=1}^m u_i h_i(x)+\sum_{j=1}^r v_j l_j(x). \tag{14}$$

特别地，当且仅当 u 满足 $u_i h_i(x)=0$ 时会取到上界。

引理4.1的证明非常简单，根据原问题的定义我们有 $h_i(x)\le 0$ 与 $l_j(x)=0$。显然 $f(x)+\sum_{i=1}^m u_i h_i(x)+\sum_{j=1}^r v_j l_j(x)\le f(x)$。显然，等式只有在 $u_i h_i(x)=0$ 时才会成立。根据这个定理，假设我们用 x^* 表示拉格朗日乘子式（13）的最优解，我们有

$$x^*=\inf_x \sup_{u,v\ge 0}\left[f(x)+\sum_{i=1}^m u_i h_i(x)+\sum_{j=1}^r v_j l_j(x)\right]. \tag{15}$$

最优解 x^* 的形式也很容易推导出来。对于一个可行解 x，显然我们有 $x^*=\inf_x f(x)$。根据引理4.1我们可以得到 $x^*=\inf_x \sup_{u,v\ge 0} f(x)+\sum_{i=1}^m u_i h_i(x)+\sum_{j=1}^r v_j l_j(x)$。另一方面，对于一个不可行解 x，显然我们有 $\sup_{u,v\ge 0} f(x)+\sum_{i=1}^m u_i h_i(x)+\sum_{j=1}^r v_j l_j(x)=\infty$，那么 $\inf_x f(x)=\infty$。综上所述，我们推导出了拉格朗日乘子式的最优解形式。

有了拉格朗日乘子式，我们就可以定义原问题的**对偶问题（dual problem）**为

$$d^*=\sup_{u,v\ge 0}\inf_x\left[f(x)+\sum_{i=1}^m u_i h_i(x)+\sum_{j=1}^r v_j l_j(x)\right] \tag{16}$$

其中 d^* 表示对偶问题的最优解。

对于对偶问题，我们可以推导出一个很重要的性质

引理 4.2. 一个优化问题的对偶问题一定是一个凸优化问题。

这也很容易证明。对偶问题考虑最大化函数 $g(u,v)=\inf_x\left[f(x)+\sum_{i=1}^m u_i h_i(x)+\sum_{j=1}^r v_j l_j(x)\right]$。由于 $u,v>0$，$h_i(x)\le 0$ 且 $l_j(x)=0$，关于拉格朗日乘子的函数 $g(u,v)$ 可以看成是一个凹函数。所以对偶问题 $\sup_{u,v\ge 0} g(u,v)$ 就是一个最大化凹函数的优化问题，即可以被看作是一个凸优化问题。对偶函数的凸性质正是我们选择对偶问题的原因。在实际解一些复杂的优化问题的时候，可能存在原问题并不是凸函数的情况。由于原问题的非凸性，我们可以解对偶问题的优化解 d^* 来作为原问题原问题最优解的近似。

引理 4.3. 原问题的最优解 x^* 与其对应对偶问题最优解 d^* 的关系为

$$d^*\le x^* \tag{17}$$

其中等号只有在原问题也是凸优化问题的情况下才成立。

在优化理论中，我们称 $d^*\le x^*$ 为优化问题的**弱对偶性**，而称 $d^*=x^*$ 为**强对偶性**。对偶问题的最优解可以被看作是原问题解的下界。特别地，当原问题为凸函数时，解对偶问题与解原问题的结果是一致的。

解对偶问题的方法也非常简单，由于对偶问题是凸函数，我们在求解时只需要求导即可。假设对偶问题为 $d(x,u,v)=\sup_{u,v\ge 0} g(u,v)$，第一步我们先对 x 求导 $\frac{\partial d(x,u,v)}{\partial x}$，令其等于 0 来推导出 x 关于 u,v 的表达式。第二步将表达式代入 $g(u,v)$，这样 $g(u,v)$ 就是只关于 u,v 的函数，对 u,v 分别求导等于 0 就能计算出 u^*,v^*,d^*。

附录 B：KKT 条件（Karush-Kuhn-Tucker conditions）

对于原问题与对偶问题的最优解 x^*, u^*, v^*，一定满足 KKT 条件。令 $L(x,u,v) = f(x) + \sum_{i=1}^{m} u_i h_i(x) + \sum_{j=1}^{r} v_j l_j(x)$，KKT 条件如下所示：

$$\begin{aligned} &\nabla_x L(x,u,v) = 0 \\ &\nabla_v L(x,u,v) = 0 \\ &u_i^* h_i(x^*) = 0 \\ &h_i(x^*) \le 0 \\ &u^* \ge 0 \end{aligned} \tag{18}$$

特别地，其中第 3 个条件被称为**对偶互补性**。这个条件说明如果 $\alpha^* > 0$，那么 $h_i(x) = 0$，相反如果 $h_i(x) < 0$，则 $\alpha = 0$。

附录 C：核方法（Kernel Method）

在分类问题的一些情况中，我们会发现在当前空间中，数据点之间并不存在明确的分界平面。但是，当我们把数据映射到一个更高维的空间之后，数据之间就会变得非常稀疏，也就更容易找到分界面。如图6的左图所示，我们并不能用一个曲线来划分这两类不同的数据点。但是当我们把数据映射到三维空间中之后（图6右图），我们就可以找到一个分类平面。核方法就是一种将低维空间中的不可分问题转化为高维空间可分问题的方法。

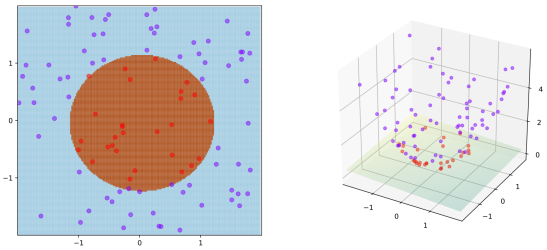

Figure 6: 核方法作用示例。

我们用 $\Phi(x) : \mathcal{X} \mapsto \mathcal{H}$ 来定义将 x 有空间 $\mathcal{X}$ 映射到高维希尔伯特空间 $\mathcal{H}$ 的一个映射函数。在原始的解空间中我们通常需要计算两个样本的内积。比如在支持向量机的对偶形式中，我们需要计算 $x_i^T x_j$。那么同样地在映射之后的高维空间中我们需要计算映射后样本的内积 $< \Phi(x_i), \Phi(x_j) >$。我们当然可以先计算 $\Phi(\cdot)$ 然后再计算内积，然而在高维空间之中计算内积的时间复杂度比较高，对于维度很高的空间来说尤其是如此。如果我们能直接知道 $< \Phi(x_i), \Phi(x_j) >$ 的形式，计算就会简化许多。假设我们定义 $K(x_i, x_j) =< \Phi(x_i), \Phi(x_j) >$，在统计学和机器学习领域中有很多预先设置并证明有效的函数形式：

- **多项式核**：$K(x_i, x_j) = (x_i^T x_j)^d$，$d \ge 1$ 为多项式的次数。
- **高斯核**：$K(x_i, x_j) = \exp(\frac{||x_i - x_j||^2}{2\sigma^2})$，其中参数 $\sigma > 0$ 为高斯核的带宽。
- **拉普拉斯核**：$K(x_i, x_j) = \exp(\frac{||x_i - x_j||}{\sigma})$，其中参数 $\sigma > 0$。
- **Sigmoid 核**：$K(x_i, x_j) = \tanh(\beta x_i^T x_j + \theta)$。其中 tanh 为双曲正切函数，$\beta > 0, \theta > 0$。

我们可以看到通过核函数我们可以直接利用低维空间的数据 x_i 与 x_j 而不是高维数据 $\Phi(x_i), \Phi(x_j)$ 来计算 $K(x_i, x_j)$，这样就省去了很多运算时间。不同的核函数会将数据映射到不同的高维空间。

我们这里简单对比一下多项式核与高斯核。多项式核的思想类似于我们之前讲过的多项式回归（polynomial regression），通过控制 d 的大小我们可以控制模型的复杂度，从而解决不可分的问题。高斯核的性能比多项式核要强。然而高斯核会将数据映射到无限维的空间中，这样导致模型的可解释性比较差。同时由于高斯核性能过于强大，还有过拟合的风险。

除了上述的一些定义好的核函数之外，我们也可以自己定义核函数。假设我们定义一个核矩阵

$$\mathcal{K} = \begin{bmatrix} < \Phi(x_1), \Phi(x_1) > & \cdots & < \Phi(x_1), \Phi(x_N) > \\ \vdots & \ddots & \vdots \\ < \Phi(x_N), \Phi(x_1) > & \cdots & < \Phi(x_N), \Phi(x_N) > \end{bmatrix} \tag{19}$$

只要我们定义的核函数对应的核矩阵 $\mathcal{K}$ 满足：(1) 对称；(2) 半正定，我们定义的核函数就是可行的。

引用

[1] UC Berkley Convex Optimization Lecture 11: https://www.stat.cmu.edu/~ryantibs/convexopt-F15/scribes/11-dual-gen-scribed.pdf.

[2] 支持向量机通俗导论：https://blog.csdn.net/v_july_v/article/details/7624837.

[3] Sequential Minimal Optimization: http://jupiter.math.nctu.edu.tw/~yuhjye/assets/file/teaching/2017_machine_learning/SMO%20algorithm.pdf.

[4] Platt, J., 1998. Sequential minimal optimization: A fast algorithm for training support vector machines.