斯坦福 ML 公开课笔记 8

本篇讲述了 SVM（Support Vector Machine,支持向量机）的剩余部分。即核技法（Kernels）、软间隔分类器（soft margin classifier）、对 SVM 求解的序列最小化算法（Sequential Minimal Optimization,SMO）以及 SVM 的一些应用。

由本篇可以看到，将笔记 7 中提到的原始/对偶问题引入到最有间隔分类器中，有两个好处，均在本章体现，其一是凡是在对偶最优问题中出现内积的地方，都可以用核替代，一定程度上可以解决非线性可分的问题，还有一个从根本上解决非线性可分问题的方法是软间隔分类器，也在本篇。其二是可以 SMO 算法对问题进行求解，非常高效。

核技法

回忆一下上篇笔记中得到的简化的最优问题，#1：

$$\max_\alpha \quad \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^{(i)}y^{(j)}\langle x^{(j)}, x^{(i)}\rangle$$

$$\text{s.t.} \quad \alpha_i \geq 0, \mathrm{i} = 1,2,\dots,\mathrm{m}$$

$$\sum_{i=1}^{m}\alpha_i y^{(i)} = 0$$

定义函数$\phi(x)$为向量之间的映射，一般是从低维映射到高维，比如在前面笔记中提到的房价和面积的关系问题中，可以定义ϕ为：

$$\phi([x]) = [x \quad x^2 \quad x^3]^T$$

其中，x 为面积。

这样，就可以将#1 问题中目标函数中的内积$\langle x^{(j)}, x^{(i)}\rangle$替换为$\langle \phi(x^{(j)}), \phi(x^{(i)})\rangle$的形式，这样就达到了将低维空间上的数据映射到高维空间上，使数据线性可分的概率变大，即不能保证在高维上一定是线性可分的，但一般情况下高维空间上比低维空间上更加线性可分。

但有些时候，经过ϕ映射后的向量的维度过高，导致映射后向量内积的计算复杂度过高。为了解决这个问题，我们正式引入核函数：

$$\mathrm{K(X,Z)} = \langle \phi(X), \phi(Z)\rangle \tag{1}$$

其中，X,Z 即为上面的$x^{(j)}, x^{(i)}$，为了简化表示，去掉了上标。

核函数的作用在于定义了核函数后，可以不用明确的定义出映射函数，就能计算两个向量在高维空间中的内积了，而且时间复杂度低。

下面举几个核函数的例子。

$$\mathrm{K(X,Z)} = (X^T Z)^2 = \left(\sum_{i=1}^{n} x_i z_i\right)\left(\sum_{j=1}^{n} x_j z_j\right) = \sum_{i=1}^{n}\sum_{j=1}^{n}(x_i x_j)(z_i z_j) \tag{2}$$

其对应的映射函数为

$$\phi(\mathrm{x}) = [x_1 x_1 \quad x_1 x_2 \quad \cdots \quad x_n x_{n-1} \quad x_n x_n]^T \tag{3}$$

一个相似的核函数如下：

$$\mathrm{K(X,Z)} = (X^T Z + c)^2 = \sum_{i,j=1}^{n}(x_i x_j)(z_i z_j) + \sum_{i=1}^{n}\sqrt{2c}x_i\sqrt{2c}z_i + c^2 \tag{4}$$

其对应的映射函数为

$$\phi(\mathrm{x}) = [x_1 x_1 \quad \dots \quad x_n x_n \quad \sqrt{2c}x_1 \quad \dots \quad \sqrt{2c}x_n \quad c]^T \tag{5}$$

一个更一般化的核函数如下：

$$\mathrm{K(X,Z)} = (X^T Z + c)^d \tag{6}$$

该核函数对应的映射函数的结果是一个$\binom{\mathrm{n+d}}{d}$大小的向量，向量中每个元素都是最高为 d 阶的变量的组合。

对于公式 2、4、6 中的核函数而言，虽然它们对应的映射函数的维度可能是 n^2 或者 n^d，意味着如果直接计算映射结果的内积的话复杂度分别为 $O(n^2)$或 $O(n^d)$，但是如果直接计算核函数的值，其复杂度均为 O(n)，这也体现了核函数降低计算量的好处。

直观上来看，如果ϕ(x)ϕ(z)在其对应的维度空间中位置接近，那么内积值 K 会很大，反之则内积会小。这意味着核函数 K 是一个向量 x 和向量 z 何等接近的度量函数，从而可以引出 SVM 中使用较广泛的高斯核：

$$\mathrm{K(x,z)} = \exp(-\frac{||x-z||^2}{2\sigma^2}) \tag{7}$$

值得一提的是，高斯核对应的映射函数ϕ是映射到无限维的。

那么，什么样的核函数才是正确的核函数，是不是所有的以 x，z 为变量的函数都能做核函数呢？即如何判断一个函数是不是能拆分成映射函数乘积的形式？前后两个问题等价的原因是核函数是由映射函数乘积得到的，因而如果核函数合法，那么必然能写成两个映射函数乘积的形式。

为了解决这个问题，首先定义核矩阵。对于一个数据集$\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$，定义一个 m*m 的矩阵 K（K 既代表核函数也代表核矩阵），K 中的每个元素定义如下：

$$K_{ij} = K(x^{(i)}, x^{(j)}) \tag{8}$$

对于核矩阵，有几个性质，首先很显然，$K_{ij}=K_{ji}$，核矩阵是一个对称（symmetric）矩阵。其次，对于任意的 m 维向量 z，可以得到：

$$\begin{aligned} z^T K z &= \sum_i \sum_j z_i K_{ij} z_j = \sum_i \sum_j z_i \phi(x^{(i)})^T \phi(x^{(j)}) z_j \\ &= \sum_i \sum_j z_i \left(\sum_k \phi_k(x^{(i)})\, \phi_k(x^{(j)}) \right) z_j \\ &= \sum_k \sum_i \sum_j z_i\, \phi_k(x^{(i)}) \phi_k(x^{(j)}) z_j \\ &= \sum_k \left(\sum_i z_i \phi_k(x^{(i)}) \right)^2 \geq 0 \end{aligned} \tag{9}$$

由此，因为 z 是任意向量，所以 K 是半正定（positive semi-definite）矩阵。事实上，这不仅是一个必要条件还是一个充分条件。因为存在一个 Mercer 定理：

给定一个 K：$\mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R}$，那么 K 是合法的核（又称 Mercer 核）的充分必要条件是对于任意一个有限数据集，对应的核矩阵是对称半正定矩阵。

对于核来说，不仅仅只存在于 SVM 内，对于任意的算法，只要计算时出现了内积的，都可以用核函数替代，从而提高在高维数据上的性能，比如感知器算法，代入后可发展为核感知器算法。这也是核函数被称为核技法的原因罢。

软间隔分类器

之前讲述最优间隔分类器时，一直强调数据是线性可分的。但是，当数据是线性不可分时，或者映射到高维空间后仍然不是线性可分，再或者即便是线性可分的但实际应用中不可避免出现噪声时，该如何处理呢？本节提供了一个比较通用的解法。

首先，对原始问题进行变形，得到#2：

$$min_{w,b} \quad \frac{1}{2}||w||^2 + \mathrm{C}\sum_{i=1}^{m}\varepsilon_i$$
$$\mathrm{s.t.} \quad y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1 - \varepsilon_i,\ \ i = 1,2,\dots,m$$
$$\varepsilon_i \geq 0, i = 1,2,\dots,m$$

由#2 可知，有些数据点可以被允许拥有比 1 小的几何间隔，但是这种情况要受到惩罚，C 为惩罚因子，是一个预设的参数。

其中，由目标函数的两部分，w 部分和惩罚部分，按照它们的指数可以分为多种组合。当 w 的指数为 n 时，称之为 Ln 正规化；当惩罚部分的指数为 n 时，称之为 Ln 损失；比如#2 中的目标函数即为 L2 正规化-L1 损失 SVC（Support Vector Classification）[1]。

按照上篇笔记中的将原始问题转化为对偶问题进行简化的例子，写出#2 对应的拉格朗日方程：

$$\mathrm{L(w,b,\varepsilon,\alpha,r)} = \frac{1}{2}||w||^2 + \mathrm{C}\sum_{i=1}^{m}\varepsilon_i$$
$$-\sum_{i=1}^{m}\alpha_i\left[y^{(i)}\left(w^T x^{(i)} + b\right) - 1 + \varepsilon_i\right] - \sum_{i=1}^{m} r_i\varepsilon_i \tag{10}$$

公式 10 中，α, r是拉格朗日乘子，均有不小于 0 的约束。

按照上篇笔记中的对偶问题的推导方式，先针对 w，b 最小化，然后再针对α最大化，得到新的对偶问题，即#3：

$$\max_{\alpha} \quad W(\alpha) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j\, y^{(i)}y^{(j)}\langle x^{(j)}, x^{(i)}\rangle$$
$$\mathrm{s.t.} \quad 0 \leq \alpha_i \leq \mathrm{C}, \mathrm{i} = 1,2,\dots,\mathrm{m}$$
$$\sum_{i=1}^{m}\alpha_i y^{(i)} = 0$$

与#1 对比得到，本问题只对α_i做了进一步的约束。求解得到α后，w 仍然按照公式$\mathrm{w} = \sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}$给出，但是截距 b 的得到方式要变化。

另一个发生变化的地方在于 KKT 中的互补条件，现在变为：

$$\alpha_i = 0 \Rightarrow y^{(i)}\left(w^T x^{(u)} + b\right) \geq 1 \tag{11}$$
$$\alpha_i = C \Rightarrow y^{(i)}\left(w^T x^{(u)} + b\right) \leq 1 \tag{12}$$
$$0 \leq \alpha_i \leq C \Rightarrow y^{(i)}\left(w^T x^{(u)} + b\right) = 1 \tag{13}$$

这些条件将在下一节用于判断 SMO 算法是否收敛。

至此，终于已经得到一个可以应用于实际的具体问题了（#1 是假设数据集线性可分，不符合实际），下面一节将对介绍对#3 问题求解的算法。

[1] Fan R E, Chang K W, Hsieh C J, et al. LIBLINEAR: A library for large linear classification[J]. The Journal of Machine Learning Research, 2008, 9: 1871-1874.

SMO 算法

坐标上升法

在介绍 SMO 算法之前，先介绍一个简化的但与 SMO 使用同一思想的算法，坐标上升法（Coordinate Ascent）。

先抛开 SVM 优化问题，看如下一个要解决的简单的新问题：

$$\max_{\alpha} W(\alpha_1, \alpha_2, \dots, \alpha_n)$$

对函数寻找最优值，之前已介绍了梯度下降法和牛顿法。现在介绍一种新方法：

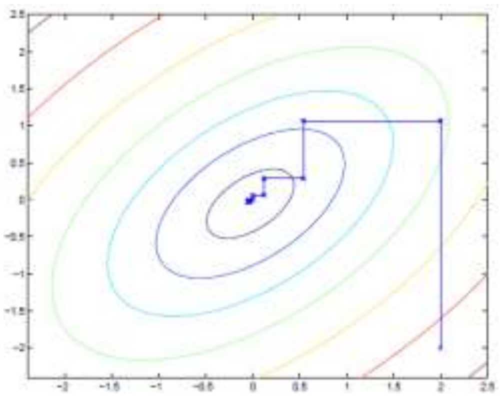
Loop Until Convergence：{

For　i=1,2…,m{

$$\alpha_i \coloneqq argmax_{\hat{\alpha}_i} W(\alpha_1, \dots, \alpha_{i-1}, \hat{\alpha}_i, \alpha_{i+1}, \dots, \alpha_n)$$

}

}



最内层循环中，当更新α_i时，保持其它的参数不变。

直观上看，这种方法比梯度下降和牛顿法迭代次数要多（一般情况下事实如此），但当能很快的求解$argmax$时，该算法仍是一个高效的算法，下面是一种运行样例图：

由图可见，当保持其它参数不变而只改变某个参数时，会使得参数的收敛方向都是平行于坐标轴的。

还有一点，这张图只有两个参数，所以才能在二维图中展示出来。

SMO 算法

SMO（Sequential Minimal Optimization）与坐标上升法不同的地方在于，由于#3 问题中有一个约束为$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$，使得当固定其他参数只改变一个参数的时候，发现剩余的那个参数是固定的，因而 SMO 算法每次选择两个参数进行优化。实际上，两个参数中可以将一个当做变量，另外一个当做该变量的函数，函数关系为：

$$\alpha_2 = y^{(2)}(-\sum_{i=3}^{m} \alpha_i y^{(i)} - \alpha_1 y^{(1)}) \tag{14}$$

公式 14 中使用$\alpha_1\alpha_2$，只是为了使符号简洁，该公式对于任意的$\alpha_i\alpha_j (i \neq j)$都适用。

SMO 算法描述为：

Repeat till Convergence{

Select two parameter $\alpha_i\alpha_j (i \neq j)$

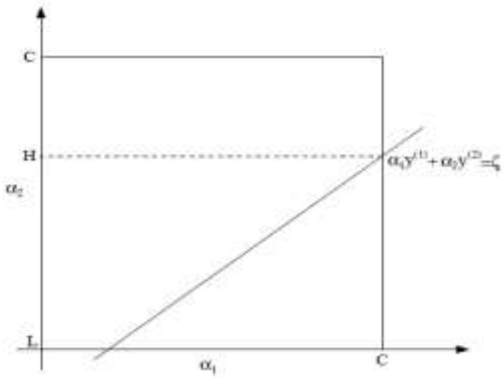
Optimize W(α) with respect to $\alpha_i\alpha_j$, holding other parameters fixed

}

循环中，在选择参数对时，使用一些启发式规则选择使全局函数增长最大的参数，具体

的启发式规则可以查看 John Platt[2]的 SMO 相关论文[3]。

前面提到，这种算法虽然迭代次数多，但当最优化时速度快，仍不失为一种高效的算法，那么对于 SVM 最优化问题，优化起来效率如何呢？对于#3 中的目标函数来说，当只保留两个参数变化时，代入公式 14 的关系，那么目标函数将变为一个二次函数，再加上$0 \le \alpha_i \le C$的条件，求最优值还是极为简单的。



当然，在计算最优值时，需要根据约束关系对自变量的取值范围进一步的计算，比如右图所示，假设以α_2为自变量，那么它的范围就是[L,H]。

至此，SVM 算法的内容就告一段落了。按照 Ng 的说法，笔记到现在才算是完成了 ML 的基础学习，大概有那么点只学了招式没学内功心法的意味吧。

SVM 的应用

SVM 作为一种分类器，自然在分类问题上大展手脚了。比如文本分类、图像分类等。下面列举几个 SVM 的实际应用。

比如手写数字识别问题，给定一张 16*16 的图片，上面写着 0-9 的 10 个数字，使用 SVM 进行判定。这本来是 NN（Neural Network）比较擅长的问题，但初次将 SVM 用于其上时效果之好令人惊讶，因为 SVM 没有图片识别的先验知识，只是依据像素就能达到很好的效果。补充一点，SVM 上使用高斯核和多项式核都能在该问题上达到和 NN 相当的效果。

再比如通过氨基酸序列对蛋白质的种类进行判别，假设有 20 中氨基酸，它们按照不同的序列可以组成不同的蛋白质，但这些氨基酸序列的长度差异很大，那么该如何设定特征呢？有一种方式是使用连续四个氨基酸序列出现次数作为向量，比如氨基酸序列 AABAABACDEF，那么可以得到 AABA:2, ABAA:1,…CDEF:1。据此，可以得到特征向量的长度为 20^4，如此大的特征向量，难以载入内存，有一种高效的动态规划算法可以解决此问题，该算法具体是什么，目前我还没查到。

与氨基酸序列识别蛋白质类似的是，字符串的识别，对于长度为 k 的字符串，如果以字母为特征，那么特征向量大小为 26^k，也需要利用 dp 算法进行解决。

[2] http://research.microsoft.com/en-us/people/jplatt/
[3] Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines.