Lecture 3：Kernel Support Vector Machine

> 课件链接：[Hsuan-Tien Lin - kernel support vector machine]
>
> Kernel Support Vector Machine(核支撑向量机)
>
> - Kernel Trick: 核技巧
> - Polynomial Kernel: 多项式核
> - Gaussian Kernel: 高斯核
> - Comparison of Kernels: 核的比较

1. Kernel Trick: 核技巧

上一章最后我们提到，SVM的对偶形式看似将原始最优化问题从$\tilde{d}+1$个变量与N个条件转化为N个变量与$N+1$个条件的最优化问题，摆脱了对$\tilde{d}$的依赖，但最后却发现，$\tilde{d}$其实并没有消失——被隐藏在了Dense的Q_D矩阵中：$q_{n,m}=y_n y_m \mathbf{z}_n^T\mathbf{z}_m$。也就是，仍然没有避开在Z空间中进行高维度内积运算。我们知道：

$$\mathbf{z}_n^T\mathbf{z}_m=\Phi(\mathbf{x}_n)^T\Phi(\mathbf{x}_m)$$

可见，计算Q矩阵的每个元素，大致可以分为两步：

- Step 1 - transform：进行特征转换，将X空间内的数据转换到Z空间；
- Step 2 - inner product：在Z空间中做内积。

我们是否可以**将两步合并为一步**？下面用**二阶多项式转换**Φ_2**(2nd order polynomial transform)**为例进行探索：

$$\Phi_2(\mathbf{x})=(1,x_1,x_2,\cdots,x_d,x_1^2,x_1x_2,\cdots,x_1x_d,x_2x_1,x_2^2,\cdots,x_2x_d,\cdots,x_d^2)$$

这里将x_1x_2与x_2x_1都写了进去，只是为了之后容易化简。现在，我们将计算任意两个X空间的样本$\mathbf{x}$与$\mathbf{x}'$转换后在Z空间里的内积：

$$\begin{aligned}\Phi_2(\mathbf{x})^T\Phi_2(\mathbf{x}')&=1+\sum_{i=1}^{d}x_ix_i'+\sum_{i=1}^{d}\sum_{j=1}^{d}x_ix_jx_i'x_j'\\&=1+\sum_{i=1}^{d}x_ix_i'+\sum_{i=1}^{d}x_ix_i'\sum_{j=1}^{d}x_jx_j'\\&=1+\mathbf{x}^T\mathbf{x}'+(\mathbf{x}^T\mathbf{x}')(\mathbf{x}^T\mathbf{x}')\end{aligned}$$

可见，对于二阶多项式转换，我们不必"按部就班"——先做转换再做内积，可以使用"**偷吃步**"——直接在X空间做内积，然后再做一步多项式乘法即可。"按部就班"的时间复杂度是$O(d^2)$，"偷吃步"的时间复杂度是$O(d)$！

因此，我们有理由相信，对于一个转换Φ，可能存在一个**核函数K**与之对应，使得：

$$\Phi(\mathbf{x})^T\Phi(\mathbf{x}')=K(\mathbf{x},\mathbf{x}')$$

上面的例子里：

$$K_{\Phi_2}(\mathbf{x},\mathbf{x}')=1+\mathbf{x}^T\mathbf{x}'+(\mathbf{x}^T\mathbf{x}')^2$$

因此，Kernel的实质就是将转换与内积合并至一步进行运算——**Kernel = Transform + Inner Product**

通过核技巧，我们便可以大大简化Q矩阵元素的计算：

$$q_{n,m} = y_n y_m \Phi(\mathbf{x}_n)^T \Phi(\mathbf{x}_m) = y_n y_m K(\mathbf{x}_n, \mathbf{x}_m)$$

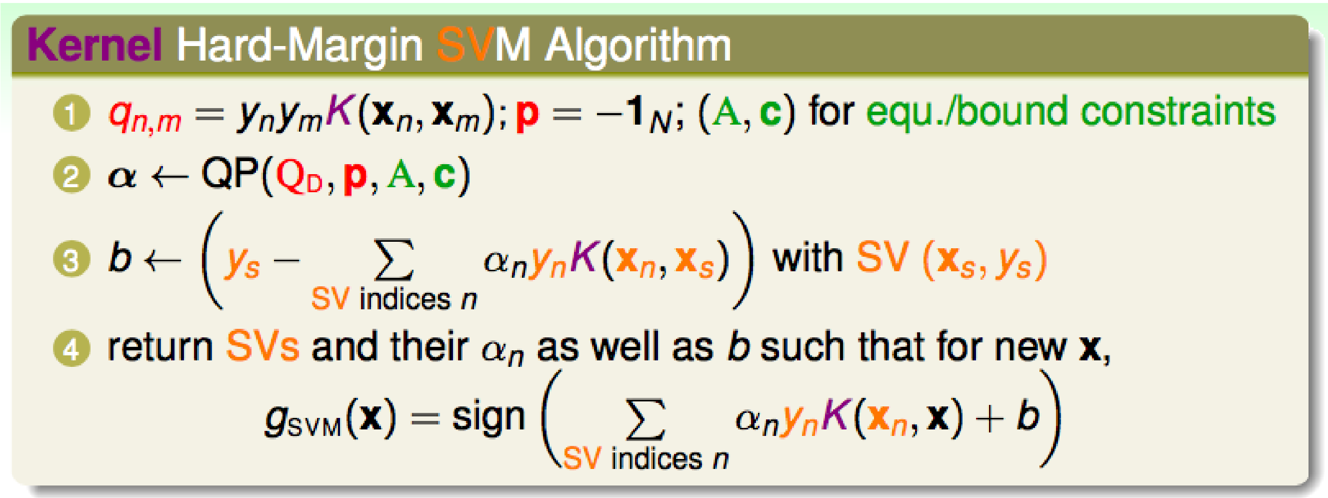
对于最优的b：

$$\begin{aligned} b &= y_s - \mathbf{w}^T \mathbf{z}_s \\ &= y_s - (\sum_{n=1}^{N} \alpha_n y_n \mathbf{z}_n)^T \mathbf{z}_s \\ &= y_s - \sum_{n=1}^{N} \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}_s) \end{aligned}$$

对于最优的SVM分类器：

$$\begin{aligned} g_{SVM}(\mathbf{x}) &= sign(\mathbf{w}^T \Phi(\mathbf{x}) + b) \\ &= sign(\sum_{n=1}^{N} \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}) + b) \end{aligned}$$

总之，使用kernel trick，我们真正做到了摆脱对Z空间维度$\tilde{d}$的依赖。**Kernel Hard-Margin SVM Algorithm**如下：



- 第一步的时间复杂度：$O(N^2) \cdot (kernel\ evaluation)$
- 第二步：N变量N+1约束的QP问题
- 第三步&第四步的时间复杂度：$O(\#SV) \cdot (kernel\ evaluation)$

2. Polynomial Kernel: 多项式核

从上一节的二阶多项式核函数，我们可以推导出更加一般化的二阶多项式核函数：

$$\Phi_2(\mathbf{x}) = (1, \sqrt{2\gamma} x_1, \cdots, \sqrt{2\gamma} x_d, \gamma x_1^2, \cdots, \gamma x_d^2)$$

对应的核函数为($\gamma > 0$)：

$$K_2(\mathbf{x}, \mathbf{x}') = 1 + 2\gamma \mathbf{x}^T \mathbf{x}' + \gamma^2 (\mathbf{x}^T \mathbf{x}')^2 = (1 + \gamma \mathbf{x}^T \mathbf{x}')^2$$

这里的Φ_2与上一节没有γ的Φ_2，power是相同的——并不是说这里的核函数多了一个参数，就表示它的能力更大——因为他们将原数据转换到的特征空间的维度一样。然而，一些东西的确发生了变化，比如**内积(inner product)**。内积改变了，那么一些几何性质也就跟着改变了，因此得到的SVM分类器可能会不一样，SVs也不一样。对于这些不一样的分类器，我们只能说它们"不一样"，但不能说谁更好、谁不好。

接着，我们可以推广到更加一般的多项式核函数：

General Polynomial Kernel

$$\begin{aligned} K_2(\mathbf{x}, \mathbf{x}') &= (\zeta + \gamma \mathbf{x}^T \mathbf{x}')^2 \text{ with } \gamma > 0, \zeta \geq 0 \\ K_3(\mathbf{x}, \mathbf{x}') &= (\zeta + \gamma \mathbf{x}^T \mathbf{x}')^3 \text{ with } \gamma > 0, \zeta \geq 0 \\ &\vdots \\ K_Q(\mathbf{x}, \mathbf{x}') &= (\zeta + \gamma \mathbf{x}^T \mathbf{x}')^Q \text{ with } \gamma > 0, \zeta \geq 0 \end{aligned}$$

因此，对于一个如上图最后一行所示的核函数(Q, ζ, γ)，它就对应着(隐藏着)一个Q阶的多项式转换。我们用它计算两个数据的函数值时，我们其实做了两步：①将两个数据映射到隐藏的那个transform的Q阶空间中去；②求取那个空间里两个向量的内积。

SVM+Polynomial Kernel = Polynomial SVM

最后，当$Q = 1, \zeta = 0, \gamma = 1$时，该函数退化为Linear Kernel，即线性核函数。线性核函数等价于在原空间里直接做内积，没有任何的transform。

Linear first。

3. Gaussian Kernel: 高斯核

预备知识——泰勒展开：

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!} (x - x_0)^n$$

exp(x)在x=0处的泰勒展开：

$$exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

假设$\mathbf{x} = (x)$，考虑核函数：

$$K(x, x') = exp(-(x - x')^2)$$

化简：

$$
\begin{aligned}
K(x,x') &= exp(-(x)^2)exp(-(x')^2)exp(2xx') \\
&= exp(-(x)^2)exp(-(x')^2)\left(\sum_{i=0}^{\infty}\frac{(2xx')^i}{i!}\right) \\
&= \sum_{i=0}^{\infty}\left(exp(-(x)^2)exp(-(x')^2)\sqrt{\frac{2^i}{i!}}\sqrt{\frac{2^i}{i!}}(x)^i(x')^i\right) \\
&= \Phi(x)^T\Phi(x')
\end{aligned}
$$

可见，这个核函数里包含一个**无限多维的转换**：

$$\Phi(x) = exp(-x^2)\cdot\left(1, \sqrt{\frac{2}{1!}}x, \sqrt{\frac{2^2}{2!}}x^2, \cdots\right)$$

更加一般的，高斯核函数是指：

$$K(\mathbf{x},\mathbf{x}') = exp(-\gamma||\mathbf{x}-\mathbf{x}'||^2)$$

其中，$\gamma > 0$。

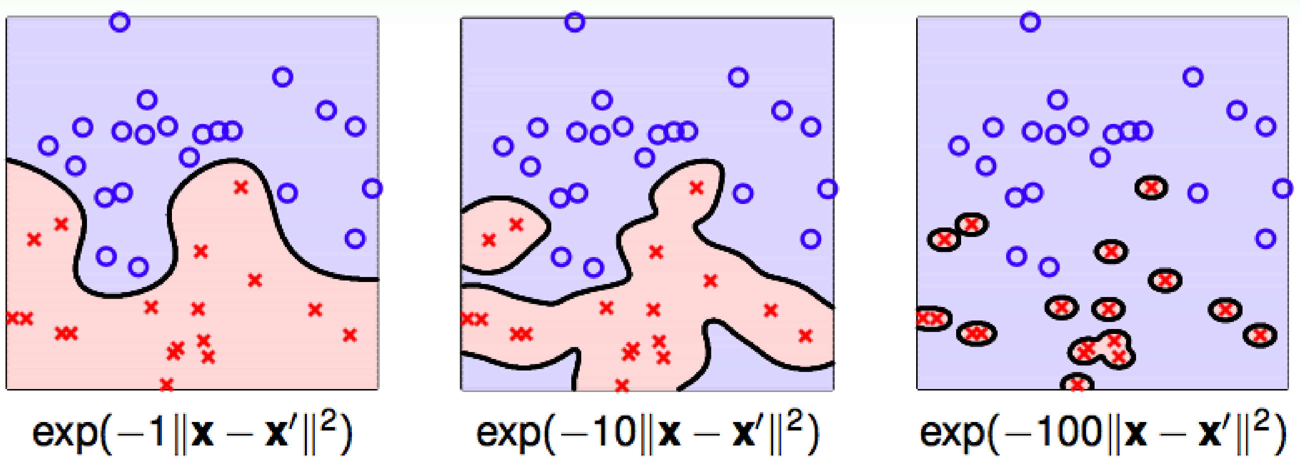
因此，Gaussian SVM的hypothesis如下：

$$
\begin{aligned}
g_{SVM}(\mathbf{x}) &= sign\left(\sum_{SV}\alpha_n y_n K(\mathbf{x}_n,\mathbf{x})+b\right) \\
&= sign\left(\sum_{SV}\alpha_n y_n exp(-\gamma||\mathbf{x}-\mathbf{x}_n||^2)+b\right)
\end{aligned}
$$

可见：

- Gaussian SVM实质是一系列高斯函数的线性组合，每一个高斯函数的中心是一个支撑向量；
- 因此又被称为Radial Basis Function (RBF) kernel。

注意，由于Gaussian十分powerful，因此可能过拟合——γ越大，越容易过拟合。



4. Comparison of Kernels: 核的比较

线性核：$K(\mathbf{x},\mathbf{x}') = \mathbf{x}^T\mathbf{x}'$

- [优]安全，不容易过拟合；
- [优]快速，有时候解primal问题反而更有效率；
- [优]解释性高，$\mathbf{w}$和SVs能提供一些信息；
- [劣]数据不一定是线性可分的。

多项式核：$K(\mathbf{x}, \mathbf{x}') = (\zeta + \gamma \mathbf{x}^T \mathbf{x}')^Q$

- [优]比线性更强一些；
- [优]当我们知道一些关于Q的信息时，很好用；
- [劣]当Q很大时会出现数值问题，如果括号内绝对值大于0，Q次方会变很大；如果括号内绝对值小于0，Q次方会变很小，接近于0；
- [劣]有三个参数要选；
- 综上，当Q较小的时候才会选择多项式核。

高斯核：$K(\mathbf{x}, \mathbf{x}') = exp(-\gamma ||\mathbf{x} - \mathbf{x}'||^2)$

- [优]最强大；
- [优]没有像多项式核那样的数值问题；
- [优]只有一个参数；
- [劣]可解释性差，因为不会算出$\mathbf{w}$；
- [劣]计算慢；
- [劣]容易过拟合。

其他核：kernel表征某种特别的相似性(similarity)，因为它是Z空间里的向量内积；然而，并不是任何相似性度量都是核函数。核函数需要满足**Mercer’s condition**——valid kernel的充分必要条件：

1. 对称，即$K(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}', \mathbf{x})$；
2. 令$k_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$，则矩阵K是半正定矩阵：

- let $k_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, the matrix K

$$
\begin{aligned}
&= \begin{bmatrix} \Phi(\mathbf{x}_1)^T\Phi(\mathbf{x}_1) & \Phi(\mathbf{x}_1)^T\Phi(\mathbf{x}_2) & \ldots & \Phi(\mathbf{x}_1)^T\Phi(\mathbf{x}_N) \\ \Phi(\mathbf{x}_2)^T\Phi(\mathbf{x}_1) & \Phi(\mathbf{x}_2)^T\Phi(\mathbf{x}_2) & \ldots & \Phi(\mathbf{x}_2)^T\Phi(\mathbf{x}_N) \\ \ldots & \ldots & \ldots & \ldots \\ \Phi(\mathbf{x}_N)^T\Phi(\mathbf{x}_1) & \Phi(\mathbf{x}_N)^T\Phi(\mathbf{x}_2) & \ldots & \Phi(\mathbf{x}_N)^T\Phi(\mathbf{x}_N) \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{z}_1 & \mathbf{z}_2 & \ldots & \mathbf{z}_N \end{bmatrix}^T \begin{bmatrix} \mathbf{z}_1 & \mathbf{z}_2 & \ldots & \mathbf{z}_N \end{bmatrix} \\
&= \mathrm{Z}\mathrm{Z}^T \text{ must always be positive semi-definite}
\end{aligned}
$$

最后：定义一个核函数虽然possible，但是hard。

5. Summary

- 核技巧的实质是将"转换"与"内积"合并为一步，避免了在高维Z空间中的内积运算；因此，一个核函数的背后"隐藏"着一个特征转换；通过核技巧，我们真正摆脱了SVM计算中对于$\tilde{d}$的依赖；
- 多项式核有三个参数，$\gamma > 0, \zeta \geq 0$，Q为阶数；当Q较小时选择多项式核效果较好；
- 高斯核(Gaussian kernel / RBF kernel)推导的核心是泰勒展开，其背后隐藏着一个无限多维的转换，因此十分强大；有一个参数$\gamma > 0$，越大，则高斯函数越尖，越易过拟合；
- 不是任何函数都可以成为核函数，成为合法核函数的充要条件是Mercer's condition。