机器学习与深度学习面试系列七（SVM）

什么是函数间隔?

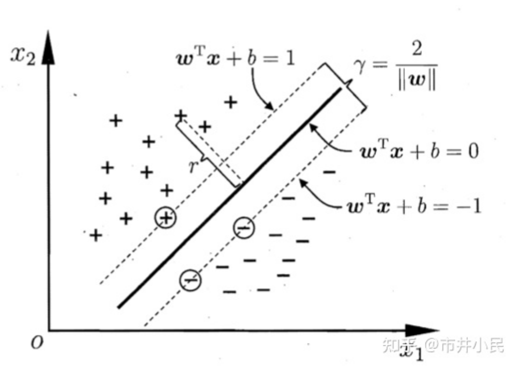
由空间上的平面公式确定超平面 $w^T x + b = 0$ ，且 $|w^T x + b|$ 表示点 x 到平面上的距离。正类负例位于分割平面两侧，因此 $y(w^T x + b)$ 可同时表示分类正确性以及距离确信度。这也就是函数间隔，其被定义为训练集中所有点到超平面距离的最小值。

什么是几何间隔?

由于成比例地缩放 w 和 b 会使得 $|w^T x + b|$ 跟着成比例缩放，因此，需要对法向量 w 加上约束，使得间隔是确定的，也就是函数间隔整体除以 ||w||，也就得到了几何间隔 $\frac{|w^T x + b|}{||w||}$ ，正类负例位于分割平面两侧，因此 $y(\frac{w^T x + b}{||w||})$ 可同时表示分类正确性以及距离确信度。

什么是支持向量?

我们期望找到一个超平面 $w^T x + b = 0$ ，使得 $y_i(\frac{w^T x_i + b}{||w||}) \geq \Delta$ 对所有的 i 都成立。两边同时除以 Δ ，得到 $y_i(\frac{w^T x_i + b}{\Delta||w||}) \geq 1$。由于 w 和 b 等比例放缩对于超平面的定义是等价的，所以我们取 $\hat{w} = \frac{w}{\Delta||w||}$ ， $\hat{b} = \frac{b}{\Delta||w||}$ ，得到 $y_i(\hat{w}^T x_i + \hat{b}) \geq 1$ 。



支持向量与间隔

距离超平面最近的这几个训练样本点使上式的等号成立，即 $y_i(\hat{w}^T x_i + \hat{b}) = 1$（ $y_i(\frac{w^T x_i + b}{||w||}) = \Delta$ ）， 它们被称为"支持向量" (support vector)。支持向量到超平面的距离为 Δ ，这就是间隔。

SVM的优化目标是什么（硬间隔）？

SVM的目标有两个：第一个是**使间隔最大化**，第二个是**使样本正确分类。**

由于 $||\hat{w}|| = ||\frac{w}{\Delta||w||}|| = \frac{1}{\Delta}$，所以要最大化间隔，就是最小化 $||\hat{w}||$ ，为了方便求解，我们可以等价的最小化 $\frac{1}{2}||\hat{w}||^2$ 。

使样本正确分类只需要满足对于所有的 i 都有 $y_i(\hat{w}^T x_i + \hat{b}) \geq 1$ 。 $\hat{w}$ 只是一个标记，下面我们不再区分 $\hat{w}$ 和 w 。

综上：

$$\min_{w,b} \frac{1}{2}||w||^2$$
$$s.t. y_i(w^T x_i + b) \geq 1 \qquad \forall i$$

SVM 为什么采用间隔最大化（与感知机的区别）？

当训练数据线性可分时，存在无穷个分离超平面可以将两类数据正确分开。感知机利用误分类最小策略，求得分离超平面，不过此时的解有无穷多个。线性可分支持向量机利用间隔最大化求得最优分离超平面，这时，解是唯一的。另一方面，此时的分隔超平面所产生的分类结果是最鲁棒的，对未知实例的泛化能力最强。

SVM的优化目标如何求解（手推SVM）？

从上面的公式看出，这是一个有约束条件的最优化问题，用拉格朗日乘子法来解决。

$$\min_{w,b} \max_{\alpha} L(w,b,\alpha) = \min_{w,b} \max_{\alpha} \frac{1}{2}||w||^2 + \sum_{i=1}^{m} \alpha_i (1 - y_i(w^T x_i + b))$$
$$s.t. \alpha_i \geq 0 \quad \forall i$$

这是一个凸优化问题，并且满足Slater定理，所以其对偶问题的解和原问题的解相同。对偶问题写作：

$$\max_{\alpha} \min_{w,b} L(w,b,\alpha) = \max_{\alpha} \min_{w,b} \frac{1}{2}||w||^2 + \sum_{i=1}^{m} \alpha_i (1 - y_i(w^T x_i + b))$$
$$s.t. \alpha_i \geq 0 \quad \forall i$$

其实就是交换了一下max和min的位置。

先求 $\min\limits_{w,b}$ ，分别对 w 和 b 求偏导，并令其等于0，得：

$$w^* = \sum_{i=1}^{m} \alpha_i y_i x_i$$

$$0 = \sum_{i=1}^{m} \alpha_i y_i$$

将其带回对偶问题的式子，得：

$$\max_{\alpha} L(w, b, \alpha) = \max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$s.t. \sum_{i=1}^{m} \alpha_i y_i = 0 \quad (\alpha_i \geq 0 \quad \forall i)$$

此时需要求解 α^*（满足上面优化目标的最优 α），利用SMO（序列最小优化）算法。SMO 的基本思路是先固定 α_i 之外的所有参数，然后求 α_i 上的极值。但由于需要满足 $\sum\limits_{i=1}^{m} \alpha_i y_i = 0$ ，所以我们每次选择两个变量 α_i 和 α_j ，负责单改变一个 α_i 不能保证这个约束条件随时成立。固定其他的m-2个 α 参数不变，每次只调整两个 α ，使得目标函数值更大。SMO算法之所以高效，是因为仅优化两个参数的过程实际上仅有一个约束条件，其中一个可由另一个表示，这样的二次规划问题具有闭式解。SMO算法具体可以参考李航《统计机器学习》7.4节。

当求出 α^* 后，我们可以得到 $w^* = \sum\limits_{i=1}^{m} \alpha_i^* y_i x_i$ ，由KKT条件可知，
$\alpha_i^*(1 - y_i({w^*}^T x_i + b^*)) = 0$ 恒成立，而 $\alpha_i^* \geq 0$ ，故当 $\alpha_i^* > 0$ 时，
$1 - y_i({w^*}^T x_i + b^*) = 0$ ，即此时样本点是支持向量。任意选择一个支持向量，求得
$b^* = \frac{1}{y_i} - {w^*}^T x$ ，由于 $y_i \in \{-1, +1\}$ ，所以 $y_i^2 = 1$ 即 $\frac{1}{y_i} = y_i$ ，带入 w^* ，得：
$b^* = y_i - \sum\limits_{i=1}^{m} \alpha_i^* y_i (x_i)^T x$ 。 w^* 和 b^* 都只依赖 $\alpha_i^* > 0$ 的样本，也就是支持向量。

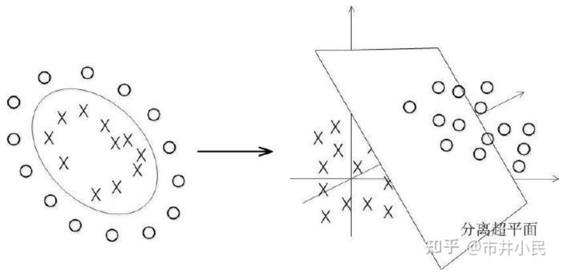
综上，超平面方程为： $\sum\limits_{i=1}^{m} \alpha_i^* y_i (x_i)^T x + b^* = 0$ ，分类决策函数可以写作：

$f(x) = sign(\sum\limits_{i=1}^{m} \alpha_i^* y_i (x_i)^T x + b^*)$ 。

核方法是什么？

事实上，大部分时候数据并不是线性可分的，这个时候满足这样条件的超平面就根本不存在。于是在线性不可分的情况下，我们通过一个映射函数 $\phi(x)$ 将低维数据映射到高维空间，而在高维空间中，数据线性可分的概率大大增加。当维度高到一定程度，一定线性可分。

例如我们有一个二维的数据 $v_1 = (x_1, y_1)$ ，有一个映射函数 $\phi(v) = \phi((x, y)) = (x^2, \sqrt{2}xy, y^2)$ 可以将二维空间数据映射到三维空间上。这只是一个很简单的例子，映射的维度越高，越容易线性可分。



引入映射函数后，上面对SVM的求解结果可以将所有的 x_i 替换为 $\phi(x_i)$ 。

$$b^* = y_i - \sum_{i=1}^{m} \alpha_i^* y_i \phi(x_i)^T \phi(x)$$

$$f(x) = sign(\sum_{i=1}^{m} \alpha_i^* y_i \phi(x_i)^T \phi(x) + b^*)$$

上面我们举的例子只是将二维数据映射到三维空间上，可以想象如果我们要做很高维度，甚至是无穷维的映射，那么 $\phi(x_i)^T \phi(x_j)$ 的计算是非常耗时的。于是我们引入一个核函数 $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ ，通过直接计算这个核函数来得到结果，而无需真的去计算这个耗时的高维映射的过程。仍然以上面这个简单的映射函数为例，我们可以定义一个核函数 $K(v_1, v_2) = (v_1^T v_2)^2$ ，通过计算这个核函数，来避免计算映射函数，这两者的结果是相等的。

$$\begin{aligned}
\phi(v_1)^T \phi(v_2) &= (x_1^2, \sqrt{2}x_1y_1, y_1^2)^T (x_2^2, \sqrt{2}x_2y_2, y_2^2) \\
&= x_1^2 x_2^2 + 2x_1x_2y_1y_2 + y_1^2 y_2^2 \\
&= (x_1x_2 + y_1y_2)^2 \\
&= (v_1^T v_2)^2 \\
&= K(v_1, v_2)
\end{aligned}$$

利用核函数，我们可以将SVM的求解结果写为：

$$b^* = y_i - \sum_{i=1}^{m} \alpha_i^* y_i K(x_i, x)$$

$$f(x) = sign(\sum_{i=1}^{m} \alpha_i^* y_i K(x_i, x) + b^*)$$

这里注意一点的是，在上述我们叙述的过程中，先定义了映射函数，再去找与其等价的核函数，是为了便于我们理解核函数的作用。实际使用时，我们是直接选择核函数，其映射函数到底是什么，其实我们并不关心。但是从数学上来说，对于任何一个核函数，都可以找到其对应的映射函数。

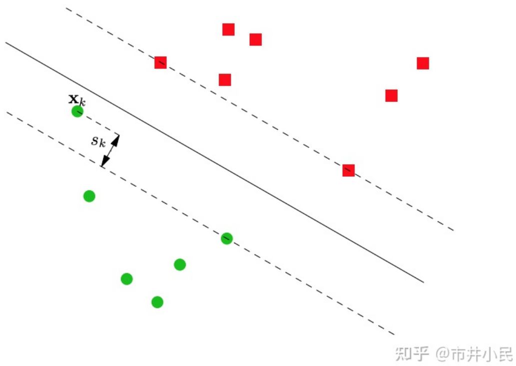
常用的核函数有哪些？有什么优缺点？分别适用于什么场景？

名称	形式	优点	缺点
线性核	$\boldsymbol{x}_i^\top \boldsymbol{x}_j$	有高效实现，不易过拟合	无法解决非线性可分问题
多项式核	$(\beta \boldsymbol{x}_i^\top \boldsymbol{x}_j + \theta)^n$	比线性核更一般，n 直接描述了被映射空间的复杂度	参数多，当 n 很大时会导致计算不稳定
RBF 核	$\exp\left(-\frac{\lVert \boldsymbol{x}_i - \boldsymbol{x}_j \rVert^2}{2\sigma^2}\right)$	只有一个参数，没有计算不稳定问题	计算慢，过拟合风险大

- 当特征维数 d 超过样本数 m 时 (文本分类问题通常是这种情况), 使用线性核;
- 当特征维数 d 比较小. 样本数 m 中等时, 使用RBF核;
- 当特征维数 d 比较小. 样本数 m 特别大时, 支持向量机性能通常不如深度神经网络

什么是软间隔?

完全线性可分是非常奢侈的，即使恰好找到了某个核函数使训练集在特征空间中线性可分，也很难断定这个貌似线性可分的结果不是由于过拟合所造成的。所以当数据集中存在一些离群点时，通过给函数间隔加上一个松弛变量，使得函数间隔有一定的弹性来包容异常点，从而增强模型的泛化能力。



软间隔松弛变量示意图

我们可以为硬间隔的SVM优化目标式一个松弛变量 s_k ，则：

$$
\begin{aligned}
&\min_{w,b} \frac{1}{2}||w||^2 + C\sum_{i=1}^{m} s_i \\
&s.t.\, y_i(w^T x_i + b) \geq 1 - s_i \qquad \forall i \\
&s.t.\, s_i \geq 0 \qquad \forall i
\end{aligned}
$$

C >0 为惩罚参数，C越大对误分类的惩罚越大，反之，越小。换言之，希望 $\frac{1}{2}||w||_2^2$ 尽量的小，误分点数量尽可能的小。通常C的取值范围是 2^{-5} 到 2^{15} ，可以通过网格搜索法寻找最佳的C。

对软间隔的优化目标使用拉格朗日乘子法，得到其对偶问题：

$$\max_{\alpha,\beta}\min_{w,b,s} L(w,b,\alpha,\beta) = \max_{\alpha,\beta}\min_{w,b,s} \frac{1}{2}||w||^2 + C\sum_{i=1}^{m} s_i + \sum_{i=1}^{m} \alpha_i(1 - s_i - y_i(w^T x_i + b)) + \sum_{i=1}^{m} \beta_i s_i$$
$$s.t. \alpha_i \geq 0 \quad \forall i$$
$$s.t. \beta_i \geq 0 \quad \forall i$$

先求 $\min\limits_{w,b,s}$ ，分别对 w 、b、 s 求偏导，并令其等于0，得：

$$w^* = \sum_{i=1}^{m} \alpha_i y_i x_i$$

$$0 = \sum_{i=1}^{m} \alpha_i y_i$$

$$0 = C - \alpha_i - \beta_i$$

由于 $\beta_i \geq 0$ ，所以 $0 \leq \alpha_i \leq C$

将其代回对偶问题目标公式，得：

$$\max_{\alpha} L(w,b,\alpha) = \max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j$$
$$s.t. \sum_{i=1}^{m} \alpha_i y_i = 0 \quad (\alpha_i \geq 0 \quad \forall i)$$
$$s.t. 0 \leq \alpha_i \leq C \quad \forall i$$

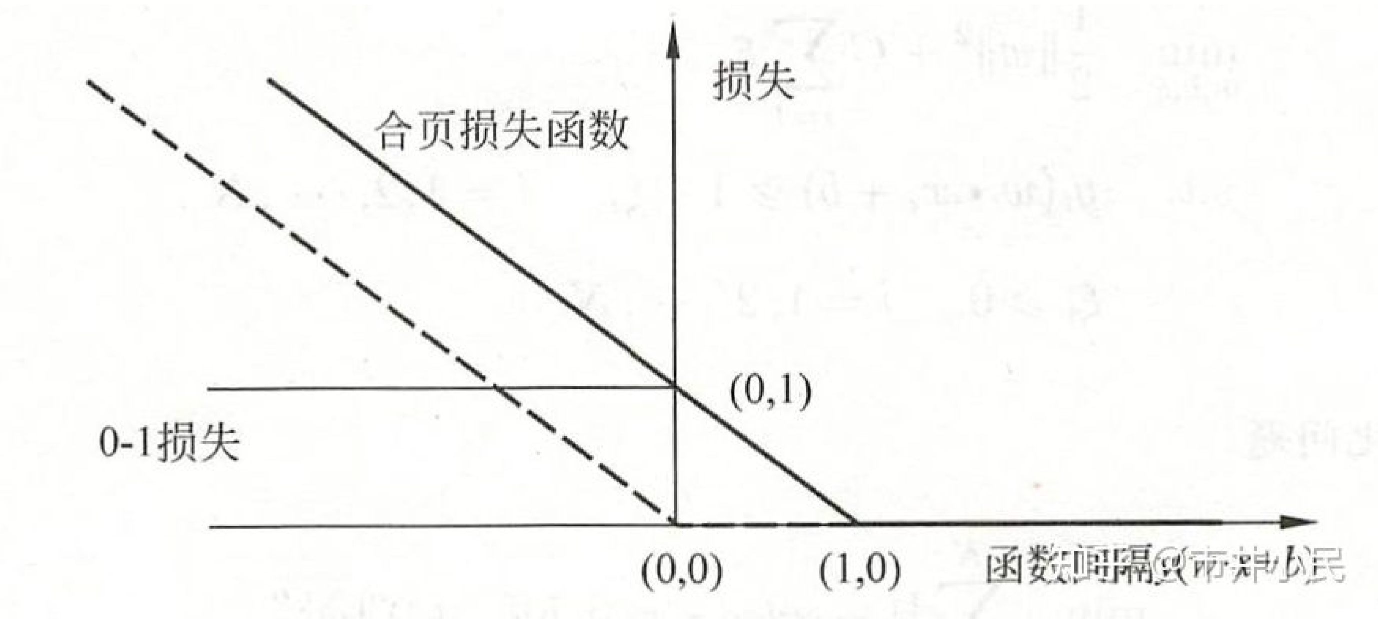
这就是软间隔最大化时的线性可分SVM的优化目标形式，和硬间隔最大化的线性可分SVM相比，我们仅仅是多了一个约束条件 $0 \leq \alpha_i \leq C$ 。我们依然可以通过SMO算法来求 α 向量就可以求出 w^* 和 b^* 了。

SVM的另一种解释?

以上我们都是从凸优化的角度推导SVM，实际上SVM还有另外一种解释，就是按照传统机器学习方法，设置一个损失函数，然后迭代的优化该损失函数求得最小值。损失函数为：

$$J=\sum_{i=1}^{m}\max\left(0,1-y_i\left(\boldsymbol{w}^{\top}\boldsymbol{x}_i+b\right)\right)+\frac{\lambda}{2}\|\boldsymbol{w}\|^2$$

它可以看作是合页损失函数加上正则化项。



合页损失函数图像如图所示，横轴是函数间隔y(w·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（也有翻译为铰链损失函数）。图中还画出了0-1损失函数，可以认为它是一个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 function）。

图中虚线显示的是感知机的损失函数[−yi(w·xi+b)]+。这时当样本点(xi,yi)被正确分类时，损失是0，否则损失是−yi(w·xi+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。

令 $\max\left(0,1-y_i\left(\boldsymbol{w}^{\top}\boldsymbol{x}_i+b\right)\right)=s_i$ ，则： $J=\sum_{i=1}^{m}s_i+\frac{\lambda}{2}\|\boldsymbol{w}\|^2$ ，取 $\lambda=\frac{1}{2C}$ ，则 $J=\frac{1}{C}(\frac{1}{2}||w||^2+C\sum_{i=1}^{m}s_i)$ ，与软间隔优化形式完全一致。

为什么SVM对缺失数据敏感?

这里说的缺失数据是指缺失某些特征数据，向量数据不完整。SVM 没有处理缺失值的策略。而 SVM 希望样本在特征空间中线性可分，所以特征空间的好坏对SVM的性能很重要。缺失特征数据将影响训练结果的好坏。

SVM的优缺点?

优点:

1. 由于SVM是一个凸优化问题，所以求得的解一定是全局最优而不是局部最优。
2. 不仅适用于线性线性问题还适用于非线性问题(用核技巧)。
3. 拥有高维样本空间的数据也能用SVM，这是因为数据集的复杂度只取决于支持向量而不是数据集的维度，这在某种意义上避免了“维数灾难”。
4. 理论基础比较完善(例如神经网络就更像一个黑盒子)。

缺点:

1. 二次规划问题求解将涉及m阶矩阵的计算(m为样本的个数), 因此SVM不适用于超大数据集。(SMO算法可以缓解这个问题)
2. 只适用于二分类问题。(SVM的推广SVR也适用于回归问题；可以通过多个SVM的组合来解决多分类问题)

LR和SVM的联系与区别

相同点

1、LR和SVM都可以处理分类问题，且一般都用于处理线性二分类问题（在改进的情况下可以处理多分类问题）
2、两个方法都可以增加不同的正则化项，如l1、l2等等。所以在很多实验中，两种算法的结果是很接近的。
3、LR和SVM都可以用来做非线性分类，只要加核函数就好。
4、LR和SVM都是线性模型，当然这里我们不要考虑核函数
5、都属于判别模型

不同点

1、LR是参数模型（假设全局数据服从伯努利分布），SVM是非参数模型（没有对全局数据假设任何分布）。
2、从目标函数来看，区别在于逻辑回归采用的是logistical loss，SVM采用的是hinge loss，这两个损失函数的目的都是增加对分类影响较大的数据点的权重，减少与分类关系较小的数据点的权重。
3、逻辑回归相对来说模型更简单，好理解，特别是大规模线性分类时比较方便。而SVM的理解和优化相对来说复杂一些，SVM转化为对偶问题后,分类只需要计算与少数几个支持向量的距离,这个在进行复杂核函数计算时优势很明显,能够大大简化模型和计算。
4、SVM不直接依赖数据分布，而LR则依赖，因为SVM只与支持向量那几个点有关系，而LR和所有点都有关系。

5、SVM依赖penalty系数（软间隔的C），实验中需要做CV（交叉验证参数）
6、SVM本身是结构风险最小化模型，而LR是经验风险最小化模型

LR和SVM选择

1. 如果Feature的数量很大，跟样本数量差不多，这时候选用LR或者是Linear Kernel的SVM
2. 如果Feature的数量比较小，样本数量一般，不算大也不算小，选用SVM+Gaussian Kernel
3. 如果Feature的数量比较小，而样本数量很多，需要手工添加一些feature变成第一种情况