支持向量机的前世今生

> $XJTU$计算机85张子诚

1.前言

(1)参考书目

- 《机器学习》(周志华,清华大学出版社)
- 《统计学习方法》(李航)
- Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines
- 机器学习技法mooc（台大林轩田）
- Deep Learning using Linear Support Vector Machines

  本文主要参考了以上资料，同时加入了自己学习过程中的一些理解，本文的逻辑组织与周志华教授的《机器学习》较为相近，由于是初学者，所以对很多地方的见解并没有那么深刻，表述也可能不严谨，还请多多指教

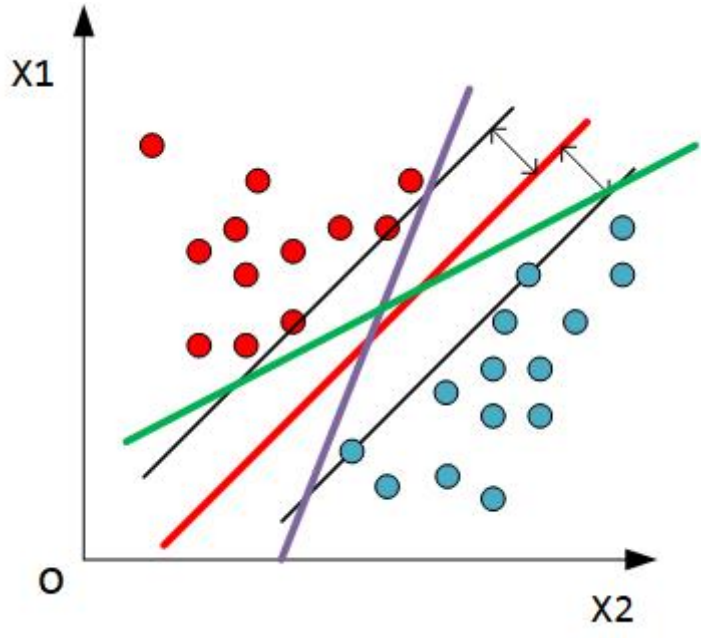
2.引入

(1)问题的引入

给定一个线性可分的训练样本集(training set)$\mathbf{D}=\{(\mathrm{x_i},\mathrm{y_i})\}_{\mathrm{i=1...N}},\mathrm{y_i}\in\{-1,1\}$,我们想对这个样本进行分类，一个很自然的想法是，我们能不能基于训练样本集$\mathbf{D}$在样本空间中寻找一个分割的超平面(hyperplane):$w^Tx+b=0$,将不同类别的样本分开,即当$y=-1$时,有$w^Tx+b<0$,当$y=1$时,有$w^Tx+b>0$。

(2)优化的目标

如下图所示:



绿色，紫色，红色的超平面似乎都可以将我们的训练样本分开，但是我们要考虑到在机器学习的过程中，相对于训练集上模型表现的有多优秀，我们更希望我们的模型能够在验证集(validation set)上有更好的表现，即所谓的泛化能力较好。直觉上，我们会觉得红色的超平面更好，因为它对噪声和扰动具有更好的稳定性。换言之，**“红色”的超平面所产生的分类结果是最鲁棒的，对未见样本的泛化能力是最强的。**

那么现在我们的目标就是寻找一条像红色的这样的泛化能力最强的超平面，那么如何来描述红色的这个超平面的特性呢？接下来，我们将一步一步的介绍。

3.最大间隔

为了描述之前我们所提到的红色的超平面的特性，我们可以直观的这样来考虑，我们希望我们的样本点到该超平面的距离越远越好，即我们希望**最大化距离超平面最近点的距离。**那么写成形式化的定义就是:

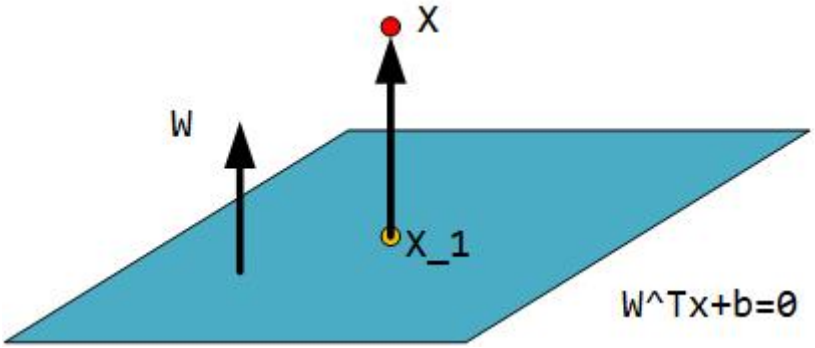
$$\max_{w,b}\min_{x}\{dist(x_i,b,w)_{i=1...N}\}$$
$$s.t.\quad y_i(w^Tx_i+b)\geq 0$$

- $dist(x_i,b,w)$ 表示第i个点到超平面:$w^Tx+b=0$的距离
- 约束条件表明训练集上的两类样本($y=1$和$y=-1$)分别位于超平面所形成的两个半空间(halfspace)中

但是，这个优化问题的定义还是模糊的，不清晰的，接下来我们就来转化该优化问题。

(1)几何间隔

在上面的优化问题中，我们定义了$dist(x_i,b,w)$函数，接下来我们将显示化该函数。我们来考虑推导空间中点到超平面的距离公式（结合下图）：



设平面外一点 x,x_1为x在超平面$w^Tx+b=0$ 的投影点，设x到$w^Tx+b=0$ 的距离为d

$\because w$ 为平面的法向量

$$x-x_1=d\frac{w}{||w||}$$
$$w^T(x-x_1)=d\frac{w^Tw}{||w||}$$
$$w^Tx_1+b=0$$

$\therefore d=\frac{w^Tx+b}{||w||}$

考虑到d的非负性，有$dist(x,w,b)=\frac{|w^Tx+b|}{||w||}$

特别的对于此问题因为$y_i\in\{+1,-1\}$,我们有$y_i(w^Tx_i+b)=|w^Tx_i+b_i|$

$$\max_{w,b}\min_{x}\frac{y_i(w^Tx_i+b)}{||w||}(i=1...N)$$
$$s.t.\quad y_i(w^Tx_i+b)\geq 0$$

> 注：此处的$\frac{y_i(w^Tx_i+b)}{||w||}$ 在很多地方定义为几何间隔

(2) 函数间隔

我们进一步的来对我们的优化问题进行转化，注意到一个有趣的事实：**对于任何超平面同时对**w,b**放大或缩小相同的倍数，并不会改变超平面,更不会改变几何间隔**，即

$$w^Tx+b=0\Longleftrightarrow\alpha w^Tx+\alpha b=0\quad(\alpha\in\mathbb{R},\alpha\neq 0)$$

我们再来看我们的优化问题，将其稍做等价变形

$$\max_{w,b}\quad\frac{1}{||w||}\min_{x}y_i(w^Tx_i+b)(i=1...N)$$
$$s.t.\quad y_i(w^Tx_i+b)\geq 0$$

我们来观察这一部分

$$\min_{x}y_i(w^Tx_i+b)$$

> $y_i(w^Tx_i+b)$ 就是很多地方定义的函数间隔

我们对w,b进行放缩，并不影响最小函数间隔的最优解x^*的获得。

这样我们先固定w,b,上式将会有一个最优值p^*，最优解x^*,这时我们不妨对w,b放缩使最优值$p^*=1$,最优解仍然为x^*。

所以我们令$\min_x y_i(w^Tx_i+b)=1$得到的优化问题：

$$\begin{aligned}\max_{w,b}\quad & \frac{1}{||w||}\\ s.t.\quad & y_i(w^Tx_i+b)\geq 0\\ & \min_x y_i(w^Tx_i+b)=1\end{aligned}$$

进一步的,等价于

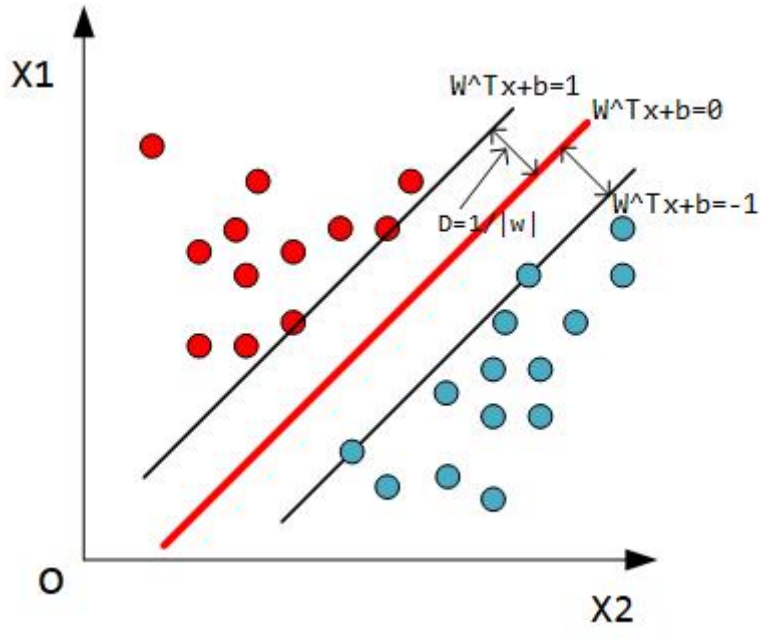
$$\begin{aligned}\max_{w,b}\quad & \frac{1}{||w||}\\ s.t.\quad & y_i(w^Tx_i+b)\geq 1\end{aligned}$$

> 注意：此处放宽约束$y_i(w^Tx_i+b)\geq 1$并不会改变最优解的取值，本质在于目标函数对于$||w||$是单减的，“最优解”取不到约束的下确界，我们总可以缩小该解，使其达到下确界，从而获得更大的目标函数。故最优解一定可以使约束取等

将其转化为一个凸优化问题(本质上是一个凸二次规划)

$$\begin{aligned}\max_{w,b}\quad & \frac{1}{2}w^Tw\\ s.t.\quad & y_i(w^Tx_i+b)\geq 1\quad i=1,\dots,N\end{aligned}$$

下图是一个直观的几何解释



4.对偶问题

上述的最后化简的优化形式，已经可以用一些解凸优化的包进行计算求解了，我们知道对于凸优化问题我们往往可以从原问题的对偶问题获得更简洁优雅的优化形式。

(1)对偶问题

原问题

$$\begin{aligned}\max_{w,b} \quad & \frac{1}{2}w^T w \\ s.t. \quad & y_i(w^T x_i + b) \geq 1 \quad i = 1, \dots, N\end{aligned}$$

Lagrange 函数：

$$L(b, w, \alpha) = \frac{1}{2}w^T w + \sum_{i=1}^{N} \alpha_i (1 - y_i(w^T x_i + b))$$

Lagrange dual 函数

$$\begin{aligned}\frac{\partial L}{\partial b} &= -\sum_{i=1}^{N} \alpha_i y_i = 0 \\ \frac{\partial L}{\partial w} &= w - \sum_{i=1}^{N} \alpha_i y_i x_i = 0 \\ g(\alpha) &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{i=1}^{N} \alpha_i \alpha_j x_i^T x_j\end{aligned}$$

对偶问题

$$\begin{aligned}\max_{\alpha} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{i=1}^{N} \alpha_i \alpha_j x_i^T x_j \\ s.t. \quad & \sum_{i=1}^{N} \alpha_i y_i = 0, \\ & \alpha_i \geq 0, \quad i = 1, \dots, N\end{aligned}$$

KKT 条件

$$\begin{aligned}& y_i(w^T x_i + b) \geq 1 \\ & \alpha_i \geq 0 \\ & w = \sum_{i=1}^{N} \alpha_i y_i x_i \\ & \sum_{i=1}^{N} y_i \alpha_i = 0 \\ & \alpha_i (1 - y_i(w^T x_i + b)) = 0\end{aligned}$$

假设函数：

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i x_i^T x + b$$

(2)支持向量

由KKT条件可以知道，对于任意的训练样本$(x_i, y_i)_{i=1..N}$，**总有**$\alpha_i = 0$**或**$y_i(w^T x_i + b) = 1$

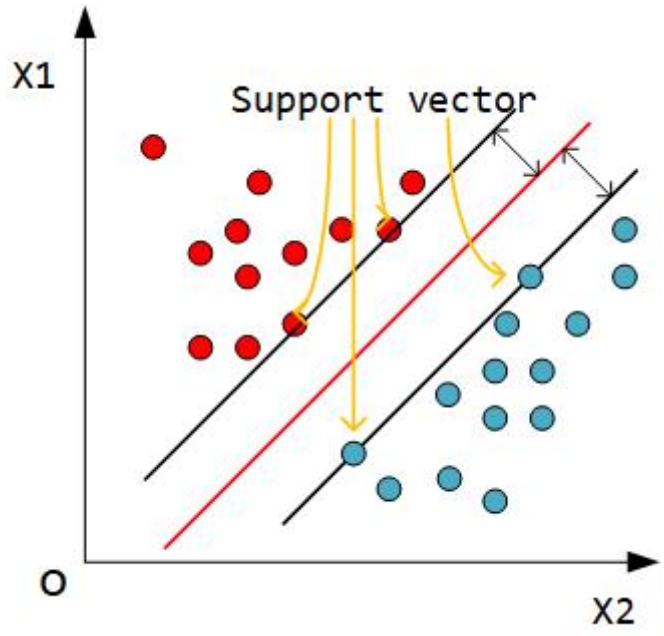
- 若$\alpha_i = 0$ 时不会对假设函数$f(x)$ 产生影响
- 若$\alpha_i > 0$ 则$y_i(w^T x_i + b) = 1$,(x_i, y_i) 位于最大间隔边界上，定义为**支持向量**

我们发现最终的参数w, b仅仅与**支持向量有关**

- $w^* = \sum_{i=1}^{N} \alpha_i y_i x_i$
- $b^* = 1/y_i - w^T x_i$ 如果$\alpha_i > 0$

> 注：实际中b^*的计算采取的是一种更加鲁棒的做法：
>
> 使用所有支持向量求解的平均值

下图显示了**支持向量**



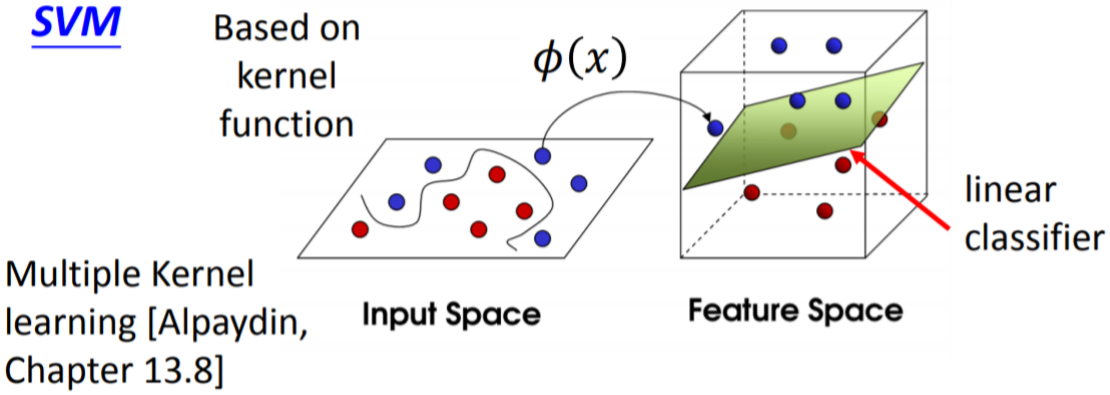
5.核函数

(1) 意义

我们首先来介绍一下我们为什么要引入核技巧(kernel trick),出于两点理由

- 我们上述的推导是在样本线性可分的情况下探讨的，但实际中我们的样本可能并不是线性可分的，最简单的比如：二维平面内的异或问题就是线性不可分的。这时候我们常常需要进行特征转换(feature transform)，我们需要在更加高维度的空间中去表征我们的特征，**核函数**可以做到这一点，并且它能使我们在**保持优化问题凸性的前提下学习非线性函数模型**
- 核函数的计算复杂度远远小于先构造$\phi(x)$将x映射到高维空间，再在内积的计算复杂度

我们先用一个图获得一些直观的感受



> 注:此图来自于台大李宏毅老师的ppt(http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2016/Lecture/SVM%20(v5).pdf)

(2) 特征映射

首先我们来看如何将我们的模型由线性的模型变为非线性的模型，假设我们的特征$\mathbf{x}$所处的是一个低维的空间，我们用$\phi(\mathbf{x})$ 来表示将$\mathbf{x}$ 映射后的处于高维空间中的特征，这样我们的模型变为：

例如,假设$x \in \mathbb{R}$我们的映射为:

$$\phi(x) = \begin{bmatrix} x \\ x^2 \\ x^3 \end{bmatrix} \in \mathbb{R}^3$$

- 假设函数

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

- 原问题

$$\begin{aligned} \max_{w,b} \quad & \frac{1}{2} w^T w \\ s.t. \quad & y_i(w^T \phi(x_i) + b) \geq 1 \quad i = 1, \ldots, N \end{aligned}$$

- 对偶问题

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i=1}^{N} \alpha_i \alpha_j \phi(x_i)^T \phi(x_j) \\ s.t. \quad & \sum_{i=1}^{N} \alpha_i y_i = 0, \\ & \alpha_i \geq 0, \quad i = 1, \ldots, N \end{aligned}$$

一个很自然的想法，我们在进行预测的时候**先将x映射为$\phi(\mathbf{x})$,再进行$\phi(\mathbf{x})^{\mathbf{T}} \phi(\mathbf{x})$内积运算**，但是这有一个问题$\phi(\mathbf{x})$ 所处的空间维度很高，有时候还可能是无穷维，这样我们在计算$\phi(\mathbf{x})^{\mathbf{T}} \phi(\mathbf{x})$时会付出很高昂的计算代价。

这时我们想能不能设想这样一个函数,它可以将**变换+内积这两步运算一次性计算出来**，从而减小计算复杂度。

(3)核函数

把我们上面的想法转化为形式化的定义就是寻找一个函数：

$$\mathbf{k}(\mathbf{x_i}, \mathbf{x_j}) = \langle \phi(\mathbf{x_i}), \phi(\mathbf{x_j}) \rangle = \phi(\mathbf{x_i})^{\mathbf{T}} \phi(\mathbf{x_j})$$

我们定义该函数为**核函数**

下面我们以一种映射为例，来说明核函数如何来降低我们计算的复杂度

例如我们设计下面的这种映射:假设$x \in \mathbb{R}^3$

$$\phi(x) = \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_1 x_3 \\ x_2 x_1 \\ x_2 x_2 \\ x_2 x_3 \\ x_3 x_1 \\ x_3 x_2 \\ x_3 x_3 \end{bmatrix} \in \mathbb{R}^3$$

下面考虑计算$\phi(x)^T \phi(x^{'})$

$$\begin{aligned}\phi(x)^T\phi(x^{'}) &= \sum_{i=1}^{3}\sum_{j=1}^{3}x_ix_jx_i^{'}x_j^{'} \\ &= \sum_{i=1}^{3}x_ix_i^{'}\sum_{j=1}^{3}x_i^{'}x_j^{'} \\ &= \left(x^Tx^{'}\right)^2\end{aligned}$$

令$\mathbf{k(x,x^{'})=\left(x^Tx^{'}\right)^2}$ 我们发现

- 先映射再内积的计算复杂度为$O(d^2)$
- 用核函数的计算复杂度为$O(d)$

> 注:d 为x的维度

所以我们发现**核函数**确实可以大大降低我们的计算复杂度

(4)常见的核函数

1.多项式核

$$\mathbf{k(x_i,x_j)=\left(x_i^Tx_j\right)^d}$$

> 注:d 表示多项式的次数

具体的说我们举例中所提到的核函数就是一个二次的多项式核，他所对应的映射的形式就如举例中的$\phi(x)$ 所示的形式

2.高斯核(RBF核)

$$\mathbf{k(x_i,x_j)}=\exp(-\frac{\mathbf{||x_i-x_j||^2}}{\mathbf{2\sigma^2}})$$

高斯核所对应的映射的特征空间是**无限维空间**(极大的增强了模型的复杂度)

(5)核矩阵

在实际中我们往往不知道合适的$\phi(\cdot)$形式是什么样子的，那么我们能不能越过寻找$\phi(\cdot)$的过程，直接构造核函数$\mathbf{k}(\cdot,\cdot)$呢?同时我们要问：究竟我们构造的函数是不是核函数呢?

下面的定理就给我们解释了什么样的函数可以称之为核函数

定理：考虑任意一个有限集合$\{x_1,\cdots,x_m\},x_i\in\mathbb{R}^n$(不一定为训练集)，定义一个核矩阵$\mathbf{K}\in\mathbb{R}^{\mathbf{m\times m}}$，其中$\mathbf{K_{ij}}=k(x_i,x_j)$,

k 是一个核函数，当且仅当$\mathbf{K}$ 是**对称半正定**的。

我们给出必要性的证明:

对于$\forall z\in\mathbb{R}^n$

$$
\begin{aligned}
z^T K z &= \sum_i \sum_j z_i K_{ij} z_j \\
&= \sum_i \sum_j z_i \phi(x_i)^T \phi(x_j) z_j \\
&= \sum_i \sum_j z_i (\sum_k \phi_k(x_i)\phi_k(x_j)) z_j \\
&= \sum_k \sum_i \sum_j z_i \phi_k(x_i)\phi_k(x_j) z_j \\
&= \sum_k (\sum_i z_i \phi_k(x_i))^2 \\
&\geq 0
\end{aligned}
$$

又因为 $K_{ij} = K_{ji} = \phi(x_i)^T \phi(x_j)$

所以$\mathbf{K}$为对称半正定的。

这个定理就给了我们设计核函数的一个准则

(6) 核函数+svm的形式

有了核函数之后如何把核函数运用到我们的模型之中？其实比较简单，只需要把模型中出现**内积**的地方换为核函数就行了。

例如:对偶问题

$$
\begin{aligned}
\max_{\alpha} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i=1}^{N} \alpha_i \alpha_j k(x_i, x_j) \\
s.t. \quad & \sum_{i=1}^{N} \alpha_i y_i = 0, \\
& \alpha_i \geq 0, \quad i = 1, \ldots, N
\end{aligned}
$$

假设函数：

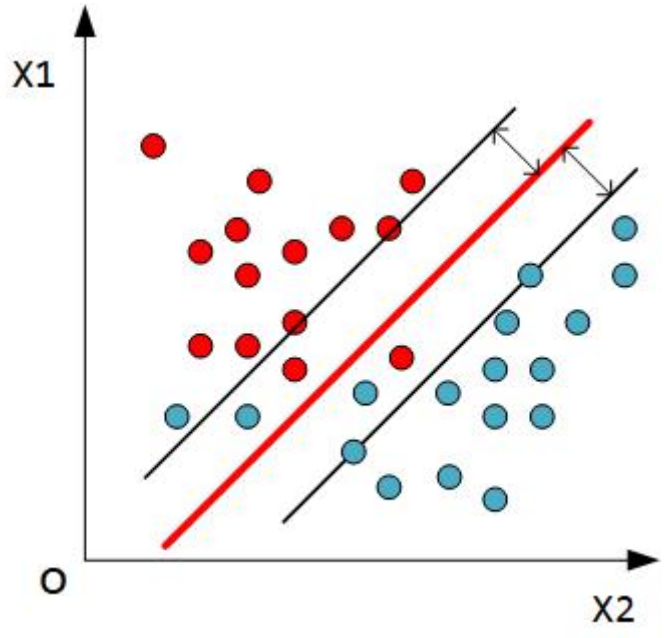
$$
f(x) = \sum_{i=1}^{N} a_i y_i k(x_i, x) + b
$$

6.软间隔

软间隔的引入也是为了解决**线性不可分问题**，退一步说，即使样本线性可分，我们也很难断定这个貌似线性可分的结果不是由于过拟合造成的，所以实际中我们一般采取软间隔形式的$\mathbf{SVM}$，所谓软间隔实际上指的是：我们允许我们的一些样本违反我们的约束条件即

$$
\text{for some i}, \quad y_i(w^T x_i + b) \leq 1
$$

如下图所示：



但我们也希望在最大化间隔的同时，不满足约束的样本条件应该尽可能的少，这样我们的优化目标可以写为以下这种形式

$$\min_{w,b} \quad \frac{1}{2}w^T w + C\sum_{i=1}^{N}\delta(y_i(w^T x_i + b) - 1)$$

其中

- δ 函数是“0/1损失函数”

$$\delta(z) = \begin{cases} 1, & \text{if } z < 0 \\ 0, & \text{otherwise} \end{cases}$$

- $C > 0$ 是一个超参数，训练前需要人为提前选择设定
- 当C很大时，意味着我们对于犯错的容忍度很小，优化函数迫使大多数样本满足约束
- 当C很小时，意味着我们对于错误的容忍度相对大一点

(1) Hinge Loss

1. Hinge Loss 的形式

由于δ 函数非凸，且不连续，我们常常用一些函数来替代它，一个常用的损失函数就是hinge loss 函数，其形式如下:

$$higheloss(z) = \max(0, 1 - z)$$

这时我们的优化问题可以变为

$$\min_{w,b} \quad \frac{1}{2}w^T w + C\sum_{i=1}^{N}\max(1 - y_i(w^T x_i + b), 0)$$

2.松弛变量

我们可以引入松弛变量 ξ 对问题进行等价变形

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}w^T w + C\sum_{i=1}^{N}\xi_i \\ s.t. \quad & y_i(w^T x_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned}$$

这就是常见的"软间隔支持向量机"，我们按着分析“硬间隔支持向量机”的方法同理可以得到其对偶问题

对偶问题：

$$\begin{aligned}\max_{\alpha} \quad & \sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j x_i^T x_j \\ s.t. \quad & \sum_{i=1}^{N}\alpha_i y_i=0 \\ & 0\le\alpha_i\le C \quad , \quad i=1\cdots N\end{aligned}$$

3.理解hinge loss

我们之前理解 SVM 是从一个几何直观出发，进而寻找最大化间隔。现在我们可以换一个角度来看待这个优化的形式。

$$\min_{w,b} \quad \frac{1}{2}w^T w+C\sum_{i=1}^{N}\max(1-y_i(w^T x_i+b),0)$$

并且可以换一种形式表示

$$\min_{w,b} \quad \frac{1}{2}\lambda w^T w+\sum_{i=1}^{N}\max(1-y_i(w^T x_i+b),0)$$

我们常常说机器学习有大致三个步骤

- 确定模型的函数空间
- 确定损失函数
- 优化损失函数

首先我们定义我们的模型的函数空间是线性函数

然后我们定义我们的损失函数，在这里我们可以将我们的损失函数分为两个部分看

- 损失项 $\sum_{i=1}^{N}\max(1-y_i(w^T x_i+b),0)$
- 正则项 $\frac{1}{2}w^T w$

损失项表明我们希望$y_i(w^T x_i+b)$的得分 越大越好，因为越大说明我们的分类越正确,这时我们不计较损失，得分小于一个间隔（这里是1）后，我们就开始计较损失了

正则项可以理解为防止过拟合，其存在可以使模型的复杂度降低

λ这个超参数用来对这两者折中

4. 优化损失函数

前面主要篇幅在说如何转化我们的优化目标，接下来我们来说明如何优化我们的目标函数，这里用到的方法仍然是机器学习和深度学习中最常用的方法**梯度下降**

梯度下降的大致流程如下

```
learning_rate=1e-3
while True:
    g=compute_gradient(x,y)
    theta=theta-learning_rate*g
```

我们不想再详细介绍梯度下降的原理，以及各种随机梯度下降的方法，我们只是介绍一下如何求解该损失函数的梯度。

> 梯度下降是一个通用方法，而对应的每个模型的梯度的求解则是每个模型的特色

首先我们得优化问题是一个凸问题，保证了局部极小等于全局极小

$$\min_{w,b} \quad \frac{1}{2}\lambda w^T w + \sum_{i=1}^{N} \max(1 - y_i(w^T x_i + b), 0)$$

> 注:因为正则项是一个二次范数的平方为凸函数，损失项为线性分段函数的非负加权和也为凸函数，所以整个损失函数为凸函数

梯度计算(只介绍w梯度的求法，b的求法类似)

令L 为损失函数，$l_i = \max(1 - y_i(w^T x_i + b), 0)$,$z_i = y_i(w^T x_i + b)$

$$\begin{aligned}\frac{\partial L}{\partial w} &= \sum_{i=1}^{N} \frac{\partial l_i}{\partial z_i}\frac{\partial z_i}{\partial w} + \lambda w \\ &= \sum_{i=1}^{N} \frac{\partial l_i}{\partial z_i} y_i x_i + \lambda w\end{aligned}$$

虽然$\max(1 - z, 0)$ 在0 处不可导，但实际中我们常常用Proximal gradient descent的方法来解决这个问题(使零处的导数为在-1到0之间任意的一个值)，常见的是

$$\frac{\partial l_i}{\partial z_i} = \begin{cases} 0, & z_i \geq 1 \\ -1, & \text{otherwise} \end{cases}$$

这样就可以计算出$\frac{\partial L}{\partial w}$,用它去更新$w$

$$w \leftarrow w - \eta \frac{\partial L}{\partial w}$$

7. 多分类的SVM LOSS

在多分类的SVM中我们对应的每一个类都存在一个相应的w_i 去描述，这样的w_i 合起来就是W，这样我们的线性函数分类器就变得很像神经网络中的线性层，是一个m 维到n维的映射

$$f(x_i; W, b) = Wx_i + b$$

输入一个样本我们的f 将输出在每个类别上的得分

而我们的Loss 这样定义，SVM 希望在正确类别上的得分，大于其他不正确分类上的得分，并且两者之差要大于一个间隔Δ,如果无法达到这个要求，我们将计较损失（意思如下图所示）

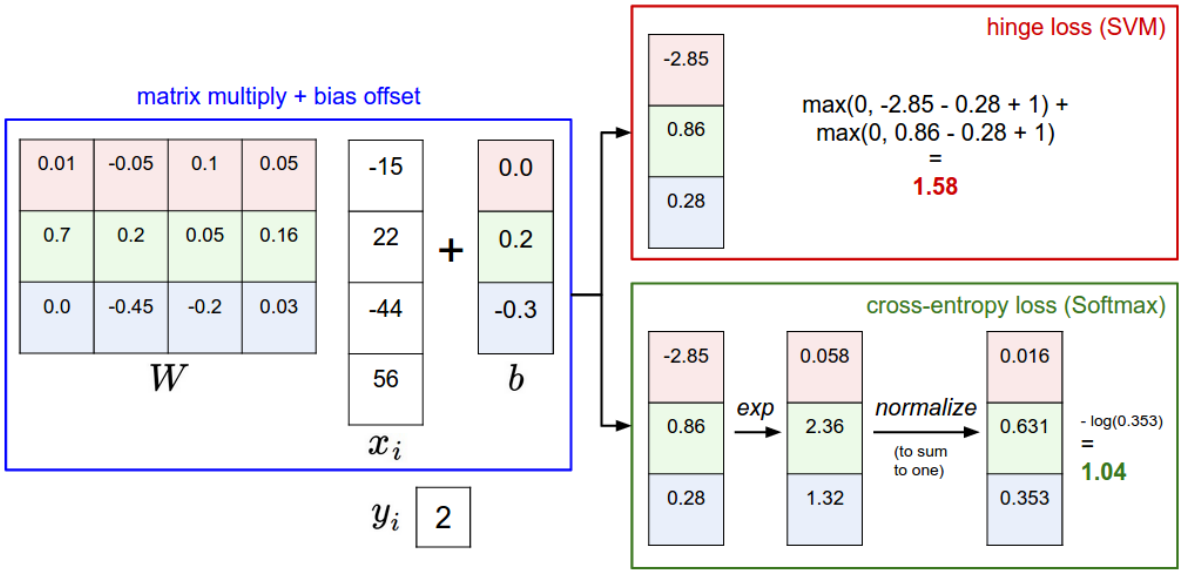
> 注:图源https://cs231n.github.io/linear-classify/

Hinge Loss

$$L_i = \sum_{j \neq y_i} \max(0, w_j^T x_i - w_{y_i}^T x_i + \Delta)$$

> 这里的Δ 通常取 1

如图阐述了L_i的计算方法

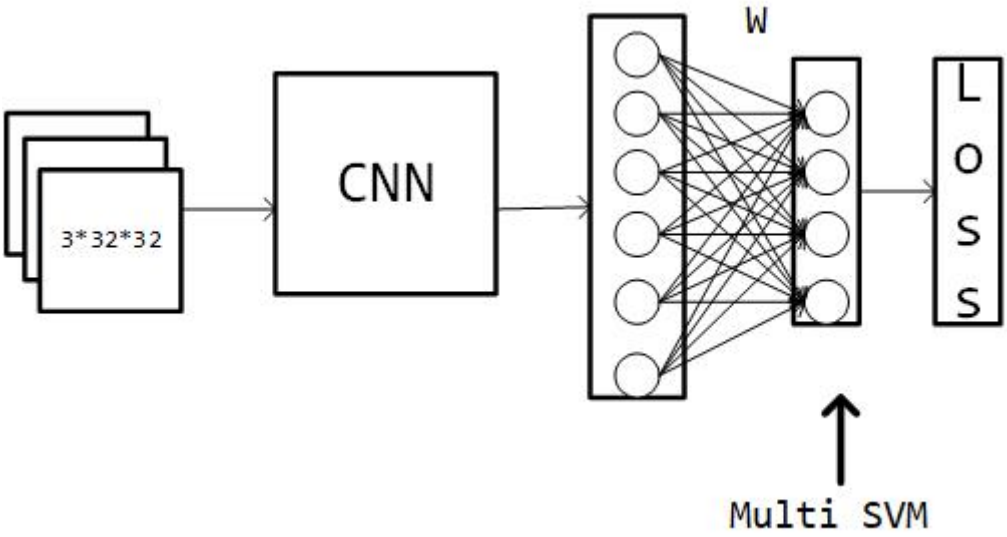


Total loss with regularization

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + \lambda \sum_k \sum_l W_{k,l}^2$$

8. Deep learning 中的svm

最后我们将讨论svm在深度学习中的应用，我们可以这样来简单的理解深度学习，原来我们在传统的机器学习中常常需要人工手动设计特征，这个过程比较繁琐，且需要一定的领域内的经验(例如在图像识别方面常常使用:Color Histogram,Histogram of Oriented Gradients,Bag of Words 等方法提取特征)而深度学习利用神经网络，让模型自己去学习如何设计特征。下图就是一个常见的CNN+SVM的架构



- 前面的卷积层其实做的就是一个特征提取的过程
- 最后的全连接层和损失函数共同构成我们的多分类SVM

具体细节可以参考下面的论文:

- Deep Learning using Linear Support Vector Machines (Yichuan Tang)

9. 小结

SVM 是监督学习中最有影响力的方法之一，而从它衍生出的核方法直接推动了统计学习的发展。即使是在深度学习大行其道的今天，SVM依旧在样本量比较小的任务中有着不可替代的作用。

在学习了解SVM中，笔者感到了它的博大精深，它的理论的优雅，自己学识尚浅，感觉仍然对其理解不深刻，希望读者能指出不足与谬误。

同时本文对SVM的介绍只是一个方面，其还有诸多变体如:SVR 等没有涉及，有兴趣的可以自行查阅