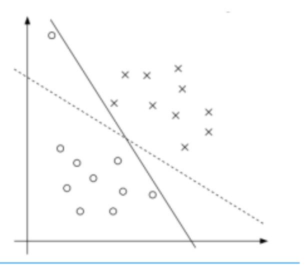
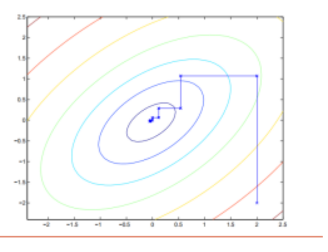
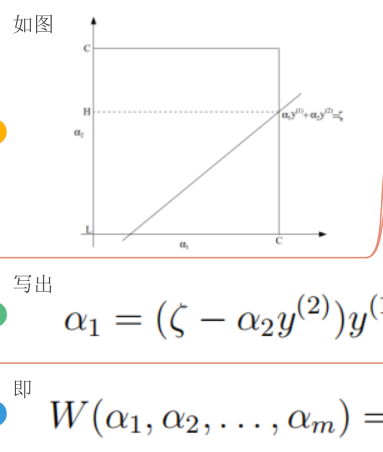
Support Vector Machine

- Kernels
  - 特征映射
    - 之前讲线性回归的时候，说过房屋的居住面积问题，我们考虑使用特征x,x^2,x^3来进行拟合一个立方函数
    - 这种将原始的输入值x映射为(x, x^2, x^3)的变换叫做 **特征映射(feature mapping)**
    - 记作
      $$\phi(x)=\begin{bmatrix} x \\ x^2 \\ x^3 \end{bmatrix}$$
    - 为什么要进行特征映射？
      - 一，为了更好地拟合
      - 二，样本可能存在线性不可分的情况，而映射到高维空间之后，往往就可分了
    - 将特征映射应用到SVM问题上， 即把所有的x替换为ϕ(x)
    - 如果原始特征内积是< x,z >，映射后为< ϕ(x),ϕ(z) >
  - 由此给出**核(Kernel)** 的定义 $$K(x,z)=\phi(x)^T\phi(z)$$
    - 这样，只需要把原来算法中的<x,z>替换为K(x,z)就可以使用特征映射ϕ进行机器学习了
    - 给定一个特征映射 ϕ，很容易就可以找出 ϕ(x) 和 ϕ(z)，然后取一下内积，就能计算 K (x, z)了。不过还不仅如此，更有意思的是，通常情况下对这个K (x, z) 的计算往往都会非常容易（very inexpensive），甚至即便 ϕ(x) 本身很不好算的时候（可能是由于向量维度极高）也如此。在这样的背景下，给定一个特征映射 ϕ，我们就可以使用 SVM（支持向量机算法）来对高纬度特征空间进行机器学习，而可能根本不用麻烦地区解出来对应的向量 ϕ(x)。
  - 接下来看一个例子
    - 设 $$K(x,z)=(x^Tz)^2$$
    - 我们可以推出
      $$\begin{aligned} K(x,z) &= \left(\sum_{i=1}^n x_iz_i\right)\left(\sum_{j=1}^n x_iz_i\right) \\ &= \sum_{i=1}^n\sum_{j=1}^n x_ix_jz_iz_j \\ &= \sum_{i,j=1}^n (x_ix_j)(z_iz_j) \end{aligned}$$
    - 也就是 $$K(x,z)=\phi(x)^T\phi(z)$$
      - 其中n=3时
        $$\phi(x)=\begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \end{bmatrix}$$
    - 这时候我们发现，计算ϕ(x)需要O(n^2)时间复杂度，而计算K(x,z)只需要O(n)级别的时间（因为K(x,z)是原始特征x,z的内积的平方）
  - 再来一个核
    $$\begin{aligned} K(x,z) &= (x^Tz+c)^2 \\ &= \sum_{i,j=1}^n (x_ix_j)(z_iz_j)+\sum_{i=1}^n(\sqrt{2c}x_i)(\sqrt{2c}z_i)+c^2. \end{aligned}$$
    - n=3时ϕ(x)为
      $$\phi(x)=\begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \\ \sqrt{2c}x_1 \\ \sqrt{2c}x_2 \\ \sqrt{2c}x_3 \\ c \end{bmatrix}$$
    - 参数 c 控制了第一组 xi 和第二组 xixj 的相对权重（relative weighting）
  - 更一般地，核 $$K(x,z)=(x^Tz+c)^d$$
    - 对应的是该维度的空间映射 $\binom{n+d}{d}$
    - 对应所有从 xi1，xi2， ...xik 一直到 d 这种形式的所有多项式
    - 然而，计算该核仍然只需O(n)时间
  - 从另一个角度看核
    - K(x,z) is large if x,z similar;
      K(x,z) is small if x,z dissimilar;(比如近似正交)
    - 这样，我们就可以把核看成是对ϕ(x) 和 ϕ(z) 的近似程度的一种度量手段，或者说是对x,z近似程度的一种度量手段
    - 选择如下函数对x,z的近似程度进行衡量
      $$K(x,z)=\exp\left(-\frac{||x-z||^2}{2\sigma^2}\right)$$
      - 如果x,z很相近，K近似为1，反之K近似为0
      - 这个核叫做高斯核，也叫做径向基函数(Radial Basis Function 简称 RBF)，对应的是一个无穷维度的特征映射。
      - 在某些特定的条件下也可以作为SVM的核
  - 进一步推广，给定某个函数 K，我们该怎样能够确定这个函数是不是一个有效的核（valid kernel）呢？例如，我们能否说如果存在着某一个特征映射 ϕ，则对于所有的 x 和 z 都有 K (x, z) = ϕ(x)T ϕ(z)？
  - 给出核矩阵的定义（Kernel matrix） $$K_{ij}=K(x^{(i)},x^{(j)})$$
    - 其中K是有效核，{x1, ... , xm}是m个点的集合，K矩阵是m*m的矩阵
    - 注：我们对K进行了重复使用，它即代表有效核，又代表矩阵
    - K是有效核可以推出
      - $$K_{ij}=K(x^{(i)},x^{(j)})=\phi(x^{(i)})^T\phi(x^{(j)})=$$
      - $$\phi(x^{(j)})^T\phi(x^{(i)})=K(x^{(j)},x^{(i)})=K_{ji}$$
      - 也就是说K是对称矩阵
    - 更进一步
      - letting $\phi_k(x)$ denote the k-th coordinate of the vector $\phi(x)$
      - 可以推出
        $$\begin{aligned} z^TKz &= \sum_i\sum_j z_iK_{ij}z_j \\ &= \sum_i\sum_j z_i\phi(x^{(i)})^T\phi(x^{(j)})z_j \\ &= \sum_i\sum_j z_i\sum_k\phi_k(x^{(i)})\phi_k(x^{(j)})z_j \\ &= \sum_k\sum_i\sum_j z_i\phi_k(x^{(i)})\phi_k(x^{(j)})z_j \\ &= \sum_k\left(\sum_i z_i\phi_k(x^{(i)})\right)^2 \\ &\geq 0. \end{aligned}$$
        - 说明K是半正定矩阵
    - 结论：如果K是有效核，则K矩阵是对称半正定矩阵
  - 默瑟核（Mercer kernel）
    - **默瑟定理**：实际上，K是对称半正定矩阵和K是有效核的互为充分必要条件
    - 这个核函数就叫做默瑟核
    - **Theorem (Mercer).** Let $K:\mathbb{R}^n\times\mathbb{R}^n\mapsto\mathbb{R}$ be given. Then for K to be a valid (Mercer) kernel, it is necessary and sufficient that for any $\{x^{(1)},\dots,x^{(m)}\}$, $(m<\infty)$, the corresponding kernel matrix is symmetric positive semi-definite.
- 正则化和不可分情况处理
  - Regularization and the non-separable case
  - 到目前为止的所有SVM内容，都是基于数据是线性可分的假设，但是这个假设有时候是不成立的，比如下图带有异常值，导致分界线出现偏移

    
  - 要想让算法适用于非线性可分的数据集，那么就要降低算法对异常值的敏感度，也就是说要允许某些点违背约束条件（函数间隔大于1）
  - 方法：加入惩罚项。使用L1正则化来重构算法，得到下列新的模型，也叫作L1软间隔SVM模型
  - $$\begin{aligned} \min_{\gamma,w,b}\ & \frac{1}{2}||w||^2+C\sum_{i=1}^m\xi_i \\ \text{s.t.}\ & y^{(i)}(w^Tx^{(i)}+b)\geq 1-\xi_i,\quad i=1,\dots,m \\ & \xi_i\geq 0,\quad i=1,\dots,m. \end{aligned}$$
    - 非负参数ξi叫做松弛变量，它允许某些样本点的函数间隔小于1，即允许某些点在最大间隔区间里面，或者某些点的函数间隔是负数。
    - 后面的CΣξi表示离群点越多，目标函数值越大，而我们要求的是最小的目标函数值。C是离群点的权重，越大表示离群点对目标函数的影响越大，也就是越不希望看到离群点。
    - 目标函数控制了离群点的数量和程度，使大部分样本仍符合条件
  - 修改后的模型对应的拉格朗日函数
    - $$\mathcal{L}(w,b,\xi,\alpha,r)=\frac{1}{2}w^Tw+C\sum_{i=1}^m\xi_i-\sum_{i=1}^m\alpha_i\left[y^{(i)}(x^Tw+b)-1+\xi_i\right]-\sum_{i=1}^m r_i\xi_i$$
    - αi 和 ri 都是拉格朗日乘数因子，>=0
  - 与往常一样，对w,b求偏导，再带回，得到该问题的对偶式
    - $$\begin{aligned} \max_\alpha\ & W(\alpha)=\sum_{i=1}^m\alpha_i-\frac{1}{2}\sum_{i,j=1}^m y^{(i)}y^{(j)}\alpha_i\alpha_j\langle x^{(i)},x^{(j)}\rangle \\ \text{s.t.}\ & 0\leq\alpha_i\leq C,\quad i=1,\dots,m \\ & \sum_{i=1}^m\alpha_iy^{(i)}=0, \end{aligned}$$
    - 还是和往常一样，可以把w用αi进行描述
    - 往常详细解法见脑图SVM的Part I的“最优边界分类器”中“求该函数的对偶形式”
    - 神奇的一点，加入L1正则化后，对对偶问题的唯一改变只是约束从 0 ≤ αi变成了 0 ≤ αi ≤ C。这里对 b□ 的计算也受到了影响而有所变动
    - KKT对偶互补条件为
      $$\begin{aligned} \alpha_i=0 &\ \Rightarrow\ y^{(i)}(w^Tx^{(i)}+b)\geq 1 \\ \alpha_i=C &\ \Rightarrow\ y^{(i)}(w^Tx^{(i)}+b)\leq 1 \\ 0<\alpha_i<C &\ \Rightarrow\ y^{(i)}(w^Tx^{(i)}+b)=1. \end{aligned}$$
  - 在下一阶段“THe SMO algorithm”中将求解该对偶式
- The SMO algorithm
  - sequential minimal optimization(序列最小化优化算法)
  - Coordinate ascent（坐标上升算法）
    - 先抛开具体问题，看这种算法
    - 假如要解决无约束优化问题 $$\max_\alpha W(\alpha_1,\alpha_2,\dots,\alpha_m)$$
    - 以前学过的优化算法有梯度下降法和牛顿法
    - 坐标上升算法 Loop until convergence: {
      For $i=1,\dots,m$, {
      $\alpha_i:=\arg\max_{\hat{\alpha}_i}W(\alpha_1,\dots,\alpha_{i-1},\hat{\alpha}_i,\alpha_{i+1},\dots,\alpha_m)$
      }
      }
      - 即每次通过优化W来调整参数αi
      - 至于调整αi的顺序，我们这里按照变量排列顺序来调整，不过还有其他更复杂的顺序
    - 坐标上升法的每一步中，移动的方向都是平行于某个坐标轴的（parallel to one of the axes），因为每次都只对一个变量进行了优化。以下是一个二维（只有α1，α2）示意图

      
  - SMO算法推导
    - SVM对偶优化问题
      $$\begin{aligned} \max_\alpha\ & W(\alpha)=\sum_{i=1}^m\alpha_i-\frac{1}{2}\sum_{i,j=1}^m y^{(i)}y^{(j)}\alpha_i\alpha_j\langle x^{(i)},x^{(j)}\rangle \\ \text{s.t.}\ & 0\leq\alpha_i\leq C,\quad i=1,\dots,m \\ & \sum_{i=1}^m\alpha_iy^{(i)}=0. \end{aligned}$$
    - 由约束可得 $$\alpha_1y^{(1)}=-\sum_{i=2}^m\alpha_iy^{(i)}.$$
      - 进而可得 $$\alpha_1=-y^{(1)}\sum_{i=2}^m\alpha_iy^{(i)}$$
      - 这一步用到了 $y^{(1)}\in\{-1,1\}$, and hence $(y^{(1)})^2=1$
    - **我们可以看出，按照坐标上升算法的思路每次优化一个参数是做不到的，因为有约束的限制，所以每次至少变动两个参数（如果只变动一个参数，αi之和=0这个条件将不再成立）**
    - 基于这种情况出现了SMO算法
      - Repeat till convergence {
        1. Select some pair α_i and α_j to update next (using a heuristic that tries to pick the two that will allow us to make the biggest progress towards the global maximum).
        2. Reoptimize $W(\alpha)$ with respect to α_i and α_j, while holding all the other α_k's $(k\neq i,j)$ fixed.

        }
      - 第一步用启发式方法选取αi,αj;
        第二步固定其他参数，确定W取极值条件下的αi,αj取值，（而αj是可以用αi进行表示的）
      - SMO之所以高效是因为固定其他参数后，对一个参数的优化过程很高效
      - 下面讨论具体方法
      - 1 根据约束得到 $$\alpha_1y^{(1)}+\alpha_2y^{(2)}=-\sum_{i=3}^m\alpha_iy^{(i)}$$
      - 2 记为 $$\alpha_1y^{(1)}+\alpha_2y^{(2)}=\zeta$$
      - 3 如图

        
        - [0,C]是第一个约束条件 在这里L=0
        - 需要注意的是，通过上面的约束条件，还能知道 L ≤ α2 ≤ H;
      - 4 写出 $$\alpha_1=(\zeta-\alpha_2y^{(2)})y^{(1)}$$
      - 5 即 $$W(\alpha_1,\alpha_2,\dots,\alpha_m)=W((\zeta-\alpha_2y^{(2)})y^{(1)},\alpha_2,\dots,\alpha_m)$$
      - 6 把 α3, ..., αm 当做常量，就能证明上面这个函数其实只是一个关于 α2 的二次函数。也就是说，可以写成 $a\alpha_2^2+b\alpha_2+c$
        - 而对于这种二次函数求极值，就非常简单了
      - 我们用如下符号表示最后的α2取值 $\alpha_2^{new,unclipped}$
      - 7 别忘了，如上图所示，α是有约束条件的，最后我们加上约束条件，得到
      - $$\alpha_2^{new}=\begin{cases} H & \text{if } \alpha_2^{new,unclipped}>H \\ \alpha_2^{new,unclipped} & \text{if } L\leq\alpha_2^{new,unclipped}\leq H \\ L & \text{if } \alpha_2^{new,unclipped}<L \end{cases}$$
      - 8 得到了α2，α1也就得到了