# K-Nearest Neighbors(K 近邻算法)

KNN 是一种无参分类器(non-parametric classifier)
对于每一个新需要分类的数据点, 执行一下几步操作:

- 选择最接近待分类点中的 K 个已经分类的点, 用 Euclidean distance() 集合距离作为评判标准
- 在 K 个点中数每一个类型中在这 K 个点的占有数量
- 新的分类点属于这 K 个点中最多的类型

由于 K 的不同取值, 结果可能有所不同

- 选取较小的 K 值会导致决策边界不稳定
- 较大的 K 值更适合分类，因为它可以平滑决策边界
- 可以做一个错误率与 K 值相关的图像, 来选择合适的 K 值

## 其他计算距离的算法

- Manhattan Distance:$d(x,y)=\sum_{i=1}^{n}|x_i-y_i|$
- Hamming Distance:找不同相加,如果$x_i$和$y_i$不相同, 距离就加 1, 否则加 0, 是一种算分类变量之间距离的办法(分类的 coding 没有实际距离含义)
- Minkowski Distance: 曼哈顿距离的扩展$D(x,y)=\left(\sum_{i=1}^{n}|x_i-y_i|^p\right)^{\frac{1}{p}}$,$\lim_{p\to\infty}$

# SVM(Support Vector Machine, 支持向量机)

是一种用于回归和分类的线性模型, 产生分割数据的线或者超平面, 最基本的 SVM 是为了找到 把数据分割成两个类型的线.

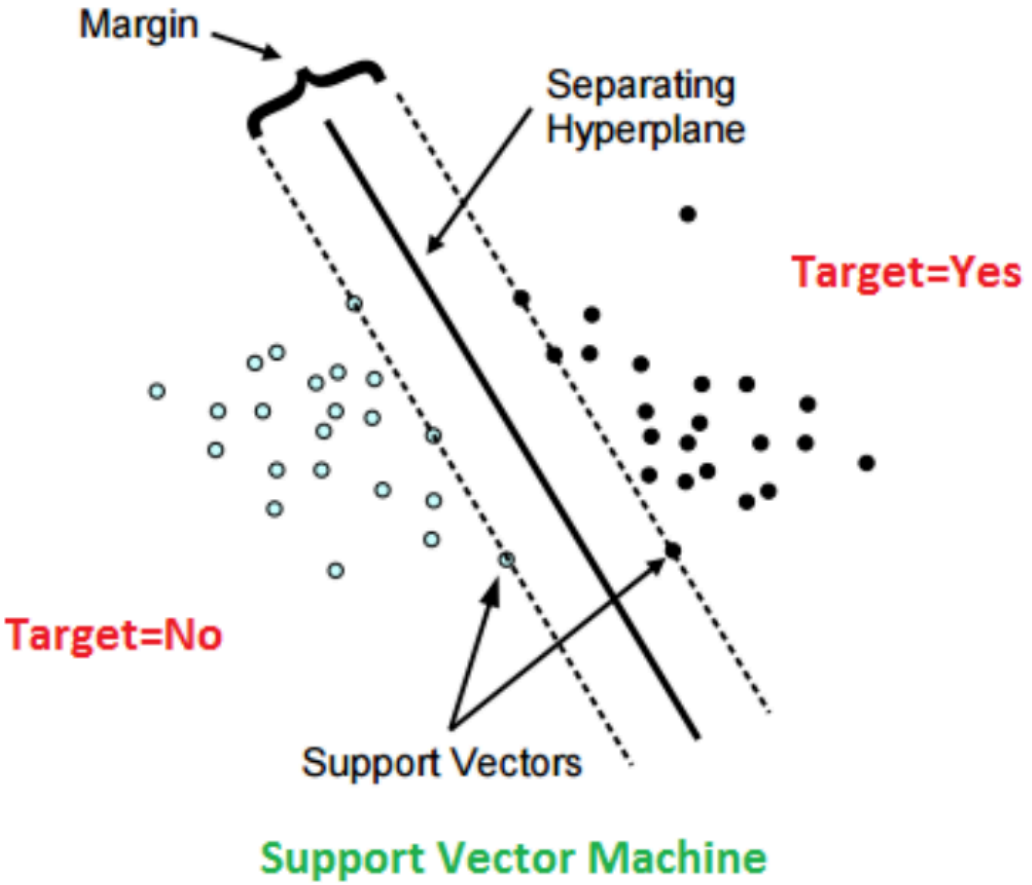


Figure 1: 超平面(hyperplane),余量(margin)的概念

Support Vector 是最接近超平面的数据点, 也是定义了 margin 的数据点, 在 SVM 算法 中, 要尽量最大化 margin

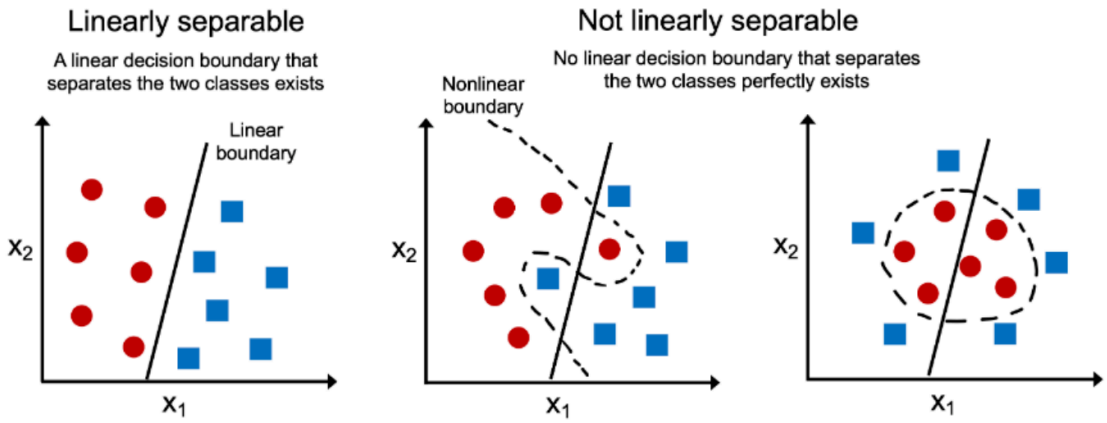


Figure 2: 不一定是能够很好的用直线分割

SVM 可以分为:

- 线性可分割 SVM
- 非线性 SVM

## 线性 SVM (见 svm.md)

关键:最大化余量(只用关注 SVM)

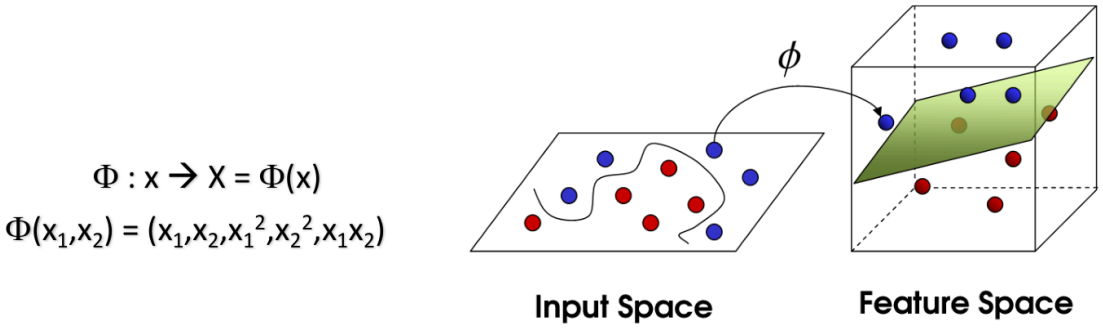


Figure 3: 对于非线性情况, 可以想办法映射成线性的特征, 常常会增加维度之类的, 这样可以找到可以分割的超平面

## Kernel function

$$K(\boldsymbol{x}, \boldsymbol{y}) = \Phi(x) \cdot \Phi(y)$$

$x$,$y$:n 维度输入

$\Phi$:把 n 维输入映射到$m$维度的函数($m \gg n$)

$\cdot$代表向量点乘

但直接映射之后再计算内积可能计算量特别大, 要用“Kernel-trick”, 可以避免显式的 映射.

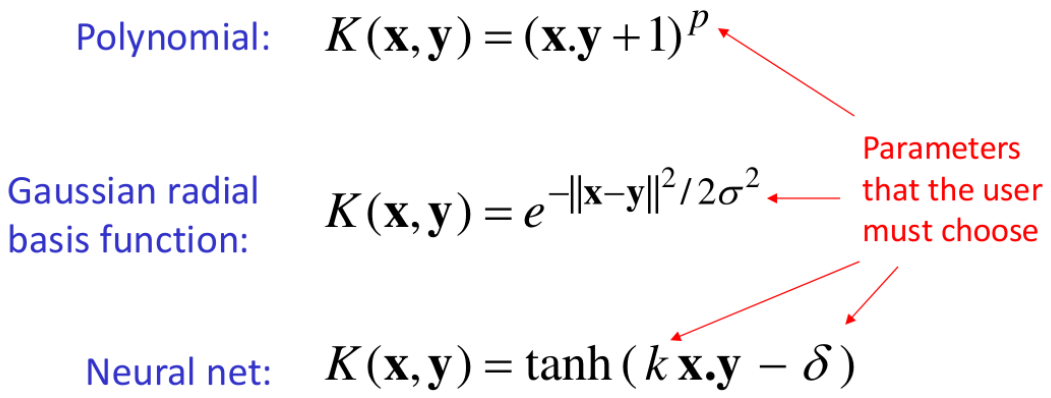


Figure 4: 常用的用于特征提取的核

## 多类型(Multi-class)SVM

要把多分类问题分解成多个二分类问题

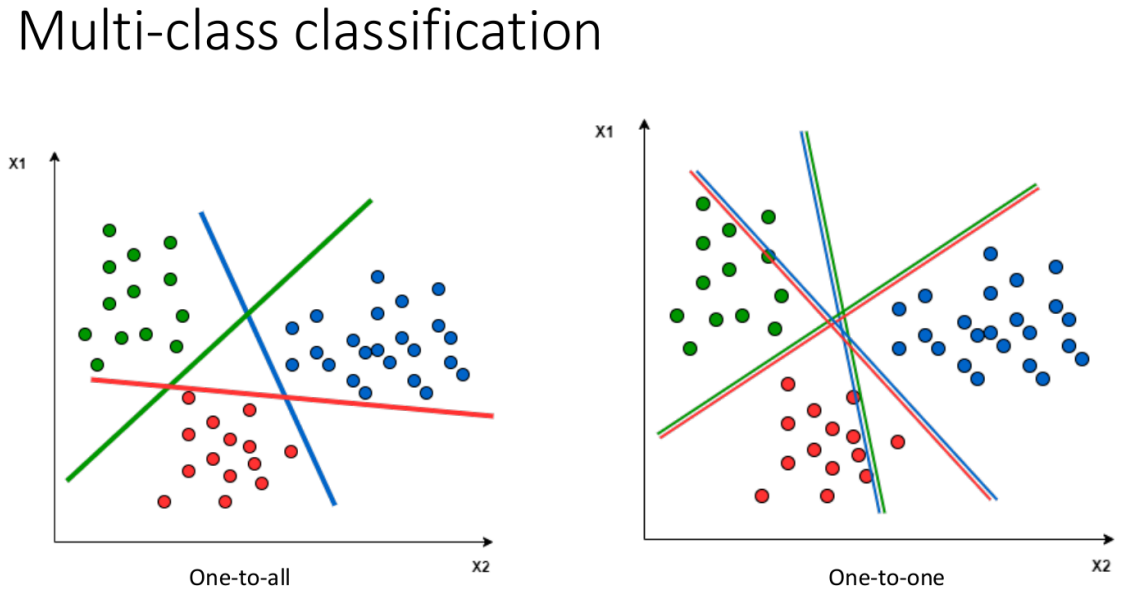


Figure 5: 一遍要吗划分一种类型和其他类型的整体分解,要吗两两划分组合

# Decision Tree(决策树)

- 非叶结点(internal nodes)代表数据集的一种特征
- 分支(branches)代表决定规则(decision rules)
- 叶节点(leaf nodes) 代表输出结果(outcome)

Pruning:剪枝,移除不想要的部分

## Decision Principle

- 建树用的是(CART, Classification and Regression Tree algorithm)

### 算法过程

asm.md

## ASM 算法

怎么选择用于建立下一个子节点的最佳属性呢?

- Information Gain:信息增益是衡量数据集在根据某个属性进行划分后熵的变化量。它表示一个属性为分类提供了多少信息, 信息增益越大，说明这个属性越适合用来划分数据集,

决策树算法总是优先选择信息增益最大的属性作为分裂节点.

$$\text{IG} = \text{原始数据的熵} - \sum \text{基于这个划分后个子集样本的加权平均熵}$$

$E(s) = -P(\text{yes}) \log_2 P(\text{yes}) - P(\text{no}) \log_2 P(\text{no})$ 交叉熵,P 是指占据的比例

- Gini Index: 倾向于选择基尼系数低的特征(类型更加集中).

$$\text{GI} = 1 - \sum_j P_j^2$$

$P_j$:类别$j$的样本比例
选择根据这个属性分裂后, 各个子树加权求和最小的结果, 把这个结果作为分裂节点.