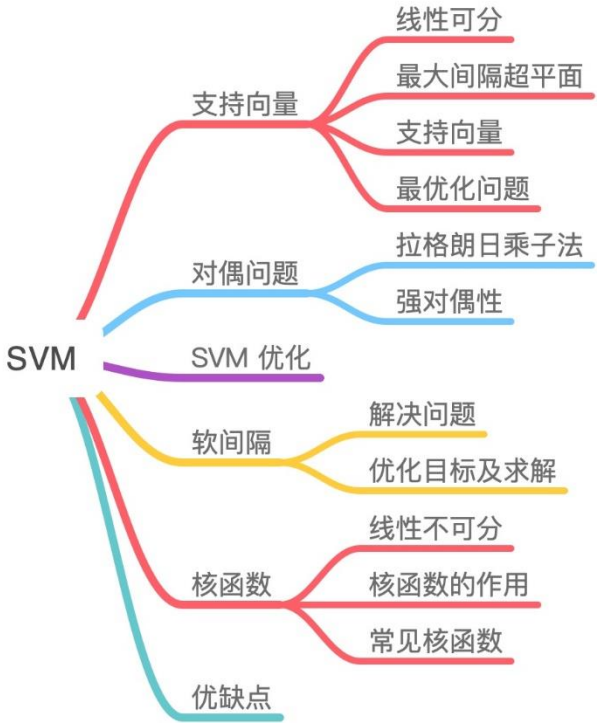
支持向量机



SVM 是一个非常优雅的算法，具有完善的数学理论，虽然如今工业界用到的不多，但还是决定花点时间去写篇文章整理一下。

SVM-支持向量机原理详解与实践

1. 前言

去年由于工作项目的需要实际运用到了 SVM 和 ANN 算法,也就是支持向量机和人工神经网络算法，主要是实现项目中的实时采集图片（工业高速摄像头采集）的图像识别的这一部分功能，虽然几经波折，但是还好最终还算顺利完成了项目的任务，忙碌一年，趁着放假有时间好好整理并总结一下，本文的内容包括：前面的部分是对支持向量机原理的分析，后半部分主要直接上手的一些实践的内容。

本文的原理部分针对支持向量机的原理，特别拉格朗日对偶性，求解拉个拉格朗日函数，以及和函数与核技巧再到软间隔和正则化等重要内容做了一些讨论。

实践部分的目标则是通过对实践时碰到的问题，调参的过程的讲解可以对前半部分讲解的 SVM 原理部分的内容有一个更深入的了解。

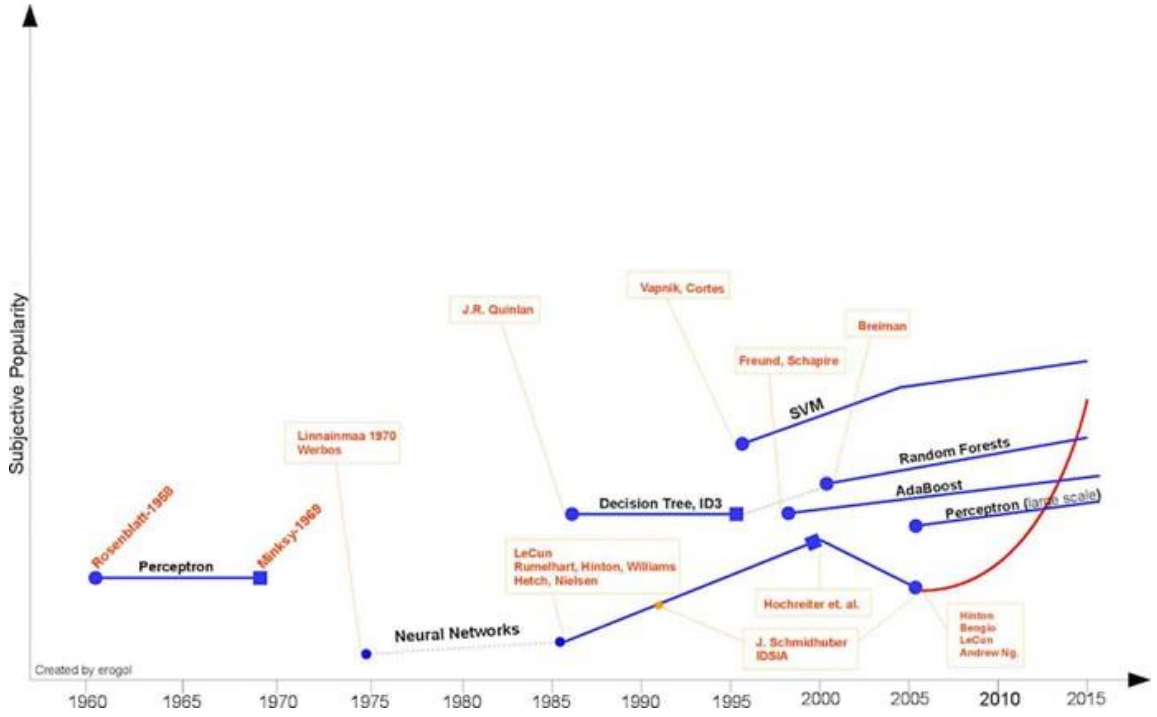
1. SVM、机器学习与深度学习

1.人工智能领域

在大数据，人工智能的时代，深度学习可以说火得一塌糊涂。美国硅谷的大公司都在布局着这个领域，而中国国内，腾讯，百度，阿里巴巴等等知名企业也都在这个领域争先发力，2017 年初，百度迎来陆奇-前微软全球执行副总裁，人工智能领域世界级的权威，要知道百度还有人工智能大牛 Andrew Ng – 吴恩达。所有迹象表明人工智能必然是继互联网之后的全球各大公司甚至国家必争的高地。

1.机器学习与深度学习

由于深度学习在大数据预测能力上的卓越表现，当下出现了深度学习是否会替代传统机器学习算法并淘汰他们的讨论，但是另一方面，大多数人仍然相信深度学习不会代替其他的模型或者算法。对于大多数的应用，像一些简单的算法如逻辑回归、支持向量机表现的已经很不错了，使用深度学习会让问题复杂化。
深度学习是可以应用到大部分领域的，但是就像前面说的，深度学习并非所有问题的最优方案，如果你的工作中有用到机器学习算法，你可以尝试传统的机器学习算法，也可以达到很好的效果。虽然现在已经有一些工作去把各领域的知识融入到深度学习中的，但这并不能完全替代原有的。



上图是一个关于机器学习算法的时间线来自于 Eren Golge。

就像在 20 世纪早期 SVM 一样，深度学习会成为主流，但首先深度学习应当解决其在大数据需求及复杂性方面的问题，这样它才会成为人们的第一选择。

1.SVM 简介

SVM(support vector machine)简单的说是一个分类器，并且是二类分类器。

- Vector：通俗说就是点，或是数据。
- Machine：也就是 classifier，也就是分类器。

SVM 作为传统机器学习的一个非常重要的分类算法，它是一种通用的前馈网络类型，最早是由 Vladimir N.Vapnik 和 Alexey Ya.Chervonenkis 在 1963 年提出，目前的版本（soft margin）是 Corinna Cortes 和 Vapnik 在 1993 年提出，1995 年发表。深度学习（2012）出现之前，SVM 被认为是机器学习中近十几年最成功表现最好的算法。

1. SVM 原理分析

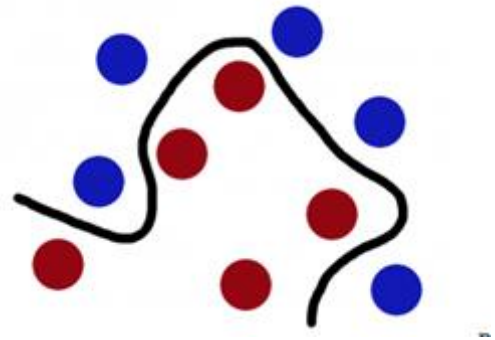
1.快速理解 SVM 原理

很多讲解 SVM 的书籍都是从原理开始讲解，如果没有相关知识的铺垫，理解起来还是比较吃力的，以下的一个例子可以让我们对 SVM 快速建立一个认知。

给定训练样本，支持向量机建立一个超平面作为决策曲面，使得正例和反例的隔离边界最大化。

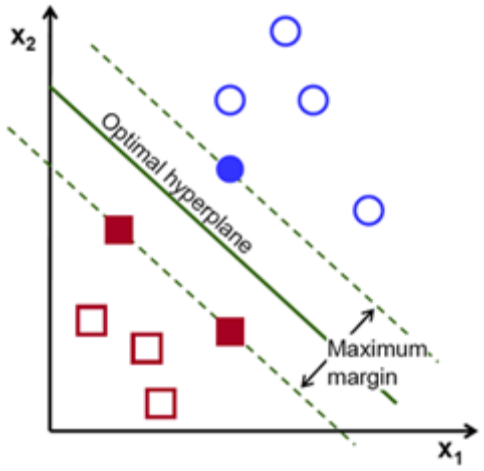
决策曲面的初步理解可以参考如下过程，

1. 如下图想象红色和蓝色的球为球台上的桌球，我们首先目的是找到一条曲线将蓝色和红色的球分开，于是我们得到一条黑色的曲线。



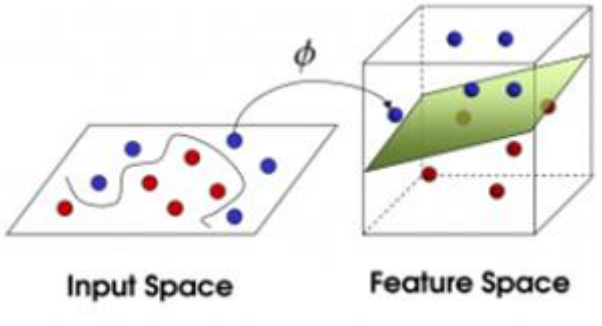
图一.

2） 为了使黑色的曲线离任意的蓝球和红球距离（也就是我们后面要提到的 margin）最大化，我们需要找到一条最优的曲线。如下图，



图二.

3） 想象一下如果这些球不是在球桌上，而是被抛向了空中，我们仍然需要将红色球和蓝色球分开，这时就需要一个曲面，而且我们需要这个曲面仍然满足跟所有任意红球和蓝球的间距的最大化。需要找到的这个曲面，就是我们后面详细了解的最优超平面。



4) 离这个曲面最近的红色球和蓝色球就是 Support Vector。

1.线性可分和线性不可分

线性可分-linearly separable, 在二维空间可以理解为可以用一条直线（一个函数）把两类型的样本隔开，被隔离开来的两类样本即为线性可分样本。同理在高维空间，可以理解为可以被一个曲面(高维函数)隔开的两类样本。

线性不可分，则可以理解为自变量和因变量之间的关系不是线性的。

实际上，线性可不分的情况更多，但是即使是非线性的样本通常也是通过高斯核函数将其映射到高维空间，在高维空间非线性的问题转化为线性可分的问题。

1.函数间隔和几何间隔

- 函数间隔 functional margin: 给定一个训练样本$(x^{(i)}, y^{(i)})$有：

$$\hat{\gamma}^{(i)} = y^{(i)}(w^T x + b)$$

函数间隔代表了特征是正例或是反例的确信度。

- 几何间隔 geometrical margin：

$$\gamma^{(i)} = \frac{(w^T x + b)}{\|w\|}$$

向量点到超平面的距离 $r = \frac{g(x)}{\|w\|}$ （其中$g(x) = w^T x + b$，后面详细介绍）

1.超平面分析与几何间隔详解

前面已经对 SVM 的原理有了一个大概的了解，并且简单介绍了函数间隔和几何间隔的概念，为了更好的理解线性可分模式下超平面，以下将进行深入的剖析推导过程，我们假设有训练样本集，期望的响应为，这里我们用类+1 和类-1 来代表，以表明样本是线性可分的。

决策曲面方程如下：

$$w^T x + b = 0$$

其中

x：输入向量，也就是样本集合中的向量；

w：是可调权值向量，每个向量可调权值；

T：转置，向量的转置；

b：偏置，超平面相对原点的偏移。

根据逻辑回归定义$w^Tx+b=0$展开其实就是：

$$w_0x_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = 0$$

其中假设约定$x_0=1$，于是$w_0x_0 = w_0$将w_0替换成 b；则有：

$$b + w_1x_1 + w_2x_2 + \cdots + w_nx_n = 0$$

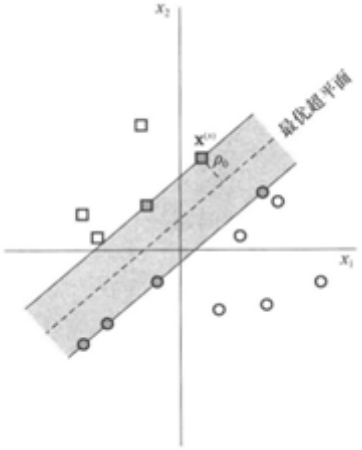
而$w_1x_1 + w_2x_2 + \cdots + w_nx_n = w^Tx$（T 是转置）所以有：

$$w^Tx+b=0$$

这里假设模式线性可分：

$$w^Tx_i + b \geq 0 \quad 当 \ d_i = +1$$
$$w^Tx_i + b < 0 \quad 当 \ d_i = -1$$

线性可分模式下最优超平面的示意图如下：



如上图所示：

- **ρ_0为分离边缘，即超平面和最近数据点的间隔。如果一个平面能使ρ_0最大，则为最优超平面。**
- **灰色的方形点和原形点就是我们所说的支持向量。**

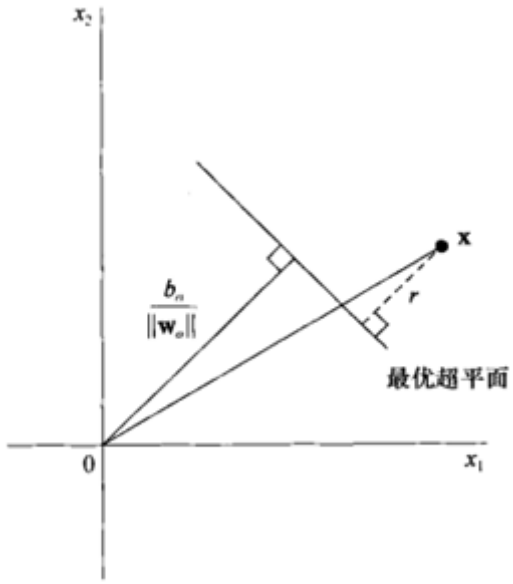
假设和向量和偏置的最优解，则最优超平面的函数为：

$$w_0^T x + b_0 = 0$$

相应的判别函数是：

$$g(x) = w_0^T x + b_0$$

以下是点 x 到最优超平面的二维示意图：



由上图可知 r 为点 x 到最优超平面的距离：

$$r = \frac{g(x)}{\|w\|}$$

那么代数距离 $r = \frac{g(x)}{\|w\|}$ 是如何得到的呢？通过将 $x = x_p + r\frac{w_0}{\|w_0\|}$ 带入

$$g(x) = w_0^T x + b_0$$

可以得到 r，其中：

- x_p 为 x 在最优超平面的正轴投影，
- $g(x_p)=0$，因为 x_p 在平面上

下面给出一种更为简单且直观的理解：

首先我们必须要知道 ***Euclidean norm*** 范数，即欧几里德范数（以下用 w 表示多维的向量）：

$$\|w\| = (w_1^2 + w_2^2 + \cdots + w_n^2)^{\frac{1}{2}} = \sqrt{w_1^2 + w_2^2 + \cdots + w_n^2}$$

再参考点 (x_0, y_0, z_0) 到面的距离公式

$$d = \frac{Ax_0 + By_0 + Cz_0 + D}{\sqrt{A^2 + B^2 + C^2}}$$

也就是

$$d = \frac{f(x_0, y_0, z_0)}{\sqrt{A^2 + B^2 + C^2}}$$

类似的,扩展到多维的 w 向量 $(w_1, w_2 \ldots w_n)$ 也是一样，代数距离 r 类似于 d，而

$g(x) = w^T x + b = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$ 类似于 ，所以展开后也就是：

$$r = \frac{g(x)}{\|w\|} = \frac{w^T x + b}{(w_1^2 + w_2^2 + \cdots + w_n^2)^{\frac{1}{2}}} = \frac{x_1 + w_2x_2 + \cdots + w_nx_n + b}{\sqrt{w_1^2 + w_2^2 + \cdots + w_n^2}}$$

对比点平面的公式，以上的 r 也就多维度空间向量的到最优超平面的距离。

再看上图，如果 x = 0 即原点则有

$$r = \frac{b}{\|w\|}$$

那么$r = \frac{b}{\|w\|}$是如何得到的呢？很简单，如上面分析的

$$r = \frac{g(x)}{\|w\|} = \frac{w^T x + b}{(w_1^2 + w_2^2 + \cdots + w_n^2)^{\frac{1}{2}}}$$

因为 x = 0，它在原点，它与任意可调权值向量 w 相乘都等于 0，于是有：

$$r = \frac{g(x)}{\|w\|} = \frac{w^T x + b}{(w_1^2 + w_2^2 + \cdots + w_n^2)^{\frac{1}{2}}} = \frac{b}{\sqrt{w_1^2 + w_2^2 + \cdots + w_n^2}} = \frac{b}{\|w\|}$$

注意 b 为偏置，只是决定了决策曲面相对原点的偏离，结合上图我们可知道：

- b > 0 则原点在最优超平面的正面；
- b < 0 则原点在最优超平面的负面；
- b = 0 则原点就在最优超平面上。

找到的这个最优超平面的参数w_0和b_0，于是在样本向量集x_i中，有一对(w_0, b_0)一定满足（因为b_0是常数，它只是决定了决策曲面相对原点的偏离）：

$$w_0^T x_i + b_0 \geq 1 \quad 当\ d_i = +1$$
$$w_0^T x_i + b_0 \leq -1 \quad 当\ d_i = -1$$

满足上式的点就是(x_i, d_i)则为支持向量，这些点距离决策曲面也就时超平面最近，时最难区分的点。于是根据点到超平面的距离公式：

$$r = \frac{g(x)}{\|w\|}$$

在超平面的正面和负面我们有任一支持向量(x_p)满足代数距离：

$$r = \frac{g(x_p)}{\|w\|} = \begin{cases} \frac{g(x_p^+)}{\|w\|} \\ \frac{g(x_p^-)}{\|w\|} \end{cases} = \begin{cases} \frac{1}{\|w\|} \\ \frac{-1}{\|w\|} \end{cases}$$

如果让ρ表示两个分离边缘的最优值，则根据上式有：

$$\rho = 2r = \frac{2}{\|w\|}$$

所以我们可以看出，如果要使得ρ最大，则就必须使得$\|w\|$最小，也就可以总结为：

最大化两个类之间的分离边缘等价于最小化权值向量 w 的欧几里得范数。

1.二次最优化

回头看我们前面提到的，给定一个训练集，我们的需求就是尝试找到一个决策边界使得几何间隔最大，回归到问题的本质那就是我们如何找到这个最大的几何间隔？ 要想要最大化间隔（margin），正如上面提到的：

最大化两个类之间的分离边缘等价于最小化权值向量 w 的欧几里得范数。

即：

$$\begin{aligned} &max_{\gamma,\omega,b}\ \check{\gamma} \\ &s.t.\ y^{(i)}\left(w^T x^{(i)} + b\right) \geq \check{\gamma}\ , i = 1, \dots, m \\ &\|w\| = 1 \end{aligned}$$

其中：$\check{\gamma}$也就是前面提到的函数间隔：$\check{\gamma}^{(i)} = y^{(i)}(w^T x + b)$，回顾几何间隔：$\gamma^{(i)} = \frac{(w^T x + b)}{\|w\|}$，约束条件$\|w\|$就是让函数间隔等于几何间隔。

或是将优化的问题转化为以下式子：

$$\begin{aligned} &max_{\gamma,\omega,b}\ \frac{\check{\gamma}}{\|w\|} \\ &s.t.\ y^{(i)}\left(w^T x^{(i)} + b\right) \geq \check{\gamma}\ , i = 1, \dots, m \end{aligned}$$

其中$\gamma = \frac{\check{\gamma}}{\|w\|}$，就是将函数间隔和几何间隔联系起来。

我们发现以上两个式子都可以表示最大化间隔的优化问题，但是我们同时也发现无论上面哪个式子都是**非凸**的，并没有现成的可用的软件来解决这两种形式的优化问题。

于是一个行之有效的优化问题的形式被提出来，注意它是一个凸函数形式，如下：

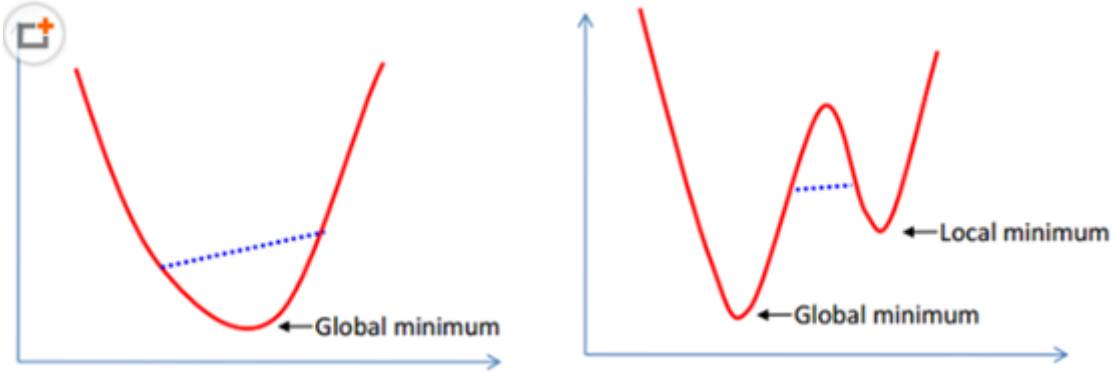
$$min_{\gamma,\omega,b}\ \frac{1}{2}\ \|w\|^2$$
$$s.t.\ y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1, i = 1, \ldots, m$$

以上的优化问题包含了一个凸二次优化对象并且线性可分，概括来说就是需找**最优超平面的二次最优化**，这个优化的问题可以用商业的凸二次规划代码来解。

凸函数：

在凸集中任取两个点连成一条直线，这条直线上的点仍然在这个集合内部，左边

凸函数局部最优就是全局最优，而右边的非凸函数的局部最优就不是全局最优了。



下面要具体介绍的拉格朗日对偶性，它可以引导我们到优化问题的对偶形式，因为对偶形式在**高维空间**有效的运用核（函数）来得到最优间隔分类器的方法中扮演了非常重要的角色。对偶形式让我们得到一个有效的算法来解决上述的优化问题并且相较通用的二次规划商业软件更好。

优化问题的对偶形式的方法简单来说就是通过 **Lagrange Duality** 变换到**对偶变量 (dual variable)**的优化问题之后，应用拉格朗日对偶性，通过求解对偶问题得到最优解，这就是线性可分条件下支持向量机的对偶算法，这样做的优点在于：

- 一是原问题的对偶问题往往更容易求解
- 二者可以自然的引入核函数，进而推广到非线性分类问题。

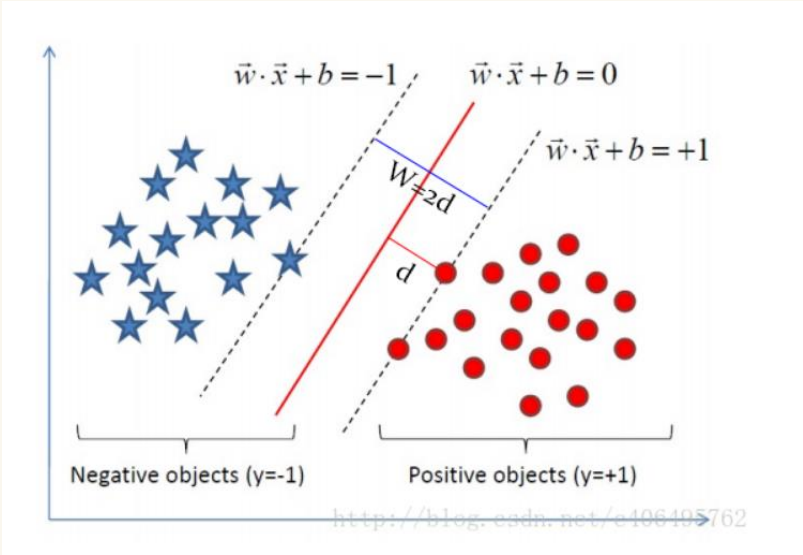
SVM 简介

SVM：Support Vector Machine，支持向量机，是一种分类算法。
同 Logistic 分类方法目的一样，SVM 试图想寻找分割线或面，将平面或空间里的样本点一分为二， 不过方法上有所不同：
1.Logistic 分割线可以是曲线曲面 （多项式函数），SVM 分割线/面只能是线型的。
2. Logistic 算法利用概率模型的最大似然公式求得风险函数，利用标准的梯度下降优化算法求个参数 Θ。
3. SVM 算法通过需要支持向量求得风险函数，利用带有约束条件的梯度下降优化算法求个参数 Θ。过程比较复杂，涉及到拉格朗日函数，kkt 条件， 拉格朗日对偶，SMO 优化算法。

4. SVM 可处理线型不可分的问题吗？ 答案是可以的，但需要将样本特征映射到一个高维空间（2 维（x1,x2）到 5 维($x_1,x_2,x_1x_2,x_1^2,x_2^2$)）然后再寻找向量支持的分割平面，在最后的章节里介绍。

1. 建模线性可分

1. 1. 寻找支持向量模型



目标的中心分割线函数为 $\omega_1X_1 + \omega_2X_2 + b= 0$
两条穿过支持向量的直线形成了间隔边界。两条边界直线的方程为 $\omega_1X_1 + \omega_2X_2 + b= -1$ 和 $\omega_1X_1 + \omega_2X_2 + b= 1$
边界与中心分割线的距离称为边界距离=d.
所有样本向量（或样本点）与中心分割线的距离的>=d. （约束条件）
SVM 算法就是在以上约束条件下，通过优化求得边界 （或称超平面，”决策平面”。）参数（w,b）使边界距离 d 最大 。

1.2. 代价函数与风险函数

$$|\omega^T x_i + \gamma| = 1 \quad \forall \text{支持向量上的样本点} x_i$$

$$|\omega^T x_i + \gamma| > 1 \quad \forall \text{非支持向量上的样本点} x_i$$

即：

$$\begin{cases} \boldsymbol{\omega}^T x_i + \gamma \geq 1 & \forall y_i = 1 \\ \boldsymbol{\omega}^T x_i + \gamma \leq -1 & \forall y_i = -1 \end{cases}$$

或一般形式：

$$y_i\left(\boldsymbol{\omega}^T x_i + \gamma\right) \geq 1 \quad \forall x_i$$

对于任意样本向量 ， 到中心分割线的距离：

$$d^* = \frac{|\boldsymbol{\omega}^T \boldsymbol{x} + \gamma|}{||\boldsymbol{\omega}||}$$

对于任意支持向量 ：

$$\mathrm{d} = \frac{1}{||\boldsymbol{\omega}||}$$

在样本点中找到支持向量（2 个以上）， 使得边界距离的最大， 并满足以上约束条件。

最大化 d, 相当于最小化 w .

所以， 代价函数为 ：

$$\min \frac{1}{2}||\omega||^2$$
$$\text{s.t. } y_i\left(\boldsymbol{\omega}^T x_i + b\right) \geq 1, i = 1,2,\ldots,n$$

s.t. 是 subject to 的缩写， 代表约束条件。

1.3. KKT 条件 和拉格朗日算子法

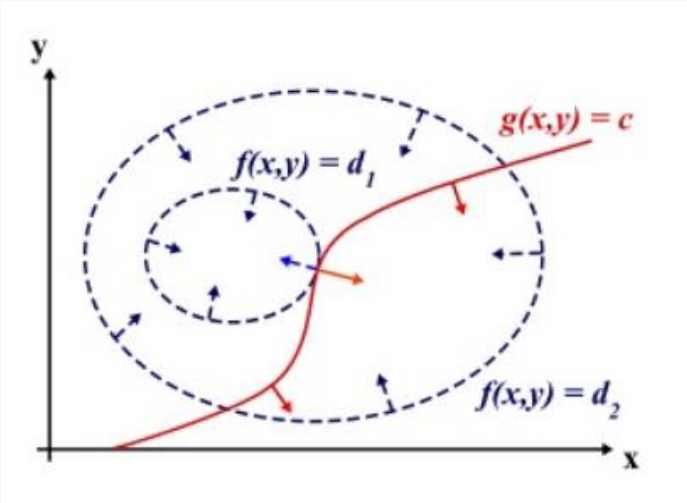
通常带有约束条件的优化问题， 需要通过加入拉格朗日算子生成拉格朗日函数去掉约束条件，从而转化为对新的拉格朗日函数的优化问题。

带有一个等式约束条件的拉格朗日算子方法可形象的用如下的等高线图来表示。

最小化 d=f(x,y) 的过程就是 d 从一个起点沿着在对于（x,y)的梯度向项点下降 （下降指的是梯度越来越趋于 0）的过程 。

任何于 f(x,y)与 g(x,y) 相交的 d 点（如 d1, d2) 都满足约束条件。

但只有 f(x,y)与 g(x,y)相切的 d (d1)才是 f(x,y)的最小值，此时的 x,y 是 d 的最优解： f'(x,y) +g'(x,y) = 0， f(x,y)和 g(x,y) 复合函数（拉格朗日函数）对 x,y 的导数为 0.



但是当原始优化问题有不等式约束条件是， 需要引入 KKT（全称是 Karush-Kuhn-Tucker） 条件。

原始有约束条件的优化问题通常是一下形式：

arg min d=f(x,y), x,y 是要优化的参数，不是样本特征和标签。

s.t. g(x,y) <= 0;

h(x,y) = 0;

拉格朗日算子方法拉格朗日函数通常是以下形式：

L(x, y，a, b,)= f(x,y) + a*g(x,y)+b*h(x, y)

第一步 ， 通过优化拉格朗日算子 a,b 使 L(x, y，a, b,)最大化得到 Θ(x,y，a*,b*) 。 得到的 Θ(x,y，a*,b*)实际是一个带有等式约束条件的优化问题。

Θ(x,y，a*,b*) = argmax L(x,y，a, b)

第二步 ， 通过优化原始目标参数（x,y）使 Θ(x,y，a*,b*) 达到最小，此时得到的 x,y 就是能使原始优化问题 f(x,y)在满足约束条件下能达到的最小值。过程如上图形象所示。

argmin Θ(x,y) = argmin argmax L(x,y,a, b)

KKT 条件为是以上过程得到最优解的必要条件。

条件一：经过拉格朗日函数 L(x,y,a, b)对 x,y 求导为零：

这个条件意味着 a,b 一旦确定，拉格朗日函数 L(x,y,a, b,)函数一定是一个凸函数， 函数值可以沿着(x,y)梯度下降为 0， 从而使等式约束和原始目标函数相切。这个是第二步优化的必要条件。
条件二：h(x，y) = 0；
条件三：α*g(x,y) = 0；

这两个条件是能保证不等式约束后优化为一个等式约束。比如对于某些样本 g(x)<0 时，相应的 a 为 0，使 α*g(x,y) = 0：当 对于某些样本 g(x)=0，相应的 a 可以是取值范围内的任意值，是可以优化的值。

总之这两个条件使 α*g(x,y) +bh(x，y) =0，从而能保证 L(a, b, x, y)，沿着 a,b 的梯度达到最大值，从而使 L(x, y,a, b)变成一个只有等式约束的问题。这个是第一步优化的必要条件。

1.4. 拉格朗日变换

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) - 1\right)$$

对每一个样本（x_i,y_i）通过引入拉格朗日算子向量 a_i（>0），从得到拉格朗日函数 L(w, b, a)， (注意 a 的分量个数是训练样本的个数， 不是样本特征的维度), x, y 为样本值， 通过调整 a 算子使函数满足 KKT 第三条件.
条件二：h(x) = 0 是满足的， 因为原始有约束条件的优化问题并没有此等式约束条件，只有不等式约束 所以 h(x) = 0 。
条件一 ： 需要 L(w, b, a)对 w 和 b, 分别求偏导数使偏导数方程为 0， 如上节所述，这是第二步优化的必要条件。当 a 一旦确定，L(w, b, a)对 w 和 b 的偏导都可以为 0 如下所示， 所以条件一，满足。

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = 0 \Rightarrow \boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0$$

条件三： $g_i(w,b,x_i, y_i) = -a_i (y_i (w^T x_i + b) -1) = -a_i (y_i u_i -1)$ ，
1.对于边界外的点, $y_i u_i -1 > 0$, 另 $a^i=0$, 从而是 $g_i(w,b,x_i, y_i) =0$， 满足条件。
2. 对于边界内的点是异常点， 应排除在优化及约束的范围内， $y_i u_i -1 < 0$， 另 $a_i=C$ （C 常数是一个很大的正数），从而 $g_i(w,b,x_i, y_i)$成为一个很大的正数， 从而不会影响 argmax L(w, b, a)的结果。满足条件。
3. 支持向量(间隔边界上样本点）$y_i u_i -1 =0$ ， $g_i(w,b,x_i, y_i)=0$. 支持向量是用来 argmin Θ(w) 的目标，而 w 依赖 a， 此时 a 的取值范围为 $0<a_i < C$. 满足条件 。
条件三可表示如下，规定 a_i 的取值从而满足条件。

$$\begin{aligned} \alpha_i = 0 &\Leftrightarrow y_i u_i \geq 1 \\ 0 < \alpha_i < C &\Leftrightarrow y_i u_i = 1 \\ \alpha_i = C &\Leftrightarrow y_i u_i \leq 1 \end{aligned}$$

1.5. 拉格朗日对偶问题

拉格朗日问题（如下）， 是一个先通过优化拉格朗日算子 a 使 L(w, b, a)最大, 在优化 w,b 使 Θ（w,b)最小的问题。

$$\min_{w,b} \theta(w) = \min_{w,b} \max_{\alpha_i \geq 0} \mathcal{L}(w,b,\alpha) = p^*$$

算子 a 分量数目巨大，如果首先对他优化，是个十分复杂到难以求解的问题。需要对问题再进行一次转换，即使用一个数学技巧：拉格朗日对偶：

$$\max_{\alpha_i \geq 0} \min_{w,b} \mathcal{L}(w,b,\alpha) = d^*$$

KKT 条件也是强对偶的比较条件，及当 KKT 条件满足时， $p^*=d^*$.

$$\max_{\alpha_i \geq 0} \min_{w,b} \mathcal{L}(w,b,\alpha) = d^*$$

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) - 1\right)$$

第一步求 L(w,b,α)关于 w 和 b 的最小值，我们分别对 w 和 b 偏导数，令其等于 0，即满足条件 1 ：

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = 0 \Rightarrow \boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0$$

将上述结果带回 L(w,b,α)得到：

$$\begin{aligned}
\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) &= \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) - 1\right] \\
&= \frac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} - \boldsymbol{w}^{\mathrm{T}}\sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i - b\sum_{i=1}^{n}\alpha_i y_i + \sum_{i=1}^{n}\alpha_i \\
&= \frac{1}{2}\boldsymbol{w}^{\mathrm{T}}\sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i - \boldsymbol{w}^{\mathrm{T}}\sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i - b\cdot 0 + \sum_{i=1}^{n}\alpha_i \\
&= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\left(\sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i\right)^{\mathrm{T}}\sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i \\
&= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j
\end{aligned}$$

从上面的最后一个式子，我们可以看出，此时的 L(w,b,α)函数只含有一个变量，即 αi。

第二步：求关于 a 最大值，从上面的式子得到

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j$$

$$s.t.\ \alpha_i \geq 0, i = 1, 2, \cdots, n$$

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

通过优化算法（满足条件三中关于 a 的取值范围）能得到 α，再根据 α，我们就可以求解出 w 和 b，进而求得我们最初的目的：找到超平面，即"决策平面"。这个优化算法叫做 SMO :表示序列最小化(Sequential Minimal Optimizaion)

满足条件三中关于 a 的取值范围，就要引入松弛变量(slack variable)C，来允许有些数据点可以处于超平面的错误的一侧。这样我们的优化目标就能保持仍然不变，但是此时我们的约束条件有所改变：

$$s.t.\quad C \geq \alpha_i \geq 0,\ \ i = 1,2,\cdots,n$$

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

1.6. SMO 算法

忽略 SMO 算法的内部原理，让我们列出下算法的具体实现步骤：

步骤 1：计算误差：

$$E_i = f(\boldsymbol{x_i}) - \mathrm{y}_i = \sum_{j=1}^{n} \alpha_j y_j \boldsymbol{x_i}^{\boldsymbol{T}} \boldsymbol{x_j} + b - \mathrm{y}_i$$

步骤 2：计算上下界 L 和 H：

$$\begin{cases} L = \max\left(0, {\alpha_j}^{old} - {\alpha_i}^{old}\right), H = \min\left(C, C + {\alpha_j}^{old} - {\alpha_i}^{old}\right) & if\ \mathrm{y}_i \neq \mathrm{y}_j \\ L = \max\left(0, {\alpha_j}^{old} + {\alpha_i}^{old} - C\right), H = \min\left(C, {\alpha_j}^{old} + {\alpha_i}^{old}\right) & if\ \mathrm{y}_i \neq \mathrm{y}_j \end{cases}$$

步骤 3：计算 η：

$$\eta = \boldsymbol{x_i}^{\boldsymbol{T}}\boldsymbol{x_i} + \boldsymbol{x_j}^{\boldsymbol{T}}\boldsymbol{x_j} - \boldsymbol{2x_i}^{\boldsymbol{T}}\boldsymbol{x_j}$$

步骤 4：更新 aj：

$$\alpha_j^{new} = \alpha_j^{old} + \frac{y_j\left(E_i - E_j\right)}{\eta}$$

步骤 5：根据取值范围修剪 aj：

$$\alpha^{new,clipped} = \begin{cases} H & if\ \alpha_2^{new} \geq H \\ \alpha_2^{new} & if\ L \leq \alpha_2^{new} \leq H \\ L & if\ \alpha_2^{new} \leq L \end{cases}$$

步骤 6：更新 ai：

$$\alpha_i^{new} = \alpha_i^{old} + y_i y_j(\alpha_j^{old} - \alpha_j^{new,clipped})$$

步骤 7：更新 b1 和 b2：

$$\begin{aligned} b_1^{new} &= b^{old} - E_i - y_i(\alpha_i{}^{new} - \alpha_i{}^{old})\boldsymbol{x_i}^{\boldsymbol{T}}\boldsymbol{x_i} - y_j(\alpha_j{}^{new} \\ &\quad - \alpha_j{}^{old})\boldsymbol{x_j}^{\boldsymbol{T}}\boldsymbol{x_i} \\ b_2^{new} &= b^{old} - E_j - y_i(\alpha_i{}^{new} - \alpha_i{}^{old})\boldsymbol{x_i}^{\boldsymbol{T}}\boldsymbol{x_j} - y_j(\alpha_j{}^{new} \\ &\quad - \alpha_j{}^{old})\boldsymbol{x_j}^{\boldsymbol{T}}\boldsymbol{x_j} \end{aligned}$$

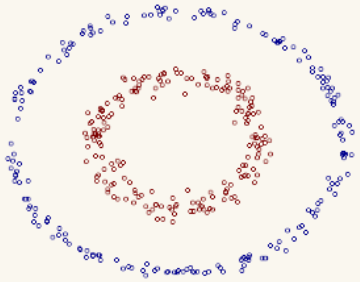
步骤 8：根据 b1 和 b2 更新 b：

$$b = \begin{cases} b_1 & 0 < \alpha_1^{new} < C \\ b_2 & 0 < \alpha_2^{new} < C \\ \dfrac{b_1 + b_2}{2} & otherwise \end{cases}$$

2. 线性不可分模型

2.1. 线性不可分模型

比如下图是一个线性不可分的问题



中心决策曲线的方程，仿佛可以写作这样的形式：

$$\omega_1 X_1 + \omega_2 X_1^2 + \omega_3 X_2 + \omega_4 X_2^2 + \omega_5 X_1 X_2 + b = 0$$

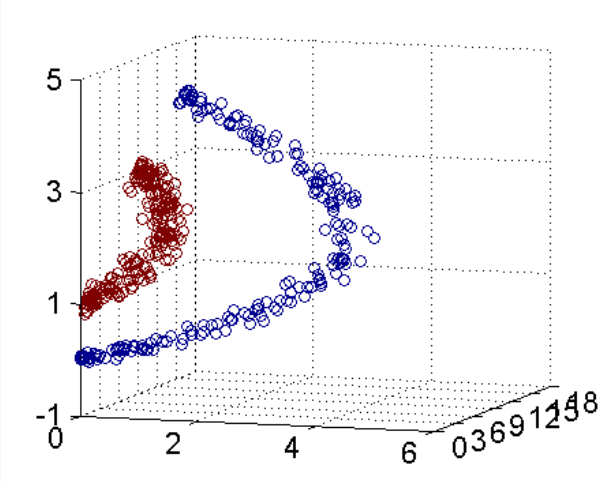
注意上面的形式，如果我们构造另外一个五维的空间，其中五个坐标的值分别为$Z_1 = X_1$，$Z_2 = X_1^2$，$Z_3 = X_2$，$Z_4 = X_2^2$，$Z_5 = X_1 X_2$，那么显然，上面的方程在新的坐标系下可以写作：

$$\sum_{i=1}^{5} \omega_i Z_i + b = 0$$

关于新的坐标Z，这正是一个 决策平面（超平面） 的方程！也就是说，如果我们做一个映射$\phi : R_2 \to R_5$，将 X按照上面的规则映射为Z，那么在新的空间中原来的数据将变成线性可分的，从而使用之前我们推导的线性分类算法就可以进行处理了。

这正是 Kernel 方法处理非线性问题的基本思想。

我只需要把它映射到$Z_1 = X_1^2$，$Z_2 = X_2^2$，$Z_3 = X_2$这样一个三维空间中即可，下图即是映射之后的结果，从坐标轴的某一个角度就可以很明显地看出，数据是可以通过一个平面来分开的 。



映射函数 ø（x）相当于把原来的分类函数

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i \langle x_i, x \rangle + b$$

映射成：

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i \langle \phi(x_i), \phi(x) \rangle + b$$

然后就可以用拉格朗日对偶问题的优化步骤及 SMO 算法求得关于 升维的关于 ø（x）的超平面的参数 w, b，使得：

$|w^T ø(x_i) + b| = 1$ 对于任意支持向量样本点 。w 的维度同 ø（x）的维度。
$|w^T ø(xi)+ b| = 1$ 对于任意非支持向量样本点。w 的维度同 ø（x）的维度。

具体计算中如果把简单的从原始空间向多维多项式空间映射，复杂度是非常高的。比如原始空间是三维，需要映射到 19 维的新空间。

简单的方法是利用整个计算过程中 $ø(x_i)$总会另为一个 $ø(x_j)$求向量内积，从而引出核函数 $K(x_i,x)$.

$$K(x_i, x) = \langle \emptyset(x_i), \emptyset(x_i) \rangle$$

核函数能够利用一些低维的计算等价高位的向量内积。

2.2. 几个核函数

- 线性核函数 ， 就是不升高维度， 只能解决线性可分的问题。

  $\kappa(x,x_i)=x \cdot x_i$
- 多项式核函数 ,能够将低维的输入空间映射到高纬的特征空间，但是多项式核函数的参数多，当多项式的阶数比较高的时候，核矩阵的元素值将趋于无穷大或者无穷小，计算复杂度会大到无法计算。

  $\kappa(x,x_i)=((x \cdot x_i)+1)^d$
- 高斯（RBF）核函数

$$\kappa(x, x_i) = exp(-\frac{||x - x_i||^2}{\delta^2})$$

高斯径向基函数是一种局部性强的核函数，其可以将一个样本映射到一个更高维的空间内，该核函数是应用最广的一个，无论大样本还是小样本都有比较好的性能，而且其相对于多项式核函数参数要少，因此大多数情况下在不知道用什么核函数的时候，优先使用高斯核函数。

- tanh 核函数

$$\kappa(x, x_i) = tanh(\eta < x, x_i > +\theta)$$

采用 sigmoid 核函数，支持向量机实现的就是一种多层神经网络。
因此，在选用核函数的时候，如果我们对我们的数据有一定的先验知识，就利用先验来选择符合数据分布的核函数；如果不知道的话，通常使用交叉验证的方法，来试用

不同的核函数，误差最下的即为效果最好的核函数，或者也可以将多个核函数结合起来，形成混合核函数。在吴恩达（Andrew Ng，斯坦福教授）的课上，也曾经给出过一系列的选择核函数的方法：

3. 模型选择

如果特征的数量大到和样本数量差不多，则选用 LR 或者线性核的 SVM；
如果特征的数量小，样本的数量正常，则选用 SVM+高斯核函数；
如果特征的数量小，而样本的数量很大，则需要手工添加一些特征从而变成第一种情况。

参考

https://blog.csdn.net/c406495762/article/details/78072313

https://blog.csdn.net/v_july_v/article/details/7624837

https://blog.csdn.net/xianlingmao/article/details/7919597

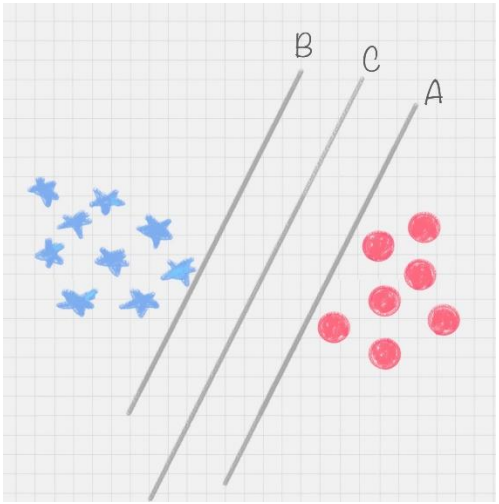
https://blog.csdn.net/batuwuhanpei/article/details/52354822

机器学习经典算法之 SVM

SVM 的英文叫 Support Vector Machine，中文名为支持向量机。它是常见的一种分类方法，在机器学习中，SVM 是有监督的学习模型。
什么是有监督的学习模型呢？它指的是我们需要事先对数据打上分类标签，这样机器就知道这个数据属于哪个分类。同样无监督学习，就是数据没有被打上分类标签，这可能是因为我们不具备先验的知识，或者打标签的成本很高。所以我们需要机器代我们部分完成这个工作，比如将数据进行聚类，方便后续人工对每个类进行分析。SVM 作为有监督的学习模型，通常可以帮我们模式识别、分类以及回归分析。
/*请尊重作者劳动成果，转载请标明原文链接：*/
/* https://www.cnblogs.com/jpcflyer/p/11082443.html * /

一、SVM 的工作原理

用 SVM 计算的过程就是帮我们找到一个超平面，能够将样本区分的过程，这个超平面就是我们的 SVM 分类器。
比如下图所示的直线 A、直线 B 和直线 C，究竟哪种才是更好的划分呢？



很明显图中的直线 B 更靠近蓝色球，但是在真实环境下，球再多一些的话，蓝色球可能就被划分到了直线 B 的右侧，被认为是红色球。同样直线 A 更靠近红色球，在真实环境下，如果红色球再多一些，也可能会被误认为是蓝色球。所以相比于直线 A 和直线 B，直线 C 的划分更优，因为它的鲁棒性更强。
那怎样才能寻找到直线 C 这个更优的答案呢？这里，我们引入一个 SVM 特有的概念： **分类间隔。**
实际上，我们的分类环境不是在二维平面中的，而是在多维空间中，这样直线 C 就变成了决策面 C。
在保证决策面不变，且分类不产生错误的情况下，我们可以移动决策面 C，直到产生两个极限的位置：如图中的决策面 A 和决策面 B。极限的位置是指，如果越过了这个位置，就会产生分类错误。这样的话，两个极限位置 A 和 B 之间的分界线 C 就是最优决策面。极限位置到最优决策面 C 之间的距离，就是“分类间隔”，英文叫做 margin。
如果我们转动这个最优决策面，你会发现可能存在多个最优决策面，它们都能把数据集正确分开，这些最优决策面的分类间隔可能是不同的，而那个拥有“最大间隔”（max margin）的决策面就是 SVM 要找的最优解。

点到超平面的距离公式

在上面这个例子中，如果我们把红蓝两种颜色的球放到一个三维空间里，你发现决策面就变成了一个平面。这里我们可以用线性函数来表示，如果在一维空间里就表示一个点，在二维空间里表示一条直线，在三维空间中代表一个平面，当然空间维数还可以更多，这样我们给这个线性函数起个名称叫做“超平面”。超平面的数学表达可以写成：

$$g(x)=\omega^{T}x+b,\text{其中}\omega,x\in\mathrm{R}^{\mathrm{n}}$$

在这个公式里，w、x 是 n 维空间里的向量，其中 x 是函数变量；w 是法向量。法向量这里指的是垂直于平面的直线所表示的向量，它决定了超平面的方向。
SVM 就是帮我们找到一个超平面 ，这个超平面能将不同的样本划分开，同时使得样本集中的点到这个分类超平面的最小距离（即分类间隔）最大化。

在这个过程中， 支持向量 就是离 分类超平面 最近的样本点，实际上如果确定了支持向量也就确定了这个超平面。所以支持向量决定了分类间隔到底是多少，而在最大间隔以外的样本点，其实对分类都没有意义。
所以说， SVM 就是求解最大分类间隔的过程，我们还需要对分类间隔的大小进行定义。
首先，我们定义某类样本集到超平面的距离是这个样本集合内的样本到超平面的最短距离。我们用 di 代表点 xi 到超平面 wxi+b=0 的欧氏距离。因此我们要求 di 的最小值，用它来代表这个样本到超平面的最短距离。di 可以用公式计算得出：

$$d_i = \frac{\left|\omega x_i + b\right|}{\left\|\omega\right\|}$$

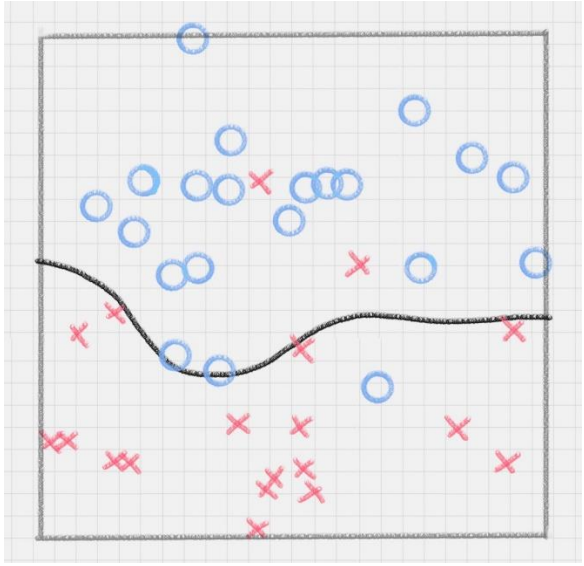
其中||w||为超平面的范数，di 的公式可以用解析几何知识进行推导，这里不做解释。

最大间隔的优化模型

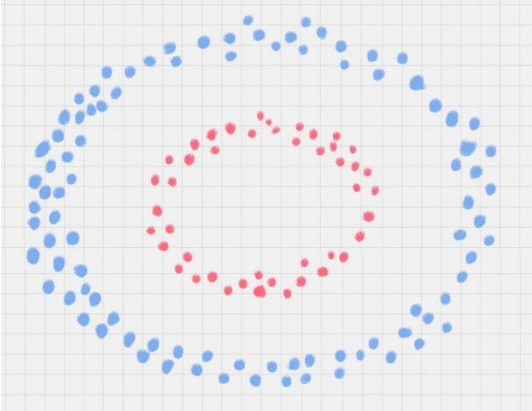
我们的目标就是找出所有分类间隔中最大的那个值对应的超平面。在数学上，这是一个凸优化问题（凸优化就是关于求凸集中的凸函数最小化的问题，这里不具体展开）。通过凸优化问题，最后可以求出最优的 w 和 b，也就是我们想要找的最优超平面。中间求解的过程会用到拉格朗日乘子，和 KKT（Karush-Kuhn-Tucker）条件。数学公式比较多，这里不进行展开。

硬间隔、软间隔和非线性 SVM

假如数据是完全的线性可分的，那么学习到的模型可以称为硬间隔支持向量机。 换个说法，硬间隔指的就是完全分类准确，不能存在分类错误的情况。软间隔，就是允许一定量的样本分类错误。
我们知道，实际工作中的数据没有那么“干净”，或多或少都会存在一些噪点。所以线性可分是个理想情况。这时，我们需要使用到软间隔 SVM（近似线性可分），比如下面这种情况：



另外还存在一种情况，就是非线性支持向量机。
比如下面的样本集就是个非线性的数据。图中的两类数据，分别分布为两个圆圈的形状。那么这种情况下，不论是多高级的分类器，只要映射函数是线性的，就没法处理，SVM 也处理不了。这时，我们需要引入一个新的概念： 核函数。它可以将样本从原始空间映射到一个更高维的特质空间中，使得样本在新的空间中线性可分 。这样我们就可以使用原来的推导来进行计算，只是所有的推导是在新的空间，而不是在原来的空间中进行。



所以在非线性 SVM 中，核函数的选择就是影响 SVM 最大的变量。最常用的核函数有线性核、多项式核、高斯核、拉普拉斯核、sigmoid 核，或者是这些核函数的组合。这些函数的区别在于映射方式的不同。通过这些核函数，我们就可以把样本空间投射到新的高维空间中。
当然软间隔和核函数的提出，都是为了方便我们对上面超平面公式中的 w* 和 b* 进行求解，从而得到最大分类间隔的超平面。

二、 用 SVM 如何解决多分类问题

SVM 本身是一个二值分类器，最初是为二分类问题设计的，也就是回答 Yes 或者是 No。而实际上我们要解决的问题，可能是多分类的情况，比如对文本进行分类，或者对图像进行识别。
针对这种情况，我们可以将多个二分类器组合起来形成一个多分类器，常见的方法有“一对多法”和“一对一法”两种。

1. 一对多法

假设我们要把物体分成 A、B、C、D 四种分类，那么我们可以先把其中的一类作为分类 1，其他类统一归为分类 2。这样我们可以构造 4 种 SVM，分别为以下的情况：

（1）样本 A 作为正集，B，C，D 作为负集；
（2）样本 B 作为正集，A，C，D 作为负集；
（3）样本 C 作为正集，A，B，D 作为负集；
（4）样本 D 作为正集，A，B，C 作为负集。

这种方法，针对 K 个分类，需要训练 K 个分类器，分类速度较快，但训练速度较慢，因为每个分类器都需要对全部样本进行训练，而且负样本数量远大于正样本数量，会造成样本不对称的情况，而且当增加新的分类，比如第 K+1 类时，需要重新对分类器进行构造。

2. 一对一法

一对一法的初衷是想在训练的时候更加灵活。我们可以在任意两类样本之间构造一个 SVM，这样针对 K 类的样本，就会有 C(k,2) 类分类器。

比如我们想要划分 A、B、C 三个类，可以构造 3 个分类器：
（1）分类器 1：A、B；
（2）分类器 2：A、C；
（3）分类器 3：B、C。
当对一个未知样本进行分类时，每一个分类器都会有一个分类结果，即为 1 票，最终得票最多的类别就是整个未知样本的类别。
这样做的好处是，如果新增一类，不需要重新训练所有的 SVM，只需要训练和新增这一类样本的分类器。而且这种方式在训练单个 SVM 模型的时候，训练速度快。
但这种方法的不足在于，分类器的个数与 K 的平方成正比，所以当 K 较大时，训练和测试的时间会比较慢。

三、 如何在 sklearn 中使用 SVM

在 Python 的 sklearn 工具包中有 SVM 算法，首先需要引用工具包：

```
1 from sklearn import svm
```

SVM 既可以做回归，也可以做分类器。
当用 SVM 做回归的时候，我们可以使用 SVR 或 LinearSVR。SVR 的英文是 Support Vector Regression。这篇文章只讲分类，这里只是简单地提一下。
当做分类器的时候，我们使用的是 SVC 或者 LinearSVC。SVC 的英文是 Support Vector Classification。
我简单说一下这两者之前的差别。
从名字上你能看出 LinearSVC 是个线性分类器，用于处理线性可分的数据，只能使用线性核函数。上一节，我讲到 SVM 是通过核函数将样本从原始空间映射到一个更高维的特质空间中，这样就使得样本在新的空间中线性可分。
如果是针对非线性的数据，需要用到 SVC。在 SVC 中，我们既可以使用到线性核函数（进行线性划分），也能使用高维的核函数（进行非线性划分）。
如何创建一个 SVM 分类器呢？
我们首先使用 SVC 的构造函数：model = svm.SVC(kernel= ‘rbf’, C=1.0, gamma= ‘auto’)，这里有三个重要的参数 kernel、C 和 gamma。
kernel 代表核函数的选择，它有四种选择，只不过默认是 rbf，即高斯核函数。
linear：线性核函数
poly：多项式核函数
rbf：高斯核函数（默认）
sigmoid：sigmoid 核函数
这四种函数代表不同的映射方式，你可能会问，在实际工作中，如何选择这 4 种核函数呢？我来给你解释一下：
线性核函数，是在数据线性可分的情况下使用的，运算速度快，效果好。不足在于它不能处理线性不可分的数据。
多项式核函数可以将数据从低维空间映射到高维空间，但参数比较多，计算量大。

高斯核函数同样可以将样本映射到高维空间，但相比于多项式核函数来说所需的参数比较少，通常性能不错，所以是默认使用的核函数。
了解深度学习的同学应该知道 sigmoid 经常用在神经网络的映射中。因此当选用 sigmoid 核函数时，SVM 实现的是多层神经网络。

上面介绍的 4 种核函数，除了第一种线性核函数外，其余 3 种都可以处理线性不可分的数据。
参数 C 代表目标函数的惩罚系数，惩罚系数指的是分错样本时的惩罚程度，默认情况下为 1.0。当 C 越大的时候，分类器的准确性越高，但同样容错率会越低，泛化能力会变差。相反，C 越小，泛化能力越强，但是准确性会降低。
参数 gamma 代表核函数的系数，默认为样本特征数的倒数，即 gamma = 1 / n_features。
在创建 SVM 分类器之后，就可以输入训练集对它进行训练。我们使用 model.fit(train_X,train_y)，传入训练集中的特征值矩阵 train_X 和分类标识 train_y。特征值矩阵就是我们在特征选择后抽取的特征值矩阵（当然你也可以用全部数据作为特征值矩阵）；分类标识就是人工事先针对每个样本标识的分类结果。这样模型会自动进行分类器的训练。我们可以使用 prediction=model.predict(test_X) 来对结果进行预测，传入测试集中的样本特征矩阵 test_X，可以得到测试集的预测分类结果 prediction。
同样我们也可以创建线性 SVM 分类器，使用 model=svm.LinearSVC()。在 LinearSVC 中没有 kernel 这个参数，限制我们只能使用线性核函数。由于 LinearSVC 对线性分类做了优化，对于数据量大的线性可分问题，使用 LinearSVC 的效率要高于 SVC。
如果你不知道数据集是否为线性，可以直接使用 SVC 类创建 SVM 分类器。
在训练和预测中，LinearSVC 和 SVC 一样，都是使用 model.fit(train_X,train_y) 和 model.predict(test_X)。

四、 如何用 SVM 进行乳腺癌检测

在了解了如何创建和使用 SVM 分类器后，我们来看一个实际的项目，数据集来自美国威斯康星州的乳腺癌诊断数据集， 点击这里进行下载。
医疗人员采集了患者乳腺肿块经过细针穿刺（FNA）后的数字化图像，并且对这些数字图像进行了特征提取，这些特征可以描述图像中的细胞核呈现。肿瘤可以分成良性和恶性。部分数据截屏如下所示：

id	diagn	radius_m	texture_s	perimeter	area_mean	smoothnes	compactne	concavity	concave p	symmetry_	fractal_c	radius_se	texture_s	perimeter	area_se	smoothnes
842302	M	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	0.2419	0.07871	1.095	0.9053	8.589	153.4	0.006399
842517	M	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	0.5435	0.7339	3.398	74.08	0.005225
84300903	M	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	0.2069	0.05999	0.7456	0.7869	4.585	94.03	0.00615
84348301	M	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	0.2597	0.09744	0.4956	1.156	3.445	27.23	0.00911
84358402	M	20.29	14.34	135.1	1297	0.1003	0.1328	0.198	0.1043	0.1809	0.05883	0.7572	0.7813	5.438	94.44	0.01149
843786	M	12.45	15.7	82.57	477.1	0.1278	0.17	0.1578	0.08089	0.2087	0.07613	0.3345	0.8902	2.217	27.19	0.00751
844359	M	18.25	19.98	119.6	1040	0.09463	0.109	0.1127	0.074	0.1794	0.05742	0.4467	0.7732	3.18	53.91	0.004314
84458202	M	13.71	20.83	90.2	577.9	0.1189	0.1645	0.09366	0.05985	0.2196	0.07451	0.5835	1.377	3.856	50.96	0.008805
844981	M	13	21.82	87.5	519.8	0.1273	0.1932	0.1859	0.09353	0.235	0.07389	0.3063	1.002	2.406	24.32	0.005731
84501001	M	12.46	24.04	83.97	475.9	0.1186	0.2396	0.2273	0.08543	0.203	0.08243	0.2976	1.599	2.039	23.94	0.007149
845636	M	16.02	23.24	102.7	797.8	0.08206	0.06669	0.03299	0.03323	0.1528	0.05697	0.3795	1.187	2.466	40.51	0.004029
84610002	M	15.78	17.89	103.6	781	0.0971	0.1292	0.09954	0.06606	0.1842	0.06082	0.5058	0.9849	3.564	54.16	0.005771
846226	M	19.17	24.8	132.4	1123	0.0974	0.2458	0.2065	0.1118	0.2397	0.078	0.9555	3.568	11.07	116.2	0.003139
846381	M	15.85	23.95	103.7	782.7	0.08401	0.1002	0.09938	0.05364	0.1847	0.05338	0.4033	1.078	2.903	36.58	0.009769
84667401	M	13.73	22.61	93.6	578.3	0.1131	0.2293	0.2128	0.08025	0.2069	0.07682	0.2121	1.169	2.061	19.21	0.006429
84799002	M	14.54	27.54	96.73	658.8	0.1139	0.1595	0.1639	0.07364	0.2303	0.07077	0.37	1.033	2.879	32.55	0.005607
848406	M	14.68	20.13	94.74	684.5	0.09867	0.072	0.07395	0.05259	0.1586	0.05922	0.4727	1.24	3.195	45.4	0.005718
84862001	M	16.13	20.68	108.1	798.8	0.117	0.2022	0.1722	0.1028	0.2164	0.07356	0.5692	1.073	3.854	54.18	0.007026
849014	M	19.81	22.15	130	1260	0.09831	0.1027	0.1479	0.09498	0.1582	0.05395	0.7582	1.017	5.865	112.4	0.006494
8510426	B	13.54	14.36	87.46	566.3	0.09779	0.08129	0.06664	0.04781	0.1885	0.05766	0.2699	0.7886	2.058	23.56	0.008462
8510653	B	13.08	15.71	85.63	520	0.1075	0.127	0.04568	0.0311	0.1967	0.06811	0.1852	0.7477	1.383	14.67	0.004097
8510824	B	9.504	12.44	60.34	273.9	0.1024	0.06492	0.02956	0.02076	0.1815	0.06905	0.2773	0.9768	1.909	15.7	0.009606
8511133	M	15.34	14.26	102.5	704.4	0.1073	0.2135	0.2077	0.09756	0.2521	0.07032	0.4388	0.7096	3.384	44.91	0.006789
851509	M	21.16	23.04	137.2	1404	0.09428	0.1022	0.1097	0.08632	0.1769	0.05278	0.6917	1.127	4.303	93.99	0.004728

数据表一共包括了 32 个字段，代表的含义如下：

字段	含义
ID	ID标识
diagnosis	M/B（M：恶性，B：良性）
radius_mean	半径（点中心到边缘的距离）平均值
texture_mean	文理（灰度值的标准差）平均值
perimeter_mean	周长 平均值
area_mean	面积 平均值
smoothness_mean	平滑程度（半径内的局部变化）平均值
compactness_mean	紧密度（=周长*周长/面积-1.0）平均值
concavity_mean	凹度（轮廓凹部的严重程度）平均值
concave points_mean	凹缝（轮廓的凹部分）平均值
symmetry_mean	对称性 平均值
fractal_dimension_mean	分形维数（=海岸线近似-1）平均值
radius_se	半径（点中心到边缘的距离）标准差
texture_se	文理（灰度值的标准差）标准差
perimeter_se	周长 标准差
area_se	面积 标准差
smoothness_se	平滑程度（半径内的局部变化）标准差
compactness_se	紧密度（=周长*周长/面积-1.0）标准差
concavity_se	凹度（轮廓凹部的严重程度）标准差
concave points_se	凹缝（轮廓的凹部分）标准差
symmetry_se	对称性标准差
fractal_dimension_se	分形维数（=海岸线近似-1）标准差
radius_worst	半径（点中心到边缘的距离）最大值
texture_worst	文理（灰度值的标准差）最大值
perimeter_worst	周长 最大值
area_worst	面积 最大值
smoothness_worst	平滑程度（半径内的局部变化）最大值
compactness_worst	紧密度（=周长*周长/面积-1.0）最大值
concavity_worst	凹度（轮廓凹部的严重程度）最大值
concave points_worst	凹缝（轮廓的凹部分）最大值
symmetry_worst	对称性 最大值
fractal_dimension_worst	分形维数（=海岸线近似-1）最大值

上面的表格中，mean 代表平均值，se 代表标准差，worst 代表最大值（3 个最大值的平均值）。每张图像都计算了相应的特征，得出了这 30 个特征值（不包括 ID 字段和分类标识结果字段 diagnosis），实际上是 10 个特征值（radius、texture、perimeter、area、smoothness、compactness、concavity、concave points、symmetry 和 fractal_dimension_mean）的 3 个维度，平均、标准差和最大值。这些特征值都保留了 4 位数字。字段中没有缺失的值。在 569 个患者中，一共有 357 个是良性，212 个是恶性。
好了，我们的目标是生成一个乳腺癌诊断的 SVM 分类器，并计算这个分类器的准确率。 首先加载数据并对数据做部分的探索：

```
# 加载数据集，你需要把数据放到目录中
data = pd.read_csv("./data.csv")

# 数据探索
# 因为数据集中列比较多，我们需要把 dataframe 中的列全部显示出来
pd.set_option('display.max_columns', None)
print(data.columns)
print(data.head(5))
print(data.describe())
```

这是部分的运行结果，完整结果你可以自己跑一下。

```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
         id diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0    842302         M        17.99         10.38          122.80     1001.0
1    842517         M        20.57         17.77          132.90     1326.0
2  84300903         M        19.69         21.25          130.00     1203.0
3  84348301         M        11.42         20.38           77.58      386.1
4  84358402         M        20.29         14.34          135.10     1297.0
```

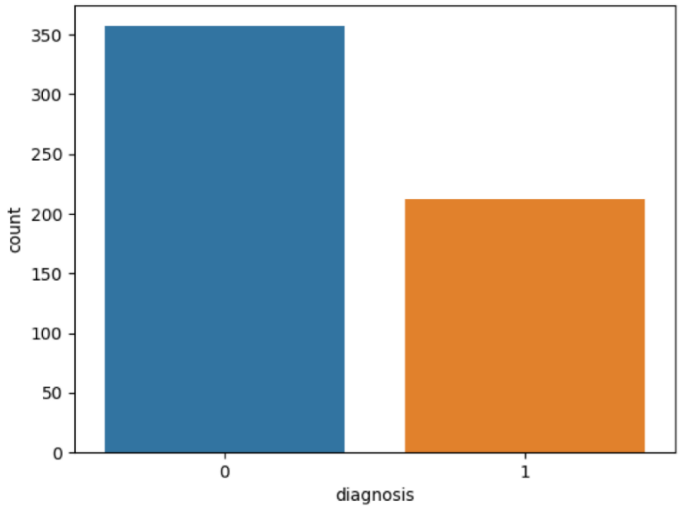
接下来，我们就要对数据进行清洗了。
运行结果中，你能看到 32 个字段里，id 是没有实际含义的，可以去掉。diagnosis 字段的取值为 B 或者 M，我们可以用 0 和 1 来替代。另外其余的 30 个字段，其实可以分成三组字段，下划线后面的 mean、se 和 worst 代表了每组字段不同的度量方式，分别是平均值、标准差和最大值。

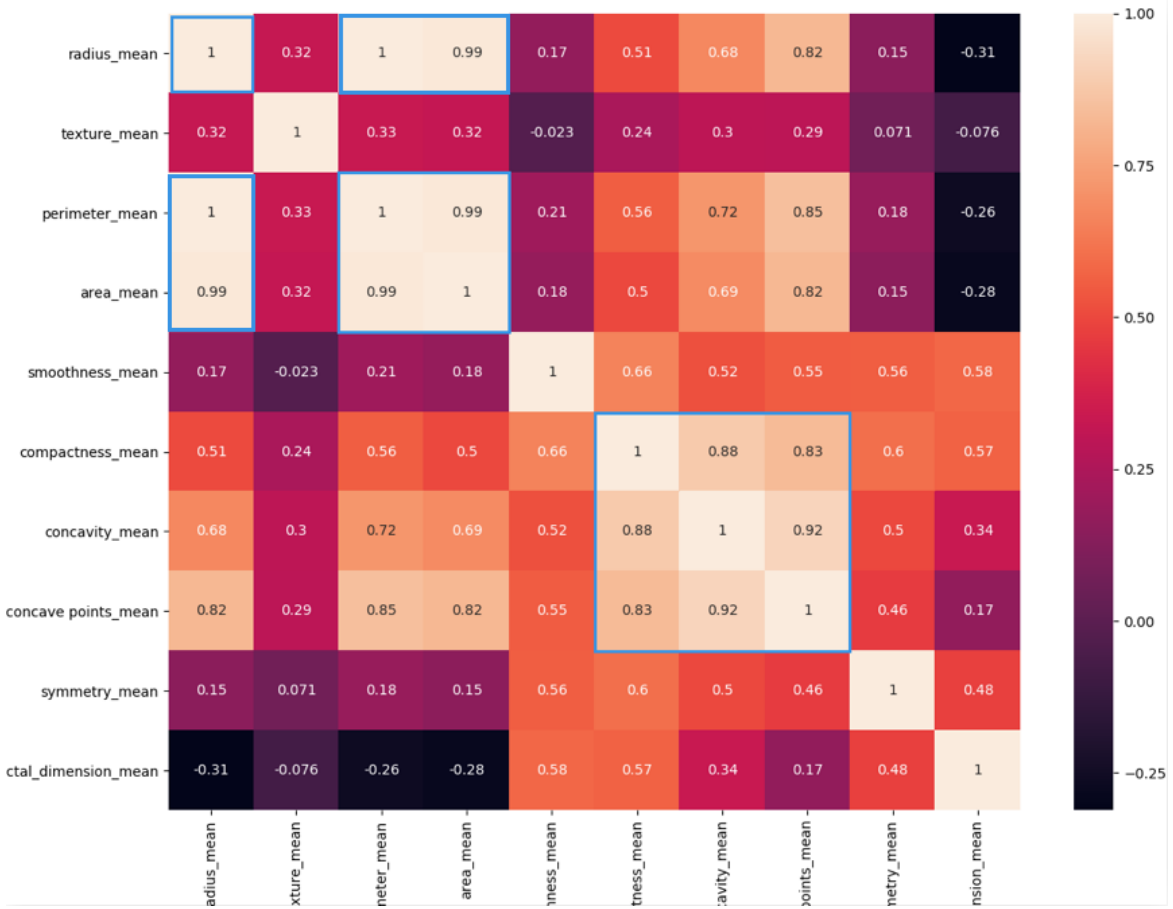
```
# 将特征字段分成 3 组
features_mean= list(data.columns[2:12])
features_se= list(data.columns[12:22])
features_worst=list(data.columns[22:32])
# 数据清洗
# ID 列没有用，删除该列
data.drop("id",axis=1,inplace=True)
# 将 B 良性替换为 0，M 恶性替换为 1
data['diagnosis']=data['diagnosis'].map({'M':1,'B':0})
```

然后我们要做特征字段的筛选，首先需要观察下 features_mean 各变量之间的关系，这里我们可以用 DataFrame 的 corr() 函数，然后用热力图帮我们可视化呈现。同样，我们也会看整体良性、恶性肿瘤的诊断情况。

```
# 将肿瘤诊断结果可视化
sns.countplot(data['diagnosis'],label="Count")
plt.show()
# 用热力图呈现 features_mean 字段之间的相关性
corr = data[features_mean].corr()
plt.figure(figsize=(14,14))
# annot=True 显示每个方格的数据
sns.heatmap(corr, annot=True)
plt.show()
```

这是运行的结果：





热力图中对角线上的为单变量自身的相关系数是 1。颜色越浅代表相关性越大。所以你能看出来 radius_mean、perimeter_mean 和 area_mean 相关性非常大，compactness_mean、concavity_mean、concave_points_mean 这三个字段也是相关的，因此我们可以取其中的一个作为代表。

那么如何进行特征选择呢?
特征选择的目的是降维，用少量的特征代表数据的特性，这样也可以增强分类器的泛化能力，避免数据过拟合。
我们能看到 mean、se 和 worst 这三组特征是对同一组内容的不同度量方式，我们可以保留 mean 这组特征，在特征选择中忽略掉 se 和 worst。同时我们能看到 mean 这组特征中，radius_mean、perimeter_mean、area_mean 这三个属性相关性大，compactness_mean、daconcavity_mean、concave points_mean 这三个属性相关性大。我们分别从这 2 类中选择 1 个属性作为代表，比如 radius_mean 和 compactness_mean。
这样我们就可以把原来的 10 个属性缩减为 6 个属性，代码如下:

```
1 # 特征选择
```

```
features_remain = ['radius_mean','texture_mean', 'smoothness_mean','compactness_mean','symmetry_mean',
'fractal_dimension_mean']
 对特征进行选择之后，我们就可以准备训练集和测试集：
# 抽取 30% 的数据作为测试集，其余作为训练集
train, test = train_test_split(data, test_size = 0.3)# in this our main data is splitted into train and test
# 抽取特征选择的数值作为训练和测试数据
train_X = train[features_remain]
train_y=train['diagnosis']
test_X= test[features_remain]
test_y =test['diagnosis']
 在训练之前，我们需要对数据进行规范化，这样让数据同在同一个量级上，避免因为维度问题造成数据误差：
# 采用 Z-Score 规范化数据，保证每个特征维度的数据均值为 0，方差为 1
ss = StandardScaler()
train_X = ss.fit_transform(train_X)
test_X = ss.transform(test_X)
 最后我们可以让 SVM 做训练和预测了：
# 创建 SVM 分类器
model = svm.SVC()
# 用训练集做训练
model.fit(train_X,train_y)
# 用测试集做预测
prediction=model.predict(test_X)
print('准确率: ', metrics.accuracy_score(prediction,test_y))
```

运行结果：

```
准确率:  0.9181286549707602
```

准确率大于 90%，说明训练结果还不错。

搜索关注微信公众号“程序员姜小白”，获取更新精彩内容哦。