lujiandong1的专栏

机器学习中的内核方法 李政轩的视频笔记

标签： 机器学习 核方法

2015-06-04 08:18 1873人阅读 评论(0) 收藏 举报

分类： 机器学习（32）

目录(?) [+]

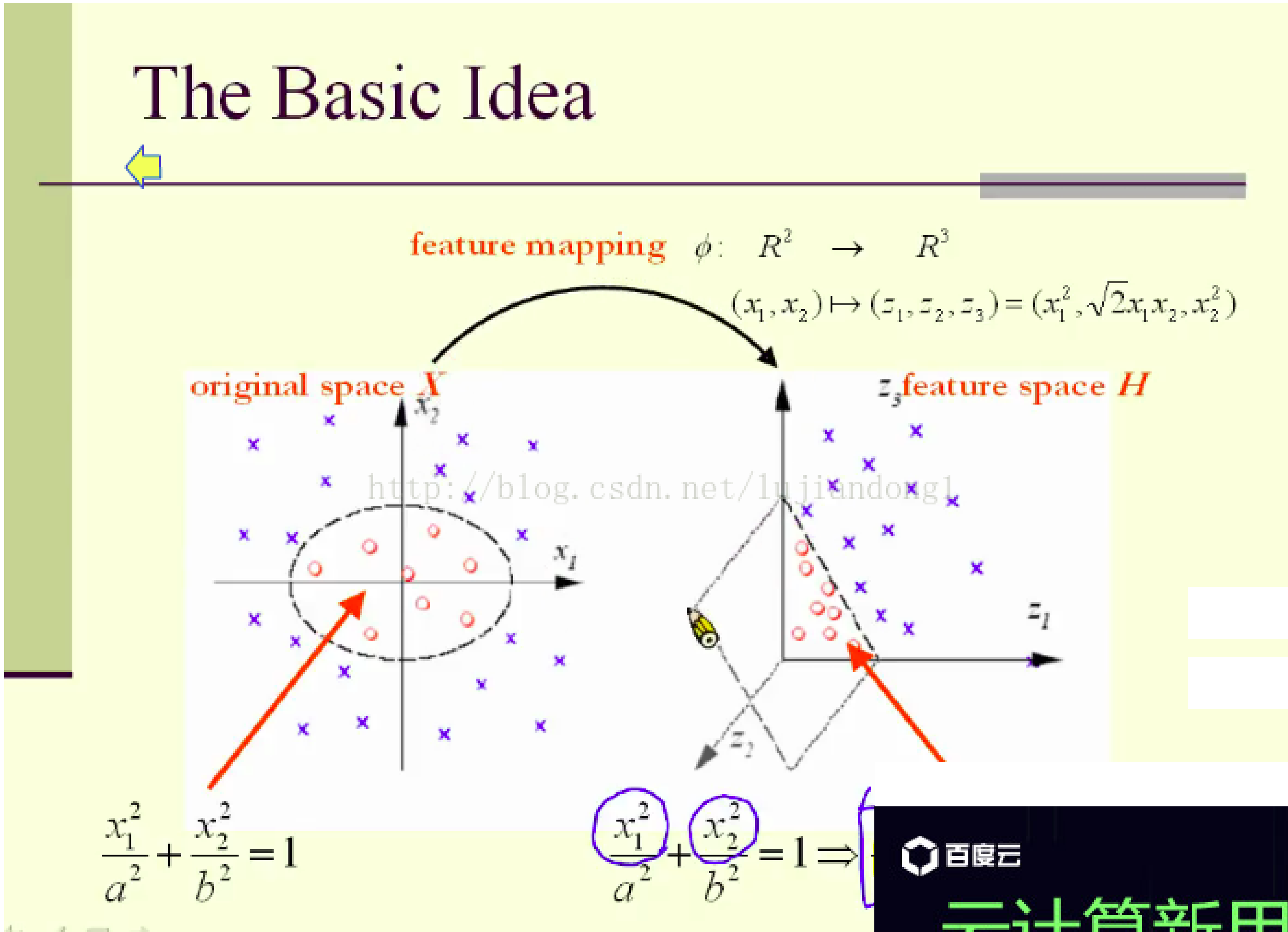
课程链接：http://www.powercam.cc/slide/6552。

Kernel Method 的基本思想：

文章搜索

文章分类

ACM DP问题 (7)
ACM其他文章 (2)
STL (5)
人生感悟 (1)
C++基础 (20)
C# (1)
ACM 贪心算法 (2)
ACM 哈希 (3)
二分法 (3)
STL 空间配置器 (1)
poj Trie树 (1)
poj 调试经验 (2)
POJ KMP (2)
poj 暴力法 (1)
C++调试 (3)
More Effective C++读书笔记 (3)
Effective C++读书笔记 (25)
机器学习 (33)
北邮人论坛 (2)
杂项 (6)
数学 (2)
opencv (3)
操作系统基本概念 (10)
Linux基础学习 (10)
程序优化 (3)
深入理解计算机系统 (5)



Kernel的基本思想是，将低维空间不可分数据映射到高纬度的空间，比如说左

是：

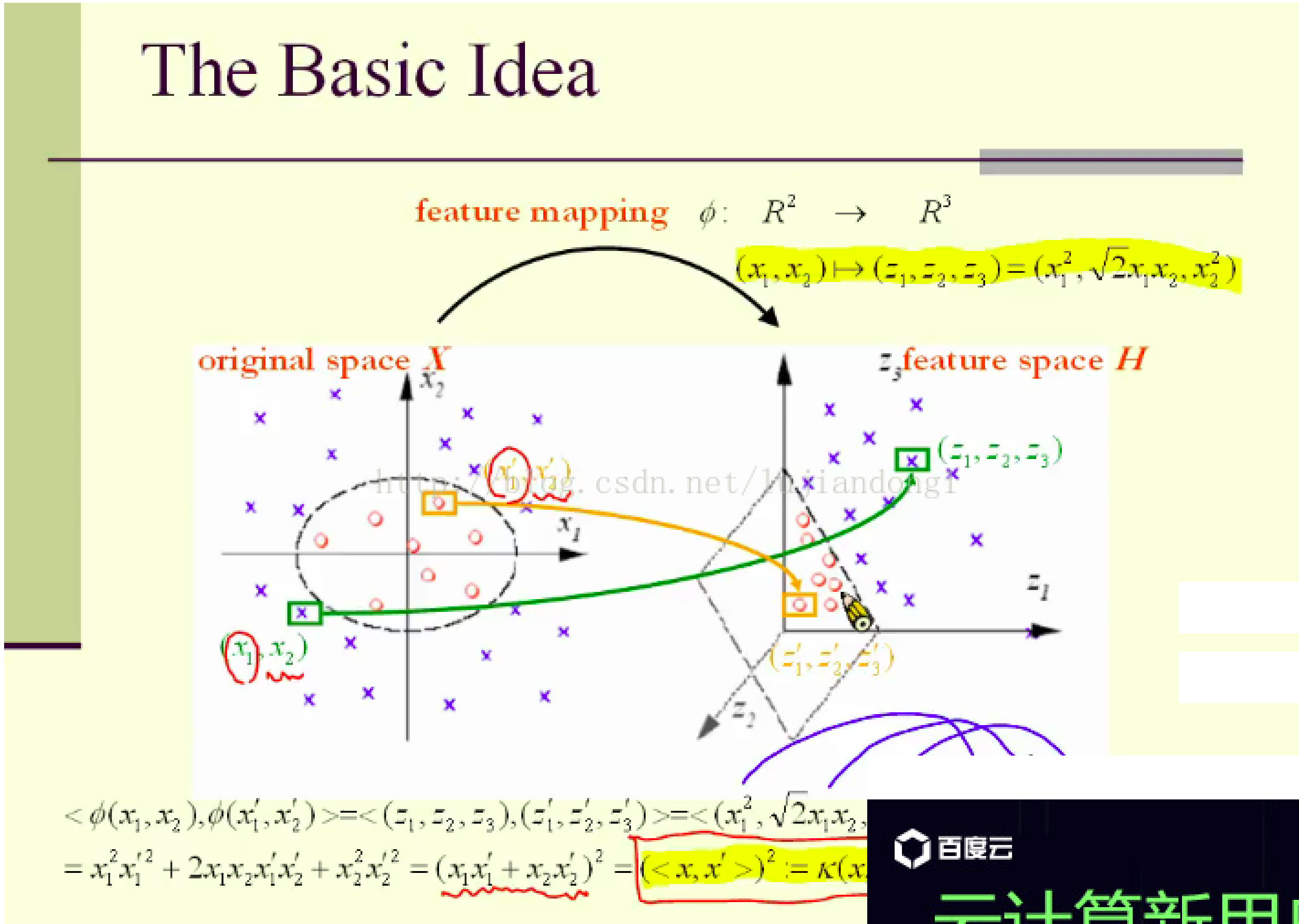
$$\frac{x_1^{\ 2}}{a^2}+\frac{x_2^{\ 2}}{b^2}=1$$

，将数据映射到三维空间，就可以得到线性的分类面，

$$\frac{1}{a^2}x_1^{\ 2}+\frac{1}{b^2}x_2^{\ 2}=1 \Rightarrow \frac{1}{a^2}\cdot z_1+0\cdot z_2+\frac{1}{b^2}\cdot z_3=1 => (x_1,x_2) \Rightarrow (z_1,z_2,z_3)=(x^2{}_1,0,x_2^2)$$

总结：在低维空间线性不可分，映射到高纬空间，线性可分的概率会增大，比如说，数据在一维空间线性可分的难度比二维空间线性可分的难度大，二维空间线性可分的难度比在三维空间线性可分的难度大。以此类推。

Kernal的另一个关键点事Kernal Function:



2维空间映射到3维空间的结果是：

$$(x_1,x_2)\to(z_1,z_2,z_3);(x_1',x_2',x_3')\to(z$$

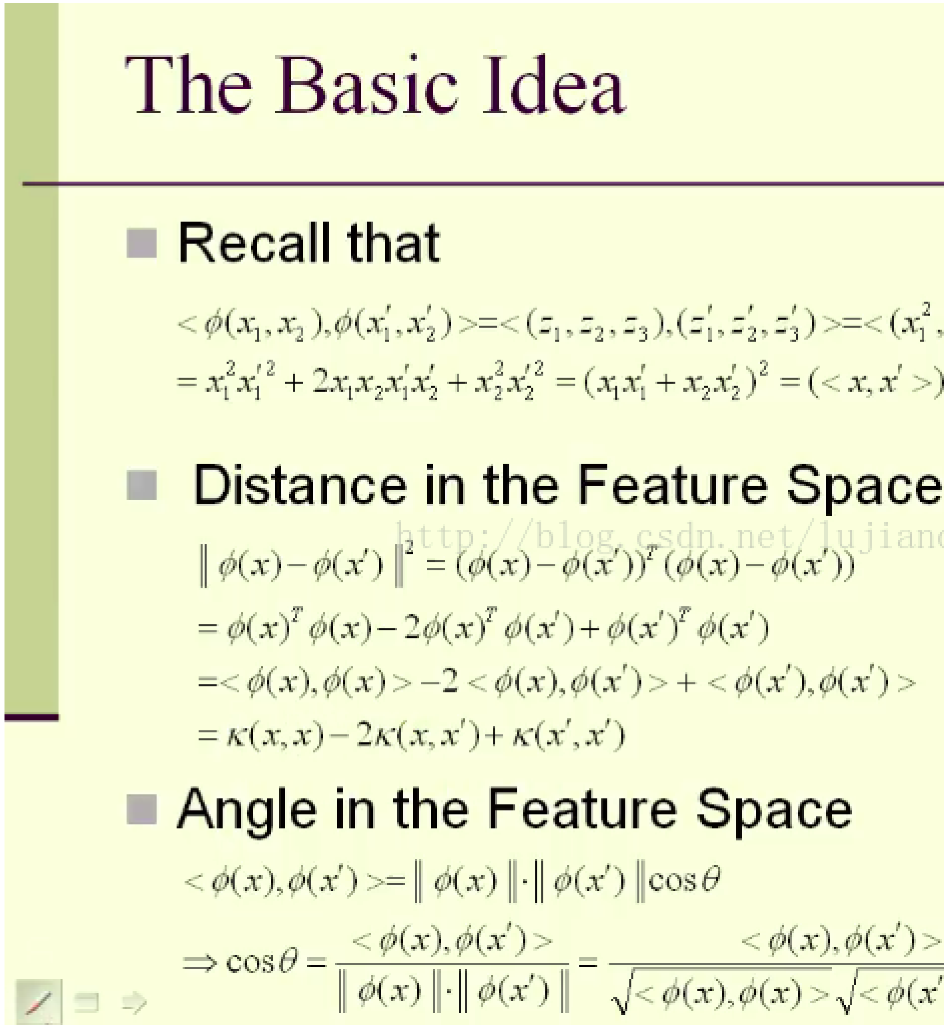
积为：

$$<\phi(x_1,x_2),\phi(x_1',x_2')>=K(x,x')$$

。也就是说，我们的内核函数K（），计算的是新空间的数据点之间的相似度。

内核方法总结：1、将数据从第维映射到更高维。2，映射到高维后，数据可以线性可分。3、kernel function可以计算高纬度空间的几何性质，角度，距离，并不需要得到映射函数 ϕ .

现在推导，可以利用kernel function计算高纬度空间的距离，角度。



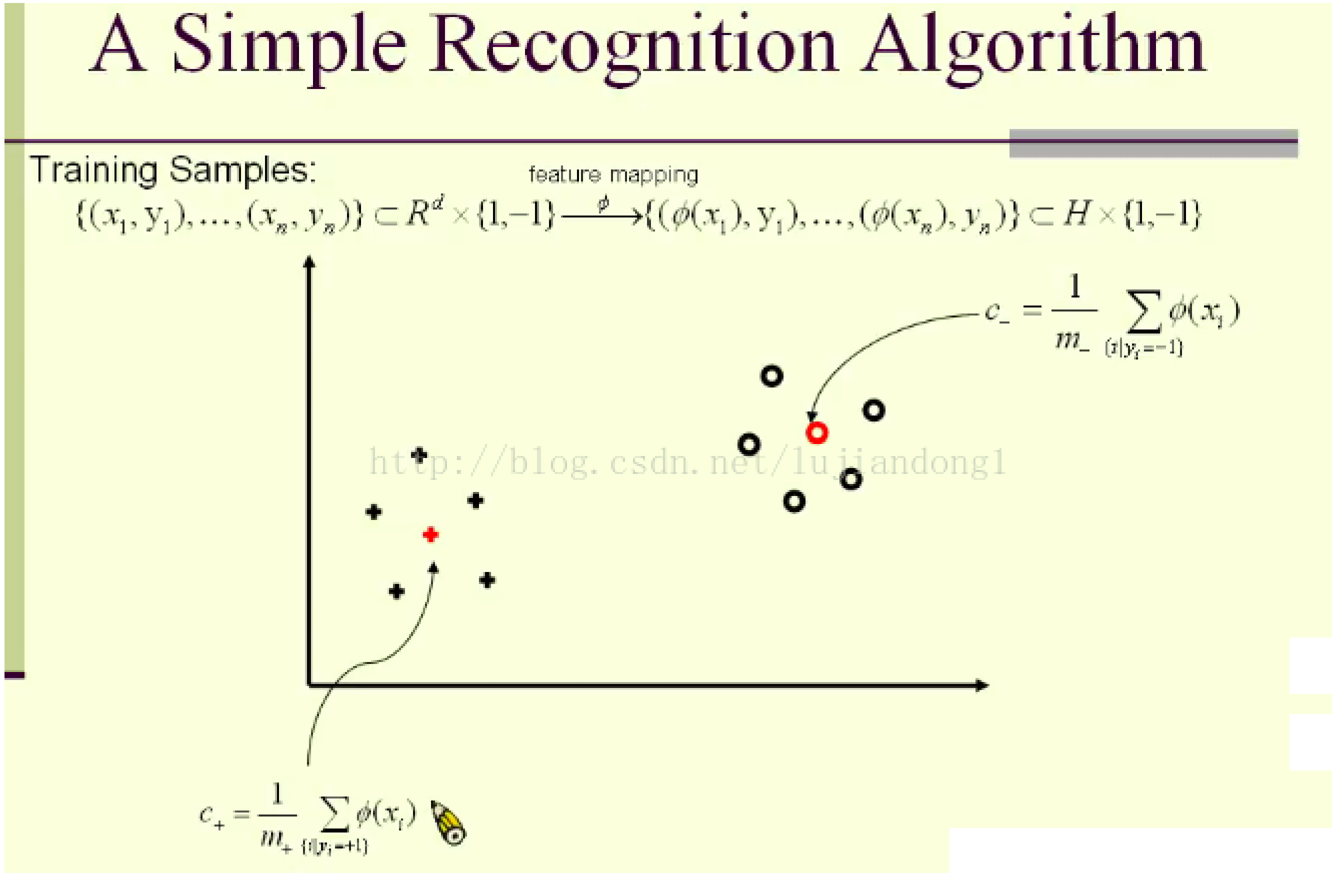
总结：在高维空间中要计算数据向量之间的角度，距离，只需要知道kernel function即可，并不需要知道 ϕ .

现在引出内积矩阵，也叫做Gram Matrix ,Kernel Matrix。Kernel Matrix其实就是把所有的样本点映射到高维空间，然后在高维空间中计算内积，形成内积矩阵。内积矩阵的计算，也只需要，kernel function,不需要映射函数

$$K = \begin{bmatrix} <\phi(x_1),\phi(x_1)> & \cdots & <\phi(x_1),\phi(x_N)> \\ \vdots & \ddots & \vdots \\ <\phi(x_N),\phi(x_1)> & \cdots & <\phi(x_N),\phi(x_N)> \end{bmatrix}$$

$$= \begin{bmatrix} \kappa(x_1,x_1) & \cdots & \kappa(x_1,x_N) \\ \vdots & \ddots & \vdots \\ \kappa(x_N,x_1) & \cdots & \kappa(x_N,x_N) \end{bmatrix}$$

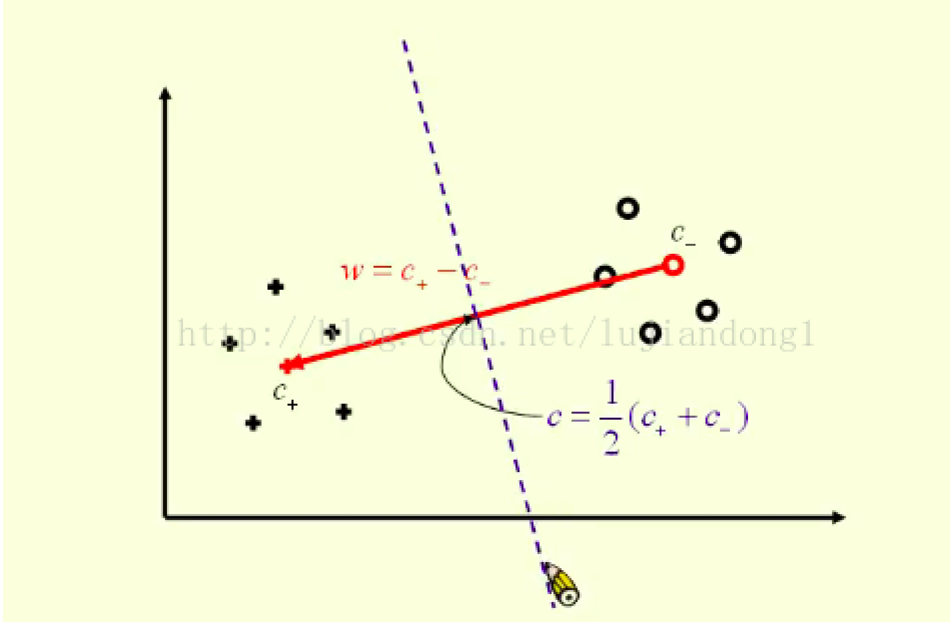
现在，举个小例子，讲解kernel function 的使用。



在地位空间中的数据点: (x_n, y_n) ,映射到高纬空间得到 $(\phi(x_n),$

高维空间中的数据分布，显而易见，在高维的空间中，数据已经是线性可分了

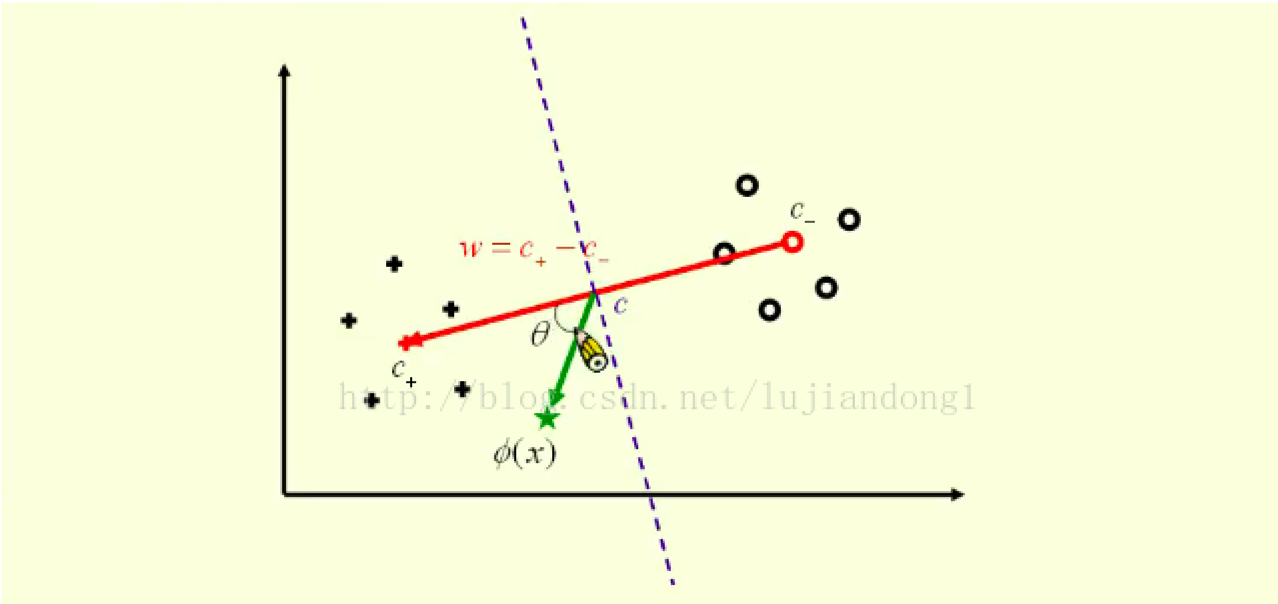
策函数呢。



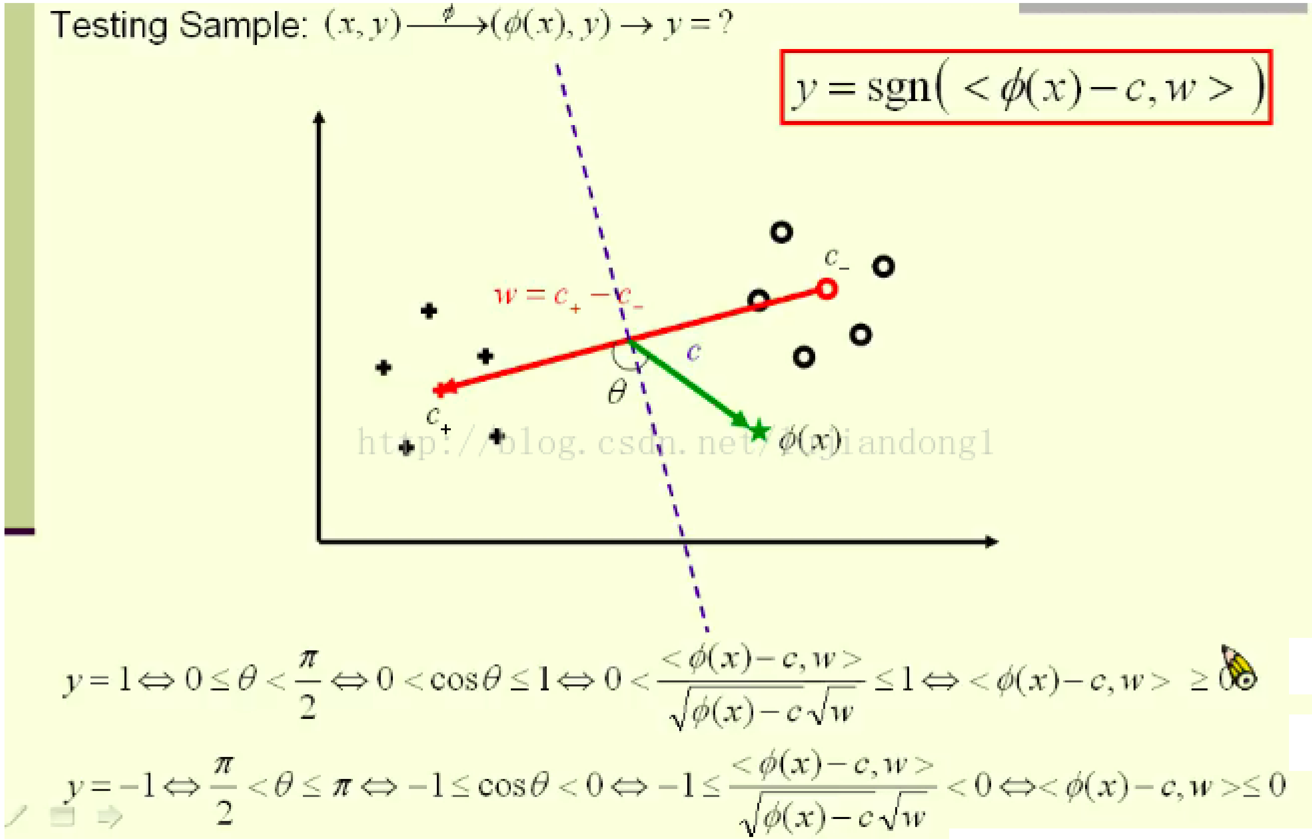
其中，$c_+ = \frac{1}{m_+} \sum_{\{i|y_i=+1\}} \phi(x_i)$，是正类别（标签值为+1的数据的中心点）$c_- = \frac{1}{m_-} \sum_{\{i|y_i=-1\}} \phi(x_i)$，是负类别（标签值为-1的数据的中心点）

。那么中间那条虚线就是对应于我们的分类平面，虚线和红线的交点坐标为，

那样就可以得到决策函数如下：



$$y=1 \Leftrightarrow 0 \le \theta < \frac{\pi}{2} \Leftrightarrow 0 < \cos\theta \le 1 \Leftrightarrow 0 < \frac{<\phi(x)-c,w>}{\sqrt{\phi(x)-c}\sqrt{w}} \le 1 \Leftrightarrow <\phi(x)-c,w> \ \ge 0$$



所有，要判断在高维的空间中，未知的点对应的表是啥，关键是计算出

$$y = \mathrm{sgn}(<\phi(x)-c,w>)$$

。那么如何用核函数计算

$$<\phi(x)-c, w> = w^T(\phi(x)-c) = w^T\phi(x) - w^T c = (c_+ - c_-)^T\phi(x) - \frac{1}{2}(c_+ - c_-)^T(c_+ + c_-)$$

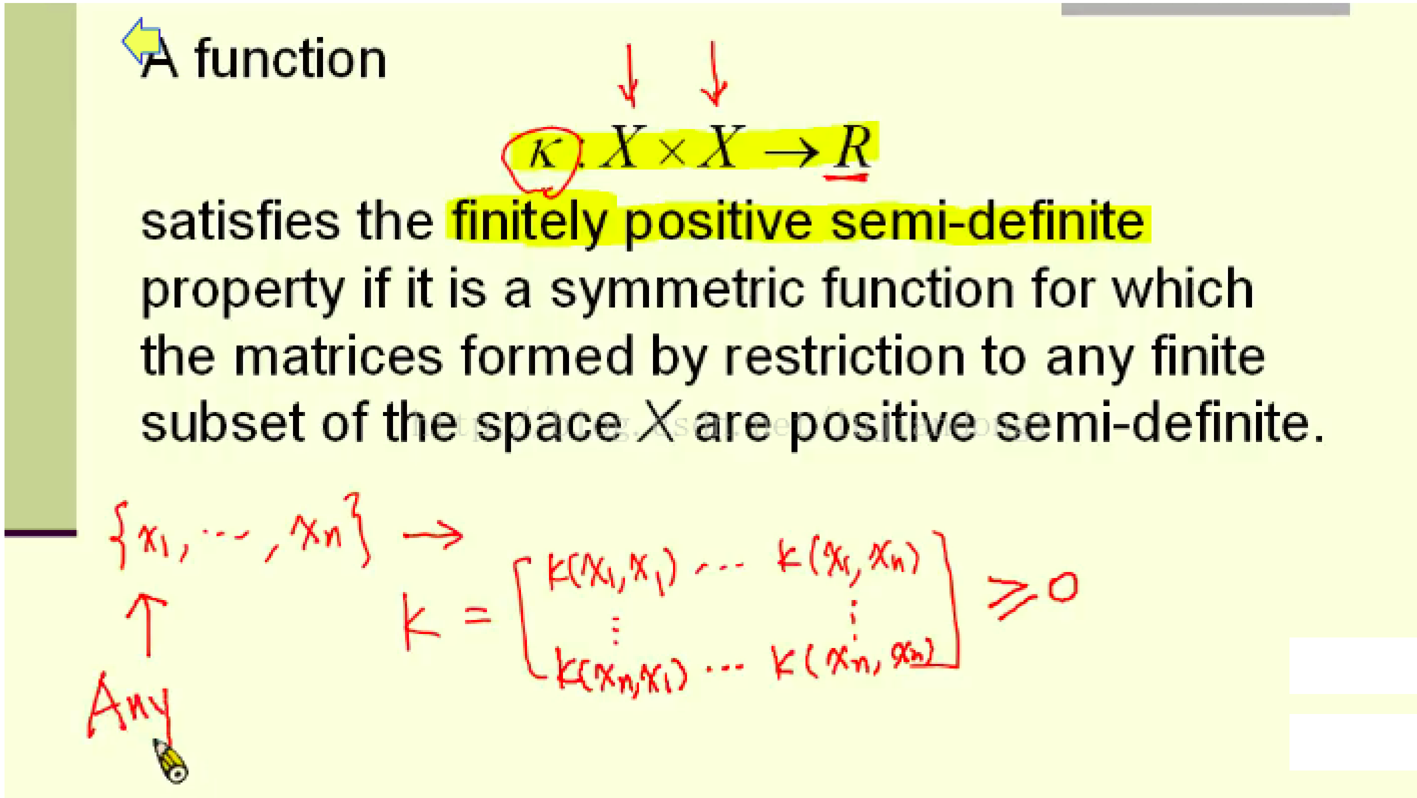
$$= \left(\frac{1}{m_+}\sum_{\{i|y_i=1\}}\phi(x_i) - \frac{1}{m_-}\sum_{\{i|y_i=-1\}}\phi(x_i)\right)^T \phi(x) - \left(\frac{1}{2}c_+^T c_+ - c_+^T c_- + \frac{1}{2}c_-^T c_-\right)$$

$$= \left(\frac{1}{m_+}\sum_{\{i|y_i=1\}}\phi(x_i)^T\phi(x) - \frac{1}{m_-}\sum_{\{i|y_i=-1\}}\phi(x_i)^T\phi(x)\right) - b$$

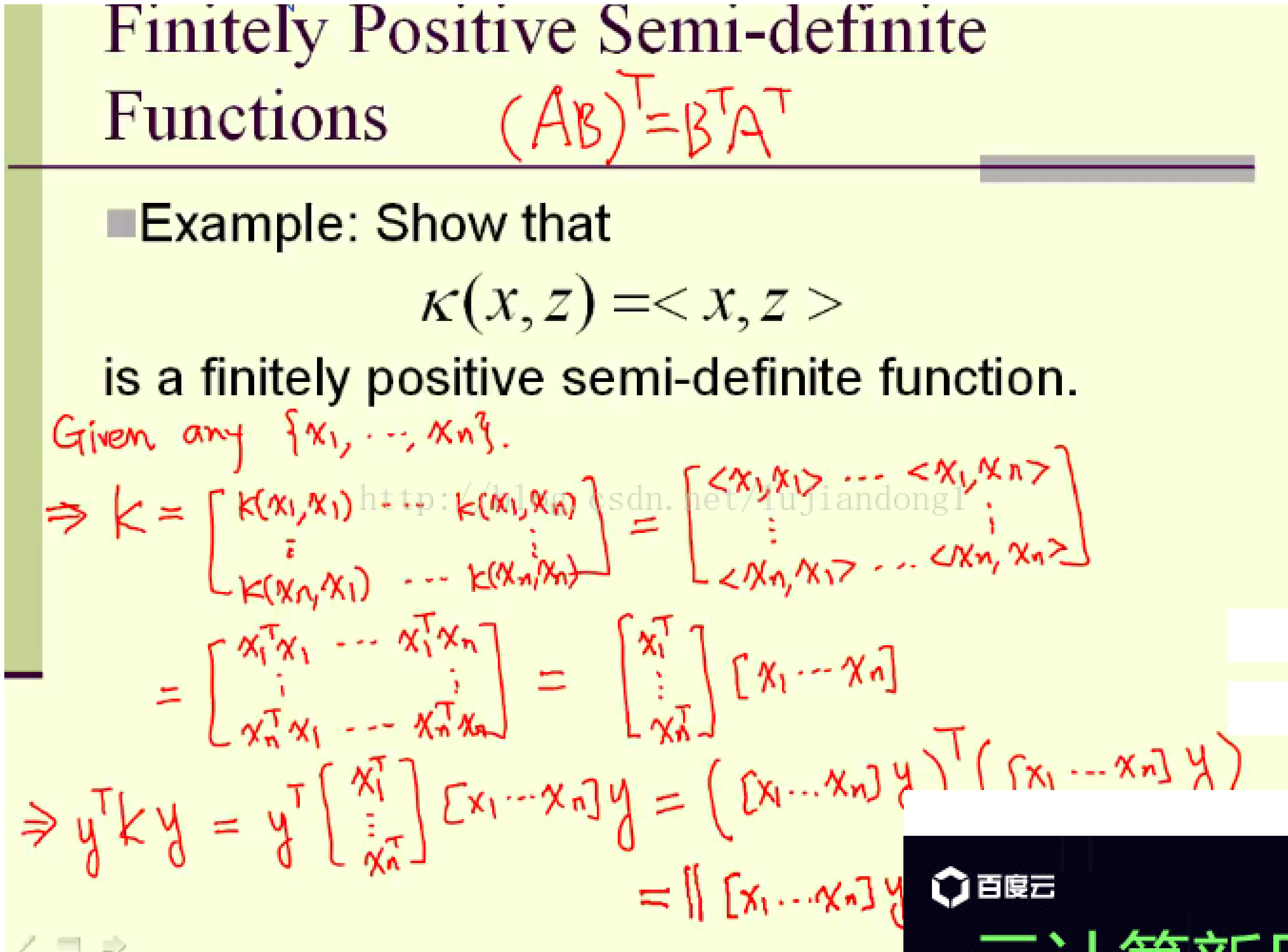
$$= \left(\frac{1}{m_+}\sum_{\{i|y_i=1\}}\kappa(x, x_i) - \frac{1}{m_-}\sum_{\{i|y_i=-1\}}\kappa(x, x_i)\right) - b$$

其中，b也是可以用kernel function函数解决的，所以，只要我们知道kernel function 我们是可以用k来计算出y的。

现在进行总结：对于kernel function 映射函数是不一定需要的，只要有kernel function即可，但是不是随便定义一个kernel function 就可以找到相应的映射,其kernel matrix 是半正定矩阵。



现证明k（x,z)=<x,z>对应的kernel function是正半定矩阵。



Dual Representation:

在读关于核映射的paper时，我们经常看到如下式子：

$$f(x) = w^T \phi($$

如果是预测问题，那么f(x)就是预测值，如果是分类问题，那么f(x)就是对应的类别。举个例子，对于刚才我们的分类问题，

$$\boxed{\begin{aligned} &y = \operatorname{sgn}(<\phi(x)-c, w>) \\ &\Leftrightarrow y = \operatorname{sgn}(f(x)),\quad f(x) = w^T\phi(x) - w^T c, \\ &w^T = \frac{1}{m_+}\sum_{\{i|y_i=+1\}}\phi(x_i) - \frac{1}{m_-}\sum_{\{i|y_i=-1\}}\phi(x_i) \end{aligned}}$$

可以看出，W是 $\phi(x_i)$ 的线性组合，所以，判决平面完全由训练样本所控制的。在统计上面有个定理，因为我们所掌握的只是训练数据，所以，我们计算处理的判决平面，投影方向等等，都是由训练样本所决定的。

$$\boxed{w = \sum_{i=1}^{N}}$$

Dual Representation

$w = \sum_{i=1}^{N} \alpha_i \phi(x_i)$

$$f(\phi(x)) = \underline{w^T \phi(x) + b}$$

$$= \left(\underline{\left(\sum_{i=1}^{N} \alpha_i \phi(x_i) \right)^T} \phi(x) \right) + b$$

$\phi \rightarrow dim=100$

$$= \left(\sum_{i=1}^{N} \alpha_i \underline{\phi(x_i)^T \phi(x)} \right) + b$$

inner product

$$= \left(\sum_{i=1}^{N} \alpha_i \underline{\kappa(x_i, x)} \right) + b$$

顶 0

踩 0

上一篇 安装scikit-learn，win7 64位（装了一下午，终于搞定了）解决ImportError: DLL load failed

下一篇 SVM的两个参数 C 和 gamma

我的同类文章

机器学习（32）

- LSTM的完整推导过程，附上.. 2016-11-16 阅读 239
- 随机森林的几个重要参数 2016-11-05 阅读 292
- 机器学习常见的六大错误 2016-11-02 阅读 111
- 安装jpype出现_jpype错误的… 2016-10-18 阅读 194
- pandas使用get_dummies进… 2016-10-17 阅读 1721
- xgboost cross_validation&自.. 2016-10-11 阅读 896
- XGBoost Stopping to Avoid … 2016-10-10 阅读 417
- AUC的理解与应用场景 2016-07-23 阅读 167
- k-means优化 & k-means距… 2016-05-05 阅读 769
- vowpal_wabbit是在单机上性… 2015-12-30 阅读 514
- 机器学习经典博客链接 2015-06-15 阅读 409

更多文章

猜你在找

统计机器学习入门——线性模型选择与正则化2
python数据分析与机器学习实战
统计机器学习入门——线性模型选择与正则化1
统计机器学习入门——分类2
统计机器学习入门——重抽样方法

R语言与机器学习中的回归方法
转R语言与机器学习中的回归方
R语言与机器学习中的回归方法
机器学习实践系列之9 - 视频结
机器学习基础林軒田笔记之七

查看评论

暂无评论

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML LBS Unity Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure S Angular Cloud Foundry Redis Scala Django Bootstrap