机器学习 第7周

法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

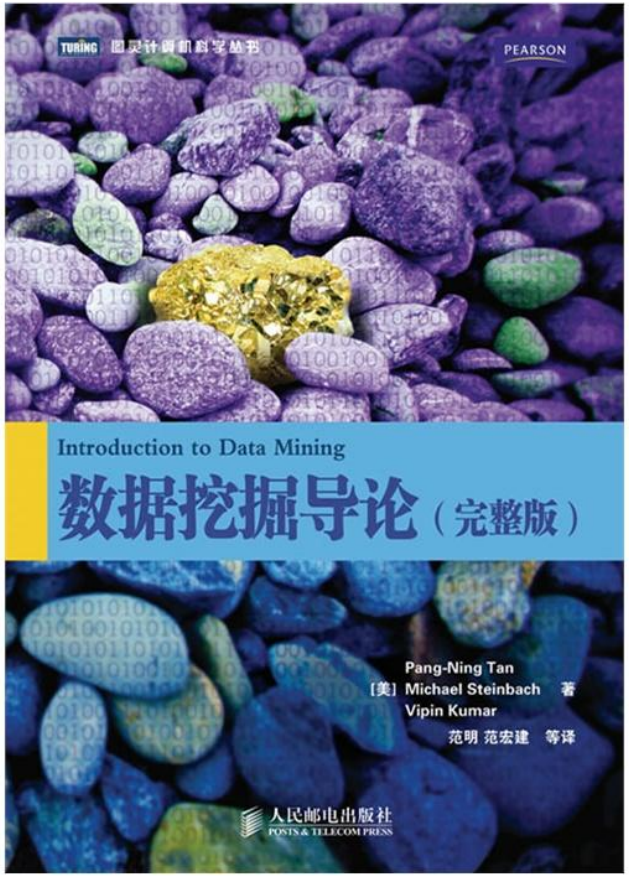
课程详情访问炼数成金培训网站

http://edu.dataguru.cn

支持向量机 SVM

- 原创性（非组合）的具有明显直观几何意义的分类算法，具有较高的准确率
- 源于Vapnik和Chervonenkis关于统计学习的早期工作（1971年），第一篇有关论文由Boser、Guyon、Vapnik发表在1992年（参考文档见韩家炜书9.10节）
- 思想直观，但细节异常复杂，内容涉及凸分析算法，核函数，神经网络等高深的领域，几乎可以写成单独的大部头专著。大部分非专业人士会觉得难以理解。
- 某名人评论：**SVM是让应用数学家真正得到应用的一种算法**

参考书

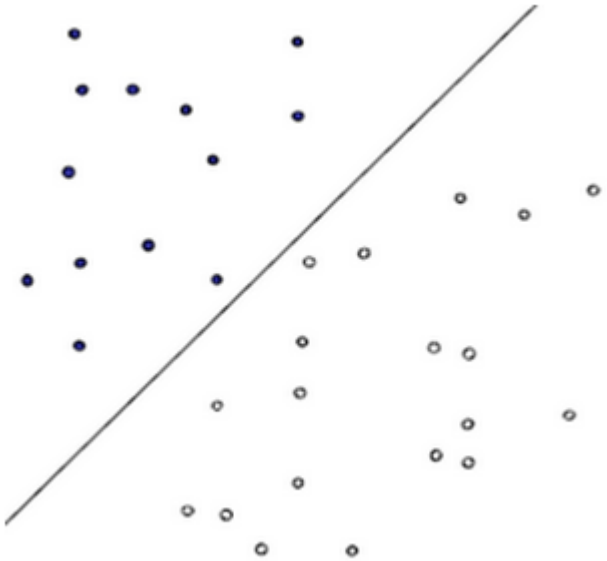

TURING 图灵计算机科学丛书
PEARSON
Introduction to Data Mining
数据挖掘导论（完整版）
Pang-Ning Tan
[美] Michael Steinbach 著
Vipin Kumar
范明 范宏建 等译
人民邮电出版社
POSTS & TELECOM PRESS

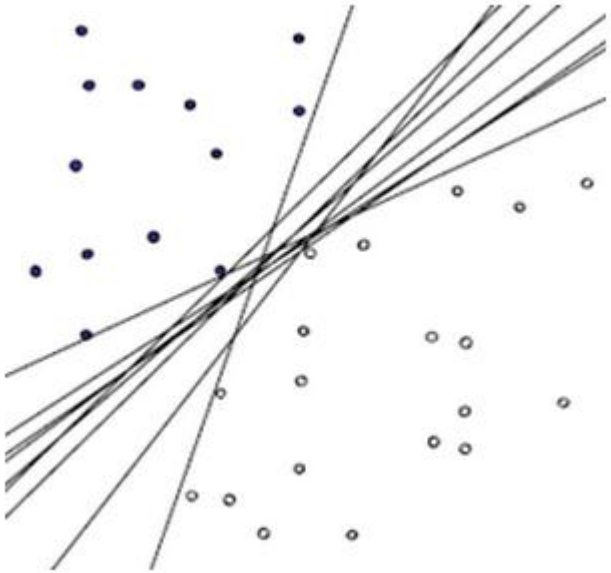
思路

- 简单情况，线性可分，把问题转化为一个凸优化问题，可以用拉格朗日乘子法简化，然后用既有的算法解决
- 复杂情况，线性不可分，用映射函数将样本投射到高维空间，使其变成线性可分的情形。利用核函数来减少高维度计算量

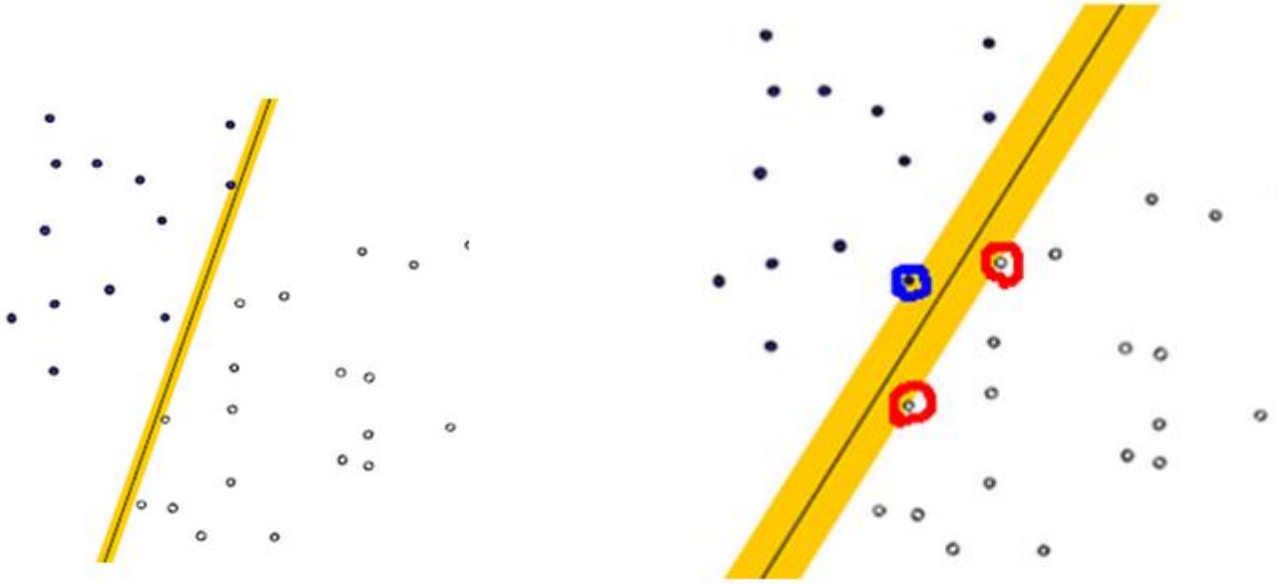
线性可分的情形

- 问题的提出：最优分离平面（决策边界）





最大边缘超平面（MMH）



最大边缘超平面（MMH）

- 决策边界边缘距离最远

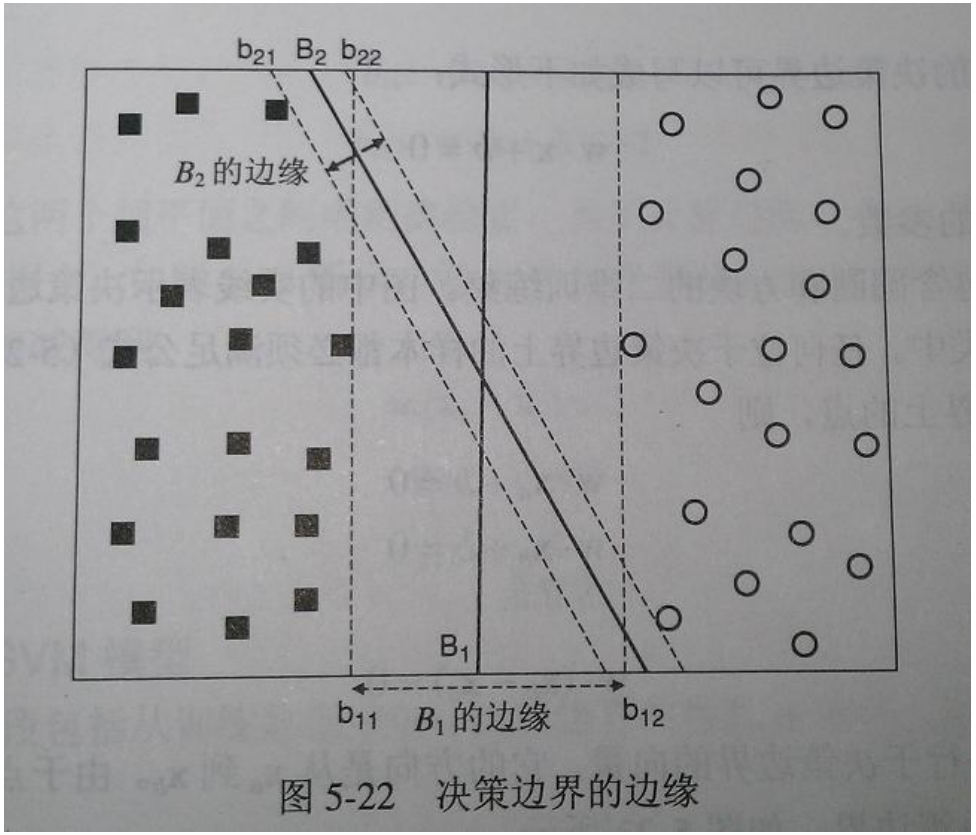
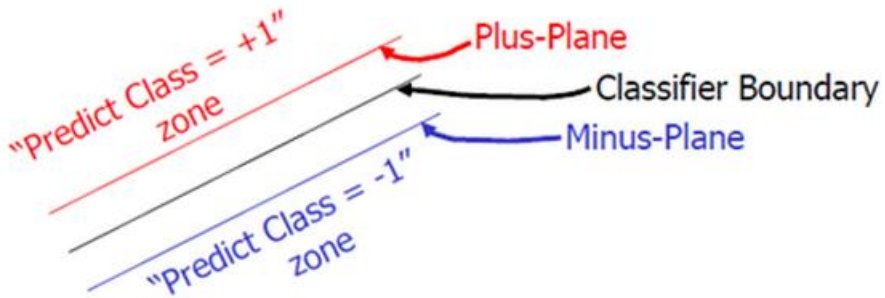
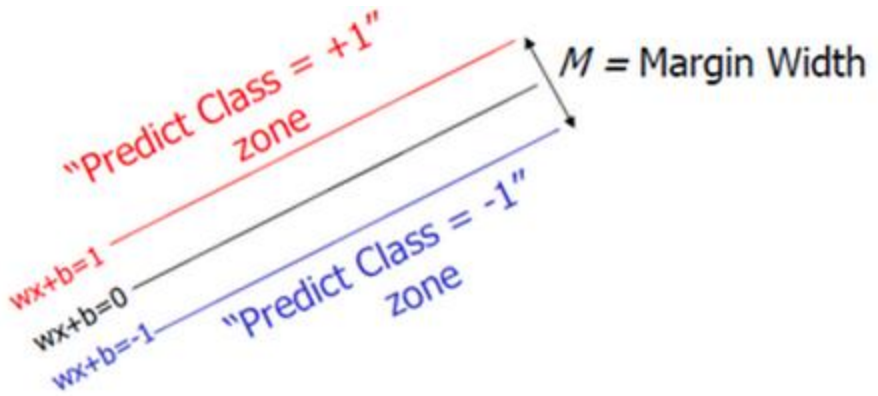


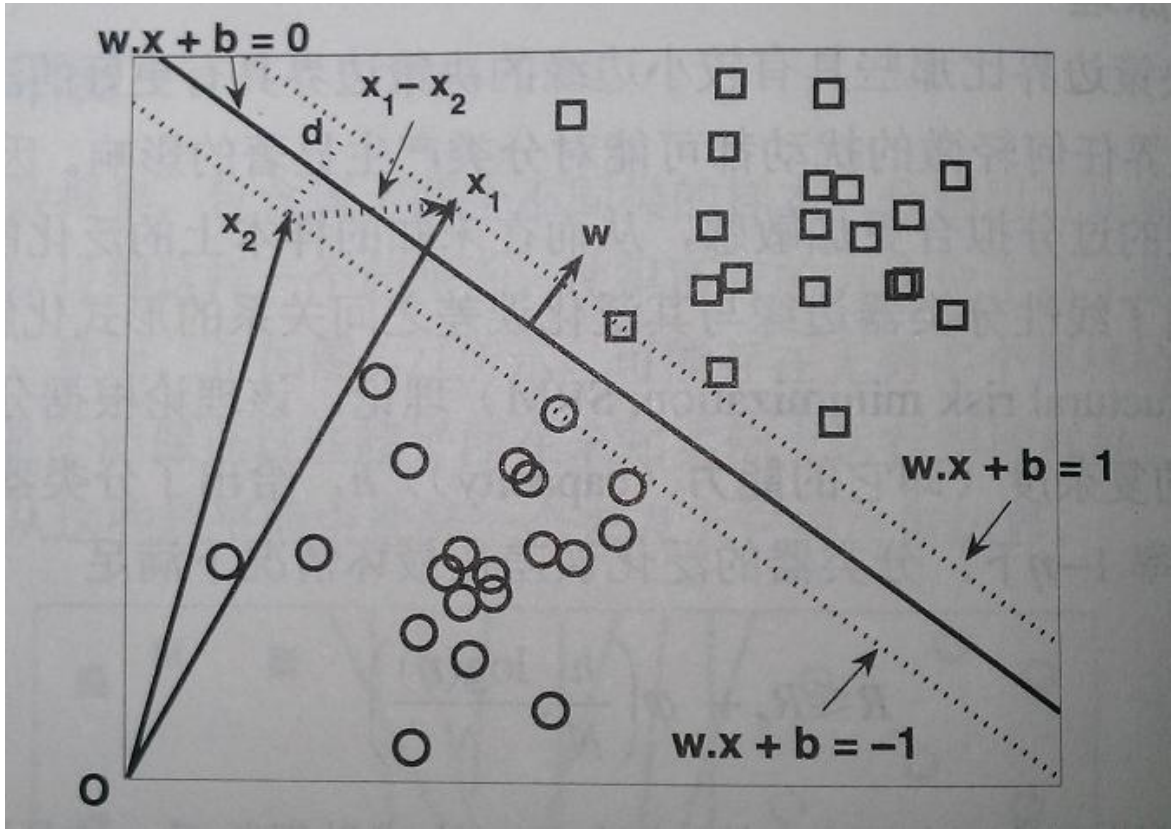
图 5-22　决策边界的边缘

最大边缘超平面（MMH）


"Predict Class = +1" zone
Plus-Plane
Classifier Boundary
Minus-Plane
"Predict Class = -1" zone


"Predict Class = +1" zone
M = Margin Width
wx+b=1
wx+b=0
wx+b=-1
"Predict Class = -1" zone

一些计算



$$b_{i1}: \ \mathbf{w}\cdot\mathbf{x}+b=1$$

$$b_{i2}: \ \mathbf{w}\cdot\mathbf{x}+b=-1$$

$$\mathbf{w}.(\mathbf{x}_1-\mathbf{x}_2)=2$$

$$\|\mathbf{w}\|\times d=2$$

$$\therefore d=\frac{2}{\|\mathbf{w}\|}$$

转化为凸优化问题

$$\begin{aligned} \mathbf{w}\cdot\mathbf{x_i}+b &\geqslant 1 \quad \text{如果 } y_i=1 \\ \mathbf{w}\cdot\mathbf{x_i}+b &\leqslant -1 \quad \text{如果 } y_i=-1 \end{aligned}$$

$$d=\frac{2}{\|\mathbf{w}\|}$$

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2}$$

$$\text{受限于} \quad y_i(\mathbf{w}\cdot\mathbf{x_i}+b)\geqslant 1, \quad i=1,2,\cdots,N$$

凸优化问题

- 可以寻求凸优化算法支持解决
- 可以使用拉格朗日乘子法继续简化

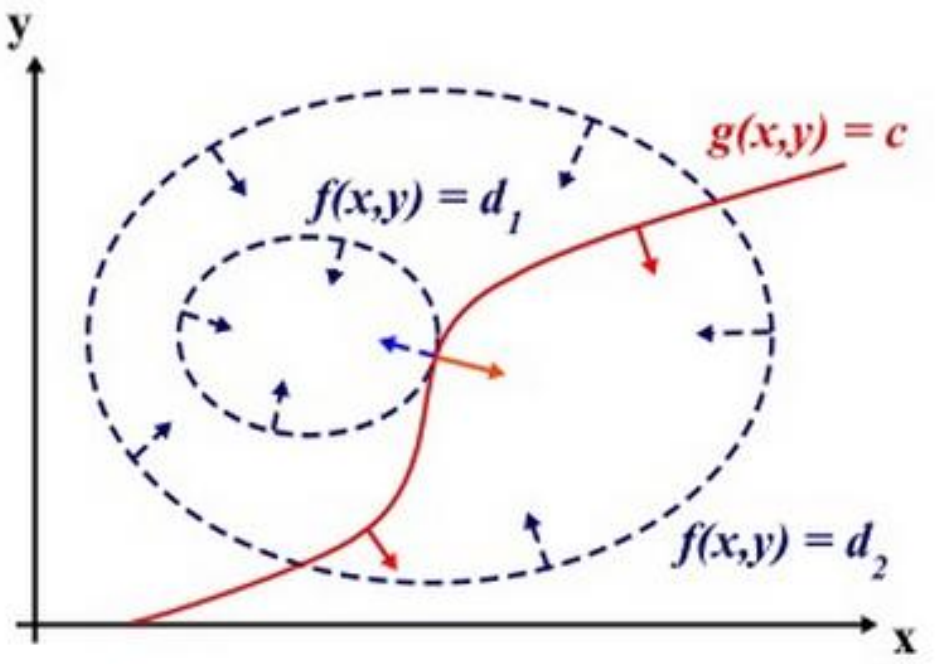
拉格朗日乘子法

$$L(w,b,a)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{n}\alpha_i(y_i(w^Tx_i+b)-1)$$

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

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

背景：拉格朗日乘子法的几何解释

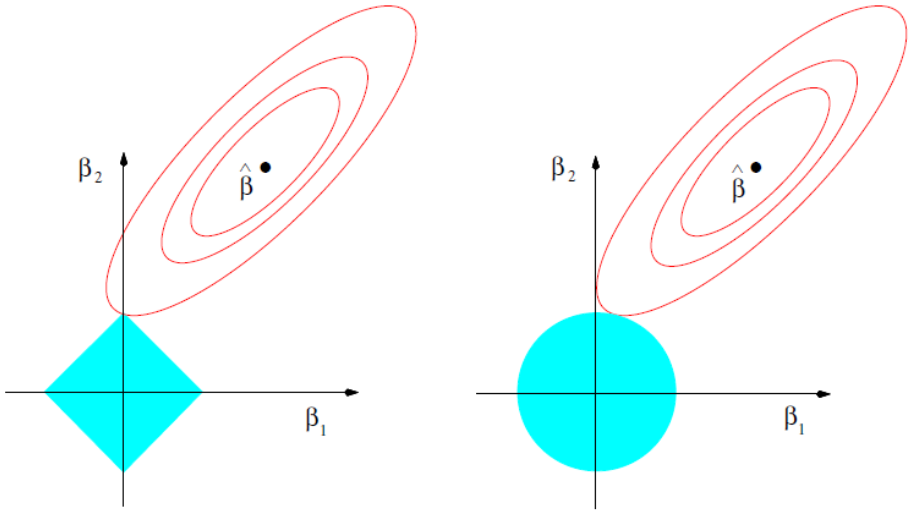


似曾相识

$$\hat{\beta}^{\text{ridge}} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}. \tag{3.41}$$

$$\hat{\beta}^{\text{ridge}} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2, \tag{3.42}$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 \le t,$$



Karush-Kuhn-Tucker 最优化条件 (KKT 条件)

$$\min. f(\mathbf{x})$$

$$\text{s.t.} \quad h_j(\mathbf{x}) = 0, j = 1,\ldots, p,$$

$$g_k(\mathbf{x}) \le 0, k = 1,\ldots, q,$$

$$\mathbf{x} \in \mathbf{X} \subset \Re^n$$

1. $h_j(\mathbf{x}_*) = 0, j = 1,\ldots, p, \; g_k(\mathbf{x}_*) \le 0, k = 1,\ldots, q,$

2. $\nabla f(\mathbf{x}_*) + \sum_{j=1}^{p} \lambda_j \nabla h_j(\mathbf{x}_*) + \sum_{k=1}^{q} \mu_k \nabla g_k(\mathbf{x}_*) = \mathbf{0},$

$\lambda_j \ne 0, \; \mu_k \ge 0, \; \mu_k g_k(\mathbf{x}_*) = 0.$

参考文章

- 关于支持向量机：
  http://www.cnblogs.com/LeftNotEasy/archive/2011/05/18/2034566.html
- 关于拉格朗日乘子法：http://blog.csdn.net/xianlingmao/article/details/7919597
- 关于KKT条件：
  http://hi.baidu.com/grandyang/item/94cd68dfdc06941e21e25099
- 求解凸优化问题的方法：
  http://wenku.baidu.com/link?url=Qwc1n8RL8GVzi0Bk_KKsru0rvm-TgyOUQvWZtrBEQVjbmrn0rNfv-SAcJgBgZ8kkx0wl9r5IC5rvEYs44fQ0p_L-KExJtvVTS3Uj4S68UpG

进一步简化为对偶问题

- 前一步得出的KKT条件中的变量太多
- 为后续引入核函数作模型准备
- 将前一步的梯度计算结果重新代入到拉格朗日函数

$$L(w,b,a)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{n}\alpha_i(y_i(w^Tx_i+b)-1)$$

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

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

$$L(w,b,\alpha)=\frac{1}{2}\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j$$

对偶问题：简化后的凸优化问题

- 比之前的凸优化问题简洁
- 可以用各种凸优化算法加以解决
- 只有支持向量参与计算，所以计算规模远低于我们的想象

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

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

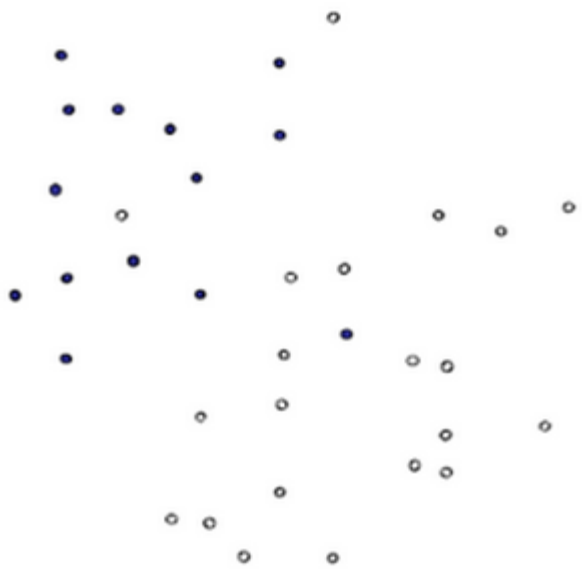
$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

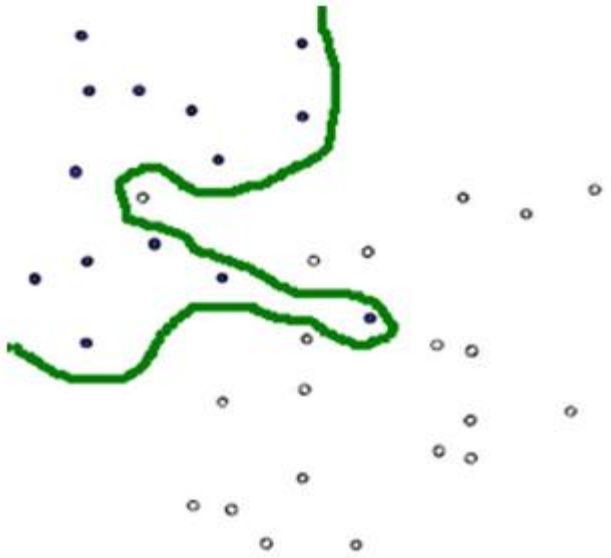
对偶问题

- 对偶公式中的未知数仅涉及拉格朗日乘子，而原问题中未知数还包含决策边界几何特征参数，未知数太多
- 待定乘子中实质有很多为0，仅在“支持向量”处不为0，所以最后的出的函数表达式远比想象中简单（但问题是预先无法知道哪些样本点是“支持向量”）

线性不可分的情形

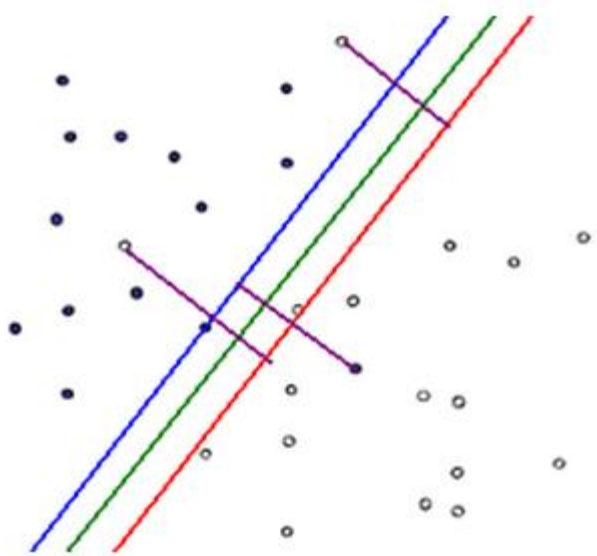
- 大部分情况都不是线性可分的
- 线性不可分时无法使用前述数学技巧
- 也可以使用加惩罚函数的方法解决：
  http://www.cnblogs.com/LeftNotEasy/archive/2011/05/18/2034566.html





松弛变量与惩罚函数

- 公式中蓝色的部分为在线性可分问题的基础上加上的惩罚函数部分，当xi在正确一边的时候，ε=0，R为全部的点的数目，C是一个由用户去指定的系数，表示对分错的点加入多少的惩罚，当C很大的时候，分错的点就会更少，但是过拟合的情况可能会比较严重，当C很小的时候，分错的点可能会很多，不过可能由此得到的模型也会不太正确，所以如何选择C是有很多学问的，不过在大部分情况下就是通过经验尝试得到的。



$$\min \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{R} \varepsilon_i$$

$$y_i (w^T x_i + b) \geq 1 - \varepsilon_i, \varepsilon_i \geq 0$$

线性不可分情形下的对偶问题

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

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

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

SMO算法

- Sequential minimal optimization
- Microsoft Research的John C. Platt在1998年提出
- 最快的二次规划优化算法
- 针对线性SVM和数据稀疏时性能更优
- 原始论文《Sequential Minimal Optimization A Fast Algorithm for Training Support Vector Machines》
- http://www.cnblogs.com/jerrylead/archive/2011/03/18/1988419.html

算法基本思路

基本思路是每次只更新两个乘子，迭代获得最终解。

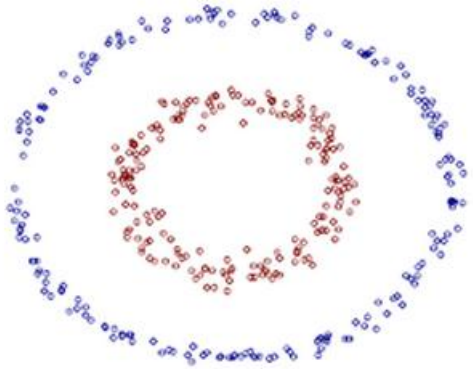
计算流程可表示如下

Repeat till convergence {

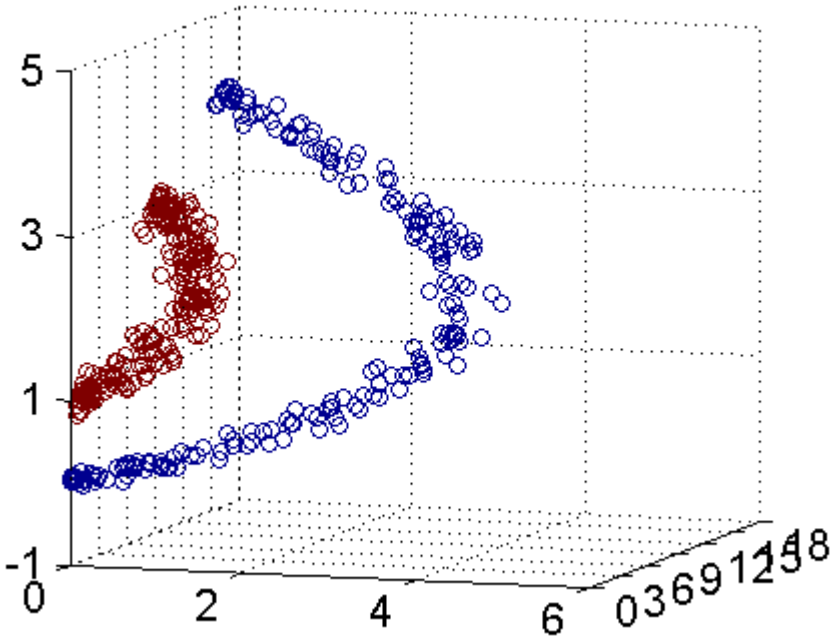
1. Select some pair α_i and α_j to update next (using a heuristic that tries to pick the two that will allow us to make the biggest progress towards the global maximum).
2. Reoptimize $W(\alpha)$ with respect to α_i and α_j, while holding all the other α_k's $(k \neq i, j)$ fixed.

}

映射至高维空间



$$z_1 = x_1^{\ 2}, z_2 = x_2^{\ 2}, z_3 = x_2$$



维度灾难

- 公式中红字的地方要使用映射后的样本向量代替做内积
- 最初的特征是n维的，我们将其映射到n^2维，然后再计算，这样需要的时间从原先的O（n）变成O（n^2）

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

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

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

$$\text{maximize} \quad L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

$$\text{subject to} \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

引入核函数

- 参考：http://www.cnblogs.com/jerrylead/archive/2011/03/18/1988406.html
- 假设n=3，见下算例
- 这时发现我们可以只计算原始样本x和z内积的平方（时间复杂度是O(n)），就等价于计算映射后高维样本的内积。

$$\phi(x) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \end{bmatrix}$$

$$K(x,z) = (x^Tz)^2$$

展开后，得

$$K(x,z) = (x^Tz)^2 = \left(\sum_{i=1}^{n} x_iz_i\right)\left(\sum_{j=1}^{n} x_jz_j\right) = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i\,x_jz_iz_j$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} (x_i\,x_j)(z_iz_j) = \phi(x)^T\phi(z)$$

另一种核函数

$$
\begin{aligned}
K(x,z) &= (x^T z + c)^2 \\
&= \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j) + \sum_{i=1}^{n} (\sqrt{2c}x_i)(\sqrt{2c}z_i) + c^2.
\end{aligned}
$$

$$
\phi(x) = \begin{bmatrix}
x_1 x_1 \\
x_1 x_2 \\
x_1 x_3 \\
x_2 x_1 \\
x_2 x_2 \\
x_2 x_3 \\
x_3 x_1 \\
x_3 x_2 \\
x_3 x_3 \\
\sqrt{2c}x_1 \\
\sqrt{2c}x_2 \\
\sqrt{2c}x_3 \\
c
\end{bmatrix}
$$

几种常用核函数

- 可以理解为描述投影样本向量的相似度

$$h\text{ 次多项式核函数}: K(X_i, X_j) = (X_i \cdot X_j + 1)^h$$

$$\text{高斯径向基函数核函数}: K(X_i, X_j) = e^{-\|X_i - X_j\|^2/2\sigma^2}$$

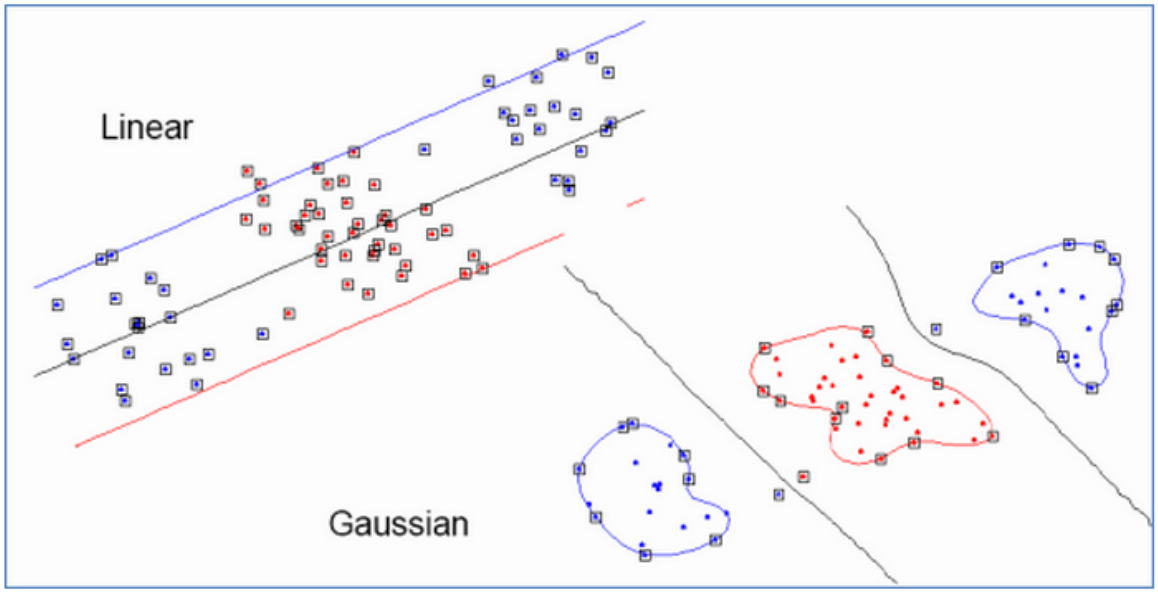
$$\text{S 型核函数}: K(X_i, X_j) = \tanh(\kappa X_i \cdot X_j - \delta)$$

$$\kappa(x_i, x_j) = (x_i . x_j + 1)^d$$

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

高斯径向基函数核函数

- 函数类似于高斯分布，因此称为高斯核函数，也叫做径向基函数(Radial Basis Function 简称RBF)。
- 它能够把原始特征映射到无穷维。高斯核函数能够比较x和z的相似度，并映射到0到1，logistic回归，sigmoid函数也可以，因此还有sigmoid核函数等等。



核函数的有效性：

问题：给定一个函数K，我们能否使用K来替代计算 $\phi(x)^T\phi(z)$，也就说，是否能够找出一个ϕ，使得对于所有的x和z，都有$K(x,z) = \phi(x)^T\phi(z)$？

比如给出了$K(x,z) = (x^Tz)^2$，是否能够认为K是一个有效的核函数。

核函数矩阵

下面来解决这个问题，给定m个训练样本$\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$，每一个$x^{(i)}$对应一个特征向量。那么，我们可以将任意两个$x^{(i)}$和$x^{(j)}$带入K中，计算得到$K_{ij} = K(x^{(i)}, x^{(j)})$。I可以从1到m，j可以从1到m，这样可以计算出m*m的核函数矩阵（Kernel Matrix）。为了方便，我们将核函数矩阵和$K(x, z)$都使用K来表示。

如果假设K是有效地核函数，那么根据核函数定义

$$K_{ij} = K(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^T \phi(x^{(j)}) = \phi(x^{(j)})^T \phi(x^{(i)}) = K(x^{(j)}, x^{(i)}) = K_{ji}$$

Mercer定理

Mercer定理：

如果函数K是$\mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$上的映射（也就是从两个n维向量映射到实数域）。那么如果K是一个有效核函数（也称为Mercer核函数），那么当且仅当对于训练样例$\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$，其相应的核函数矩阵是对称半正定的。

在拉格朗日函数中使用核函数简化

$$\max \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$0 \le \alpha_i \le C \quad \sum_i \alpha_i y_i = 0$$

$$\text{maximize} \quad L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

$$\text{subject to} \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

R中实验SVM

- 使用e1071包

源代码

```
data(iris)
attach(iris)
## classification mode
# default with factor response:
model <- svm(Species ~ ., data = iris)
# alternatively the traditional interface:
x <- subset(iris, select = -Species)
y <- Species
model <- svm(x, y)
print(model)
summary(model)
```

源代码

```
# test with train data
pred <- predict(model, x)
# (same as:)
pred <- fitted(model)
# Check accuracy:
table(pred, y)
# compute decision values and probabilities:
pred <- predict(model, x, decision.values = TRUE)
attr(pred, "decision.values")[1:4,]
# visualize (classes by color, SV by crosses):
plot(cmdscale(dist(iris[,-5])),
col = as.integer(iris[,5]),
pch = c("o","+")[1:150 %in% model$index + 1])
```

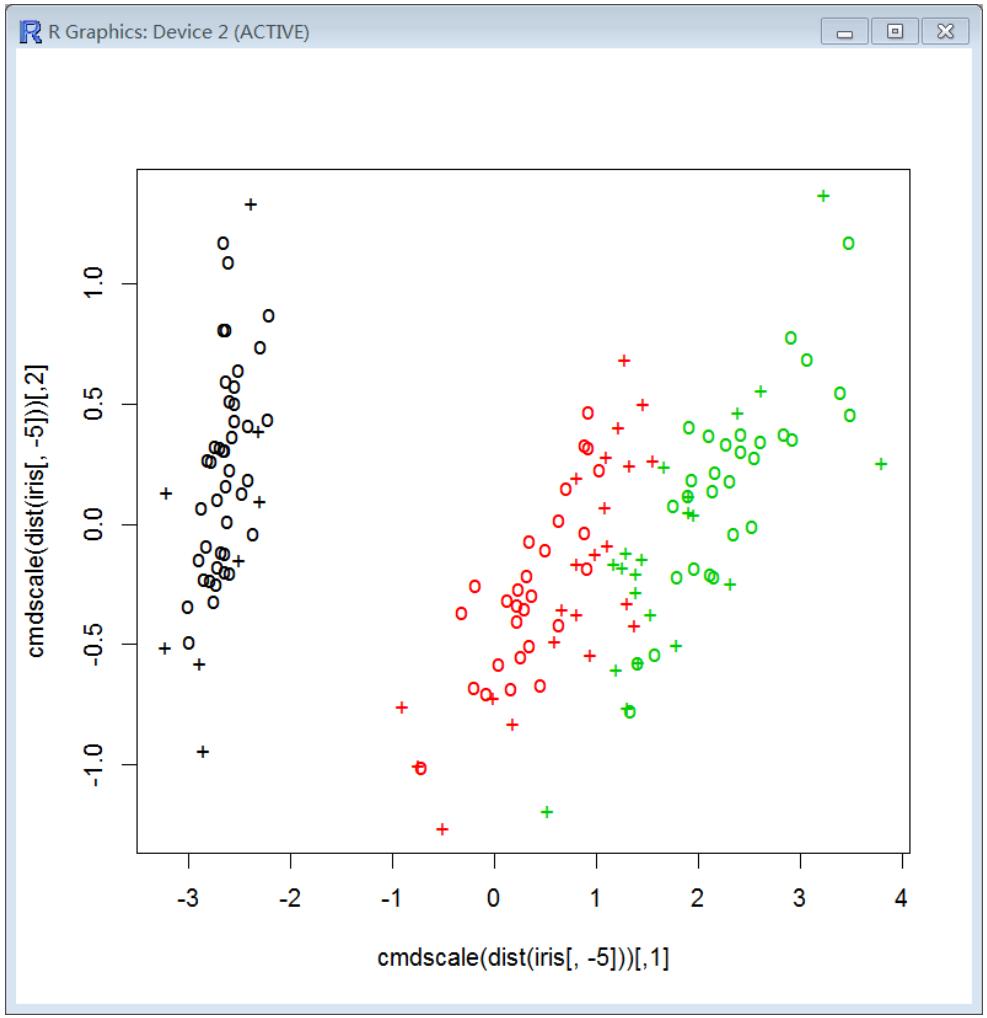
实验结果

```
> data(iris)
> attach(iris)
> x <- subset(iris, select = -Species)
> y <- Species
> model <- svm(x, y)
> print(model)

Call:
svm.default(x = x, y = y)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.25

Number of Support Vectors:  51
```

实验结果

```
> pred <- predict(model, x)
> # (same as:)
> pred <- fitted(model)
> table(pred, y)
            y
pred         setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         48         2
  virginica       0          2        48
> pred <- predict(model, x, decision.values = TRUE)
> attr(pred, "decision.values")[1:4,]
  setosa/versicolor setosa/virginica versicolor/virginica
1          1.196152         1.091757            0.6708810
2          1.064621         1.056185            0.8483518
3          1.180842         1.074542            0.6439798
4          1.110699         1.053012            0.6782041
> plot(cmdscale(dist(iris[,-5])),
+ col = as.integer(iris[,5]),
+ pch = c("o","+")[1:150 %in% model$index + 1])
> 
```

实验结果


R Graphics: Device 2 (ACTIVE)
cmdscale(dist(iris[, -5]))[,2]
1.0
0.5
0.0
-0.5
-1.0
-3
-2
-1
0
1
2
3
4
cmdscale(dist(iris[, -5]))[,1]

libsvm

- SVM用于模式识别或回归时，SVM方法及其参数、核函数及其参数的选择，目前国际上还没有形成一个统一的模式，也就是说最优SVM算法参数选择还只能是凭借经验、实验对比、大范围的搜寻或者利用软件包提供的交互检验功能进行寻优。
- LIBSVM是台湾大学林智仁(Lin Chih-Jen)副教授等开发设计的一个简单、易于使用和快速有效的SVM模式识别与回归的软件包。
- LibSVM是以源代码和可执行文件两种方式给出的。如果是Windows系列操作系统，可以直接使用软件包提供的程序，也可以进行修改编译；如果是Unix类系统，必须自己编译，软件包中提供了编译格式文件。
- 该软件包可在http://www.csie.ntu.edu.tw/~cjlin/免费获得。
- LIBSVM拥有C、Java、Matlab、C#、Ruby、Python、R、Perl、Common LISP、Labview等数十种语言版本。最常使用的是C、Matlab、Java和命令行(c语言编译的工具)的版本。

libsvm

www.csie.ntu.edu.tw/~cjlin/

Welcome to Chih-Jen Lin's Home Page

Research

- Machine Learning:
  Support vector machines (SVM), large-scale data classification, and machine learning software design.
- Operations Research:
  Large-scale Nonlinear Optimization.
- We always welcome new students to join our lab.

Software and research projects

- LIBSVM : a simple and easy-to-use support vector machines tool for classification (C-SVC, nu-SVC), regression (epsilon-SVR, nu-SVR), and distribution estimation.
  It includes a GUI for both classification and regression. Version 1.0 released in April 2000. NEW Current Version: 3.18, April 2014.
  See software document published in ACM TIST: pdf, ps.gz, ACM digital lib.
- LIBLINEAR: a library for large linear classification. It is very suitable for document classification. Version 1.0 released in April 2007. NEW Current Version: 1.94, November 2013.
- LibShortText: a library for short-text classification and analysis. Version 1.0 released in February 2013. NEW Current Version: 1.1, September 2013.
- LIBMF: LIBMF: A Matrix-factorization Library for Recommender Systems Version 1.0 released in September 2013. NEW Current Version: 1.1, April 2014.
- BSVM : a simple decomposition method for support vector machines for large classification.
  It has faster convergence in difficult cases. Version 1.0 released in January 2000. NEW Current Version: 2.08, June 2012.
- NMF : a simple and easy-to-use MATLAB code for Nonnegative Matrix Factorization.
  Version 1.0 released in October 2005. NEW Current Version: 1.01, December 2005.
- TRON: Trust region Newton Method for large bound-constrained optimization, released June 1999. We also provide an AMPL interface in NEOS.
- Other miscellaneous software

炼数成金逆向收费式网络课程

- Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。
- 关于逆向收费式网络的详情，请看我们的培训网站 http://edu.dataguru.cn

Thanks

FAQ时间