[Perceptron 1](#_Toc12920)

[1.1 perceptron model 1](#_Toc17694)

[1.2 perceptron learning strategy 2](#_Toc14209)

[1.3 perceptron algorithm 3](#_Toc22022)

[1.4 perceptron algorithm by python 3](#_Toc31578)

[1.4.1 algorithm overview 3](#_Toc8939)

[1.4.2 generate training data 4](#_Toc22269)

[1.4.3 Perceptron Class 5](#_Toc9762)

[1.4 perceptron algorithm convergence 6](#_Toc29587)

[1.4 perceptron dual form 6](#_Toc9167)

# Perceptron

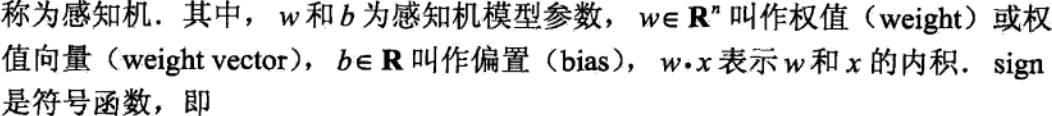
## 1.1 perceptron model

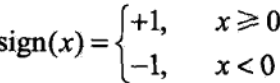
感知机是**二分类的线性分类模型**，其输入为实例的**特征向量**，输出为实例的类别取+1、-1。感知机将输入空间分为正负两类的**超平面**，称为**判别模型**。感知机的学习目的在于求出最佳超平面，由此导入基于误分类的**损失函数**。利用**随机梯度下降法**对损失函数进行最小化，求得感知机模型。

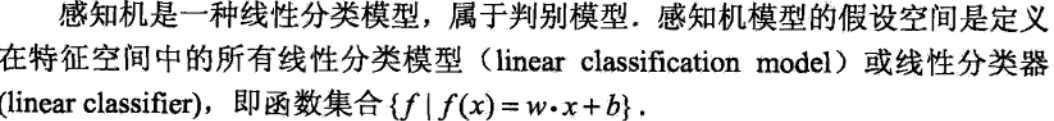
感知机是神经网络与支持向量机的基础。

定义 （**感知机**）： 假设输入空间(特征空间)是，输出空间是Y={1，-1}。输入表示实例的特征向量，对应于输入空间（特征空间）的点；输出表示实例的类别。由输入空间到输出空间的如下函数

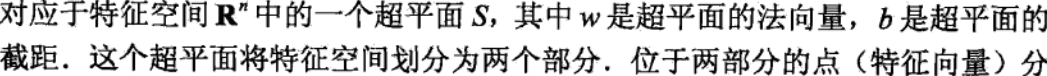
f(x)=sign(w.x+b)



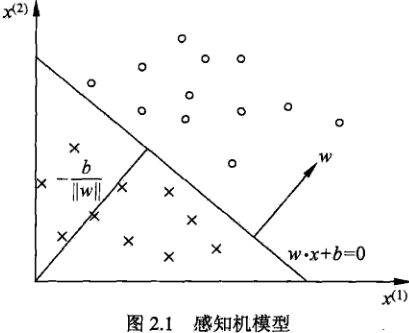




感知机的几何解释：线性方程w.x+b=0

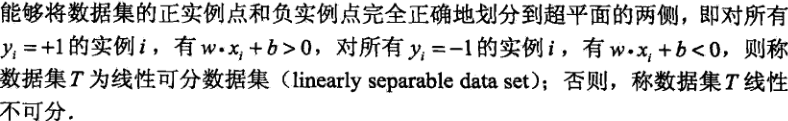




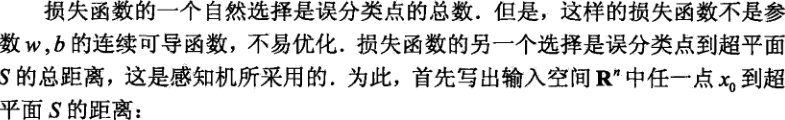


## 1.2 perceptron learning strategy

定义：**数据集的线性可分** 对于一个给定的数据集



假设数据是线性可分的，那如何找到这样一个超平面呢？即确定感知机模型参数w、b。

由此我们需要一个学习策略，即定义（**经验**）**损失函数**并将损失函数最小化。





误分类数据（本来y=1结果却识别为了-1，而-1识别为了1）





这样，假设超平面S的误分类点集合为M，那么所有的误分类点到超平面S的总距离为，不考虑，就得到感知机学习的损失函数.

给定数据集



损失函数定义为：

其中M为误分类点的集合，这个损失函数就是感知机学习的**经验风险函数。**

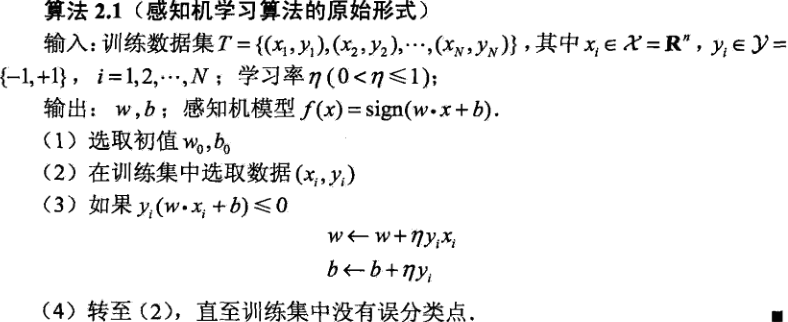
## 1.3 perceptron algorithm

感知机学习问题转化为了求损失函数的最优化问题。感知机学习算法是误分类驱动的，于是考虑使用，随机梯度下降法。

随机梯度下降法：

1. 随机选择一个超平面
2. 计算梯度
3. 

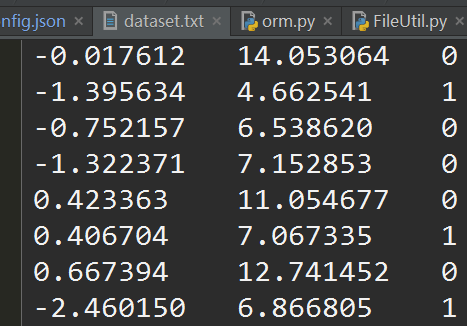
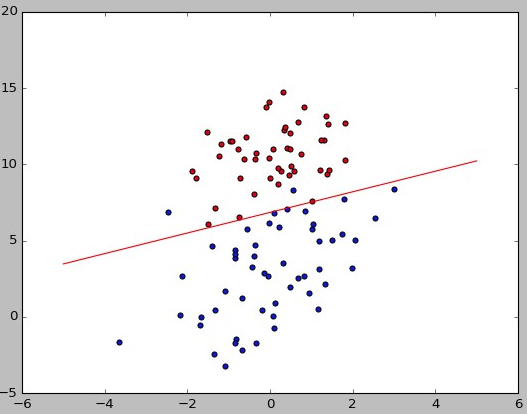


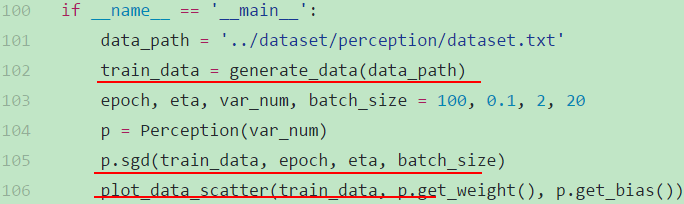


## 1.4 perceptron algorithm by python

### 1.4.1 algorithm overview

问题：使用感知机做二分类问题，数据集如下前两列为维数，最后一列为分类结果(为方便在准备数据的阶段，把0转化为了-1)。右图中红蓝点为数据的散点图，而红色的直线代表感知机生成的划分平面。

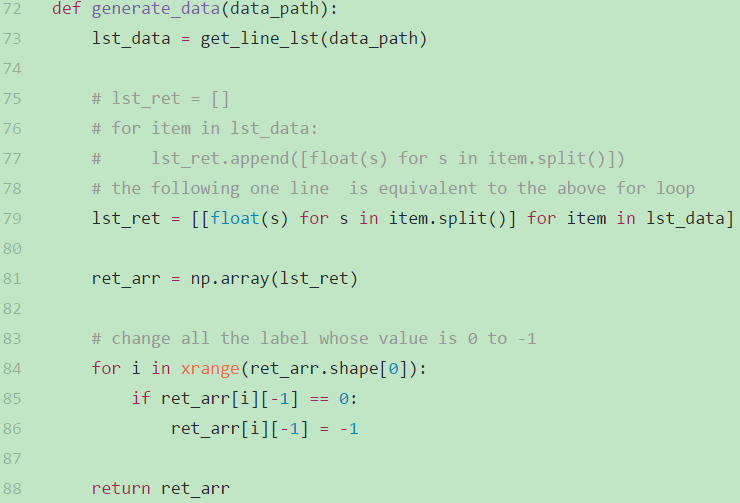
 

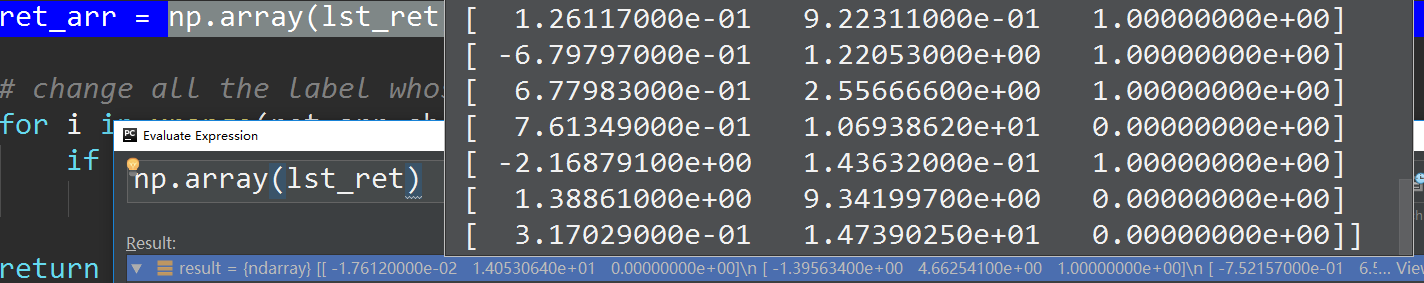


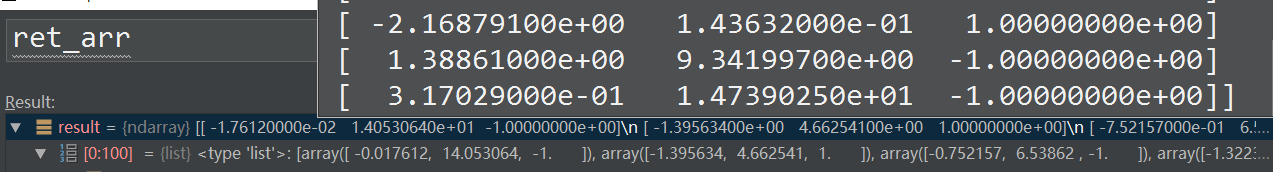
代码的总体思路：

1. 从dataset.txt读取数据，生成训练数据形式
2. 生成感知机类，以及随机梯度下降法参数，并训练网络
3. 画出训练数据集的散点图，以及最终生成的分类平面

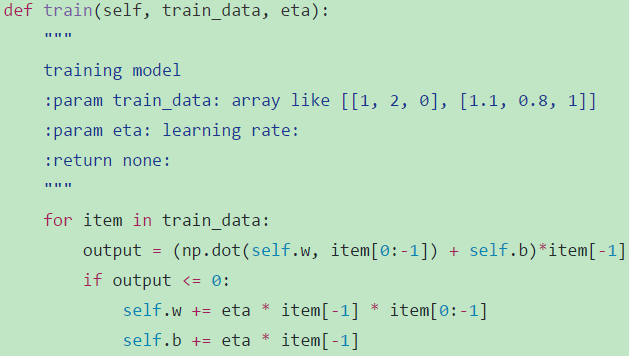
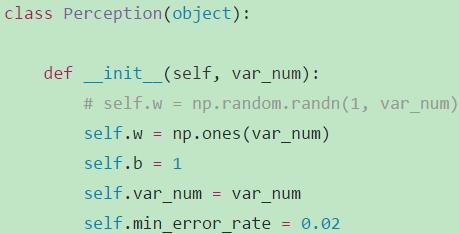
### 1.4.2 generate training data

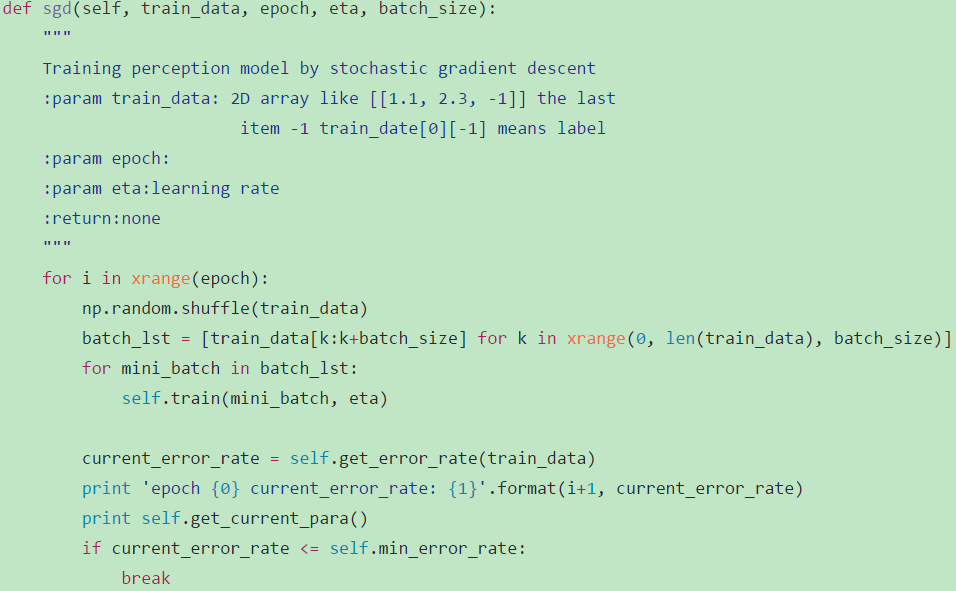


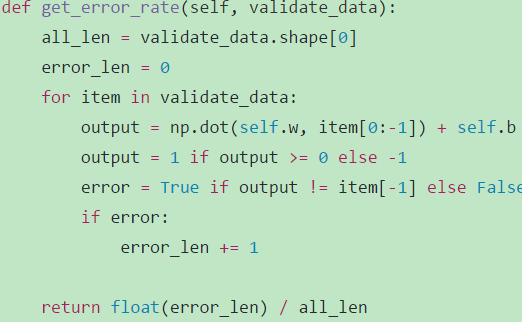




### 1.4.3 Perceptron Class







## 1.4 perceptron algorithm convergence

算法的收敛性证明：先略过。

## 1.4 perceptron dual form

感知机的对偶形式。

对偶的目的：让计算变得简单高效

# Principal Component Analysis

## 2.1 basic theory

PCA是一种无监督的学习方式，是一种很常用的降维方法。在数据信息损失最小的情况下，将数据的特征数量由n，通过映射到另一个空间的方式，变为k(k<n)。

### 2.1.1 Mathematical Background

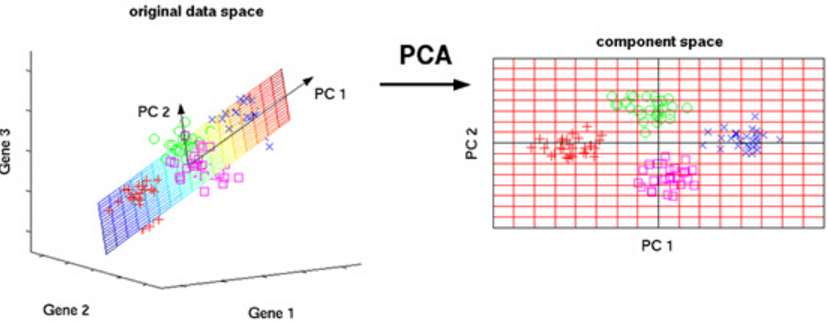
下表1是某些学生的语文、数学、物理、化学成绩统计：



首先，假设这些科目成绩不相关，也就是说某一科目考多少分与其他科目没有关系。那么一眼就能看出来，数学、物理、化学这三门课的成绩构成了这组数据的主成分（很显然，数学作为第一主成分，因为数学成绩拉的最开）。为什么一眼能看出来？因为坐标轴选对了！下面再看一组学生的数学、物理、化学、语文、历史、英语成绩统计，见表2，还能不能一眼看出来：



数据太多了，以至于看起来有些凌乱！也就是说，无法直接看出这组数据的主成分，因为在坐标系下这组数据分布的很散乱。究其原因，是因为无法拨开遮住肉眼的迷雾~如果把这些数据在相应的空间中表示出来，也许你就能换一个观察角度找出主成分。如下图1所示：



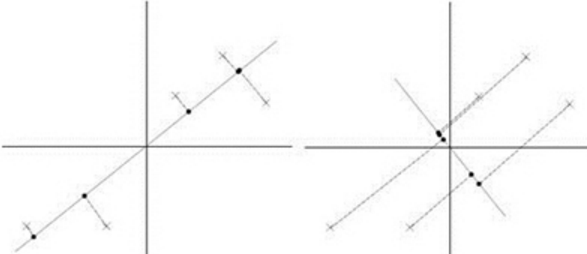
但是，对于更高维的数据，能想象其分布吗？就算能描述分布，如何精确地找到这些主成分的轴？如何衡量你提取的主成分到底占了整个数据的多少信息？所以，我们就要用到主成分分析的处理方法。

### 2.1.2 max variance theory

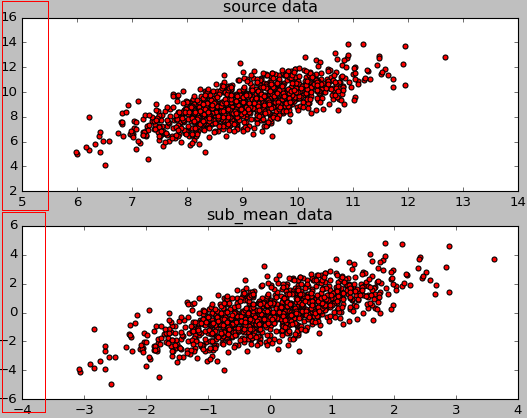
在信号处理中认为信号具有较大的方差，噪声有较小的方差，信噪比就是信号与噪声的方差比，越大越好。如前面的图，样本在u1上的投影方差较大，在u2上的投影方差较小，那么可认为u2上的投影是由噪声引起的。

因此我们认为，最好的k维特征是将n维样本点转换为k维后，每一维上的样本方差都很大。

比如我们将下图中的5个点投影到某一维上，这里用一条过原点的直线表示（数据已经中心化）：



假设我们选择两条不同的直线做投影，那么左右两条中哪个好呢？根据我们之前的方差最大化理论，左边的好，因为投影后的样本点之间方差最大（也可以说是投影的绝对值之和最大）

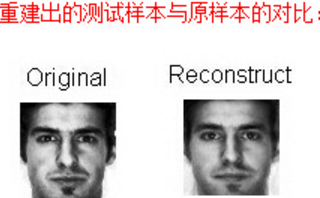


可以看到原始数据，以及求取平均值然后减去平均值，数据集相当于向下平移。

### 2.1.3 practical example

PCA将n个特征降维到k个，可以用来进行数据压缩，例如100维的向量最后可以用10维来表示，那么压缩率为90%。同样图像处理领域的KL变换使用PCA做图像压缩，人脸检测和匹配

...



可见测试样本为人脸的样本的重建误差显然小于非人脸的重建误差。

## 2.2 algorithm step

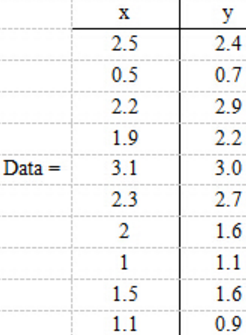
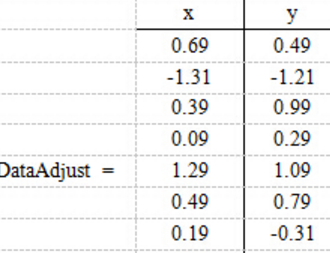
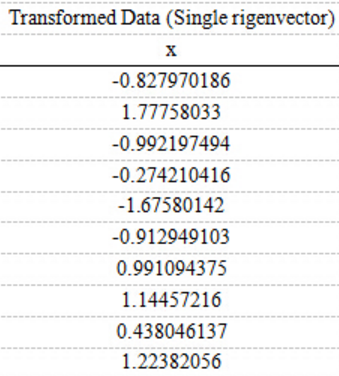
### 2.2.1 overview

The overview step :

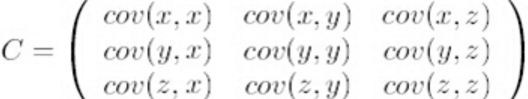
1. Remove the mean
2. Compute the covariance matrix
3. Find the eigenvalues and eigenvectors of the covariance matrix
4. Sort the eigenvalues from largest to smallest
5. Take the top N eigenvectors
6. Transform the data into the new space created by the top N eigenvectors

下面给个简单大概的示例：

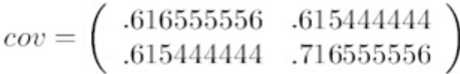
第一步，分别求x和y的平均值，然后对于所有的样例，都减去对应的均值。这里x的均值是1.81，y的均值是1.91，那么一个样例减去均值后即为（0.69,0.49），得到

第二步，求特征协方差矩阵，如果数据是3维，那么协方差矩阵是

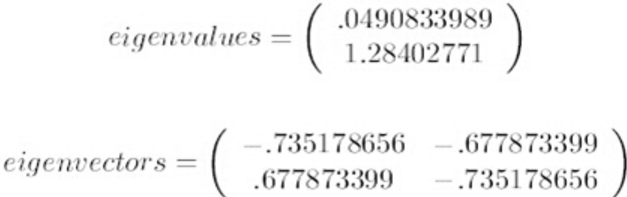


这里只有x和y，求解得



对角线上分别是x和y的方差，非对角线上是协方差。协方差是衡量两个变量同时变化的变化程度。协方差大于0表示x和y若一个增，另一个也增；小于0表示一个增，一个减。如果ｘ和ｙ是统计独立的，那么二者之间的协方差就是０；但是协方差是０，并不能说明ｘ和ｙ是独立的。协方差绝对值越大，两者对彼此的影响越大，反之越小。协方差是没有单位的量，因此，如果同样的两个变量所采用的量纲发生变化，它们的协方差也会产生树枝上的变化。

第三步，求协方差的特征值和特征向量，得到



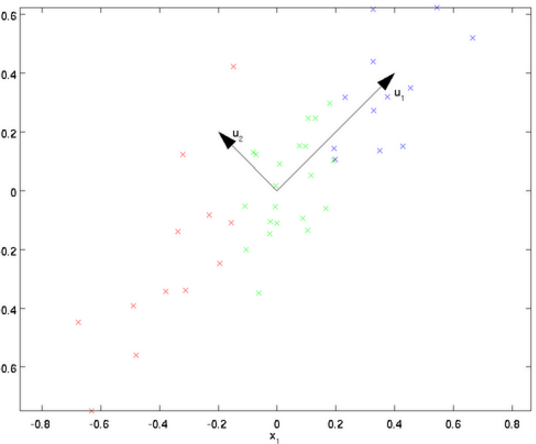
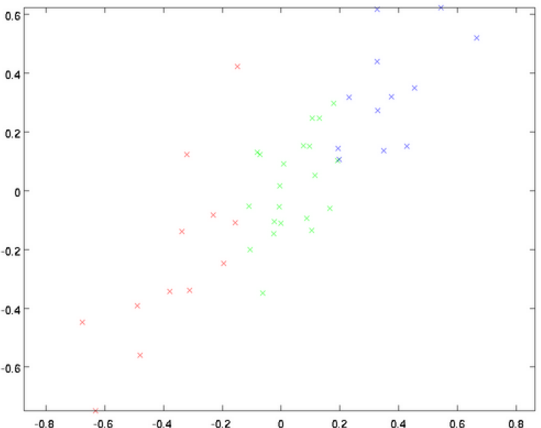
第四步，将特征值按照从大到小的顺序排序，选择其中最大的k个，然后将其对应的k个特征向量分别作为列向量组成特征向量矩阵。

这里特征值只有两个，我们选择其中最大的那个，这里是1.28402771，对应的特征向量是(-0.677873399, -0.735178656)T。

第五步，将样本点投影到选取的特征向量上。假设样例数为m，特征数为n，减去均值后的样本矩阵为DataAdjust(m\*n)，协方差矩阵是n\*n，选取的k个特征向量组成的矩阵为EigenVectors(n\*k)。那么投影后的数据FinalData为

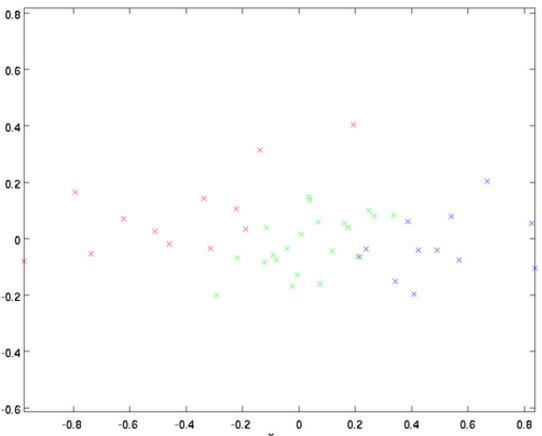
FinalData(10\*1) = DataAdjust(10\*2矩阵) \* 特征向量(-0.677873399, -0.735178656)T

得到的结果(见上面的图左右边的)

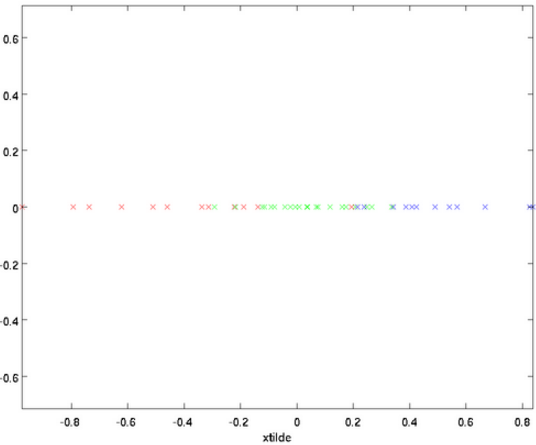
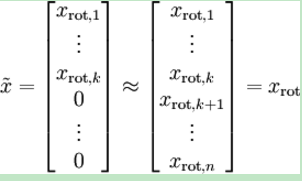


Rotating the Data

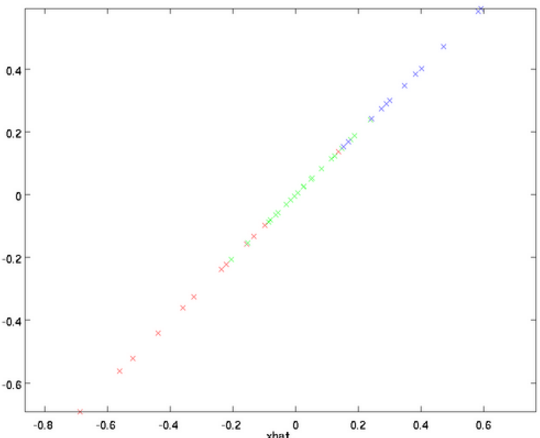
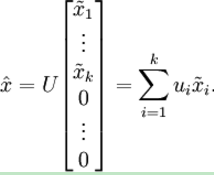
Thus, we can represent x in the  basis by computing



Reducing the Data Dimension



Recovering an Approximation of the Data



## 2.3 python code

### 2.2.1 overview

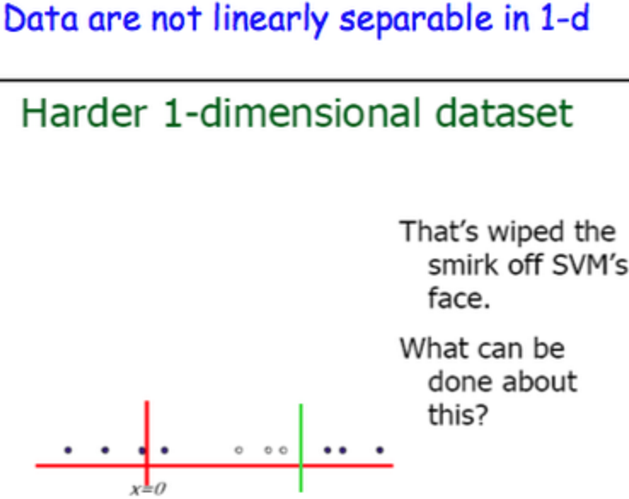
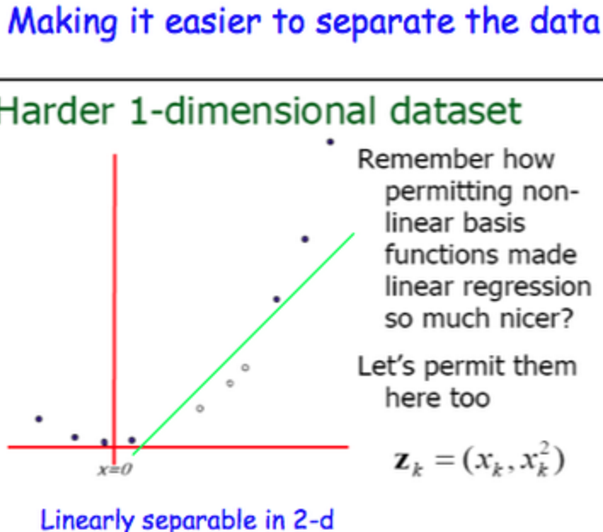
### 2.2.2 load\_data

### 2.2.3 overview

# SVM

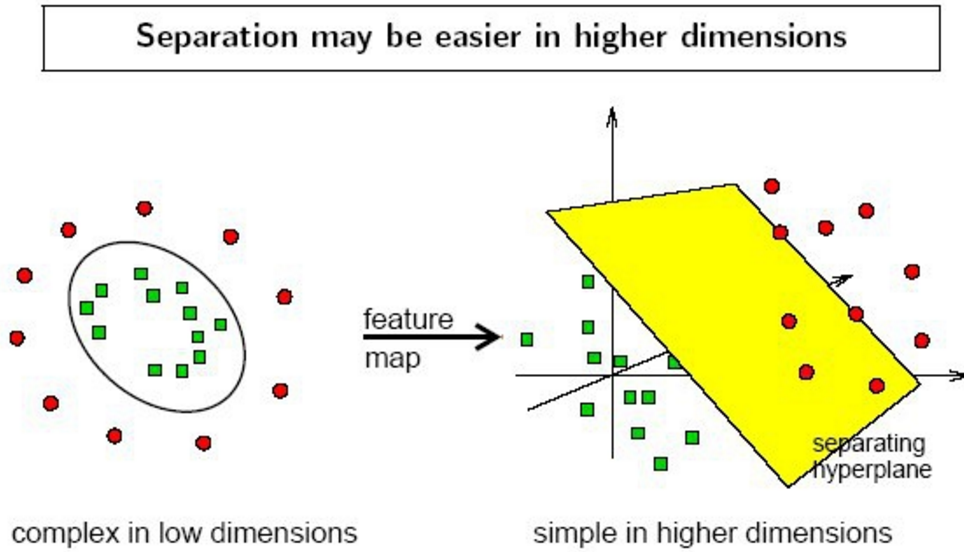
## 2.1 kernel function

比如我们有一个一维的数据分布是如下图的样子，你想把它用一个直线来分开，你发现是不可能的，因为他们是间隔的。所以不论你画在哪，比如绿色竖线，都不可能把两个类分开。

但是我们使用一个简单的升维的方法，把原来一维的空间投射到二维中，x->(x, x^2)。比如: 0->(0,0) 1->(1,1) 2->(2,4) 这时候就线性可分了

再举个例子，在一个二维平面里面，这样的情况是不可能只用一个平面来分类的，但是只要把它投射到三维的球体上，就可能很轻易地分类。



理论上，由于train set是有限的，当你把data投射到无限维度的空间上是一定可以在train set上完美分类的，至于在test set上当然就呵呵了。记得要选取合适（试试各种）kernel function来“避免过拟合”。

很多算法，都是要基于一个“距离”的概念。比如：层次聚类要先聚离得近的点，然后再逐渐把更远的点往里聚。支持向量机要找一个东西，让两组点的边界离得最远。那么问题就来了：什么叫距离？

你可以简单地使用两点之间几何距离，就是sqrt(x2 + y2 + z2 + .....)。但是有的时候，用别的距离定义会有特别的效果。**定义这个距离的东西，就叫核函数。**

核函数只是满足某些必要条件的函数，其作用要与具体的算法结合才能显示出来。我来简明说一下SVM中核技巧(kernel trick)的作用，一句话概括的话，就是降低计算的复杂度，甚至把不可能的计算变为可能。

核函数有如下一个性质：

其中是对x做变换的函数，有些变换会将样本映射到更高维的空间，如果这个高维空间内与是线性可分的，那么我们就做了一次成功的变换。核函数是二元函数，输入是变换之前的两个向量，其输出与两个向量变换之后的内积相等（这个性质非常重要）。

而我们知道，求解SVM时，其原始形式(这里我们假设已经对原始的输入做了变换，即输入模型的样本变成了。

如果我们把原始样本从十维空间变换到一万维的空间，那么求解该问题的时间复杂度提升了1000倍或者更多，我们知道有些变换可以将样本换边到无穷维空间，那么这种变化之后直接是不可求解的。

使用核技巧之后，学习是隐式地在特征空间进行的，不需要显式地定义特征空间和映射函数(李航)

