教你Machine Learning玩转金融，从入门到放弃（下）

对冲笔记 2017-06-24

（接上篇）

2.4 Classification

接下来我们要介绍的就是Classification了。
classification跟regression的区别就是Y的类型不同，regression是Y的具体数值的预测，比如涨跌幅度，而classification是对于单纯分类的预测，比如Y的涨(Y值取1)或者跌(Y值取0)。

但是classification里面有一个地方不好处理，那就是如果想要预测的Y不仅仅只有两个分类怎么办，比如当Y有三个值甚至更多。因为如果大于两个值的话就不能简单将Y设置为0，1，2，但是Y的这三个分类值在实际的意义上不一定是等距的，如果我们将其设置成0，1，2的话是默认这三个分类之间的距离相等，但在实际过程中Y不同分类结果之间的距离很难用数字去衡量。
为了简化，此处只考虑Y有两类: 是(Y=1)或否(Y=0)的情况。

logistic regression

而classification里面最出名的一个方法就是logistic regression。这里不要被regression的名字所误导，之所以取的是regression的名字是因为用的是类似regression的方法，但是由于这里Y是离散的(不是具体连续的数值，而是比如说二选一，涨或者跌)，所以对于Y的处理上有一些变化。

下面就是logistic regression的公式推导:

Mathematical Model for Logistic Regression

Consider a binary classification problem, where $y \in \{0,1\}$. Elements of our training set of size m, can be referenced as $(\underline{x}^{(i)}, y^{(i)}), \forall i \in \{1, \dots, m\}$. In logistic regression, we use the sigmoid (or logit) function $g(z) = \frac{1}{1+e^{-z}}$ to define $h_{\underline{\theta}}(\underline{x}) = \frac{1}{1+e^{-\underline{\theta}^T \underline{x}}}$.

Here, we use the probabilistic approach to model $P(y = 1 | \underline{x}; \underline{\theta}) = h_{\underline{\theta}}(\underline{x}) \in [0,1]$, and $P(y = 0 | \underline{x}; \underline{\theta}) = 1 - h_{\underline{\theta}}(\underline{x})$. In other words, we model as $Y \sim Binomial(m, h_{\underline{\theta}}(\underline{x}))$.

This yields a likelihood function of $L(\underline{\theta}) = p(y|X; \underline{\theta}) = \prod_{i=1}^{m} \left(h_{\underline{\theta}}(\underline{x}^{(i)}) \right)^{y^{(i)}} (1 - h_{\underline{\theta}}(\underline{x}^{(i)}))^{1-y^{(i)}}$. Optimizing the log-likelihood $l(\underline{\theta}) = \log L(\underline{\theta})$ leads to the LMS rule

$\theta_j \leftarrow \theta_j + \alpha (y^{(i)} - h_{\underline{\theta}}(\underline{x}^{(i)})) x_j^{(i)}$. In practice, implementations try to solve for $l'(\underline{\theta}) = 0$ using the Newton-Raphson method as

$\underline{\theta} \leftarrow \underline{\theta} - H^{-1} \nabla_{\underline{\theta}} l(\underline{\theta})$,

where H is the Hessian matrix defined as

$H_{ij} = \frac{\partial^2 l(\underline{\theta})}{\partial \theta_i \partial \theta_j}$.

This implementation of logistic regression involving the Hessian matrix is called Fisher Scoring.

Other points to note are:

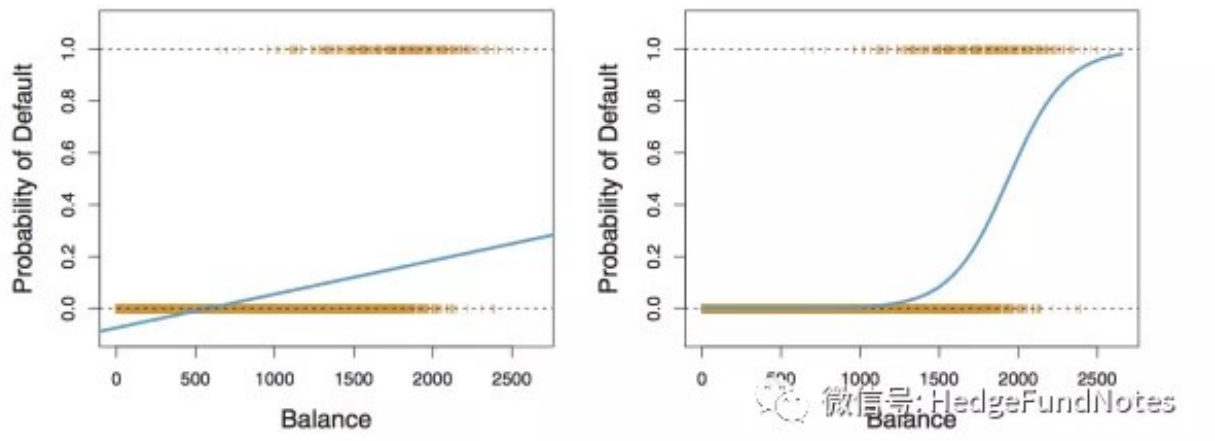
- Extension to $K \neq 2$ is available through soft-max regression.
- Probit regression is obtained by modifying the modeling assumption to $Y \sim Binomial\left(m, \Phi(\underline{\theta}^T \underline{x})\right)$, where $\Phi(.)$ is the cumulative distribution function for the N(0,1) distribution.
- In practice, it is common to use L1 or L2 regularization with logistic regression.

是不是看一圈之后懵逼了~

这里用的是maximum likelihood function的推导，因为涉及到了矩阵运算，所以这里为了让读者更轻松一点暂时省去~

我来用另一种不是那么准确但是更容易接受的方法解释一下，举个例子就明白了。

先上一张图，



我们现在呢，手上有信用卡的数据，知道每一个人有没有违约还有他们信用卡欠款的数目，想要研究的问题呢就是违约概率(Y)和信用卡欠款数目(X)之间的关系。因为这里一个人是不是违约只有是或者否两种结果，不可能存在第三种情

况，所以正好适合我们这里分类classification的问题。

左边的图呢，那一条蓝线就是直接用linear regression来fit Y和X的结果。公式如下：

也就是基于一个人的信用卡欠款X，我们想要预测的这个人违约的概率p(X)。这里由于是用的regression，所以p(X)是一个连续的数值。

我们从上图中可以看到明显这个model对于实际非常不符合。
一个是可以看到预测非常不准，即使我们将纵轴probability of Default>0.5看做是，probability of Default<0.5看做否，这个fitting的结果在balance的整个横轴跨度内基本上都是属于否这一类。但实际上从图中我们可以看到当balance这个横轴数值在2000左右的时候，其纵轴probability of Default大部分其实是1，也就是会违约。
另一个问题就是由于linear model是一条线，所以这条线对于横轴balance上的某些点，会使得纵轴probability of Default甚至有小于0和大于1的情况。我们这里讨论的是概率，只会属于0和1之间，所以小于0和大于1的预测在这里明显是不对的。

所以为了解决这两个问题，我们可以想办法把左边图的那条linear曲线做一定的变换mapping到另一种形式，使得其只属于0到1这个区间。

我们需要经过两个步骤:
首先呢，我们要想办法把linear function的$\beta 0+\beta 1\cdot X$的值(负无穷到正无穷)，变成全部是正数，也就是从0到正无穷。有些读者肯定想到了，那就是利用指数函数exponential function。
将其变为$e^{(\beta 0+\beta 1\cdot X)}$，这样所有的结果都会是0到正无穷。

但是这样还不够，因为我们想要的是将其压缩在0到1之间。所以我们可以同时对上面linear model的右边p(X)进行如下操作:

$$\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta_1 X}.$$

这样就保证了p(X)只会属于0到1这个区间。如果对于这个推导过程不熟悉的，可以自行任意带入负无穷到正无穷间的值取代上述方程中β0+β1·X的位置，检查概率p(X)是否都是落在0到1这个区间。如果发现不是，那么恭喜你，你颠覆了数学，我请你吃饭，管饱~

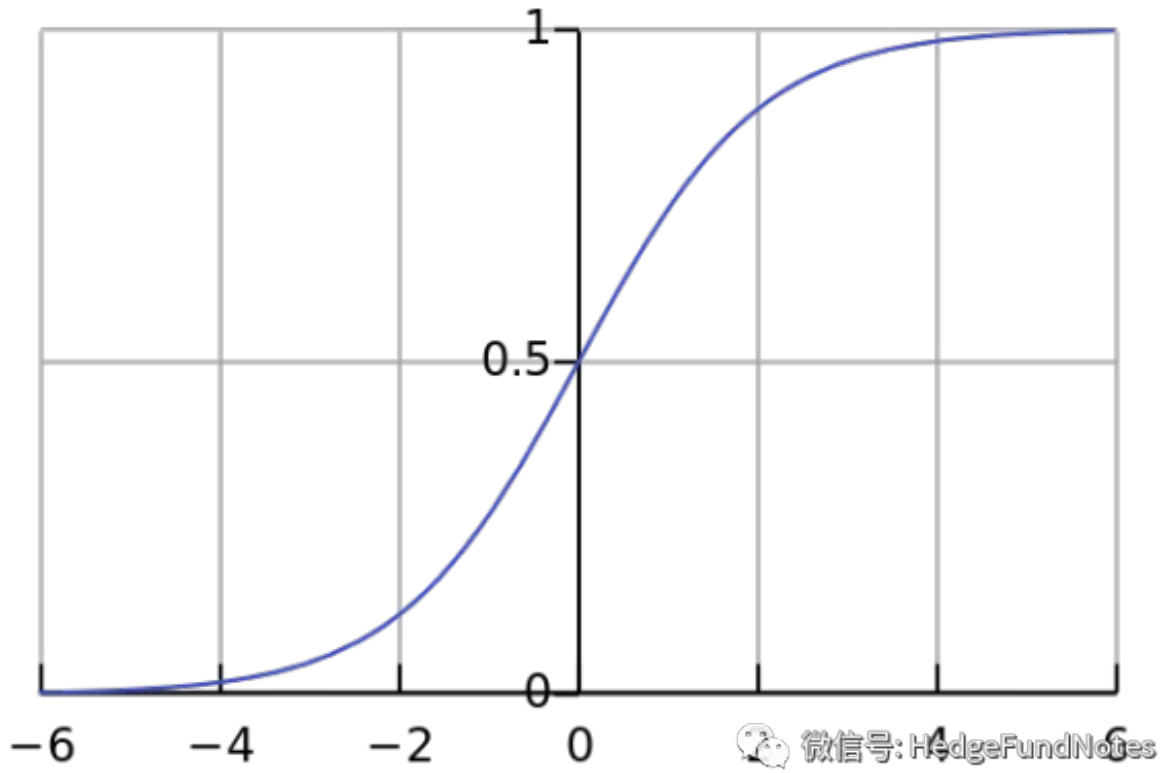
至于为什么叫logistic　regression，看下面这个式子就明白了(完全等同于上一个式子):

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}.$$

然后我们可以看到数学中标准的logistic function的形式是这样:

$$f(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x}$$

它的图形是长这样:



是不是发现正好符合我们想要的需求呢，随着横坐标X的变化，Y永远都被限制在0和1之间。
并且我们把标准的logistic function中的x换成我们的linear function里面的β0+β1·X之后，就完全是我们推导出来的结果。一颗赛艇！
所以这里这个方法是把linear function的内容β0+β1·X嵌入到了标准logistic function里面，自然这个方法就叫logistic regression了。

经过上面的介绍和推导，我们用logistic regression的时候呢也只会关注这个最终形式:

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}.$$

我们输入进去的数据是因子X和相应的Y(也就是是1或者否0)，fitting的结果得到的是β0和β1。这样利用这个model，如果有任何的X，我们就可以预测出p(X)的值。如果当概率p(X)>0.5的时候我们就说Y属于"是"，当概率p(X)<0.5的时候我们就说Y属于"否"。

至于怎么得到最合适的β0和β1，其实就是上面那个看得懵逼的maximum

likelihood的推导图。如果大家还有印象的话，前面linear regression确定参数时候用的是least square approach得到linear model的各个系数β0，β1，β2...，其本质上是maximum likelihood的一种特殊情况。区别就是least square approach比较好理解和表示，而变成non-linear model的情况需要变换成较为抽象的矩阵运算，也就是maximum likelihood的推导。

接下来我们看一个金融中实际应用的例子:

这里JP Morgan用到的是期权(后面我会详细介绍期权是一个什么东西和volatility surface，如果想先弄清楚可以先跳到最后的科普期权部分)，内容是对于某个股票一直sell 1M ATM(At The Money)的call，并且不断rolling到下一个月。
因为是sell option，所以可以一直收取premium拿到期。至于为什么是ATM的call，因为在at-the-money的时候，时间价值是最juicy的，也就是自己overwrite的时候可以收的premium会最大。如下图我们可以看到100% at-the-money的时候是最大化收益的。除之此外，我们还可以看到一个有意思的现象，整体来说102%的收益比98%的低，105%的比95%的低，原因是volatility surface其中一个维度的volatility skew现象导致在相同程度的moneyness情况下，ITM会比OTM的implied volatility更大。还有一个现象就是102%的比98%波动更大，105%的比95%波动更大，原因是ITM的已经包含了intrinsic value增加了缓冲区，但trade-off就是虽然ITM的会风险更小但是upside potential也会受到限制(我们可以看到102%和105%一开始其实是比98%和95%收益更高的，但当掉下去的时候也掉的更多，所以OTM的波动更大)。

Exhibit 5: Daily Rolls of One Week Options – Strikes: 95%, 98%, 100%, 102 & 105%

但是JPMorgan取的是1M的rolling，事实上tenor最小的weekly maturity会效果最好(如下图)，不知道为什么不用。具体的解释是跟volatility surface的另一个维度term structure有关了，最后期权科普部分会详细介绍。

Exhibit 7: Daily Rolls of ATM Options - 1W, 1M, 3M Maturities

Overall IRR	
S&P TR	2.66%
Index 1W	9.03%
Index 1M	8.65%
Index 3M	7.23%

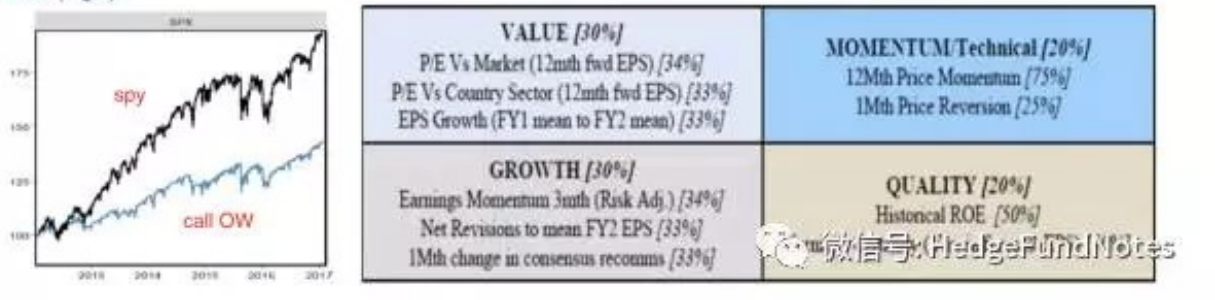
Flat Market	Bullish Market	Bearish Market	Very Bullish Market	
8.14%	14.65%	-33.76%	38.17%	S&P TR
11.73%	11.16%	-8.23%	25.02%	Index 1W
11.43%	10.70%	-12.34%	30.54%	Index 1M
10.53%	9.06%	-19.17%	35.98%	Index 3M

VIX Levels				
23.81	24.17	80.86	52.65	Max
13.67	13.04	31.22	26.23	Average
10.23	9.89	16.12	16.12	

虽然有那么多好处，但由于是单边卖期权，所以存在的风险就是股票的价格会往反方向走。比如说这里如果股票价格一直猛涨的话，还这么卖期权会死的很惨。所以风险就在于预测这个股票的表现强弱。

当然这里衡量股票强弱并不是直接预测股票的价格，而是JPM自己通过各个factor综合构造的一个分数。
如下：

Figure 56: Performance of call OW strategies vs underlying equity indices (left), 10-factor model along with weights as used in JPM's Q-Score Model (Right)



左边的图就是这个“call OW策略” VS sp500大盘的ETF(spy)的表现。

有些人可能会问了，这策略表现这么差，有啥用?
股票不一定都会疯涨嘛，总会有不同sector轮转的时候。而且要知道现在美股是超级大长牛，但是股票不会涨上天对吧，总会有经济周期超过均衡太多而掉下来的时候。所以如果我们能用这个策略无缝切换的话那不是就都可以抓住了么?
(当然，比较困难...)

右边的图就是形容JMP是怎么给股票打分来决定其表现好坏的。我们可以看到一共分为4大块儿，Value，Momentum/Technical，Growth和Quality，分别在这个股票的得分里面有不同的权重。每一块儿呢又是由其下面的小factor通过不同的权重构成。

比如对于每一个股票我们有这些10个小factors的数值，按照他们自己的权重可以算出这4个大的factor的数值，然后再按照这些大的factor的权重算出来一个最终的得分作为衡量股票表现强弱的标准。应该是得分越高，这个股票的表现越好。但由于我们的策略是不希望股票的表现过于强势，所以当股票表现差的时候会trigger我们的策略(否则就单纯long？或者加入其它的策略也可以)。

OK，现在开始解释怎么用logistic regression来做这件事。
我们首先需要弄清楚我们的Y和Xi分别是什么。这里对于Y呢，我们想要预测的是“call OW策略”是否优于单纯做多这个股票的概率，所以也就是当样本数据中“call OW策略”的收益高于单纯做多的时候Y的取值是1，低于单纯做多的时候Y的取值就是0。对于Xi，这里每个股票的这10个小的factor作为不同的因子Xi。
对于样本数据，由于金融的数据不是很多，所以想要减少model的对于数据本身的依赖程度variance，我们可以用前面bootstrap的方法来抽样得到很多个不同的sample dataset，每一个sample dataset都用logistic regression来拟合，自然每一个sample dataset我们都可以得到一个logistic regression的方程，也就是每一个sample dataset我们都可以得到一套这10个因子的系数，即他们

每个因子分别对每一个logistic regression预测概率的贡献程度(感觉自己好啰嗦，这样的解释比较白话了吧...)。最终的结果就是对这些不同sample dataset拟合的model的结果取平均值。(如果对研究某一个系数感兴趣就是将这个系数取平均值，如果是对最终的预测概率Y感兴趣就用每一个sample dataset的预测概率Y取平均值，实际上也就是取感兴趣东西的distribution的mean)
对于系数的理解可以参照linear regression，系数的正负就是这个因子对于预测结果的影响方向，系数的绝对值大小就是其影响的大小。

bootstrap其实是为了多次重复试验获取Y或各个因子前面系数的distribution(或者quantile，因为distribution本身就含有quantile的信息)，也可以用来降低variance。因为是很多次重复试验的平均值，因为bootstrap之后获得分布的均值会比用全sample仅做一次regression的结果的variance更小。简单来说，就是做无数次实验，然后每一次实验都得到一个结果，将这些结果全部扔进histogram里面，就变成了这个想要观察结果的分布图(这个结果可以是Y，也可以是各个因子前面的系数，下面Figure 57里面10个因子的系数应该就是取平均值之后的结果)。

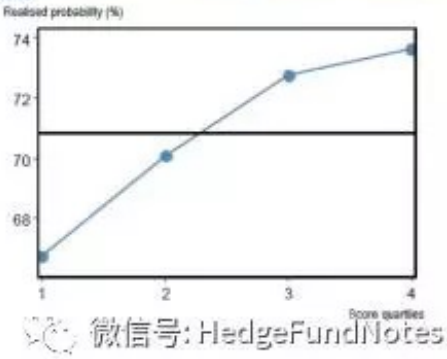
下图结果的意思是我们需要避免3M Realized Volatility这个因子高的股票，但是Earnings Certainty和12M Price Momentum这两个因子是对股票有利的。

然后通过logistic regression，我们就通过这10个Xi和其系数可以预测出来“call OW策略”的收益优于单纯做多这个股票的概率。并且可以自己自行定义，当概率超过某一个值(不一定是50%，可以更严格比如80%，90%都可以，其实这个也可以用统计的方法来求得最优的一个benchmark)的时候trigger这个“call OW策略”，如果小于这个概率值的话就单纯做多这个股票。

Figure 57: Logistic Regression: Coefficients and Z-value (left), Model score from logistic regression (into quartiles) vs. realized probability of call OW outperforming the equity index; Out-of-sample results covering calendar year 2016

Factor	Coef Estimate	Z-value
3M Realised Volatility	-0.36	-6.1
Historical ROE	-0.06	-1.5
1M Price Momentum	-0.05	-1.2
1Y Earnings Yield vs. Country Sector	-0.08	-1.2
EPS Growth	-0.05	-1.0
1Y Earnings Yield	-0.05	-0.8
Earnings Momentum 3M	-0.03	-0.5
Net revisions to Mean FY2 EPS	-0.02	-0.4
1M Change in Consensus Recs	0.00	-0.1
Earnings Certainty	0.01	0.2
12M Price Momentum	0.11	2.3

Source: J.P.Morgan Macro QDS



上图里面我其实有一个疑惑的地方，就是这个地方的score quantile(Model

score from logistic regression)应该跟前面提到的10个factor的Q-score是不一样的。因为上面右图中随着这个score的增大，“call OW策略”优于单纯做多这个股票的概率越大，也表示着预测的股票的表现会随着这个score的增大而越来越差才对。
但实际上Q-Score的值越高，预测的收益率就越高。
Q-Score里面也有我们用到的这10个相同的因子(因为这10个因子就是从Q-Score的model里面来的)，所以我们可以像前面Figure 56里表示的那样根据权重算出来那4个大的factor，再把这些大的factor结合起来变成一个综合的分数来衡量这个股票的强弱程度。但这里似乎没有用到这个方法来衡量股票的强弱。
我稍微搜了一下JPMorgan以前的报告，Q-score具体是长这个样子:

Q-Snapshot: Benetton Group S.p.A.

Quant Return Drivers (a Score >50% indicates company ranks 'above average')

Score 0% (worst) to 100% (best)	vs Country	vs Industry	Raw Value
Value			
P/E Vs Market (12mth fwd EPS)	74%	63%	0.9x
P/E Vs Sector (12mth fwd EPS)	90%	82%	0.6x
EPS Growth (forecast)	20%	11%	-11.3%
Value Score	75%	44%	
Price Momentum			
12 Month Price Momentum	64%	50%	-27.2%
1 Month Price Reversion	64%	50%	9.8%
Momentum Score	67%	51%	
Quality			
Return On Equity (forecast)	52%	20%	9.0%
Earnings Risk (Variation in Consensus)	83%	32%	0.13
Quality Score	63%	27%	
Earnings & Sentiment			
Earnings Momentum 3mth (risk adjusted)	23%	28%	-116.3
1 Mth Change in Avg Recom.	59%	51%	0.02
Net Revisions FY2 EPS	46%	52%	0%
Earnings & Sentiment Score	44%	49%	
COMPOSITE Q-SCORE* (0% To 100%)	**66%**	**38%**	

The higher the Q-Score the higher the one month expected return。

这里明显和Figure 57右图不符合。所以现在的问题就变成了Figure 57右图中的横轴"call OW score"是怎么计算的。但我也不知道...感兴趣的读者可以自行研究...可能在文章给的JPMorgan Market的link里，但如果没有access的就看不到那个报告了。

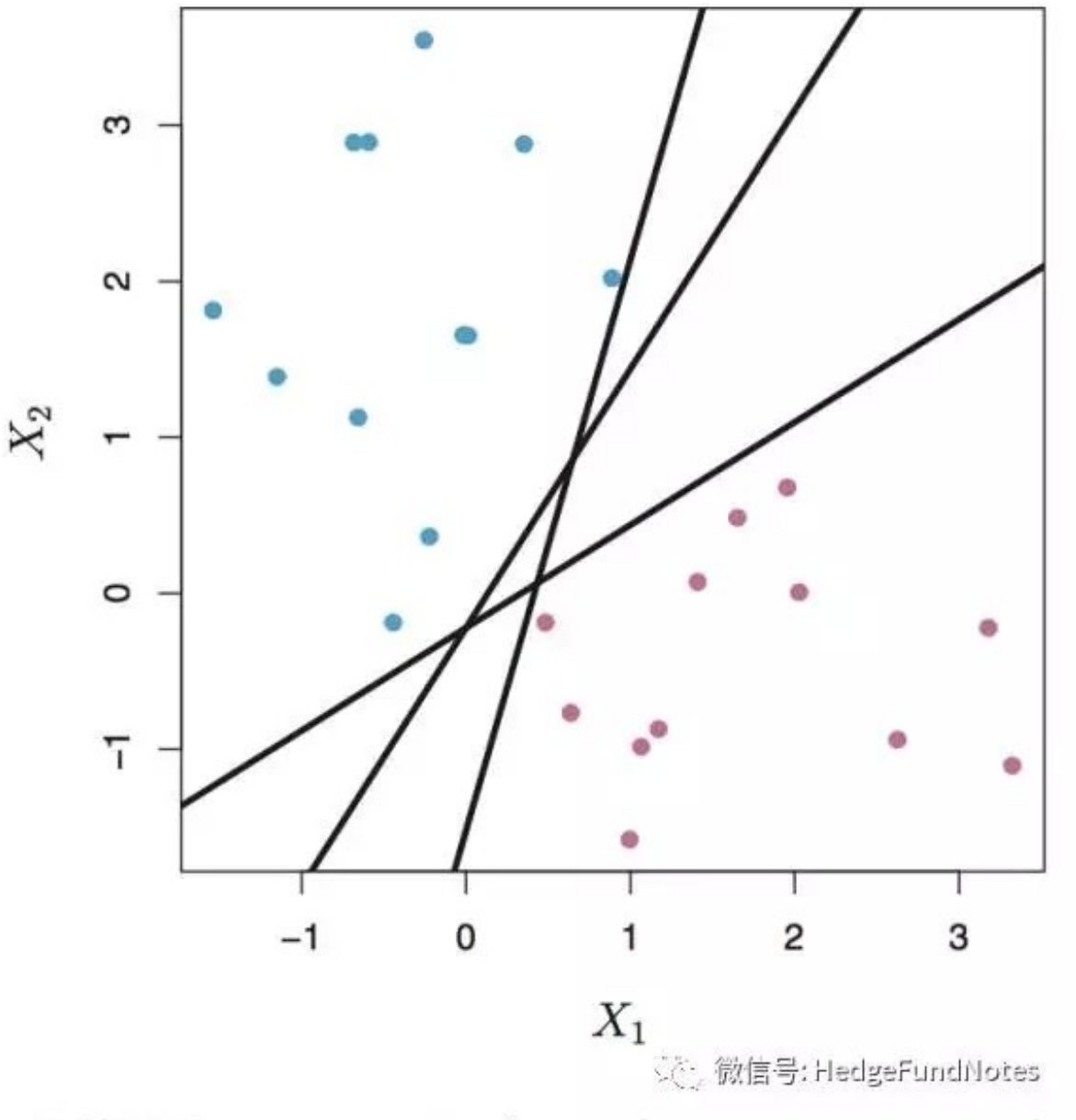
这大概就是怎么运用logistic regression来处理是和否的问题。

当然，任何模型都有其适用范围，logistic regression也不例外。
logistic regression的缺点就是当处理很多个变量Xi的时候，不能很好的解决Xi之间相关性的问题，这个需要很注意。

Support Vector Machine

接下来我们可以看看另一个classification的方法，就是很著名的Support Vector Machine(SVM)。

首先我们可以看一张图：



这张图表示的东西呢，就是我们有一堆观察点，每一个观察点都有Y和X1，X2的值。这里Y呢因为是classification，所以只有两个值，一个蓝色(我们分配的值给+1)，一个是红色(我们分配的值是-1)。X1的值就是横轴，X2的值就是竖轴。

所以每一个观察的点都有Y，X1，X2这三个属性，画在X1和X2的平面上面，Y的值用颜色来表示。
我们想要做的事情呢，就是用一条线把这些点区分开来(譬如图上的三条黑线都可以把蓝色的点和红色的点分开了)。由于这是二维平面上面的一条线，所以其表示的方程就是:

$$\beta 0+\beta 1\cdot X1+\beta 2\cdot X2 = 0$$

这一条线确定之后，系数β0，β1，β2也就确定了，以为其只是用来确定这一条线的系数，并且在这条线上面的点的坐标(X1, X2)都满足这个方程。

所以利用这样一条线，我们就可以用来区分这一条线两边的点了。
当一个点的坐标(X1, X2)满足 $\beta 0+\beta 1\cdot X1+\beta 2\cdot X2 > 0$ 的时候 ，就处于这条线的上方，也就是蓝色(Y=+1)。反之，如果满足 $\beta 0+\beta 1\cdot X1+\beta 2\cdot X2 < 0$ 的时候，这个点就在这条线的下方，也就是红色(Y=-1)。

这样我们就将这两种情况合并成一个式子来表示这条线两边所有点需要满足的条件:
$Y\cdot(\beta 0+\beta 1\cdot X1+\beta 2\cdot X2) > 0$

但是这里就出现了一个问题，图中有三条线，这三条线都可以将蓝色和红色的点分开，那我们怎么对比这三条线的好坏呢~
这个问题也很好解决，那就是我们希望这两边所有的点离这条线的距离尽可能的远，因为这样就说明这两边的点被这条线区分的最开，区分的结果自然也就更加可信，也就是使两边的点到这条线的垂直距离尽可能的大。

为了使得他们分得更开，我们可以将大于0这个条件更加严格一点，使得
$Y\cdot(\beta 0+\beta 1\cdot X1+\beta 2\cdot X2) > M$，
M是某一个正值，这样也就给两边都增加了缓冲区，使得两边的点距离这条线的距离全部都大于M，并且我们希望这个M的值越大越好。(因为M的值越大，说明两边离这条线的距离最近的点越远，也就保证了其他所有点都会离得更远)

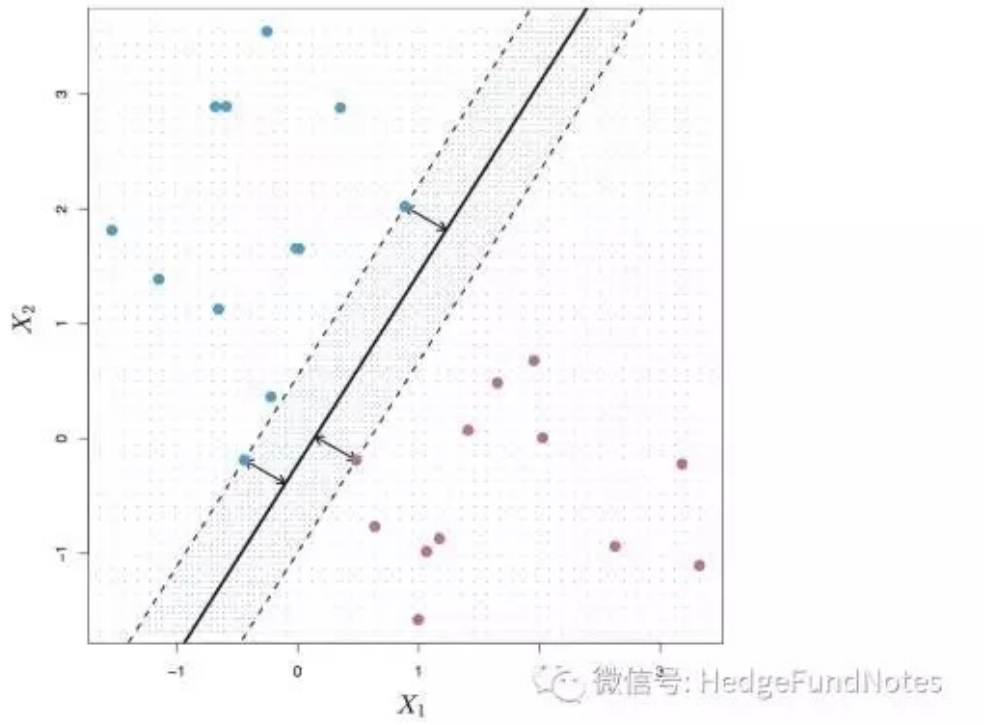
数学上的表达式是这样(这里i是指的每一个点，系数β的和被标准化为1，这样normalize之后可以和M值作对比):

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{maximize}}\, M$$

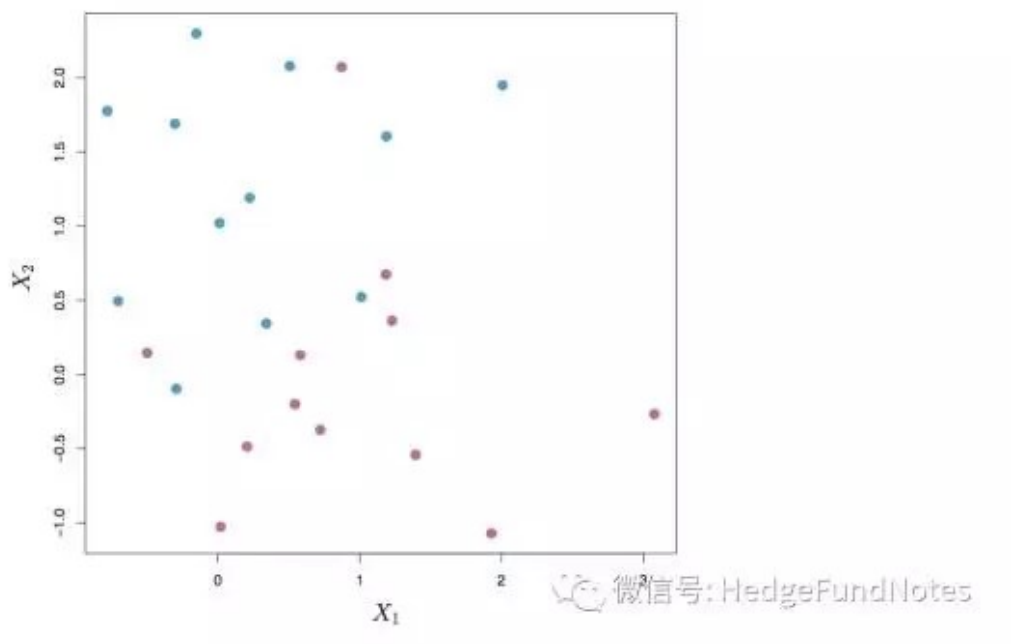
$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \ldots, n.$$

然后就变成了如下这张图，中间那条黑色的实线就是之前M=0的情况，而两条虚线就是方程加上某个正数M之后的效果((其实就是对中间这条黑线分别上下平移距离M)。还需要注意的是，这里M并不是我们人为给的一个值，而是上面的数学式对所有的点经过一定的optimization之后得到的结果，各个Xi系数β的值也是一样经过optimization之后的结果。



但是，不要以为这样就可以解决问题了，因为实际情况并非这么完美。很可能会产生如下的情况，蓝色和红色的点都混杂在一起，就无法再用一条直线完全将这些点都分开。



那我们怎么办呢?
这里我们就要进入真正的Support Vector Machine的内容了。

但具体应该怎么做呢?
也许就有人会想到，那如果我把M的值变成负数会怎么样?
这个其实也就是下面要说的，我们可以在上面的数学式子里面加入一定的错误率(也就是那些错误被线分类的点，反应在式子里面就是M加上一个负数)，也就是允许这条线两边的蓝色区域允许有红色的点，红色区域允许有蓝色的点，但是需要将这些错误率保持在自己能接受的一定的大小之内，毕竟我们希望其错误率越小越好(最理想的情况就是这条线两边的点完全没有错误率，也就是最开始完全分开的情况，但是现实中基本上不存在)，所以可以将其作为一个penalty ϵ加入上面数学的表达式中。

并且我们可以把这个penalty变得更加有效，使得其跟缓冲区结合起来。我们可以看如下的图，

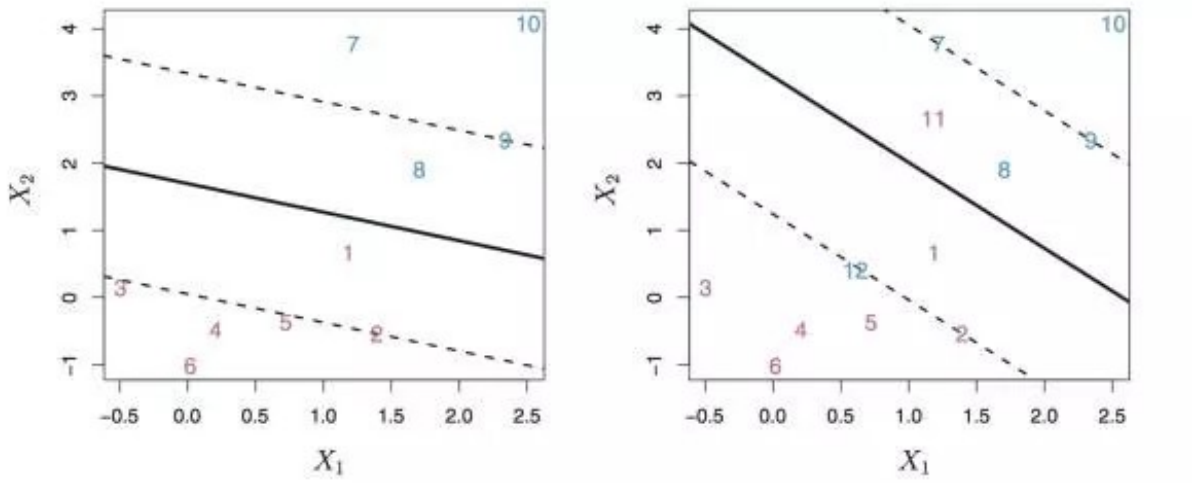


FIGURE 9.6. Left: *A support vector classifier was fit to a small data set. The hyperplane is shown as a solid line and the margins are shown as dashed lines.* Purple observations: *Observations* 3, 4, 5, *and* 6 *are on the correct side of the margin, observation* 2 *is on the margin, and observation 1 is on the wrong side of the margin.* Blue observations: *Observations* 7 *and* 10 *are on the correct side of the margin, observation* 9 *is on the margin, and observation* 8 *is on the wrong side of the margin. No observations are on the wrong side of the hyperplane.* Right: *Same as left panel with two additional points,* 11 *and* 12. *These two observations are on the wrong side of the hyperplane and the wrong side of the margin.*

上面的左图中，和前面一样，我们可以看到中间的那条实线代表的是M=0，虚线代表的是蓝色和红色两边的缓冲区。这样所有的点就被划分在了4个区域里面:
蓝色一边: 在缓冲区内，在缓冲区外
红色一边: 在缓冲区内，在缓冲区外
这样我们就可以给予不同区域的点不同的penalty的值ϵ。

因为这实际上和前面一样，都是optimization的问题，所以我们需要先给ϵ在不同区域内不同的loop范围求得其最佳值。
当蓝色的点在蓝色这一边的缓冲区之外，或者红色的点在红色这一边的缓冲区之外的时候，我们将ϵ的值设置为0(因为这是最理想的情况，没有penalty)。
当蓝色的点在蓝色这一边的缓冲区之内，或者红色的点在红色这一边的缓冲区之内的时候，我们将ϵ的值设置为$0<\epsilon<1$(因为在缓冲区内的话就开始有一定的分类错误风险，我们设立缓冲区的目的就是为了使得所有的点分的越开)
当蓝色的点在红色那一边的时候，或者红色的点在蓝色那一边的时候，我们将ϵ的值设置为$1<\epsilon$(这里因为已经形成了错误的分类，所以将ϵ的值设置的更大一点，并且在错误的一端错的越远这个penalty的值就越大，对整体分类效率的负面影响也就越大)

所以我们就有下面的数学表示:

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M$$

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

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

$$\epsilon_i \geq 0, \ \sum_{i=1}^{n} \epsilon_i \leq C,$$

这里中间那一条分类线其实和M还有penalty ϵ无关，因为其本身就是当M=0时候的结果。这里变化的只是最优缓冲区的确定，并且这是整体所有点都考虑进来之后的最优缓冲区。

这里ϵ就是上面说的的penalty。第三个式子右边的M·(1-ϵ)表示的是加入ϵ这个penalty之后改良过的每一个点距离中间那条线的“距离”，在不同的区域内的时候由于ϵ的值不一样，分类越错误，M·(1-ϵ)的值也就越小，错误分类的时候其值甚至是负数。但我们是想要其越大，因为其代表了所有点距离中间这条线的”距离"，所以我们需要将整体的ϵ之和控制在一定的范围之内。

这里的C就是我们自己设置用来控制所有在缓冲区甚至错误分类的点的ϵ总和。我们想要整体的(潜在错误:缓冲区内 和实际错误: 在另一个相反区域内)错误率越小，也就是想要C的值越小。当我们设置C为0的时候，也就是让所有的ϵ都是0，也就是一开始最理想的情况，0容忍率。当C设置的越大，也就表示我们对于错误的容忍程度越来越大。

同样，我们希望的是通过这个数学式的optimization得到所有点共享的一个M值(因为存在有错误分类的情况，所有这里的M就不是前面理想情况下距离中间那条线最近的那个点的距离了，而是一个真正使得整个分类最优化的缓冲区域，我们称作Margin)和Xi的β值，还有每一个点分别的ϵ值。ϵ进行optimization逼近的时候的跨度取决于这个点在哪个区域，实际上也是取决于M的取值，因为M代表的就是margin的范围大小。我们自然是希望M的值越大越好，因为其代表的是分类的有效程度，又不能变得很大，因为加入ϵ之后，当M越大的时候包括进margin的点就越多，也使得ϵ不为0的点就越多，这样分类的效率就越被侵蚀。

所以这其实是一个M值和被包括进margin区域的点的ϵ(还有本身就是错误分类的点的ϵ)总和C之间的一个平衡。

我们还可以从上面的式子看出来一个现象，那就是只有在缓冲区内和错误分类的点才会对M的值(或者说优化的缓冲区大小)产生影响。这些点其实就是被称作support vector，他们的作用就是确定最优的缓冲区，我猜名字的来由也就是他们是用来support将两边区分开来的区域。因为其实我们只需要保证这些点同时距离中间那条分类的直线最远即可，其他在缓冲区之外的点都比这些点离中间的分类线更远，所以也同时保证了缓冲区外的点都更能被分类。这里我们也可以想到，由于在缓冲区外面的点的penalty ϵ为0，所以对于缓冲区的大小其实没有影响，有影响的仅仅只是ϵ不等于0的缓冲区内和错误分类的点(当然这里有一个feedback loop，因为只有确定了缓冲区M的值之后才能知道哪些点在缓冲区外面，所以整体才是属于全局优化的问题)。

缓冲区的重要意义在于，在缓冲区之外的点，我们可以非常确信是这一边的分类，所以我们是希望缓冲区外面的点越多越好，我们对于所有点的整体判断成功率就越高。所以当我们将我们对于错误的容忍程度C的值降低的时候，缓冲区的大小也在收缩。如下图我们可以看到随着C的值减小，缓冲区的宽度也在一直变窄:

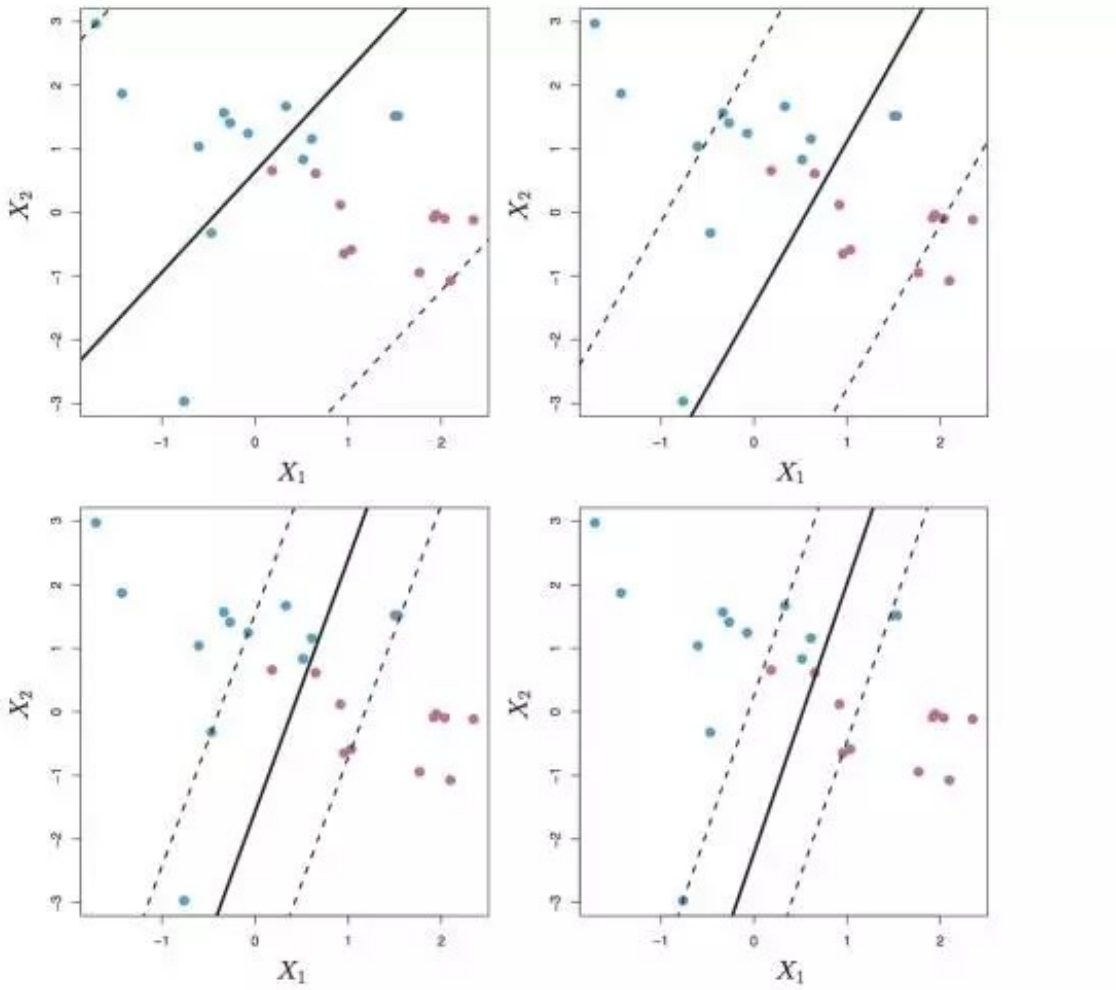


FIGURE 9.7. *A support vector classifier was fit using four different values of the tuning parameter C in (9.12)–(9.15). The largest value of C was used in the top left panel, and smaller values were used in the top right, bottom left, and bottom right panels. When C is large, then there is a high tolerance for observations being on the wrong side of the margin, and so the margin will be large. As C decreases, the tolerance for observations being on the wrong side of the margin decreases, and the margin narrows.*

这里我们给予的参数C做的事情其实就是统计模型中最核心的bias和variance的tradeoff。

当C很小的时候，我们对错误分类的容忍程度小(当设置C=0的时候，其实是零容忍)，缓冲区会很窄M的值很小，很少的点会在缓冲区内或者被错误分类(即缓冲区被很少的点所决定)，这样其实是使得整体的分类效果更好(bias很小)，但牺牲的就是model的灵活性，也就是variance很大，对于out-of-sample的数据的分类效率会很低。直观的理解上来说，如果我们将缓冲区内的点稍微变一下，对于缓冲区的影响是非常大的，也就说明这个缓冲区对于数据的依赖是非常大的。

同理，当C很大的时候，错误的容忍程度大，缓冲区会很宽M的值很大，然后几乎所有的点都被包括进了缓冲区，所以整体的分类效果会很差(bias大)，但是对于数据的依赖性降低(variance小)。这里的直观理解也类似，如果我们改变一些缓冲区内点的值，对缓冲区的影响非常小，因为缓冲区取决于其他更多的点，也

就是对于数据的改变不是那么敏感。
这两种情况我们都不希望发生，我们需要在bias和variance之间找一个平衡点，所以我们就需要用前面介绍的cross-validation的方式来确定最优的C的值。

OK，经过上面的过程，基于我们所有的数据，还有设置合适的C值，就可以得到中间那条分类线和缓冲区了，然后就可以根据他们对于out-sample的点的X1，X2的值进行Y的分类预测。

实际情况其实还经常会更加复杂。上面讲的SVM是仅仅针对linear的情况(也就是直线就可以帮助进行分类)。但如果是如下的情况呢？很明显这样线性的预测就不再适用了。

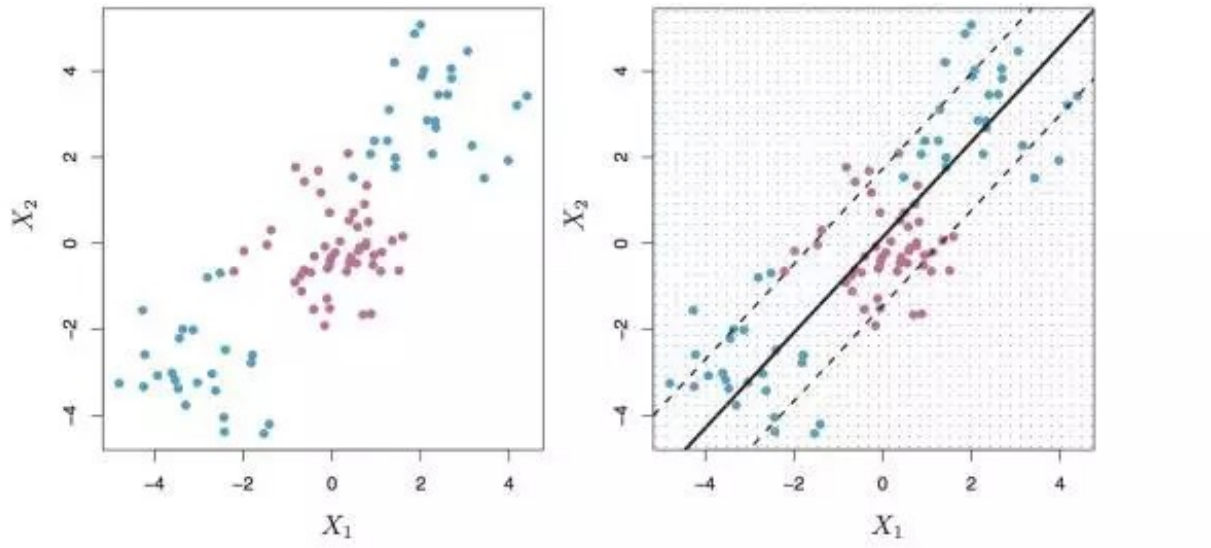


FIGURE 9.8. Left: *The observations fall into two classes, with a non-linear boundary between them.* Right: *The support vector classifier seeks a linear boundary, and consequently performs very poorly.*

这里我们就需要将线性的空间拓展或者经过映射变成高维非线性的空间(图中的例子是2维，因为只有X1和X2这2个变量，为了得到non-linear的关系，我们需要将X1和X2进行升维，譬如简单的加入另外的因子X1和X2的平方值，开方值，log值等等形式，复杂的就是直接将每一对观察值本身作为因子或者说kernel，将直线变成曲线甚至是圆等等形式)，如下图所示:

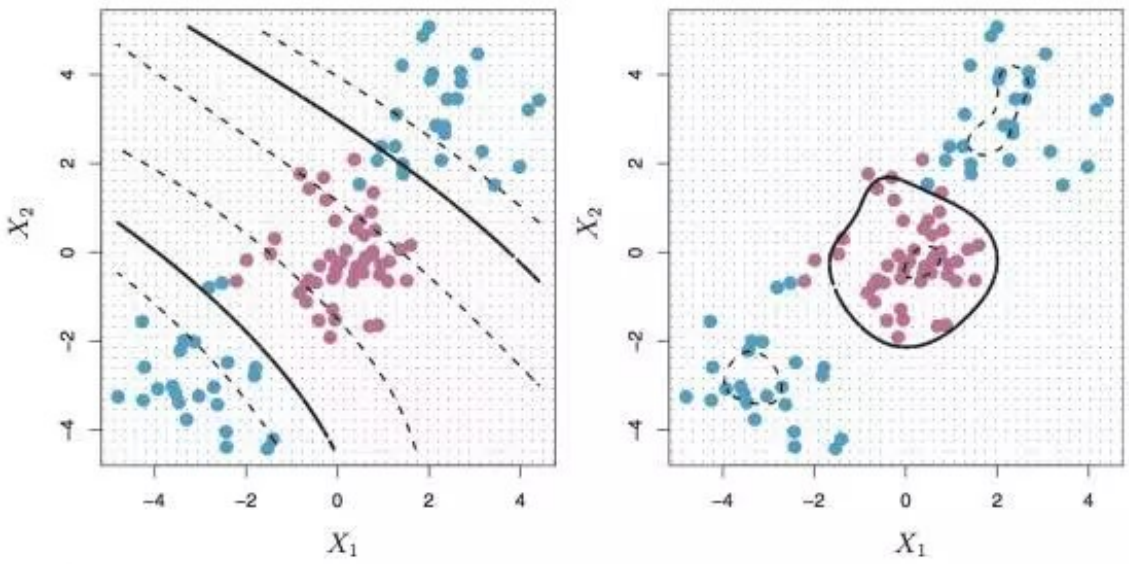


FIGURE 9.9. Left: *An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from Figure 9.8, resulting in a far more appropriate decision rule.* Right: *An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.*

我们可以看到，这样non-linear的SVM分类方式对于上图更为合适，在实际操作中kernel的具体形式可以自行在R中选择(具体选择哪一种更合适可能就取决于自己对于想要处理的dataset性质的了解了)。

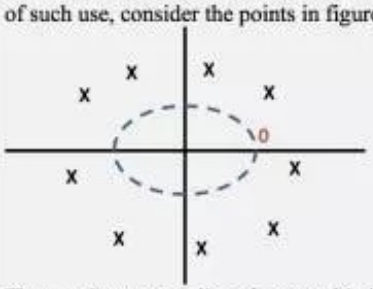
Kernel Trick Within Support Vector Machine

We describe the kernel trick in this section.

Given a set of input *attributes*, say $\underline{x} = [x_1, x_2, x_3]$, one often finds it useful to map these to a set of *features*, say $\phi(\underline{x}) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ \vdots \\ x_3x_2 \\ x_3x_3 \end{bmatrix}$.

Such a mapping into higher dimensions can often improve the performance of algorithms for function approximation. In many algorithms, one can reduce the functional dependence on input attributes to just inner products, i.e. the optimization function depends on the training example $\underline{x}^{(i)}$ only through the form of $< \underline{x}^{(i)}, \underline{c} >$, where $\underline{c}$ is another vector. In this case, the mapping to the higher-dimensional feature space creates the inner product $< \phi(\underline{x}^{(i)}), \phi(\underline{c}) >$. The kernel trick refers to the observation that the inner product $k(\underline{x}^{(i)}, \underline{c}) = < \phi(\underline{x}^{(i)}), \phi(\underline{c}) >$ can often be computed very efficiently.

As an example, consider the mapping ϕ as defined above. A naïve computation of $< \phi(\underline{x}^{(i)}), \phi(\underline{c}) >$ would be an $O(n^2)$ operation. But noting that $< \phi(\underline{x}^{(i)}), \phi(\underline{c}) > = (\underline{x}^{(i)T}\underline{c})^2 = k(\underline{x}^{(i)}, \underline{c})$, we can compute the value in just O(n) time. This implies that an algorithm designed for use in a low-dimensional space of attributes can function even in a high-dimensional space of features, merely by replacing all inner products of the type $< \underline{x}^{(i)}, \underline{c} >$ by $k(\underline{x}^{(i)}, \underline{c})$. As an example of such use, consider the points in figure.



These points are not linearly separable in two dimensions, but mapping them to higher dimensions, say three, will make them so. In practice, this kernel trick is widely used to improve the performance of algorithms as diverse as the perceptron to support vector machines to Principal Component Analysis. The kernel trick enables linear learning algorithms to learn a non-linear boundary without specifying an explicit mapping. One need not stop at a finite-dimensional mapping. The commonly used kernel function is the Gaussian Radial Basis Function, $k\left(\underline{x} - \underline{y}\right) = e^{\frac{\|\underline{x}-\underline{y}\|^2}{2}}$, which corresponds to an infinite-dimensional mapping function ϕ and yet is computable in just O(n) time.

To decide what forms are acceptable for the kernel function $k(\underline{x}, \underline{y})$, one defers to Mercer's theorem. Simply put, it states that for a function k: $\mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ to be a valid kernel function, it is necessary and sufficient that for any set of vectors $\{\underline{x}^{(1)}, \underline{x}^{(2)}, \ldots, \underline{x}^{(n)}\}$, the corresponding kernel matrix $K_{ij} = k(\underline{x}^{(i)}, \underline{x}^{(j)})$ is symmetric and positive definite.

上面就是具体kernel来升维的数学推导过程，感兴趣的童鞋可以仔细推敲。如果不感兴趣的，也可以装作懂了的样子，继续往下看其具体在外汇option上应用的例子了~

这里的主角是外汇期权，1M ATM EURUSD options，
我们来看看Y和变量Xi分别是什么。这里的Y是由rolling这个期权的long position产生的PnL(profit and loss)决定的，并且是分类的类型。
当PnL<-20 bps的时候，Y的分类是"vol sell"，这里vol的意思是波动率volatility，后面的期权部分会详细介绍。
当PnL>20 bps的时候，Y的分类是"vol buy"
当-20<PnL<20 bps的时候，Y的分类是"neutral"

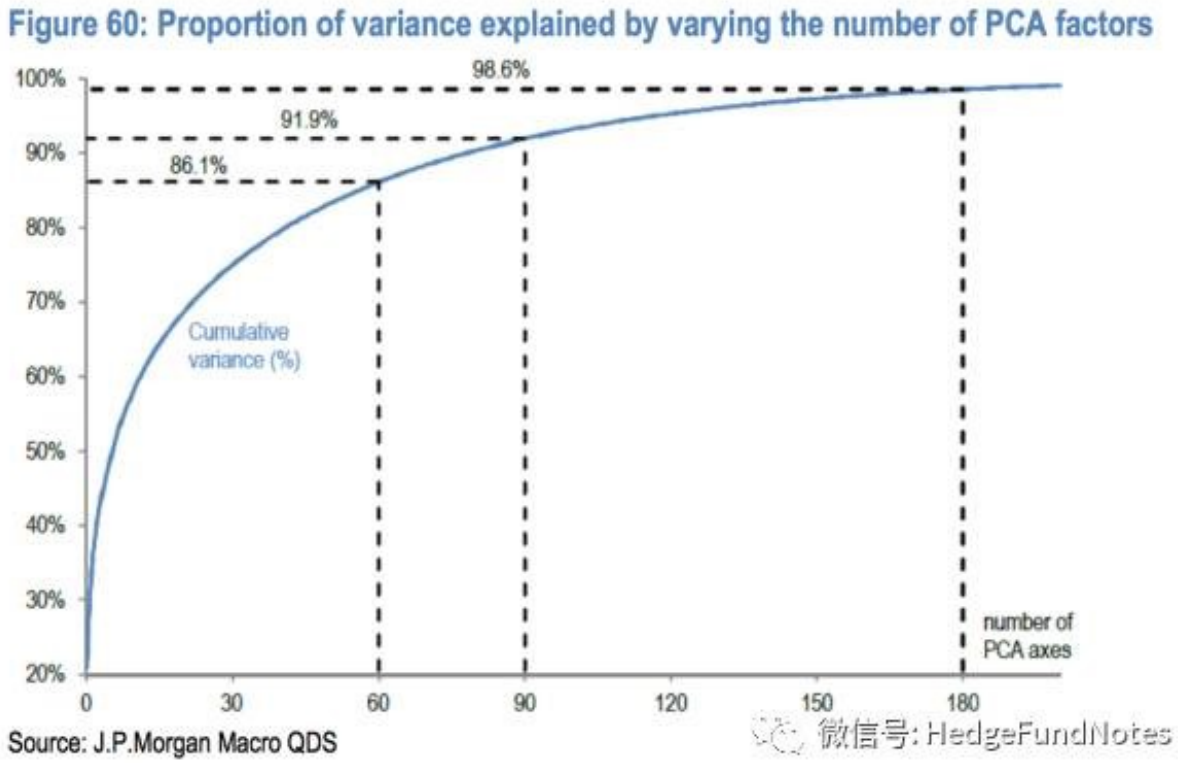
这里的Xi呢，是由如下377个不同的指标构成(由于这些指标的scale不一样，所以依然需要对其normalization处理):

Figure 59: List of input variables for analysis of FX volatility strategy

Market data type	Market data	Level	1 week change	1M change	Count
FX realised vols	2M realised vols in USD vs G10, MXN, BRL, ZAR, TRY, INR and KRW	X	X	X	45
FX ATM vols	1M, 3M and 1Y ATM in same pairs	X	X	X	135
FX skews	3M 25D RRs	X	X	X	45
FX spots	FX Spots in 15 G10 and EM USD pairs		X	X	30
Depo rates / Basis	3M FX Forward drops		X	X	30
Interest Rates	10Y Gov yields: US, Japan, UK, Germany, France, Italy, Spain, Australia		X	X	16
Equity Indices	S&P, Nikkei, FTSE, E-Stoxx, ASX, Mexbol, Bovespa, KOSPI, Hang Seng		X	X	18
Commodities	Gold and Brent spot		X	X	4
Credit spreads	CDX IG and HY, iTraxx spread indices		X	X	6
EASI indices	Global, US, CAD, EU, UK, CHF, NOK, SEK, Japan, AU, NZ, China		X	X	24
IMM positions	USD, EUR, JPY, GBP, CHF, CAD, AUD, NZD, MXN, RUB, Gold		X	X	24

Source: J.P.Morgan Macro QDS

然后对于数据呢，我们是取的过去10年daily的数据，一共是2609个点。为了防止overfitting，我们用Principle Component Analysis(PCA，后面也会详细介绍)来对Xi进行降维(因为这里的Xi有370个，过于多了)，我们通过PCA选取最重要的60个，90个，180个变量Zi(这里的Zi是由全部的Xi线性合成的大因子，被称作主成分因子)。如下图:



PCA呢，是一种对变量Xi非常多的情况进行降维的方式，下面unsupervised的统计方法里面我们会细致讲这个方法是具体怎么降维的。主要的思路就是将所有的Xi线性组合成若干个主成分因子Zi，使得其能最大程度的代表所有的信息。上图表示的就是这些最主要贡献的Xi积累起来对于Y的解释程度，也就是这些选中的Xi所包含的信息能涵盖Y的多少信息。横坐标就是由PCA选出来的主成分因子的数目，纵坐标就是对Y的解释程度。
例如在图里我们可以看到，当我们选择前60个最大贡献主成分因子的时候，可以解释86.1%的Y；当增加到前90个因子的时候，可以解释91.9%；当选取前180个因子的时候，可以解释98.6%，相当于我们用了180个主成分因子就代表了原始370个因子的91.9%的信息，使得计算和model复杂程度都大大减少。

有些盆友可能会觉得有疑惑，前面的linear　regression的lasso方法似乎也可以用来降维，为什么这里要用PCA这个方法呢。
我觉得有两个原因:
lasso方法需要利用连续的Y的值去做regression才能进行降维，而这里Y是分类不是数值，并且PCA的方法仅仅只需要Xi本身的信息就可以直接对Xi进行线性组合来降维。
第二个原因是这里的因子数目太多，有370个，如果用lasso的linear regression直接做回归会使得模型不稳定和过拟合。

对于数据，我们将其分成两部分，前8年的数据作为training sample，后2年的数据作为test sample，并且对于training sample用了10-fold cross-validation的方式去train model。

10-fold cross-validation其实是k-fold cross-validation中k=10的情况，其内容是将training sample随机均分成10组。然后每次选出一组作为validation set(其实也就是用来做test sample)，这一组选出来之后剩下的数据全部作为training set来train模型，模型确定之后再应用在那选出的一组上求得MSE(可以当做test error)。这是一次实验。接着按照同样的方式将这10组的每一组都作为validation set进行同样的步骤，每一组都可以求得一个MSE的值。然后最终的结果就是将这10组MSE的值取平均，得到这个model最终的MSE。

但是这里为什么要用10-fold cross-validation我比较疑惑。或者说这里的10-fold cross-validation是用来得到10个Y的预测，然后平均一下得到最终的Y？不太确定10-fold cross-validation是不是可以直接用来对Y求平均值。一般10-fold cross-validation只是用来求得平均的MSE，或者说根据某一个变化的参数画出来最小MSE时候(一般由于这个参数是penalty因子，代表的是bias和variance之间的tradeoff，自然MSE的图会是U型)这个最优参数的位置。

然后得到如下的结果:

Figure 61: Prediction accuracies of supervised learning models on out-of-sample data from 2014 to 2016

Algorithm	Raw Data	Normalised	PCA 180	PCA 90	PCA 60
kNN	69.0%	83.9%	83.7%	83.3%	84.1%
SVC (polynomial kernel - degree 3)	59.5%	74.1%	82.4%	84.1%	84.1%
SVC (linear kernal)	62.1%	80.8%	83.3%	83.3%	82.4%
Ridge Regression	81.0%	80.8%	73.9%	68.4%	69.3%
Gaussian NB	67.2%	68.4%	69.2%	69.5%	69.3%
Linear Discriminant Analysis	81.2%	81.2%	73.4%	68.6%	68.2%
Logistic Regression	79.3%	79.9%	74.1%	68.6%	68.0%
CART Decision Tree	75.7%	76.1%	61.9%	68.6%	67.6%
PAC Regression	48.5%	76.6%	63.2%	65.7%	60.3%
SGD Regression	41.2%	76.4%	69.3%	64.9%	57.5%

Source: J.P.Morgan Macro QDS

我们可以看到KNN，SVC(也就是SVM，kernel是三阶的polynomial方程)还有

SVC(kernel是线性的方程)这三种model表现最好。

预测的结果跟现实的结果对比如下图:

Figure 62: Performance of Support Vector classifier on test set

		Predicted			
		Long	Neutral	Short	Recall
Actual	Long	**30**	26	0	53.6%
	Neutral	4	**304**	16	93.8%
	Short	0	46	**96**	67.6%
	Precision	88.2%	80.9%	85.7%	

Source: J.P.Morgan Macro QDS

我们可以看到SVM的方法预测准确率还是比较高的(也就是一般预测的都是对的)，但是抓住机会的次数(或者说预测trigger开仓的次数)不多。
所以策略的话可以着重利用高准确率这方面，或者重仓。

上面我们了解了两种处理classification问题的方法，一个是logistic regression，一个是SVM。他们的比较是当分类的点非常分散的时候SVM的表现更好(图上的表现就是相同类别的点会聚集在一起，并且不同类别的点尽量分开)，当各个类的点混杂在一起的时候logistic regression更加合适(高度混杂在一起的时候SVM就无能为力了)。
SVM可以看作是一种空间上的mapping，所以当不同类的点被糅杂在一起而不是分团聚集在一起的时候(聚集在一起不一定是指在linear情况的在线的同一边，也可以是聚集成圆等等形状，通过mapping到另一个合适的更高维空间依然可以转化成linear的情况来进行处理，继而再mapping回去降维)，会难以找到一个合适的映射空间处理问题。

2.5 Unsupervised Learning(包括期权相关科普)

Principal Component Analysis

Unsupervised方法和Supervised的区别呢，就是在应用的过程中有没有涉及到Y。
如果有Y，那就是supervised，更进一步如果Y的取值是连续的数值，比如涨跌幅度，那就是quantitative，比如regression的方法；如果Y的取值是discrete分立的类，那么就是qualitative，比如classification的方法。
如果没有Y，那就是属于Unsupervised方法。因为没有Y，所以Unsupervised的方法仅仅操作因子Xi本身。

这里unsupervised由于没有Y，所以并不像supervised一样是用于预测某个值或者某个类，而是主要是用于研究Xi之间的关系。所以Unsupervised方法有一个好处就是可以作为应用supervised方法之前对数据本身的预处理，数据本身的信息对于后期我们选择supervised方法进行预测Y有很大的帮助。所以unsupervised方法又起到exploratory data analysis的作用。
但是我们也要知道unsupervised方法有一个问题，正因为没有Y，所以我们没有办法像supervised方法那样验证我们的结果是否正确。
相对于supervised方法，unsupervised方法还有一个好处就是可以处理当因子Xi非常多(比如成百上千个)但是数据量又不是那么多的时候的情况。如果直接用传统的regression方法就会造成模型很不稳定，所以可以先利用unsupervised里面的Principal Component Analysis(PCA)方法将Xi进行降维，下面将会介绍到。

好了，大概unsupervised方法的背景就介绍到这，现在介绍一下其具体内容。

Unsupervised里面分为两块儿，一块儿是Factor Analysis，另一块儿是clustering。

首先呢，我们介绍Factor Analysis这一块儿，其中比较重要的Principal Component Analysis(PCA)方法。

为了更好的理解PCA的应用，首先简单介绍一下PCA的原理。
从名字Principal Component Analysis我们就可以大概看出来这个方法想干嘛，中文翻译过来叫做主成分分析。自然，其想要研究的就是主要的成分，换句话说，就是用最少的变量描述最多的信息
这里和前面lasso的方法有点不一样，lasso是直接把因子Xi前面的系数变为0，也就是直接把这个因子剔除的方式来给因子Xi降维，
而PCA是将很多的因子Xi进行线性组合，变成若干几个Zi来代表尽量多的信息。(这里一定要注意，PCA并不是像lasso一样将Xi的系数变为0的方式来降维，而是通过找到能最大代表数据信息的Zi，每一个Zi都含有所有Xi的信息，只不过他们前面的系数不一样)
因为如果当因子非常多(有些fund的strategy pool里有上百万个因子)的时候，我们可以通过PCA将这些因子构造成若干个主因子来进行操作。

并且在统计界有一个恐怖的传说，叫做"curse of dimensionality”。
因为因子Xi是我们人为主观判断可能会跟Y有关系才加上去的，那么有些人会想，因子的数目不就越多越好？毕竟最怕的就是在预测Y的时候没有将真正有贡献的因子考虑进来，前提错了的话后面不管怎么弄fancy的model都是错的，所以那如果把能想到的因子全部都加进去这样不就万事大吉了么，一个都不能少！这样的想法是没错，漏掉重要的因子很可能会对Y的预测效果产生巨大的影响。但是凡事都是一个平衡，如果在模型里面因子加入了太多的话，即使把真的有贡献的因子加了进去，也会淹没在茫茫的因子中(因为加入的没有贡献的因子更多，并且由于每一个因子都有一定的权重会一起影响这个model，相对来看，有贡献的因子对于模型的解释能力其实是被稀释了)。
用统计的黑话来说就是，加入好的因子虽然会使得模型整体的bias下降(毕竟因子的信息更全面了)，但是损失的是整个模型的variance会变大，也就是增加了其对于数据本身的依赖性，导致overfitting。特别是当因子的数目甚至大于我们数据点的数目的时候，那基本上无解了~(no kidding，是方程真的木有唯一解了...)

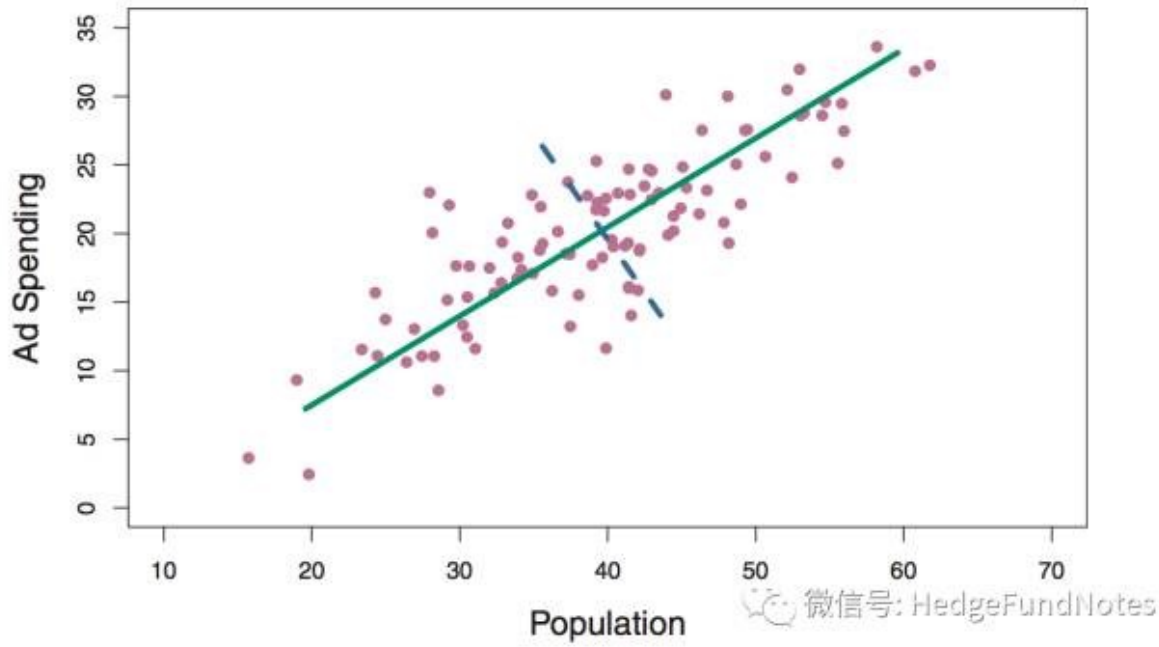
所以这里对于Xi降维的重要性就体现出来了，一个就是像lasso做的只留下几个能产生最多贡献的因子，另一个就是将所有因子线性组合成若干个可以代表这些因子信息的大因子，虽然这样同时也扔掉了没有被包括进来的因子可能包含的真实的有效信息，但由于减少不那么重要的因子数目，降低整个模型的variance，虽然会一定程度上牺牲模型的bias，但是总体来看效率还是提高了。

这样，通过PCA的降维方式，我们就更清楚数据信息背后的结构和主要的贡献来源，使得后续supervised方法的fitting更加有效率。

除此之外，PCA还能将这些主要的因子线性组合起来，变成若干个大的因子(实际上这些大因子就叫做Principal Components，主成分)，用这些大因子代替之前的单个小因子Xi来尽量解释最多的Xi的信息。
因为同样对于高维(Xi非常多)的情况，还有一个问题就是当因子数目多了之后，因子之间的相关性问题会变得非常严重(我们可以想象因子足够多的时候，他们之间的信息很难会是完全exclusive的，特别是像金融市场，不同因子之间基本上都会有一定的相关性)。一个解决办法就是将这些相关联的因子结合起来变成一个因子，也就是这里PCA可以做的事情。并且PCA可以保证这些得到的大因子之间是线性不相关的，也就是在方向上是垂直的。PCA通过正交变换将一组可能存在相关性的变量(若干Xi)转换为一组线性不相关的变量(变成Xi的线性组合Zi)，除去对结果影响不大的特征，转换后的这组变量叫主成分。

好了，接下来就是数学上的解释了。为了避免公式的推导，直接用上图吧。

祭出遇到PCA必拿出来的一张图:



这里的横轴和纵轴分别是X1和X2两个变量，然后所有的点都画在这样一个二维

的平面内。
现在呢，我们想要用一个变量去尽量包括这些点最多的信息，PCA也就是做这件事，图中的绿色哪条线就是PCA的结果。
怎么解释呢？
可以这么看，一个理解是最小二乘法，也就是这些点距离这条直线的距离最短，就像regression一样，最理想的情况就是这些点全部都在这条线上，自然这条线就能完全描述这些点的信息。

另一个理解就是最大方差理论，尽量使这些点在这条线或者说这个方向上的variance最大，也就是让这些点向这条直线做投影，要尽量使得这些点的投影距离最远。
为啥要最远呢？因为最远可以表示有更丰富的信息落在这一条线上面，这些点的区分度也越大。举个例子，

“举个例子，比如现在有10个人的数据，每条数据有两个维：出生地，性别，10个人的出生地各不相同，5男5女。从感觉上看，哪一维所包含的信息量大一些？是不是出生地呢？如果我一说出生地，是不是马上就可以找到那个人了呢？因此出生地各不相同嘛，而性别呢？因此性别只有2种，因此会有大量重复的值，也就是很多人的值都是一样的，这个属性对于人的刻画效果不好，比如你找一个性别是男的人，可以找到5个，这5个人从性别这一维上看，是完全一样的，无法区分。从PCA的角度来看，出生地这个维的方差就要比性别这个维的方差要大，而方差大就说明包含的信息比较丰富，不单一，这样的特征是我们希望留下的，因此就这个例子来看，PCA更倾向于保留出生地这个特征。

那么是不是简单的就把需要保留的特征留下来，不需要保留的特征去掉就行了呢？比如说刚才那个人的数据的例子，是不是直接把性别这个特征去掉就完了？远远不止那么简单，PCA是把原特征映射到一个低维空间上，使得特征在这个低维空间中的方差最大，如果是直接去掉特征维的话，那么剩下的特征不一定是低维空间中方差最大的。这里要注意一个问题，继续刚才那个人的数据的例子，有2维，性别和出生地，如果用PCA把它降成1维，那么这一维是什么？这是没有定义的，也就是说，这一维既不是性别，也不是出生地，它已经没有定义了，我们只知道它是1维选择出来的特征而已，但这并不影响我们对特征的使用。”

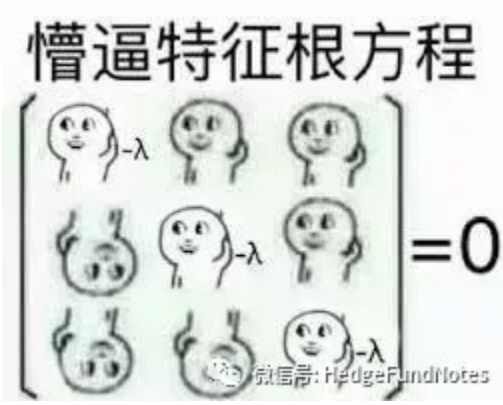
说了这么多，现在我们大概知道PCA是在干一件什么事情了，就是通过线性组合

高维空间中Xi的方式构造出若干个Zi来对数据进行降维，并且找到尽量能代表所有点信息的几个主方向(也就是几个构造的大因子，一般第一个大因子包含的信息最多，然后信息度浓度次递减)。

好了，接下来说PCA的具体应用。
这里PCA是应用在期权的implied volatility surface上面。

首先我们来看一张implied volatility surface的美图:

什么？又一脸懵逼？



看来这里需要稍微介绍一下期权才行呢...

首先，能读到这儿的我充分相信大家的实力，就直接上传说中的BS公式了(摘选自wiki)。

Black–Scholes formula [edit]

The Black–Scholes formula calculates the price of European put and call options. This price is consistent with the Black–Scholes equation as above; this follows since the formula can be obtained by solving the equation for the corresponding terminal and boundary conditions.

The value of a call option for a non-dividend-paying underlying stock in terms of the Black–Scholes parameters is:

$$C(S_t, t) = N(d_1)S_t - N(d_2)Ke^{-r(T-t)}$$
$$d_1 = \frac{1}{\sigma\sqrt{T-t}}\left[\ln\left(\frac{S_t}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)\right]$$
$$d_2 = d_1 - \sigma\sqrt{T-t}$$

The price of a corresponding put option based on put-call parity is:

$$P(S_t, t) = Ke^{-r(T-t)} - S_t + C(S_t, t)$$
$$= N(-d_2)Ke^{-r(T-t)} - N(-d_1)S_t$$

For both, as above:

- $N(\cdot)$ is the cumulative distribution function of the standard normal distribution
- $T - t$ is the time to maturity (expressed in years)
- S_t is the spot price of the underlying asset
- K is the strike price
- r is the risk free rate (annual rate, expressed in terms of continuous compounding)
- σ is the volatility of returns of the underlying asset

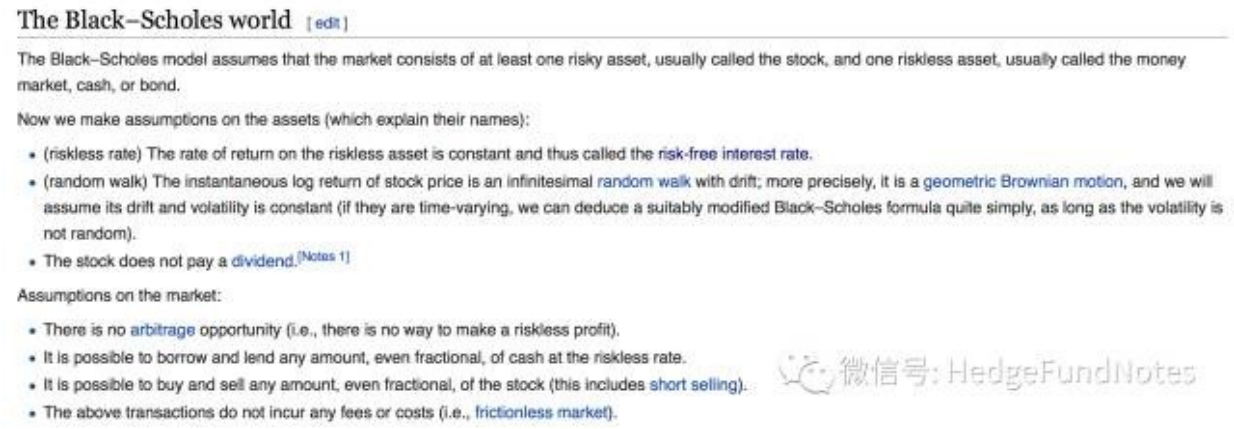
期权定价的核心几乎就是这两个公式，Call的定价和put-call parity的问题。

我们看第一个式子Call的定价C(St， t)，我们可以看到这个式子中有5个参数。对于固定某个时间T，我们知道这个call的strike price执行价格K，也知道这个call的maturity到期时间(T-t)，也知道risk free rate无风险利率r，正态分布N(·)的形式自然也都知道，剩下的就是volatility σ不知道，所以现在我们就得到了一个volatility的方程来表示Call的价格C(St, t)。

volatility是一个什么东西呢？volatility是波动率，也就是期权underlying的标的价格的标准差。你如果问我标准差怎么算，我只能说去Google一下，计算讲的太细节估计都看睡着了...

所以假如我们知道volatility呢，我们就可以知道这个Call的理论价格，也就可以衡量当市场上Call的实际价格是高还是低，高了呢我们就可以卖，低了呢我们就可以买。
可惜天上不会掉馅饼，因为我们不知道volatility...

接下来还得知道BS公式的假设前提:



The Black–Scholes world [edit]

The Black–Scholes model assumes that the market consists of at least one risky asset, usually called the stock, and one riskless asset, usually called the money market, cash, or bond.

Now we make assumptions on the assets (which explain their names):

- (riskless rate) The rate of return on the riskless asset is constant and thus called the risk-free interest rate.
- (random walk) The instantaneous log return of stock price is an infinitesimal random walk with drift; more precisely, it is a geometric Brownian motion, and we will assume its drift and volatility is constant (if they are time-varying, we can deduce a suitably modified Black–Scholes formula quite simply, as long as the volatility is not random).
- The stock does not pay a dividend.[Notes 1]

Assumptions on the market:

- There is no arbitrage opportunity (i.e., there is no way to make a riskless profit).
- It is possible to borrow and lend any amount, even fractional, of cash at the riskless rate.
- It is possible to buy and sell any amount, even fractional, of the stock (this includes short selling).
- The above transactions do not incur any fees or costs (i.e., frictionless market).

我们就可以看出BS模型的一个弊病了，那就是它假设underlying标的价格的波动率volatility是一个常数，跟strike price还有maturity没有半毛钱关系。
这明显是不符合实际情况的，因为市场会对对于不同的strike price和maturity有不同的预期和偏好，导致市场对于未来波动率的预期也会不一样。

但是！
我们虽然不能用波动率来算出来期权的理论价格，可是我们可以反过来呀~
我们有期权的市场价格，还有BS公式里面其他一系列的变量，比如strike price K，期权的maturity(T-t)，也知道risk free rate r等等，就可以反过来求出来理论上的波动率。这样我们就可以知道市场对于未来的underlying标的的波动率预期了。

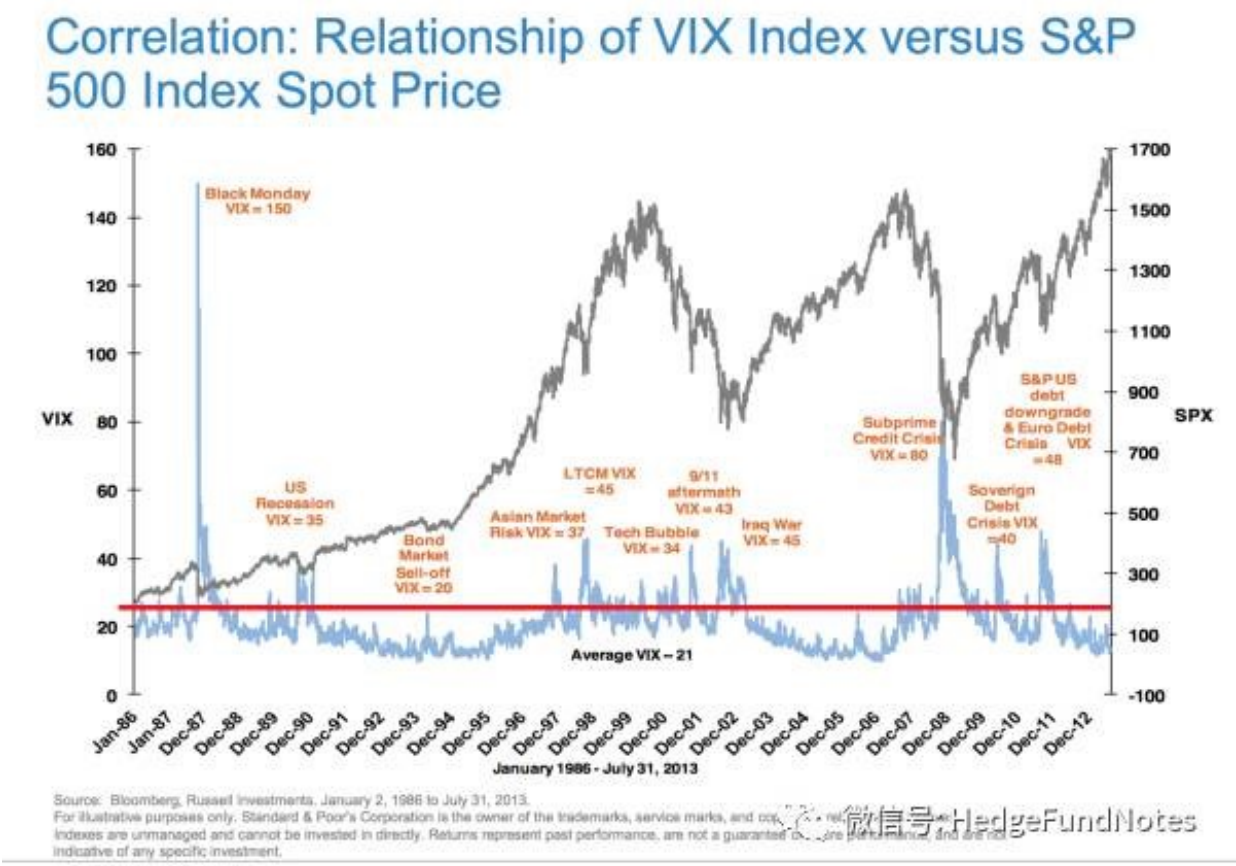
具体怎么算出来的，简单一点说就是当其他变量已知也就是固定的时候，期权的价格和波动率是一个单调递增的关系(如果对Greek有了解的，其实就是因为期权价格对波动率求一阶导的Vega是正值)，所以不断试σ的值就可以最终match到当下的期权价格上面，自然也就是对应的波动率的解了。

看到这儿，很多人心里是不是在想，“尼玛，这货说了这么多，implied volatility到底有啥用？”

首先呢，implied volatility对于期权玩家的重要性不言而喻。
假设市场的每个参与者都使用BS模型定价，由于模型所需的其他输入值比如执行价格、所剩期限、标的价格还有无风险利率都是相对固定的，人跟人对于期权价格的看法就是取决于他们对于未来这个资产的波动率了。假设有一个人巴菲特附体如有神助，对于未来资产波动率的预测百发百中如入无人之境，那么他(她)就

可以根据自己预测的期权的价格跟实际的期权价格作对比来低买高卖，赚的就是别人对波动率的预测都很弱鸡。
但是可惜的是，这种外星人是不可能存在的，不然早就是世界首富了。当然如果有个人弄出来一个模型可以准确预测资产未来的波动率，那也超神了。

所以对于我们一般战斗力只有5的地球人怎么办呢？
我们可以反其道而行之，利用将已经是事实的市场的期权价格代入BS公式中算出来大家对于未来波动率的预期，也就是隐含波动率。因为期权价格是市场上供求关系平衡下的产物，是买卖双方博弈后的结果，所以隐含波动率反映的是未来市场对标的产品波动率的看法。
但是，算出来有什么用呢？
我们除了隐含波动率，还有一个波动率叫做历史波动率。这个东西其实算出来也挺简单的，只要我们有underlying标的的价格变化历史数据，就可以用统计的公式算出来标准差，也就是波动率。而且我们发现一件什么事情呢，那就是历史的年化波动率是处于一个区间，会有回归均值的一个特性(做swing交易的同学应该最熟悉了)，并且离开越远回归的速度就越快。比如说有一个股票，历史波动率都是在10%到30%之间，这样我们就可以知道从现在期权市场价格算出来的隐含波动率在历史上是处于高还是低的位置，同样可以做到低隐含波动率的时候买期权高隐含波动率的时候卖期权。但这里有一个问题就是历史波动率的区间，不代表就真的不会飞出区间去，当发生重大事件或者传说中的黑天鹅的时候隐含波动率也会变很极端，比如Black　Monday的时候飞到了150%，08年金融危机的时候也飞到了80%。



当然，做均值回归有点投机的感觉。有的同学可能觉得太low，我可是做价值投资的，只是想利用期权对冲一下风险。
同样，隐含波动率也很有用的！因为当确认了自己想投资的标的方向之后，可以把隐含波动率跟历史波动率作对比，权衡一下此时买入期权是不是划得来。也可以对不同strike　price和maturity的期权进行横向比较，毕竟都是针对同一个资产，自然希望在满足了自己的需求买(卖)的情况下期权的价格越低(高)越好咯。

除此之外，对于直接交易underlying标的的交易员，我们也可以利用期权来帮助自己对标的的未来走势做出一定的判断。比如我们可以根据市场上对于同一个标的的所有不同strike price和maturity的期权价格，算出来每一个strike price和maturity对应的隐含波动率，其实也就是市场对于这个标的未来波动预期。然后将这个算出来的隐含波动率代入比如说隐含波动率二叉树模型，再算出来市场对于标的未来走势的预期，因为期权的价格本身就是已经是集中了市场上所有的有效信息，所以隐含波动率其实也包括了所有的博弈信息，利用其推导出来的未来走势就可以帮助我们对直接trade标的有一个大概的方向和市场预判。

好了，对于基本的期权概念应该有一个大概的sense了，接下来我们就来详细解释一下之前那张volatility surface的图。

(此图是在之前读伽玛交易员一篇文章存下来，08年危机时候的volatility surface，感谢之~)

期权交易员对volatility surface都会非常熟悉，因为这一定程度上就是自己根据市场情况做trade和进行风险对冲的基础。

volatility surface呢，是一个三维图，Z轴是隐含波动率，X和Y轴分别是moneyness和term(也许不同的软件会有不同的名字，这里借鉴Bloomberg的表示)。moneyness的意思就是期权strike price和underlying标的的价格的ratio，这里以call为例，分为in-the-money(strike price小于underlying标的的价格，ratio小于100%的区间)，at-the-money(strike price等于underlying标的的价格，也就是图中的ratio等于100%的位置)还有out-of-the-money(strike price大于underlying标的的价格，ratio大于100%的区间)这三种状态。如果是put的话strike price和underlying标的价格的关系则是反过来。term的意思呢，就是不同的到期日。

所以呢，我们可以把未来每一个到期日的每一个strike price和其算出来的隐含波动率画在一张3D的图上，这个就是volatility surface了。上图中右上角的是volatility VS term的二维横截面曲线，右下角是volatility VS moneyness的二维横截面曲线。可以说volatility surface包含了绝大部分我们需要了解的volatility的信息，而上面也介绍了volatility又是期权的核心，自然volatility surface的重要性也就不言而喻了。

既然volatility surface这么重要，那么如果我们可以预测未来的volatility surface是不是就相当于我们可以预测implied volatility进行交易了呢?

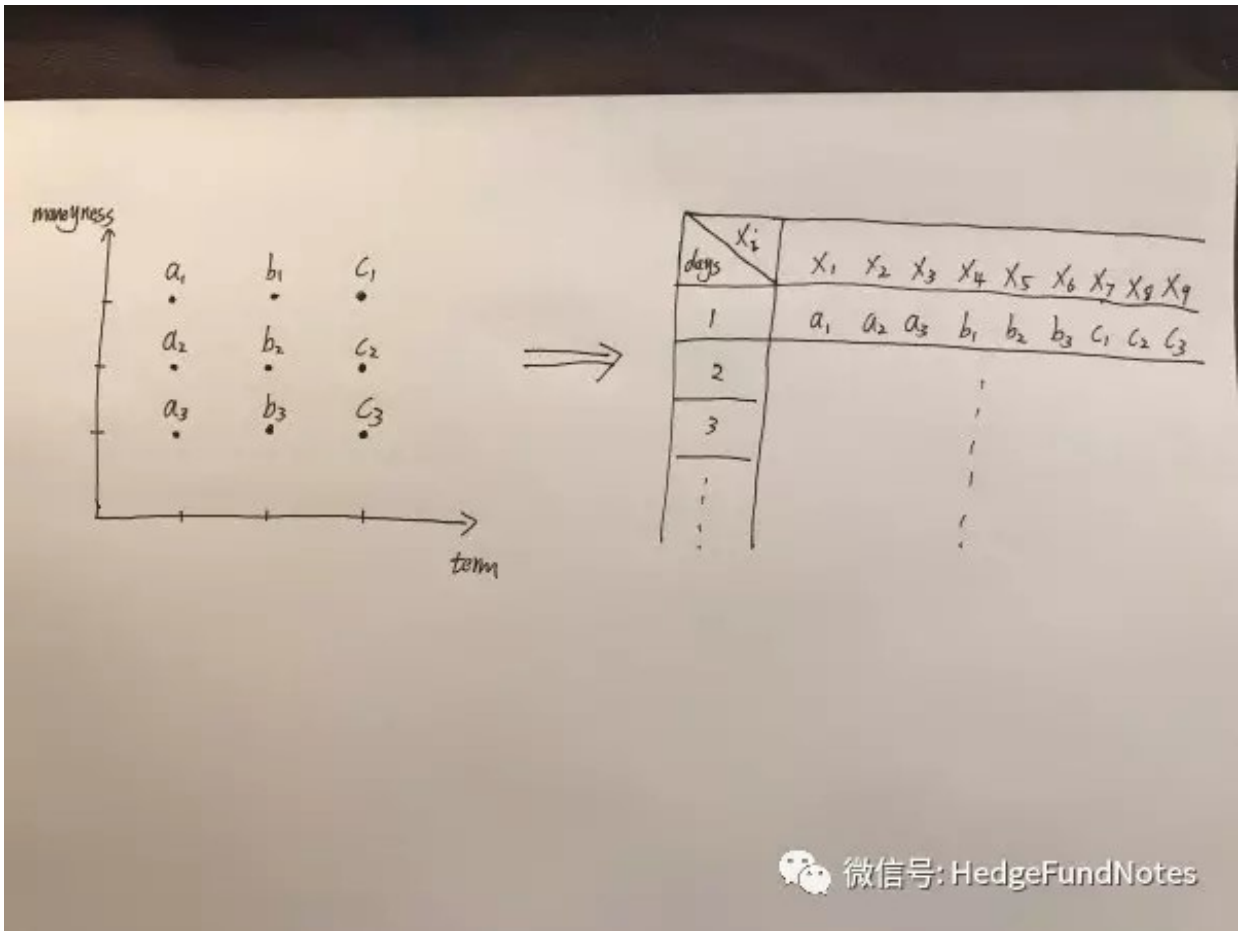
因为如果仅仅根据历史波动率进行交易，很明显我们丢掉了其他的很多有用的东西，仅仅只留下了波动率大小这一个信息。如果根据整个volatility surface的历史数据进行分析，对我们的预判的帮助会更大。在知乎上面看到说成功的期权交易，是做整个曲面的变化，这涉及的是曲面的smile/skew(下面会介绍)，和曲面的整体高低位置，前者贡献利润的10%，后者贡献利润的90%。

好了，铺垫了这么多，接下来介绍利用PCA来分析volatility surface的信息。

这里选择的underlying标的是USDJPY(美元兑日元)，至于为什么选这个，我感觉可能是想了解美日间carry trade的情况，而且日元是外汇市场非常重要的组成部分。

接着我们看看具体怎么将PCA应用在USDJPY的volatility surface上面。
volatility surface是一个三维的图像，Z轴是隐含波动率implied volatility，X和Y轴分别是moneyness和term。
对于PCA我们知道它做的事情是将所有的变量Xi线性组合成几个大的因子(主成分)Zi，所以问题就来了，我们的Xi到底哪些。
这里我们有的是implied volatility，moneyness还有term的信息。但我们并不能将他们全部作为Xi，因为虽然volatility是连续的数值，但term和moneyness的信息是固定的点。
所以我感觉这里有一个trick就是将每天的volatility的变化值作为Xi的值，而moneyness还有term的信息作为Xi的位置。

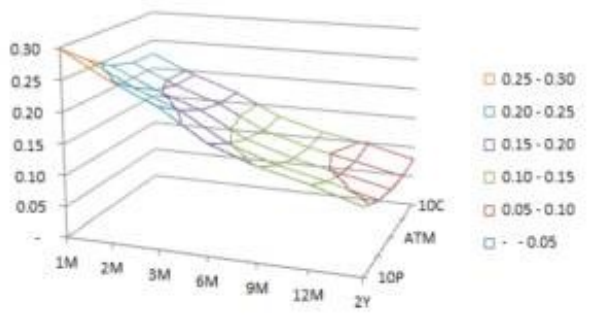
估计这么讲会有点混乱，画一张图就清楚了。

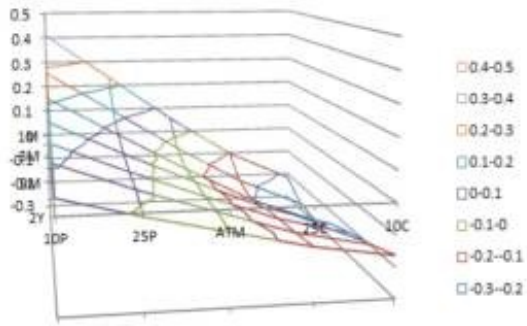


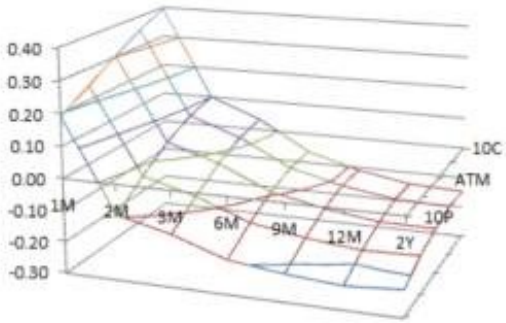
我们这里可以将所有的点都拉成一条线作为Xi，然后每天都有一列数据Xi。这样处理我们就将整个volatility surface变为了一列变量Xi，因为每天都有一个volatility surface，所以每天可以作为一个observation。

这样就将volatility surface的问题转化成了PCA的问题，我们想要找若干个由Xi线性组合成的大因子Zi，使得这些个Zi的方向最大程度的代表所有的点的信息(每天可以看做是这个Xi维空间上的一个点)。换做实际问题的解释，这些Xi的信息其实就是每天的整个volatility surface的信息，然后找到的由Xi线性组合成的大因子Zi的volatility surface可以最大的程度代表历史上所有天数的volatility surface的信息。

报告里面呢，选取了前三个主成分Z1，Z2和Z3，分别的图如下:





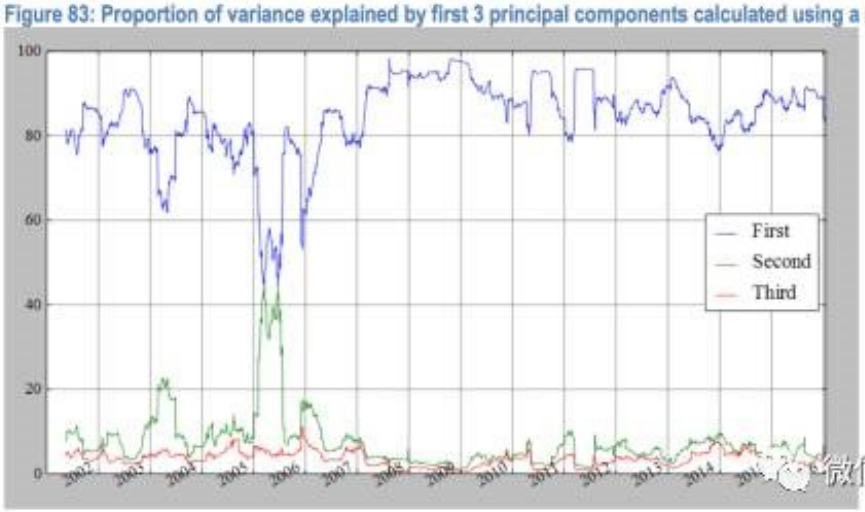


Number of principal components	Percentage of variance explained (%)
1	88.04
2	4.82
3	3.66
4	1.85
5	0.60

然后PCA还有一个好处呢，就是可以算出来每一个合成的大因子(主成分)对于所有点的解释程度(上图右下)，也就是包含了数据中多少比例的信息。具体的方法叫做“proportion of variance explained(PVE)”，做法就是算出来这个大因子的variance和数据整体的variance之间的比例。
所以这里我们就找到了前三个大因子的贡献率分别是88%，4.8%和3.66%，

下图就是这三个主成分大因子在历史上不同时间的解释能力。可以看到第一个主成分的解释能力保持恒定，并且几乎都在80%以上。

Figure 83: Proportion of variance explained by first 3 principal components calculated using a rolling 90 day covariance matrix



知道了能代表所有历史上volatility surface的这3个主成分之后我们自然能做的事情就很多了。
比如依据第一个主成分来判断delta hedge ratio(这里delta是期权价格对于

underlying标的价格的一阶导，可以当做期权的价格对underlying价格变化的灵敏程度)，也就是hedge delta使得整个portfolio独立于underlying标的价格变化(但实际上delta hedge只能解决一阶的hedge，二阶gamma的hedge因为是非线性所以只能依靠另外的期权来进行hedge)。还可以把前几个主成分单独拿出来，对剩下的residual做均值0的回归。

并且从这三个主成分大因子得到的volatility surface图中我们就可以看出来一些信息，对于固定maturity的情况下当moneyness是in-the-money的时候，implied volatility是很高的，说明了随着in-the-money的程度市场对于USDJPY的看跌程度是越来越大的。如果对于外汇市场了解的盆友可能就知道了，这就是因为日元往往被当做避险货币。

为了避免观众对外汇市场不熟悉，这里讲点日元这个特殊货币的背景，那就是日元具有避险功能。我大概总结了一下，仅作参考。

(1) 日本的常年低息政策，我们都知道在经济不好的情况下就会降息，如果已经是低息的话降息的空间就有限，利率降的越多货币越弱势，所以日元贬值的空间有限

(2) 同时也是由于日本低息的环境，所以国际环境好的时候很多人是借入日元然后买入其他高息的货币进行carry trader套利(就像美元QE的时候美国利息低大家会借入美元买入人民币投资中国市场)，但是一旦国际环境恶化，买入的货币国家会实行降息然后贬值，投资的风险喜好也变为保守，这样套利就没有办法继续维持，大家就会卖出买入的其他国家货币然后买回日元还掉以前介入日元的债务，这样就进一步推高日元和日本的汇率，同时也压低其他国家的货币。

(3) 日本的不管政府或者民间的资本在国际环境好的时候喜欢投资海外的资产，譬如政府的养老金基金，当外部环境变差的时候，这部分资金也会卖出资产回流日本换成日元，即使不换回日元也会买入看多日元的衍生品(期权或者期货等)来对冲风险，都会进一步推高日元。

(4) 日元的池子大，流通性好，外部环境变好之后随时可以再卖掉

(5) 大家只要一出事就做多日元已经是习惯了。。。所以形成了一个自我实现的循环。

其实上述对于固定maturity的情况下volatility随着moneyness变化的现象就是

有名的“volatility smile(skew)”。具体这里不详细说，因为模型假设的是underlying标的的收益率并符合标准的正态分布，但实际上市场上由于不同的风险偏好产生了期权的尖峰肥尾现象，肥尾的方向也就代表了implied volatility的smile(skew)方向。产生的原因也有很多paper探讨，暂时没有一个固定的说法。也许后面有时间的话我会写一写期权的notes，不过看老板push我科研的程度了...

对于期权有一点需要注意或者可以利用，因为涨和跌并不是对称的，涨的时候总是缓慢的波动率低的，跌的时候总是摧枯拉朽波动率高的，期权价格呢又跟波动率有关，所以呢我们可以利用这一点。譬如说买put，一方面如果资产真的下跌，put本身由于moneyness和intrinsic value会增值；另一方面资产下跌的时候会引起恐慌增加波动率，也会使得期权增值，所以其实两方面的动力会一起做贡献让put的价格飞起来很快。还记得前段时间的50 cent先生么？大家一开始都嘲笑这哥们脑子有病每天给市场送钱，现在才发现这哥们真牛逼。其实吧，他就是利用了期权波动率不对称这一点，疯狂在波动率低的时候扫货买便宜的put，一旦市场有点风吹草动导致哪怕一天的恐慌，向下的利润是非常丰厚的，远远超过前期裸买put的成本投入。当然，别人是建立在对当时整个市场情绪非常敏感的正确判断前提下才敢这么单边做期权，还冒着成为各国基金经理永远嘲笑的二逼的风险，这种勇气我还是很佩服的，实至名归。对于不了解资本市场的同学呢，不要看了之后脑门一热以为股神附体就all in了，成功了是big short不成功就变big idiot，老婆本都亏光了不要怪我，我代表华夏三千万单身汉同志欢迎你的加入~

PCA除了在期权上面的应用之外，最常见的做法是用来分解portfolio的risk，也就是找到不同主成分大因子的beta(就是市场系数那个beta...)进行分配风险。大概的结论如下:

Figure 84: Performance of PCA factors as derived from cross-asset risk premia (value, momentum, carry, short-volatility across equity, bond, currency and commodity complex)

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
Average (%)	8.9	1.0	2.9	3.1	5.6	1.0	8.0	4.3	1.6	8.1
CAGR (%)	6.5	-0.8	1.4	1.8	5.1	0.4	7.8	3.9	1.1	8.0
STDev (%)	22.8	18.7	17.6	15.6	11.9	11.5	9.6	10.1	9.7	8.5
MaxDD (%)	-63.8	-72.1	-56.8	-57.2	-34.6	-37.3	-47.9	-39.0	-37.4	-26.8
MaxDDur (in yrs)	9.1	30.2	24.6	18.8	15.1	23.3	5.4	9.0	23.9	4.1
Sharpe Ratio	0.39	0.05	0.16	0.20	0.47	0.09	0.83	0.43	0.16	0.96
Sortino Ratio	0.61	0.07	0.24	0.28	0.92	0.13	1.37	0.66	0.24	1.67
Calmar Ratio	0.31	0.02	0.18	0.30	0.47	0.07	0.58	0.47	0.14	1.08
Pain Ratio	0.39	0.03	0.09	0.10	0.47	0.05	1.26	0.57	0.09	1.99
Correl with SPX	0.23	-0.54	-0.03	0.22	0.20	0.54	0.00	-0.20	-0.07	0.30
Correl with UST	0.09	0.14	0.01	0.68	0.11	-0.03	0.05	-0.01	0.27	-0.01
Skewness	0.05	0.20	0.00	-0.50	3.16	0.03	-0.35	-0.26	-0.18	-0.12
Kurtosis	2.28	2.98	2.00	2.10	32.65	1.11	2.69	2.77	0.57	0.97

	PC11	PC12	PC13	PC14	PC15	PC16	PC17	PC18	PC19	PC20
Average (%)	2.0	0.1	6.5	2.7	2.6	0.8	8.6	3.2	1.1	2.1
CAGR (%)	1.7	-0.2	6.4	2.5	2.4	0.6	8.8	3.1	1.0	2.1
STDev (%)	8.7	7.5	7.8	7.0	6.6	6.2	5.7	5.3	4.2	2.8
MaxDD (%)	-30.3	-46.3	-24.1	-41.8	-21.0	-28.4	-8.0	-17.0	-14.1	-6.3
MaxDDur (in yrs)	11.8	32.8	4.9	27.8	12.8	14.1	1.8	8.0	9.5	3.8
Sharpe Ratio	0.23	0.02	0.84	0.38	0.39	0.13	1.50	0.61	0.26	0.74
Sortino Ratio	0.35	0.02	1.81	0.64	0.59	0.19	3.05	0.97	0.38	1.25
Calmar Ratio	0.16	0.02	0.85	0.56	0.14	0.13	1.55	1.19	0.17	0.55
Pain Ratio	0.22	0.00	1.89	0.14	0.48	0.08	7.58	0.98	0.30	1.77
Correl with SPX	-0.12	0.08	0.04	-0.09	-0.18	0.07	-0.09	0.19	0.14	0.04
Correl with UST	-0.01	-0.07	0.38	0.05	0.05	0.48	-0.10	0.13	-0.08	-0.03
Skewness	-0.06	-0.26	3.05	0.83	-0.15	0.00	0.20	0.34	-0.14	-0.08
Kurtosis	1.94	3.44	30.97	5.71	0.86	1.85	[illegible]	[illegible]	[illegible]	[illegible]

Source: J.P.Morgan Macro QDS

具体细节参见JPM在2013年的一个报告: https://www.cmegroup.com/education/files/jpm-systematic-strategies-2013-12-11-1277971.pdf

对于clustering的方法呢，也就是聚类。报告里面讲的应用细节不是很多，不知道具体金融领域是怎么应用的，而且里面直接进行对比的聚类方法太多，好多我也没见过，就不赘述了...

好了，主要的统计方法和应用就讲完了。能读到这儿的简直真爱...么么哒!

三. 随便结个尾

结尾就扯点最近的感受吧。
最近发现大家都越来越担心自己的工作是不是以后会被机器人取代，特别是人类在围棋上面被毫无悬念的横扫之后。
不学习就直接被淘汰，譬如美国的工人阶层就是一个最明显的例子，即使trump将工业的工作机会强行弄回了美国，但长久以往人类对于工人技术的需求会越来越小，最终会成为书中的历史。
但被机器人取代还是相对比较长期的趋势，而短期更需要担心的还是人和人之间的竞争。

有没有人想过为什么现在社会竞争越来越激烈，我觉得可能分为两部分，一部分是人和科技的竞争，另一部分就是人和人之间的竞争。
人和人之间的竞争加剧很大一个原因就是互联网。因为互联网降低了获得知识的门槛，使得任何人都可以通过互联网低成本地习得所需的知识参与竞争，而原地不动的人自然就会被淘汰掉。
在以前，一个人只能干一类事，因为技能被自己的生活环境局限住了，只能接触到身边力所能及的技能知识，所以一辈子可能都只接触到这一个技能和行业，但这种原始的环境实际上也是一种护城河，保护了自己相对他人的优势。但是现在互联网打破了这种隔离，把所有人都放在了同一个平面靠实力直接竞争，所有想学习的人都可以轻而易举的学习到感兴趣的领域，譬如编程和machine learning等，技能的获得几乎没有成本。这样一来，竞争的就是每个人对于知识的理解深度和自身的努力程度了，不再具有之前由于生活圈子的局限而产生的护城河，只要利用好了互联网资源，任何人都可以将自己PK下去。
这也是为什么每个人都会焦虑，因为没有安全感，自己在工作上不再是不可取代，世界上任何一个人都有可能取代自己，甚至比自己要求的工资便宜得多，所以给自己所在公司一个理由，why me?
为什么兴趣会越来越重要？因为大批对自己领域感兴趣的人可以毫无阻碍的进入自己所在的领域，还比自己更加有兴趣钻研这个领域，长期以往自己积累的优势自然就会逐渐消失。越觉得自己工作毫无挑战性，那说明自己工作的进入门槛越低，自己在未来竞争中的优势越小。
那怎么办？除了不断学习没其他办法，总不能怪别人太努力吧，特别对于中国这样一个自古就强调努力的民族。
很多人不知道自己的兴趣在哪儿，甚至进入社会之后依然在茫然。我觉得吧，其

实想找到自己的的兴趣只有唯一的一个办法。那就是一个字，试！找不到兴趣的原因无他，就是因为自己接触的东西太少了，被自己画的圈限制住了。只有保持好奇心不断去尝试没有接触过的东西，我们才能在对比中找到相对喜欢的东西，自己的视野也在寻找中变宽广了。只有找到自己喜欢的东西才能真心投入时间进去，才能一直积极主动的保持竞争力。
互联网降低学习成本不可阻挡，学习资源只会越来越廉价，知识也会在竞争中不断快速进化，不进则退也只会愈演愈烈，我们能把握住的只有不断学习的能力，找到自己的兴趣和合理的定位，利用优势建立相应的护城河，不然最终只有被取代的命运，无论是被人还是机器人。machine learning固然重要，learning machine似乎更为重要...

怎么建立起自己的优势呢?
平时少看几篇散乱的文章，多看几篇系统阐述的书。
学一个领域最关键的是建立自己的理解系统，而不是死记硬背散落的知识点。只有真的把各个点通过自己理解的逻辑连成线和网才叫真的懂了，这个才是永远不会忘的东西，也是真正的核心竞争力，具体每个点的细节并不是那么重要。
说实话不要觉得看看别人写的文章自己就了解很多了，其实了解的都是皮毛和结果，并不是真正的内功。想要真的形成自己的一套东西，必须扎扎实实的思考出整个框架和逻辑，广度都是建立在一定深度之后，没有深度作为基础的广度就是空中楼阁，遇到实际难题一碰就散。

虽然现在崇尚学科交叉还有复合型人才，但说实话，我现在的体会是切忌把技能点加太多。
我自己就走了弯路，什么都想学，结果什么学的都是皮毛。
之前我是对于任何事情都是非常好奇，物理金融统计量化心理哲学甚至各种运动和做菜，什么都想学并且都想做好，后来发现任何技能点都是需要长时间全身心投入才能做到极致的，而人的精力是有限的，不可能真的什么都学会，反而会把自己仅有的时间碎片化。而我又是一个不喜欢事情只做一半的性格，结果就导致放在waiting list里面的领域越来越多，徒然给自己压力。

对于绝大部分人来说，就像投资一样，如果你把资金全部分散化，确实分散了风险，但永远不可能有很好的收益。一个是深度，一个是广度，也就是收益和风险一个硬币的两面性。从个人来说相当于自己把总量不变的时间分配给不同的技能，虽然确实让自己更加有综合能力，但同时损害的也是自己长久发展的潜力。

全部分散不好，但是集中在一个技能上面同样也会有非常大的风险，时代毕竟一直在变化，谁也保不准自己的技能在未来是不是会被淘汰。当然不排除有人的edge就是重仓，但前提是你的深度可以超过大部分跟你同台竞争的对手，如果不是，那只能说是无谓增加了风险。

很少有人既做到深入专研，又做到对领域外也了如指掌。即使在金融领域，也是在深度的基础上不断积累出的广度，而不是一开始的目标就是广度。不然只会懂得皮毛，并且三心二意。
所以人最重要的就是发现自己擅长的地方，并且从自己擅长的技能里面挑最擅长或者最喜欢的一两个专注的去做，这样才能扎扎实实的make a difference。这种balance的trade-off其实是生活里非常简单的道理，但是往往最容易忽略。

“吾生也有涯，而知也无涯。以有涯随无涯，殆已！”

与君共勉~

下个月要做美国高能物理年会的报告，也还有好多quant的书和报告都没来得及看，所以写的略潦草请见谅哈~这只是我自己统计和金融学习过程中的一个总结，如果还能帮助到同样对machine leanring应用在金融上面感兴趣的童鞋那就最好咯，还是那句话，知识是少有的越分享越多的东西，希望大家也都乐于分享，对自己也是一个总结夯实的过程。想要一起讨论的童鞋也非常欢迎呀，我的微信号是693960928~

最后附上一句话，

“***让你遇到麻烦的不是未知，而是你确信的事并非如你所想***” — 马克吐温

送给处在正开始深刻变革的历史转折点的我们。

参考文献与推荐书籍:

1.《Machine Learning and Alternative Data Approach to Investing》JP Morgan
2.《Systematic Strategies Across Asset Classes》JP Morgan
3.《Momentum Strategies Across Asset Classes》JP Morgan
4.《An Introduction toStatistical Learning》
5.《The Elements of Statistical Learning》
6.《Active Portfolio Management》
7.《Quantitative Equity Portfolio》
8. 若干知乎问答
9. 裂墙感谢身边给予极大帮助的盆友们!
10. 还有我自己...嘚瑟一个，哈哈