教你Machine Learning玩转金融，从入门到放弃（上）

江海 对冲笔记 2017-06-24

呐，当你们看到这么搞个大新闻的题目的时候肯定内心的活动是这样的

但是我保证你们看完之后一定会变成

学习宏观和machine　learning之后发现这两个真是天生一对，这也是我为什么想要写这篇文章。
(对宏观感兴趣的童鞋可以参见我之前在知乎上写的一篇自学笔记《Macro入门note》)

接触金融的这两三年发现大家都喜欢吐槽宏观报告，报告里一条条理由说的都对很有道理令人信服，但结论总是变成了random walk甚至反向指标。

我也一直觉得这个现象挺有意思，后来学了machine　learning之后从统计的角度想了想可能是这个原因:
宏观系统本身非常复杂，变量因子多，即使逻辑线以线性关系为主，但因子和因子之间互相还有correlation和dependence使其结构更加复杂，况且还有人这个非理性的参与者randomly影响，掺入了很多有内部结构的噪音。最重要的是因子的权重还是不断动态变化的，即使报告里说的理由都是对的，但是如果并不是当下主要的driver因子，那结论也会是错的。

大家都喜欢说趋势和周期，但到底是个什么没人说得清楚。
我现在理解的趋势只是一种市场运动的表象结果，深层原因是在趋势背后的driver是各个不同因子的相互作用(说起来有点像我们高能物理里基本粒子之间的对撞interaction呢~)。长的趋势有长的因子，短的趋势有短因子。周期则是一个框架将无数因子进行分类和重组，长周期将长期的因子给于动态权重互相作用在一起组成长周期的系统，中短周期亦如是，然后在不同的时间跨度上共同描述我们的整个经济机器的运行。宏观是势，微观是机。

而且对于不同市场，因子也会不一样。譬如国内股市散户多，所以非常容易产生一些很神奇的事情，前几天看了一篇文章就说比如股票分拆都可以算一个很强的因子，因为分拆之后价格变低而国内散户多，大家都觉得低的价格会涨起来。这是很没有根据的，公司的价值怎么可能因为拆股了就市值翻了几番，但是就是发

生了。就是因为国内都是炒预期，并且大部分人由于不专业都是跟随其他人，自然而然预期就很容易就成型，并硬生生产生了自我实现的趋势。但是这个趋势只是空中楼阁，并非真实存在的基本面变化。
这可能也是为什么国内市场的波动大，波动大并非因为市场经常有分歧，反而是因为市场太容易形成趋势，并且这个趋势很容易就被证伪而戛然而止，所以形成的趋势非常脆弱，一旦不能继续维持之后就会反弹式向反方向运动，即强化版的反身性。

好的trader本身就一直在市场里面，即使对长的宏观因子不甚了解，但像A股这样抓住短期市场情绪的driver因子也够收益了。既能做好交易又能动态抓住长远宏观因子从而抓住大趋势的人就简直是凤毛麟角了，一般也都直接去买方当大佬自己赚钱了。

当然，也不是说报告就不用看了，其实看报告是一个不断丰富自己factor pool的过程，但是怎么判断其权重就得从平时对金融系统的观察和思考里面来了。
市场中每时每刻都有不同的声音，大部分时候其实大家说的都有道理，并且都可以在教科书和历史中找到对应，而最大的问题并非是判断谁对谁错，而是判断当下甚至未来给予每个factor的权重是多少，和历史的最大不同也并非是共有的ingredient不同，而是其配比不同，并且这个配比是动态变化的。我觉得这也是一个trader或者投资经理水平高低的决定性因素了。
分析师(特别是买方!)如果想要以后有更广阔的空间不仅需要了解所有影响的因子，更需要掌握每个时段各个因子的权重，想要做好portfolio还是得对因子和市场同时都理解很深刻才行。

引用马克思的一句话，"发展的观点看问题，善抓主要矛盾”。

也正是因为金融市场因子繁多并且动态的特性，machine leaning作为一个分析

工具更有用武之地，因为各种统计方法可以帮助我们筛选出因子，并且能给出这些因子的权重，帮助基金经理和fund做出更合理的决定。
并且我已经看到了一个趋势，那就是随着数据越来越多，machine learning之类工具的运用也越来越多，信息的处理速度也越来越快，之前可能需要花很长时间和人力去收集和分析出来的东西，也许现在就只是建个模run几个参数就可以得到。这样的话市场的反应速度也会越来越快，以前传统的方式因为结论总是滞后就越来越没有价值，毕竟策略的时效性在金融市场是很重要的。对于个人而言，如果自己没有相应对数据进行处理的能力(这里不单单包括machine learning)，那么相对其他人的edge就会越来越被限制，直到被市场淘汰。
那也许就有人会问，以后是不是金融领域就全部招程序员和data scientist，不需要基金经理了呢？其实也不是，因为金融的核心还是对市场的intuition，处理数据的技术只是作为增加攻击力的武器，如果仅仅单纯依靠武器很容易伤到自己。想舞倚天剑屠龙刀，那也得有张无忌的内功才行对吧~
所以我觉得以后对portfolio manager的要求会越来越高，首先最重要的是具有传统金融市场的intuition，同时还需要掌握怎样用技术处理数据得到自己想要的信息(当然这部分其实也可以交给下面的quant去做，但至少需要了解大概是怎么回事，也就是这篇文章科普的目的了)。

好了，讲了这么多，其实就是围绕一个主题：这篇文章很重要！哈哈。

还有一件事，上次写Macro入门的时候好多人吐槽中英文夹杂的问题。
说实话...我是故意的。
很多时候用中文很难准确表达一个英文的意思，硬翻译过来反而让理解更加困扰，至少不是那么直观。就拿物理来说，本科时候都是翻译过来的中文教材，很多词本来在英文里面是非常易懂的，但翻译成中文之后就完全搞不懂了。金融和统计也一样，我希望尽量保留原汁原味的知识和这个英文单词的语境。而且英文这个东西的重要性我也不需要再复述，特别是现在竞争压力最大金融和科技行业本身就是发源于英语国家，想要在激烈竞争中不落后与人，从英文学习到一手的信息是很重要的。如果永远都只读别人翻译成中文的东西(还得依仗别人的心情和信达雅的文学程度)，无端的增加了自己的学习难度，并且由于获得的信息一直都是滞后的永远都会处于信息上的劣势。

铺垫完背景，接下来就进入正题咯~

本文分为三部分

一. 金融和统计背景介绍

二. machine learning各个方法和在trading上的应用

2.1 Supervised Learning: Regressions

2.2 Non-Parametric Regression: K-Nearest Neighbor and LOESS

2.3 Tree Based Method（Random Forest 和 Extreme Gradient Boosting）

2.4 Classification（Logistic Regression 和 Support Vector Machine）

2.5 Unsupervised Learning（Principal Component Analysis 和 期权科普）

三. 随便结个尾

首先声明，修炼此功的基本功如下，如果不具备这些基础强行练功而走火入魔的话，我敬你是条汉子~

1.数学
2.数学
3.牢记第一，第二条

一. 金融和统计背景介绍

呐，首先的首先，在用machine learning建模的时候一定要分清楚X和Y。

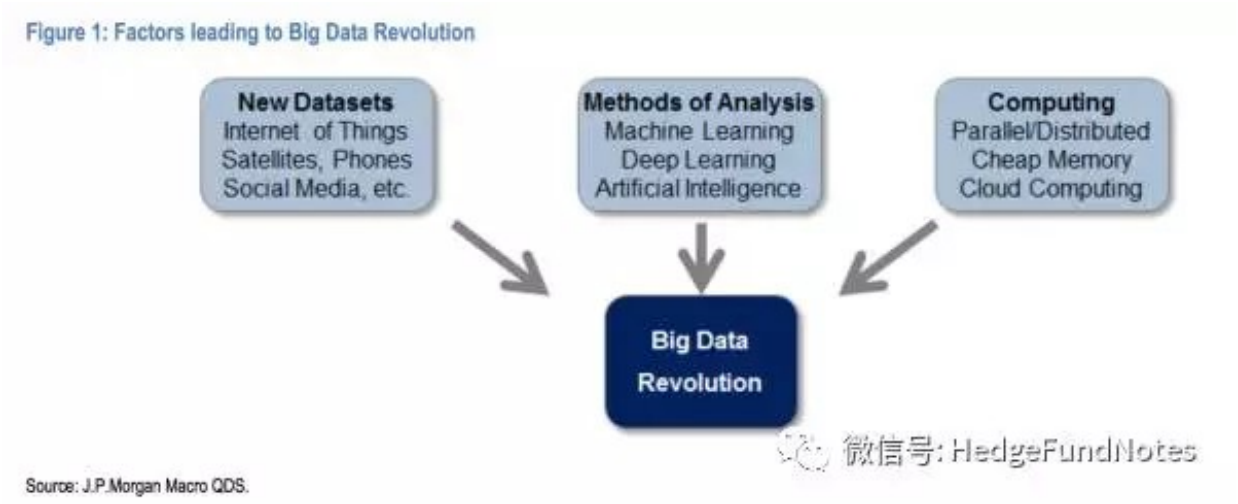
大部分时候我们利用统计方法要做的事情就是，利用Xi去预测Y。也就是属于supervised情况(既有X又有Y)。
这里Y一般是涨跌幅或者涨跌的方向，Xi一般就是对Y起影响作用的因子。

最重要的是我们要先弄清楚我们的Xi是什么，我们要预测的Y又是什么。至于Xi是linear还是quadratic，都仅仅是模型处理的细节。

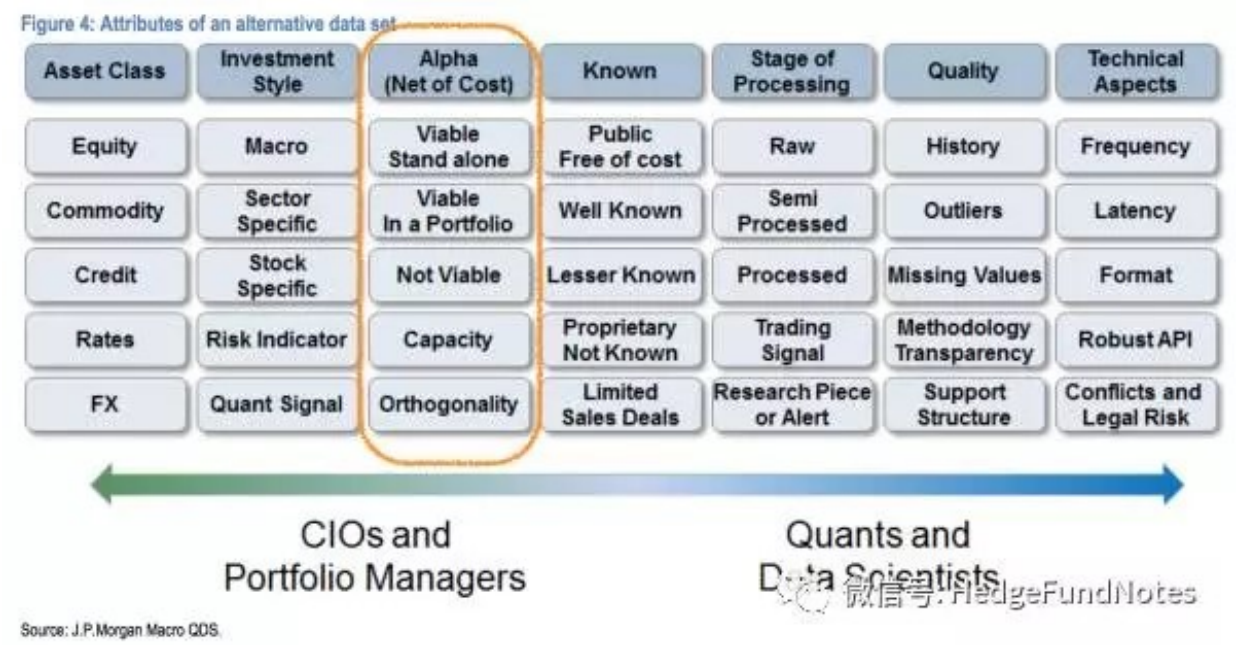
还有另一部分时候，我们想要了解的是Xi与Xi之间的关系，所以用到的就是unsupervised方法(因为这里没有Y)。

这篇文章里金融上的实例都是用的JPMorgan近期一个非常火的报告，《Machine Learning and Alternative Data Approach to Investing》，至少大家会信服一点吧...

我选了几张图介绍当下Big Data和金融方面的背景:



具体金融上的大概分类:



金融市场不同sharp ratio程度的因子对应的应用方式:

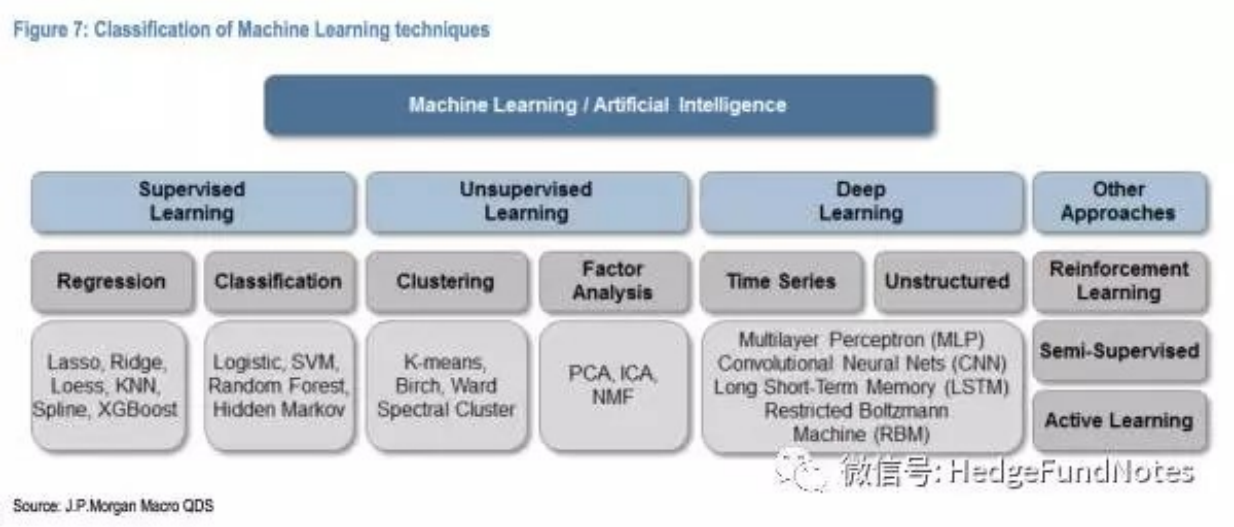
上面这张图其实挺重要的，潜台词其实说的就是怎么判断单个因子的好坏，和怎么用这个因子。
市场上大部分为人们所知的因子都不能成为可以独立运行的策略(也就是图中的not viable as StandAlone Sharp, but viable in a portfolio context for quants)，因为知道的人越多就越不可能成为可以独立运行的策略，但是能加入策略的因子都是有足够大的sharp ratio，然后利用machine learning的方式(也许是unsupervised的主成分分析的方式)组合起来变成一个更strong的策略。

大概有几步，
首先需要得到可靠处理好的data(比如直接从交易所拿到的量价数据etc和做数据清洗之类)，
第二步从数据中利用machine learning得到sharp rato足够大的的单个因子或者策略Xi，然后把这些因子组合起来形成策略池。
第三步就是给这些选出来的因子分配动态的权重组合成一个portfolio，可以是简单的线性分配weights，也可以是高维甚至多重嵌套(这里更复杂的combine因子可能就不能用简单的线性supervised和unsupervised模型给weights去操作，我猜可能需要其他的方法)。
最后一步就是看这些合成起来的总因子的回测表现，算出来总的sharp ratio。

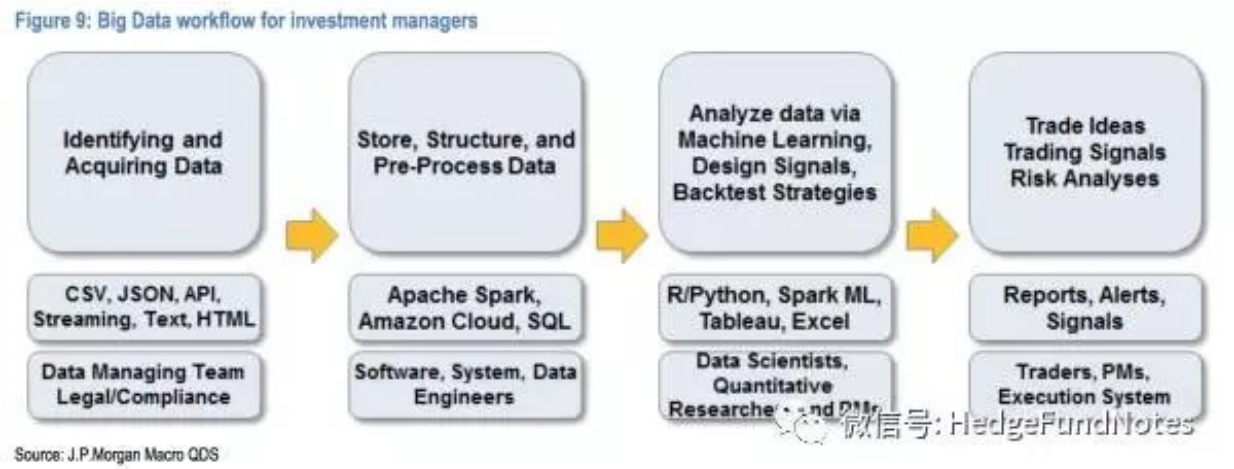
至于给权重的办法，
大概的传统办法就是投资组合，Markowitz Mean-Variance Model和经过改

进更为主观也更flexible的Black-Litterman model。
还有的就是动态的投资组合方式，具体我也还在努力啃active portfolio management那本圣经...也许到时候会写写这方面(如果到时我还活着...)。
至于machine learning来进行动态给权重，我也不甚了解...希望有热心的读者可以指导哈！

下面这张图就是不同machine learning方法处理不同问题的分类:



下图是big data应用在金融上的标准化工作流程~



金融想要解决的问题和其对应的machine learning方法:

Figure 41: Typical tasks and frequently used Machine Learning methods

Question	Data Analysis Technique
Given set of inputs, predict asset price direction	Support Vector Classifier, Logistic Regression, Lasso Regression, etc.
How will a sharp move in one asset affect other assets?	Impulse Response Function, Granger Causality
Is an asset diverging from other related assets?	One-vs-rest classification
Which assets move together?	Affinity Propagation, Manifold Embedding
What factors are driving asset price?	Principal Component Analysis, Independent Component Analysis
Is the asset move excessive, and will it revert?	
What is the current market regime?	Soft-max classification, Hidden Markov Model
What is the probability of an event?	Decision Tree, Random Forest
What are the most common signs of market stress?	K-means clustering
Find signals in noisy data	Low-pass filters, SVM
Predict volatility based on a large number of input variables	Restricted Boltzmann Machine, SVM
What is the sentiment of an article / text?	Bag of words
What is the topic of an article/text?	Term/InverseDocument Frequency
Counting objects in an image (satellite, drone, etc)	Convolutional Neural Nets
What should be optimal execution speed?	Reinforcement Learning using Partially Observed Markov [illegible]

Source: J.P.Morgan Macro QDS

我们可以看到右边有很多的统计方法，每一个都得了解背后的机理，所以这真是一个长期的学习过程呢。

接下来在介绍具体的machine　learning方法和应用之前，需要介绍统计领域一个最最重要的概念，那就是bias和variance和trade-off，换句话说也就是overfitting的问题。

我们用统计方法的一般流程是，首先我们将的dataset分成两部分，一部分是training sample，是用来确定(train) model的；另一部分是test sample(或者validation sample)，是用来看这个model效果的。

If we are in a data-rich situation, the best approach for both problems is to randomly divide the dataset into three parts: a training set, a validation set, and a test set. The training set is used to fit the models; the validation set is used to estimate prediction error for model selection; the test set is used for assessment of the generalization error of the final chosen model. Ideally, the test set should be kept in a "vault," and be brought out only at the end of the data analysis. Suppose instead that we use the test-set repeatedly, choosing the model with smallest test-set error. Then the test set error of the final chosen model will underestimate the true test error, sometimes substantially.

It is difficult to give a general rule on how to choose the number of observations in each of the three parts, as this depends on the signal-to-noise ratio in the data and the training sample size. A typical split might be 50% for training, and 25% each for validation and testing:

在各个统计方法里面，很难找到一个方法是好到完全dominate其他所有的方法，一般都是各有利弊，主要的原因就是因为我们有bias和variance和trade off。

直观一点理解的话，variance代表的是第一步选出来的model对于我们用的training sample的依赖性。我们自然是希望从training sample中选出的模型能很好的fit整个dataset，但比如我们重新做一次实验，将我们的dataset分成新的training sample和test sample，如果从training sample得到的model是跟第一次得到的model完全不一样，那么就说明这个model并不能很好的描述我们的数据。这个对于training sample的选择的依赖性就是variance，我们是希望它越小越好。

对于bias呢，就是我们得到model之后在test sample上面测试这个model的好坏。一般越是复杂的model，在training sample上预测的效果越好(因为overfitting)，也就是bias越小。但是同时越复杂的model对于training sample数据的依赖性就越强，很可能换一个training sample就得到完全不一样的model，这样模型会非常不稳定，因为统计上我们是假设所有的数据是从同一个真实的模型f(x)中产生出来的，我们的任务就是从已有的数据中找到最接近这个真实的模型f(x)的模型。

所以我们希望得到bias和variance之间的平衡点，也就是希望bias和variance之

和最小(MSE，Mean Squared Error就是用来estimate他们之和的一个指标)。

我们可以看一张bia和variance直观的图:
图中的点是从真实的model f(x)产生出来的，我们想做的就是用不同flexibility程度的统计方法去fit这些点，看看哪种方法的拟合效果越好。

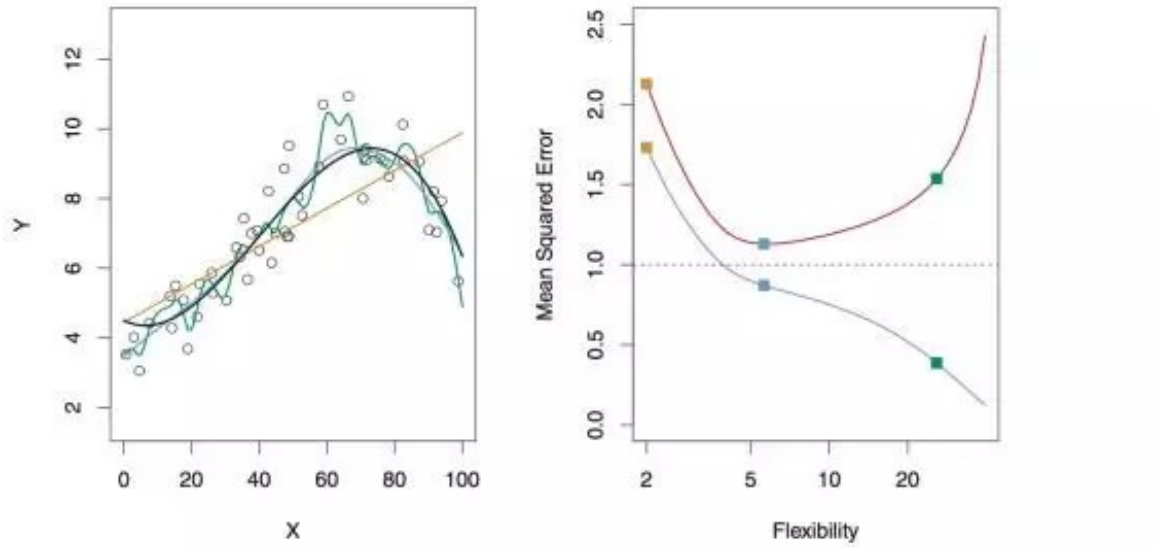


FIGURE 2.9. Left: *Data simulated from f, shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves).* Right: *Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line). Squares represent the training and test MSEs for the three fits shown in the left-hand panel.*

这里左图中黄线是linear regression模型，蓝线和绿线是smoothing spline模型。
右图是flexibility(可以看做对于data的依赖性)为横坐标，他们各自的bias和variance之和也就是MSE为纵坐标。红色那条线就是将模型用在test sample得到的test MSE的结果，我们可以看到是一个U型，也正是说明了bias和variance之间是有一个平衡位置使得MSE最小。linear regression模型(黄色的点)的MSE比较大，因为左图中可以明显看到其对这些点的线性回归效果并不好，但是是不是最扭曲的绿线smoothing spline模型就效果最好呢，也不是，我们可以看到红线上绿色的点的MSE大小随着flexibility的程度又翘上去了，反而是蓝色的模型在bias和variance之间取得了平衡。
至于灰线，它是training MSE，我们一般不关心，因为training set的信息只是用来获得model的，对于衡量model的预测效果不能提供有用的信息。

所以我们可以看到并不是模型越flexible越扭曲越好，而是需要在bias和

variance之间进行一个平衡。

下面还有变化了真实的model f(x)的情况下，同样用这三种model拟合的MSE结果:

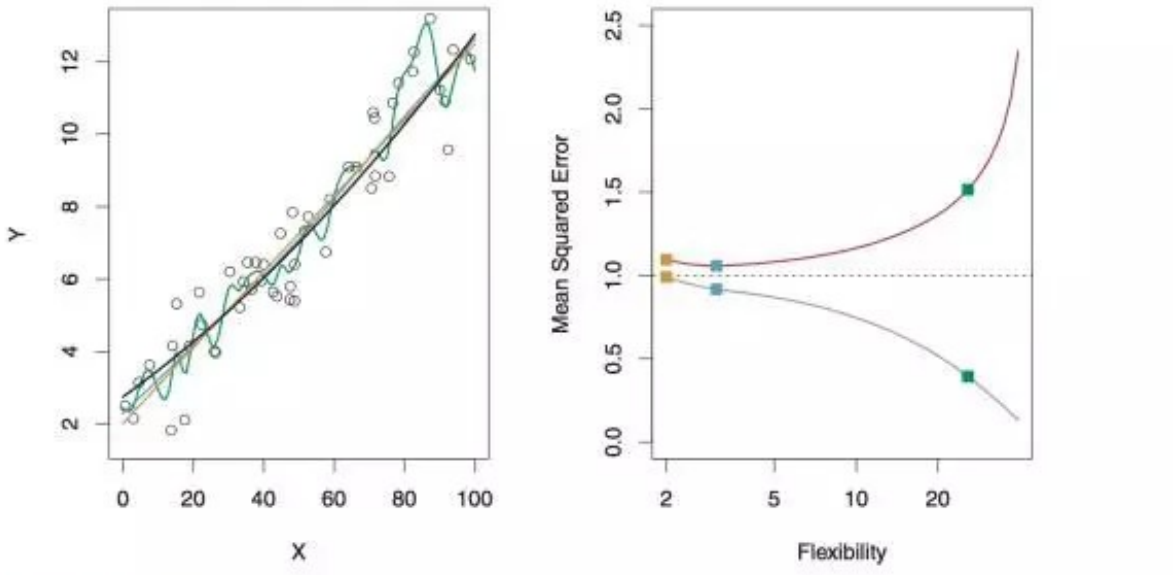


FIGURE 2.10. *Details are as in Figure 2.9, using a different true f that is much closer to linear. In this setting, linear regression provides a very good fit to the data.*

上图是f(x)接近linear model的情况。

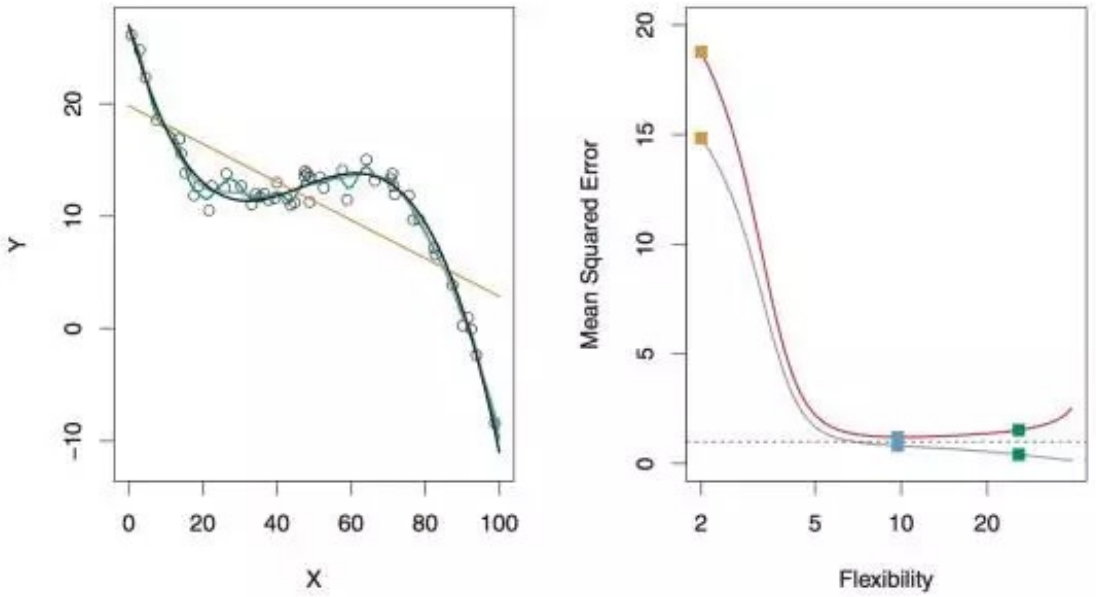


FIGURE 2.11. *Details are as in Figure 2.9, using a different f that is far from linear. In this setting, linear regression provides a ve…*

上图是 f(x)为非线性model的情况。

我们可以看到如果我们的model跟真实的f(x)的形式越match，那么拟合的效果就好越好。

接下来我们看看理论上公式是怎么推导出bias和variance的(这块儿内容可能需要一定统计知识，如果不感兴趣的也可以跳过，但是想要真的理解这两个东西还是得从最本质的公式入手):

$$E\left[(Y-\hat{f}(X))^2\right] = [Bias(\hat{f})]^2 + Variance(\hat{f}) + IrreducibleError$$

Here, bias is defined as

$$Bias(\hat{f}) = E_X\left[f(X) - \hat{f}(X)\right]$$

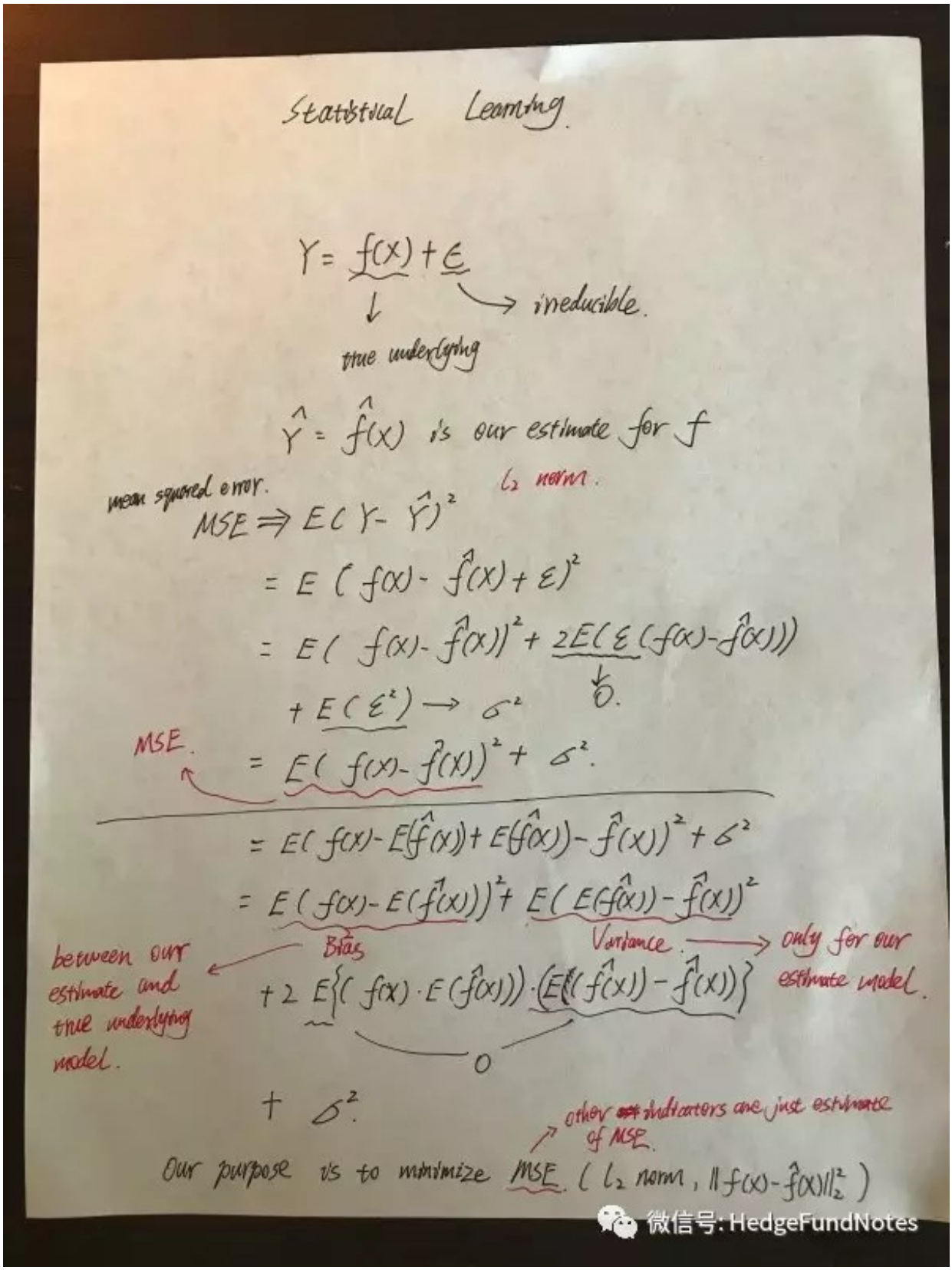
And variance is defined as

$$Variance(\hat{f}) = E_X\left(\left[\hat{f}(X)\right]^2\right) - \left(E_X[\hat{f}(X)]\right)^2$$

And irreducible, random error is

$$IrreducibleError = \sigma^2$$

我也把之前写的推导草稿贴出来吧...



从上面的公式里面我们可以看到看到Bias和Variance分别是代表了什么和是怎么算出来的，其实bias和variance都是针对training sample取的期望值。

统计从理论上的做法是假设在dataset之下有一个fundamental的真实model

f(x)，然后Y是由f(x)加上一个不可消除irreducible的随机量ε构成，ε是符合正态分布的一个随机数。然后我们是想要从data sample中找到一个model f^hat(x)使其尽量靠近真实的那个model f(x)。

上面写的MSE其实是expected test MSE(注意我是用的箭头，不是等号)，就是在我们已有的样本基础上得到的我们的model f^hat(x)和真实model f(x)之间的平方差的期望值的估计。求期望呢，实际上是对f^hat(x)求期望，因为整个MSE的式子里面唯一的变量就是我们的model f^hat(x)，而f^hat(x)又是从training sample得到的，所以我们对f^hat(x)求期望实际上是对training sample的信息求期望。

bias呢，就是真实值f(x)和f^hat(x)的expectation之间的差。所以是基于数据求出来的error。如果过拟合了呢，就会使得这里的error会非常小，也就是bias会很小。
variance呢，是我们模型f^hat(x)本身的expectation和它本身之间的差值的期望值，也就是仅仅跟我们的model选取有关，如果过拟合了training sample呢，那么我们选取的model就会非常volatile，也就是这里的variance会非常大。

这里的MSE仅仅是针对x0这一个点，真正的MSE其实是应该遍历所有的点，也就是通过无数次随机选取training sample得到的distribution，再对这么多training sample的distribution求期望得到的值。但在现实中我们不可能做到取无数组training set，所以我们只能用sample去estimate整个population的MSE。我们就用test sample中的所有点作为整体MSE的估计值，或者用cross-validation的方式来随机取一定数目的training sample和test sample，并且对于这些test sample的test MSE求平均来估计整体的MSE。详细内容可以见《the elements of statistical learning》中的5.5.2和7.3这两节。

并且在实际中因为我们不可能知道产生这些点的真实model f(x)，所以是不可能单独观察bias和variance的，我们能得到的只有他们的和MSE的estimate，也就是那条U型的曲线，从中我们可以找到最低点，也就是bias和variance最平衡的model。
对于U型曲线，实际上左半边的U型是我们的model不断逼近真实model的结果，也就是bias在一直减小并且减小的部分大于variance增加的部分，所以使得整体MSE越来越小，直到到达某一个最优点；然后当flexibility继续增大的时

候，我们的model就会开始用越来越多的parameters去拟合training sample中的irreducible error，但这些error由于是random的噪音，实际上是对我们的预测没有任何帮助的，这也就是overfitting的来源，就使得我们选取的model非常volatile让variance增大的部分大于bias减小的部分，也就导致了U型曲线的右半边随着flexibility的增加使得总体的MSE又翘上去了。

除此了bias和variance的trade-off之外，还有一个trade-off是prediction accuracy和model interpretability。有些复杂的model(比如非参或者deeplearning等等模型)预测起来奇准无比，但是理解起来会不是那么直观，甚至完全不知道内部发生了什么，在应用的时候就会造成困扰，比如别人问你为什么是这个结果的时候我们是说不出个所以然来的，无法令人信服；另一方面，像简单的model比如线性模型就非常好解释是怎么预测的，每一个变量都很直观，但是问题就是很多时候预测效果不好，特别是对非线性的问题。这个也是需要针对实际情况进行取舍。

看到这里是不是已经一脸懵逼了呢~

不要怕！马上就到应用部分了！

二. machine learning各个方法和在trading上的应用

这里报告里面只大概介绍了各个方法怎么运用，但是对于一般没有统计基础的读者来说还缺少了事前对于每个方法本身的介绍。我这里想做的事情就是先从统计的角度介绍每一个方法，然后再解释其在金融上面怎么应用(大部分的书都是要么单纯关于统计的推导，要么直接写金融，但我觉得其合起来一次性顺着看下来其实是最符合理解整个过程)。

每一个方法我都会大概的介绍其数学推导的机理和方向(为了尽量使得每一个方法简洁易懂，所以需要牺牲一定程度的严谨性，希望写的不准确的地方大家见谅)，并且加上我自己的理解给出其相应在金融上具体是怎么应用。
还得强调一下，这里并不是说这些推导不重要所以才省去，而是极其重要以至于很难在一篇文章里面用文字说清楚，推导的细节才是这些方法的核心和精华，所以如果不想只懂皮毛的话还是需要细心去思考每一个方法的构建过程，毕竟其应用的利弊和假设条件都隐藏在芸芸的细节之中。

首先我们需要知道一些区分统计方法的概念:
一个是parametric和non-parametric，一个是supervised和unsupervised，还有一个是regression和classification，分别是代表了问题的不同的特性或者要求。

parametric和non-parametric的区别是model中有没有参数。
supervised和unsupervised的区别是有没有Y，比如我们只研究Xi之间关系的话就是属于unsupervised。
regression和classification的区别是在有Y的supervised方法下面，Y是定量的(quantitative，比如涨跌幅度)还是定性的(qualitative，比如涨跌方向)。

接下来我们就开始介绍各个不同的统计方法和其应用:

2.1 Supervised Learning: Regressions

对于supervised方法，我们最重要的就是要弄清楚我们想要predict的是什么，我们有的因子又是什么；也就是分别弄清楚在实际问题里面Y是什么和X是什么。

Penalized Regression Techniques: Lasso, Ridge, and Elastic Net

对于处理线性的问题，我们可以用ridge还有lasso
好处是非常易于理解背后的关系，坏处就是对于non-linear的问题fitting效果不好，弱点还有变量Xi数目比较多，或者变量之间有correlation的时候也会效果不好。但是这个是所有regression方法的基础，所以必须掌握。

一般的形式如下：

In ordinary linear regression, we forecast the value y to be a linear combination of the inputs $x_1, x_2, \ldots, x_n$. In short we assume that variable y has 'Betas' to a number of variables x (plus some random noise).

$$y = \beta_0 + \sum_{i=1}^{n} \beta_i x_i + \varepsilon$$

Xi是变量，β是相应变量的系数，ε是随机的一个不可消除的误差，Y是我们想要预测的目标。

比如一个具体的形式是

这里的US growth，EM growth，US HY Bonds，US Equities，Global Equities是五个变量Xi，想要预测的Y就是Asset Price。

这里我们可以看到US Equities和Global Equities前面的系数相对其他变量大很多，这就说明了这两个变量对于这个Asset Price的影响最大，当然US Equities是正面的影响，Global Equities是负面的影响。

这里我们就可以发现一个问题，那就是比如像EM growth这个变量前面的系数非常小，说明这个因子对于Asset Price的贡献不大，所以如果依然把这个因子加入model的会就会导致过拟合，因为本来这个因子是不应该在model里面结果我们却放进去了，自然就导致变量更多，model更复杂增加了其对于data的dependence，也就是variance。

当变量比较少的时候我们自然可以手动把这些不需要的因子剔除掉，但是如果变量有成百上千甚至上万的话，人工来选择因子就会非常繁琐，而且还容易造成错误。有没有一种方法可以自动帮我们做这件事情呢？有，就是接下来我们说到的Lasso方法。

As we saw in the numerical example above, this minimization is not stable and can yield spurious and/or large values of betas. One way to prevent that from occurring is to change the objective function in the minimization above. Instead of minimizing the least-squares objective, we can modify it by adding a penalty term that reflects our aversion towards complex models with large 'betas'. If we change the objective to include a penalty term equal to the absolute value of the beta coefficients, i.e.

$$\text{Lasso: Minimize Historical Sum of } \left(y - \left(\beta_0 + \sum_{i=1}^{n} \beta_i x_i\right)\right)^2 + \alpha \sum_{i=1}^{n} |\beta_i|,$$

then the optimizer will set 'unnecessary' and very large betas to zero. The addition of a penalty term equal to the absolute value of the coefficients is called L_1 regularization and the modified linear regression procedure is called Lasso (or LASSO). By concentrating only on the most relevant predictors, Lasso performs an implicit feature selection for us[30].

L1 norm

The objective function for Lasso can be understood as follows:

- If we set $\alpha = 0$, then we recover the coefficients of ordinary linear regression
- As α increases, we choose a smaller and smaller set of predictors, concentrating only on the most important ones

Lasso做的事情就是加入一个penalty因子，使得自动将相关性不强的因子自动筛选掉，也就是使得其前面的系数变为0。

然后还有一个方法，就是ridge。唯一不同的地方就在于ridge的penalty函数是系数的平方之和，而不是Lasso里面系数绝对值之和。

其实他们做的事情都比较类似，都是为了减小不相关因子的系数大小。

但是这里有一个问题，那就是如果有一个因子确实是一个非常强的因子，并且前面系数非常大。如果用Lasso和Ridge的话就会使得其前面的系数减小，那这样的话不是underestimate这个因子的重要性了么？毕竟系数越大的，在这个penalty下面会减小的越多。

对于这个很大系数的主要因子的问题，理论上可以证明出Lasso和Ridge并不会首先将系数大的主要因子系数变为最小，而是首先将不那么重要的因子的系数变小。可以从主观的感觉上面来理解，如果将主要因子的系数降低的话，会造成公式前面那个RSS这一项的值变得很大，这样即使后面penalty会减小，但整体的和不一定会降低，反而可能会升高(毕竟penalty只是一个调节的项，RSS才是想要降低的主体)。但是如果降低的是不那么重要的因子的系数，就会使得前面RSS的值增加不多，但是后面的penalty会降低不少，特别是penalty前面系数a足够大的情况下。所以综合而言，Lasso的作用是把不重要的因子剔除出去(也就是使得其前面的系数变为0)。

Lasso相对于Ridge的优势也正在于可以将主要因子的系数直接降为0，这样也就是一个选因子的model。

关于Lasso和Ridge之间的对比，至于为什么Lasso可以将不重要的因子的系数降为0而Ridge不行，我们可以看下面这个图更加直观：

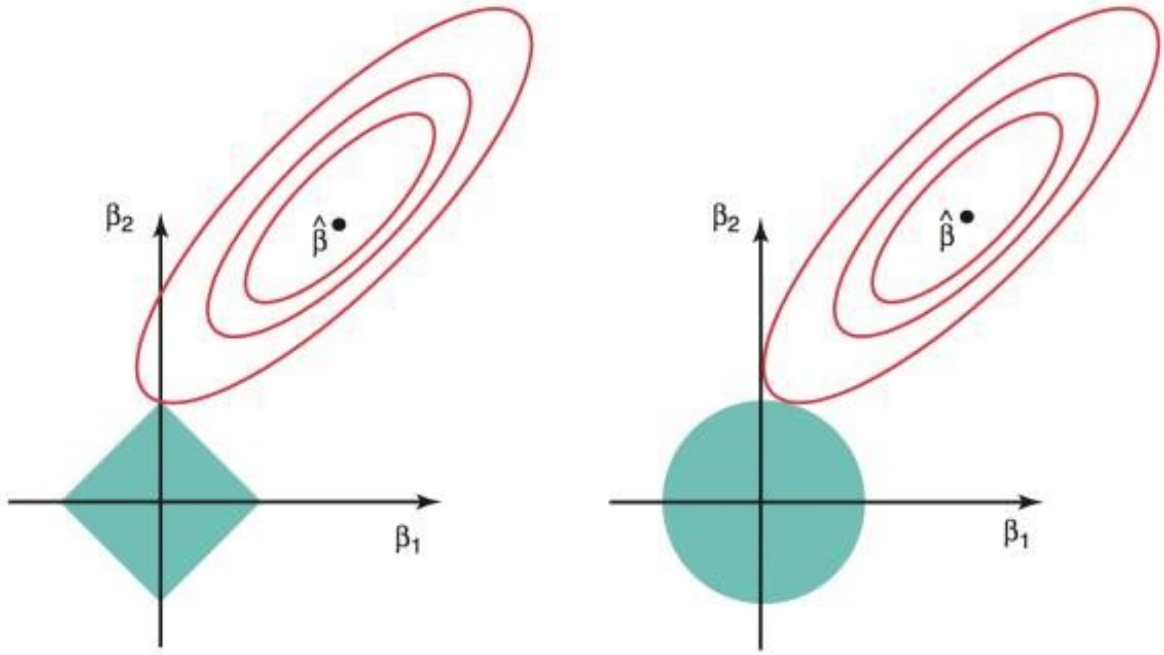


FIGURE 6.7. *Contours of the error and constraint functions for the lasso* (left) *and ridge regression* (right). *The solid blue areas are the constraint regions,* $|\beta_1| + |\beta_2| \leq s$ *and* $\beta_1^2 + \beta_2^2 \leq s$, *while the red ellipses are the contours of the RSS.*

假设我们只有两个因子X1和X2，他们分别的系数是β1和β2。然后上面那个椭圆就是前面那个loss function RSS画在β1和β2的平面上，每一个椭圆代表的是不同的RSS的值，并且椭圆越小的时候RSS越小，也就是我们想要其越小越好。对于Lasso和Ridge，因为前面那一项RSS都是用同一套数据经过同样的least square方法计算出来，所以椭圆的形状对于Lasso和Ridge都是一样的。不一样的仅仅是第二项，也就是阴影的面积。

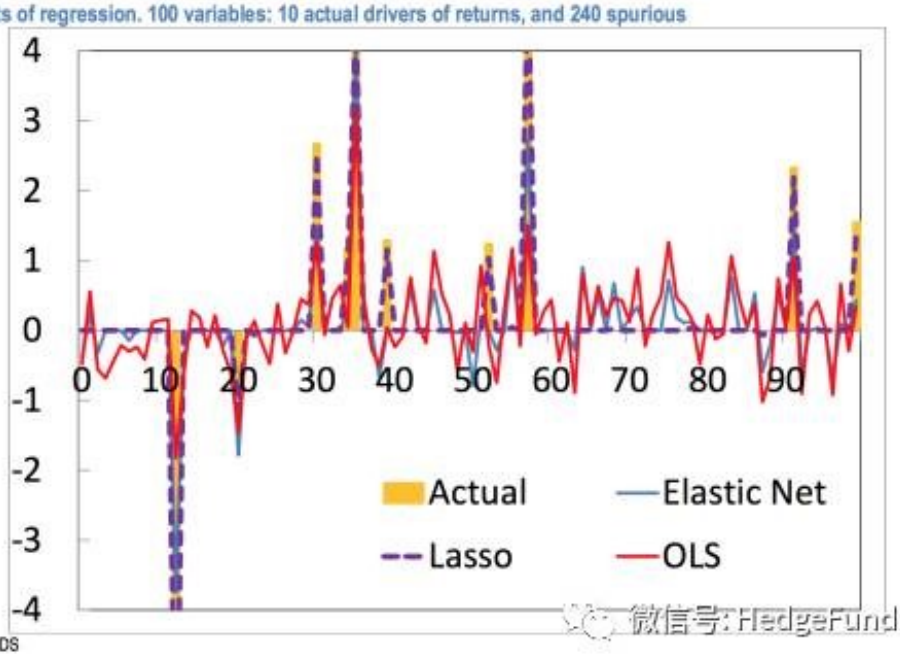
一方面我们想要这个椭圆越小越好，最好的情况就是直接缩小成中间的点β hat。
另一方面，我们又想要阴影部分的面积最小，因为我们所加的penalty一定程度上就代表了阴影的面积。
所以两个方面的平衡下，我们就需要找到这个椭圆和阴影面积的切点，这样可以使得椭圆足够小，这个阴影的面积也足够小。
现在我们知道了切点是我们需要找的点，这个点的坐标是(β1, β2)。

我们可以看到Lasso的情况是正方形的阴影面积和椭圆相切，由于正方形是有四个突出的顶点，所以这四个顶点更容易和椭圆相切。而这四个顶点的坐标都在X或者Y的坐标轴上，也就是要么$\beta 1$被留下，要么$\beta 2$被留下，其实这就是为什么可以将被去掉的因子的系数变为0(因为切点在坐标轴上)。

而Ridge的情况是，由于阴影面积是一个圆，所以并没有突出的顶点可以更容易跟椭圆相切，故想要正好相切在坐标轴上是非常困难的，需要RSS的形状满足非常极端的条件才行。

所以如果想要选择因子的时候，我们可以利用Lasso的方法来获得足够重要的因子。

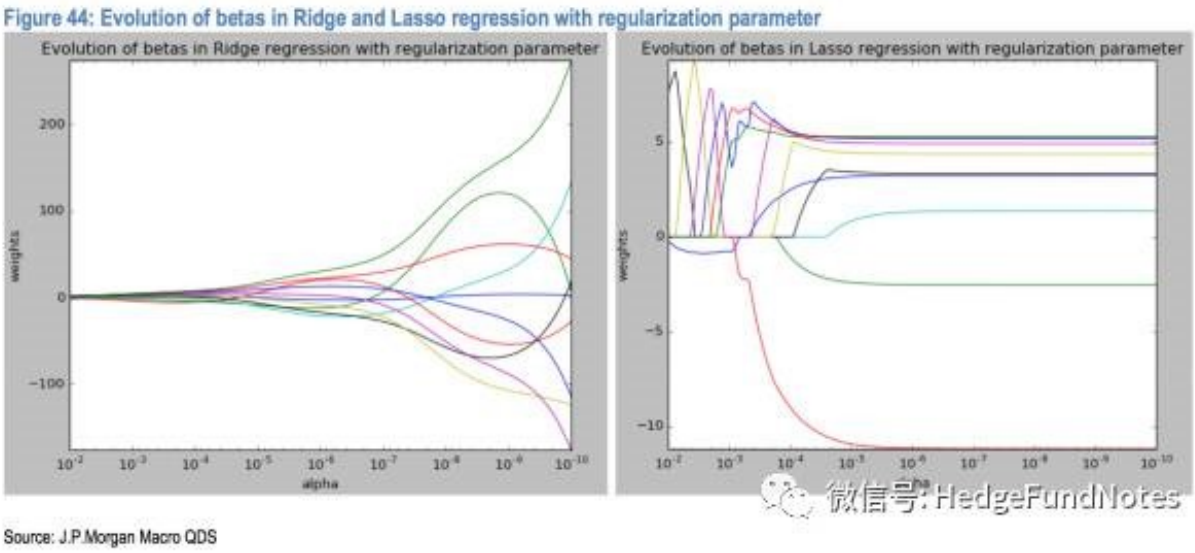
Figure 43: Coefficients of regression. 100 variables: 10 actual drivers of returns, and 240 spurious



Source: J.P.Morgan Macro QDS

从上面这张图我们可以看到Lasso基本上把真实Actual的主要因子都抓住了，并且给予的系数(权重)基本上和真实值一致。而OLS，也就是最基本的least square的方法(仅仅只有Lasso的第一项)会分配权重给全部的因子，这样就造成了overfiting。也可以看到Lasso里面加上的penalty这一项对于筛选重要的因子的还是很有效的。

下面是penalty因子系数α的大小(X轴)和得到的每个因子的系数大小(Y轴)，



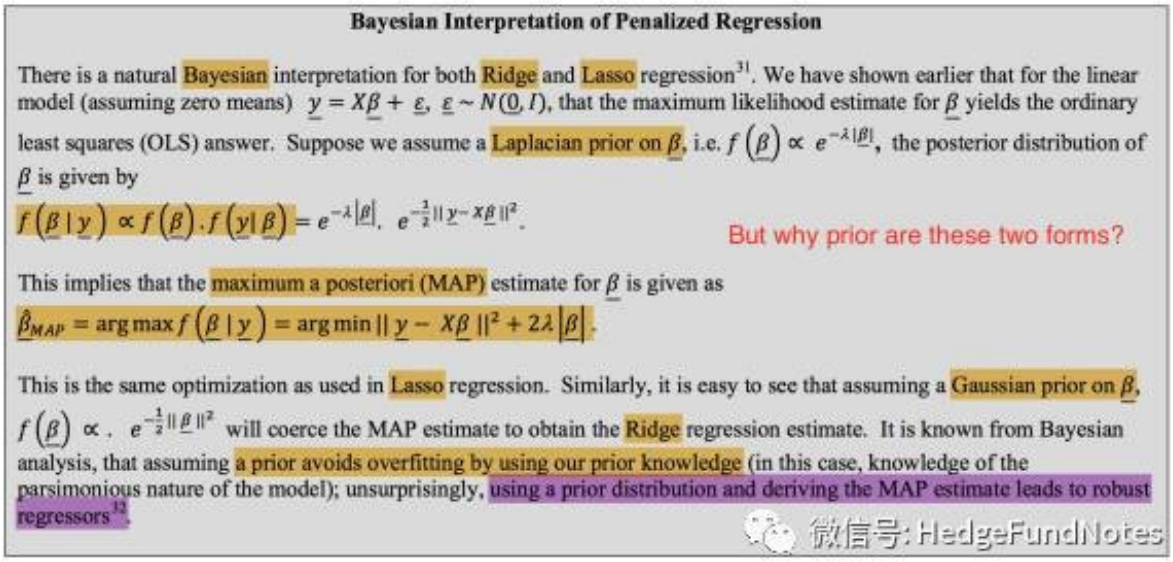
我们可以看到α的值从小到大(X轴从右往左看)的时候，右边的Lasso model会让不同的因子在不同的时期消失。也就是随着α的值增大(也就是penalty的程度越来越大)，不断有因子的系数被Lasso归0，也就是剩下的因子的重要性越来越大。最后剩下几个影响非常大的因子。
而左边Ridge的只能让这些因子的系数一起变小，最左边α值也就是penalty非常大的时候，几乎所有的因子系数都变成非常小，这样显然不是我们想要的结果。

Bayesian和ridge还有lasso的关系：

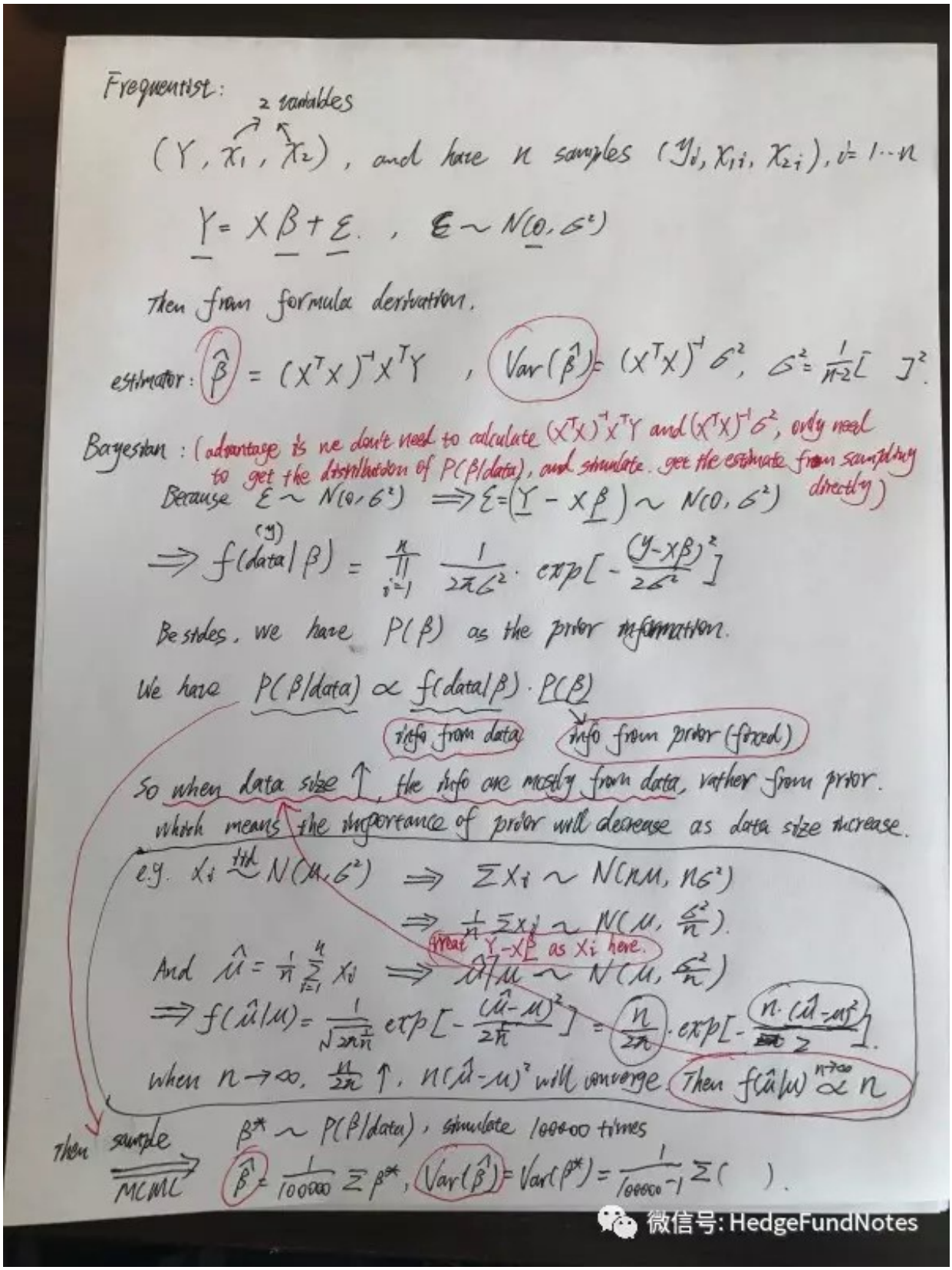
ridge还有lasso都可以由贝叶斯推导出来，只是需要改变贝叶斯里面prior的distribution的形式，ridge需要在贝叶斯里面把prior改变成laplacian，lasso需要把prior改变成gaussian。

理论上来说，这两种prior distribution的贝叶斯得出的结果应该是和ridge和lasso的结果一致。

这里用贝叶斯导出的原因是贝叶斯不需要复杂的计算公式，仅仅需要的到P(u|data)之后做simulation，可以直接从generate出来的sample里面得到系数beta的均值和方差，而不需要像frequentist那样用公式来求。(具体公式的推导在附图中)



并且贝叶斯的方法当data足够大的时候，prior的distribution其实重要性会越来越小(我自己也胡乱写了传统统计方法frequentist和下面Bayesian的证明)。不过data的size什么才叫做足够大，这是一个问题。



接下来我们介绍一个利用Lasso的例子，

We illustrate an application of Lasso by estimating the 1-day returns of assets in a cross-asset momentum model. For more background on trend following strategies see our CTA primer. We attempt to predict the returns of 4 assets: S&P 500, 7-10Y Treasury Bond Index, US dollar (DXY) and Gold. For predictor variables, we choose lagged 1M, 3M, 6M and 12M returns of these same 4 assets, yielding a total of 16 variables. To calibrate the model, we used a rolling window of 500 trading days (~2y); re-calibration was performed once every 3 months. The model was used to predict the next day's return. If the next day predicted return was positive, we went long the asset, otherwise we shorted it. Prior to regression, all inputs were standardized to avoid the problem of input features being of different scales. Performance of this momentum strategy is shown in tables below for each of the assets. Note that each of the momentum strategies performed long only, ie position in their respective asset.

这里我们想要预测的Y有4个，S&P 500，10-year UST，US dollar(DXY)，还有黄金gold。

并且我们选取了4个Xi，就是他们各自过去1M，3M，6M还有12M的收益。

这样我们对每一个Y都有4个Xi作为变量。我们想要预测的Y是此种asset第二天的收益。如果是大于0就做多，小于0就做空。
比如我们现在单独看S&P500。
我们利用的dataset是滚动的500个交易日，也就是对于S&P500有500个data值，每个data的值是一个5维空间内的一个点(Y， X1， X2， X3， X4)，也就是(S&P500第二天的收益，S&P500当天之前1个月的总收益，过去3个月的总收益，过去6个月的总收益还有过去12个月的总收益)。我们想要做的就是利用一个Lasso模型去fit得到这500个点，使得其对于这500个点fitting的最好，也就是RSS+penalty的和最小。

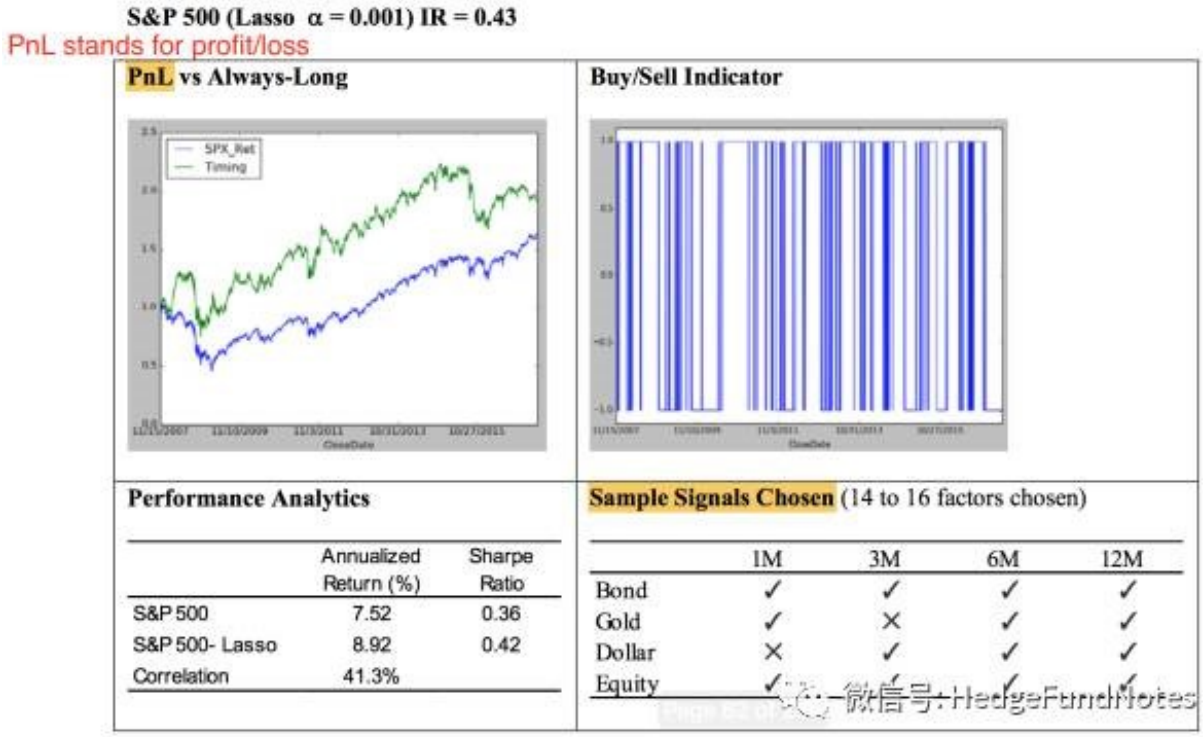
这样我们就在5维空间内找到了一条线，根据坐标(X1，X2，X3，X4)还有其系数我们就可以预测出Y，也就是在当天(接近)收盘的时候，算出来S&P500当天之前1个月的总收益，过去3个月的总收益，过去6个月的总收益还有过去12个月的总收益，也就是在5维空间里面加入今天这个点，并且因为是rolling window，所以将最初始的第一个点(今天之前的第499天)去掉，得到空间中新的500个点，重新获得一个新的Lasso model，并且根据这个update之后的model去预测第二天的收益。

具体应用中似乎是3个月重新renewal一次model。也就是并不是每天都会加入一个点再去除最旧的一个点，而是在三个月之内用同一套model结合这三个月每天的(X1，X2，X3，X4)去预测第二天的收益。等到了三个月之后，就一次性加入这三个月所有的点，去除掉最旧的相同数目的点，重新进行建模得到一个新的Lasso model，然后再用这个model结合接下来三个月每天的(X1，X2，X3，X4)来预测第二天的收益，一直rolling下去。

这里需要注意的是(Y, X1, X2, X3, X4)都需要standardize(具体怎么standardize需要弄清楚，后面很多方法都会预先将Xi standardize)，因为我们知道跨度不一样其值也会有不同的scale。

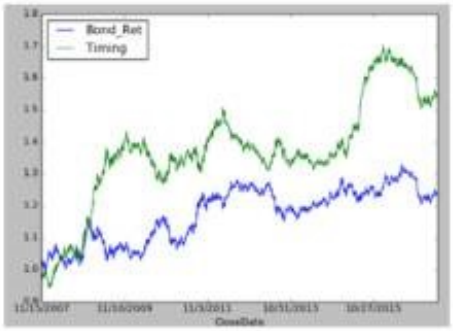
但这里是仅仅用了(X1，X2，X3，X4)作为变量，但实际上我们有4个不同的asset Y，S&P 500，10-year UST，US dollar(DXY)，还有黄金gold，
所以可以将这4个Y的(X1, X2, X3, X4)都设置成共用的16个变量(X1, X2, X3, X4, ...X16)。然后用这同一套Xi对这四种asset进行regression，每一种asset就都有一个Lasso model进行预测。

结果譬如下图，

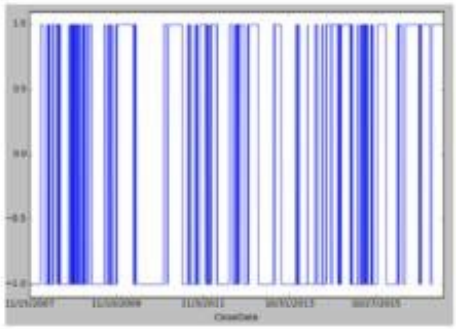


我们可以看到预测S&P500的lasso model用到了14个factors。并且单纯做多的收益和sharp ratio都比直接买S&P500高。
同样我们可以看到其他asset的结果。

PnL vs Always-Long



Buy/Sell Indicator



Performance Analytics

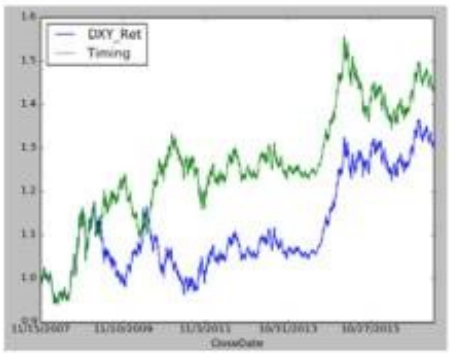
	Annualized Return (%)	Sharpe Ratio
IEF	2.30	0.32
IEF- Lasso	4.86	0.67
Correlation	-19.4%	

Sample Signals Chosen(12 to 16 factors chosen)

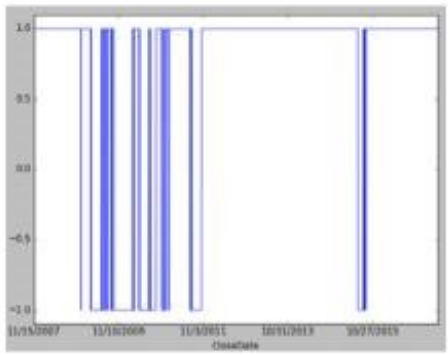
	1M	3M	6M	12M
Bond	✓	✓	✓	✓
Gold	✓	✓	✓	✓
Dollar	✓	✓	✓	✓
Equity	✓	×	✓	×

Result for DXY - Lasso ($\alpha = 0.05$) yields IR = 0.50

PnL vs Always-Long



Buy/Sell Indicator



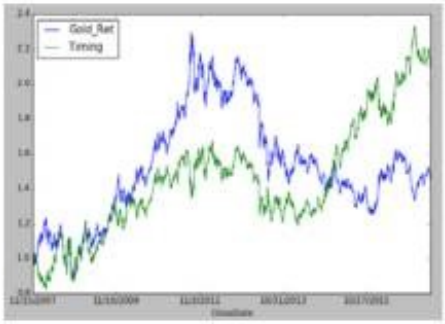
Performance Analytics

	Annualized Return (%)	Sharpe Ratio
DXY	3.22	0.38
DXY- Lasso	4.20	0.49
Correlation	58.9%	

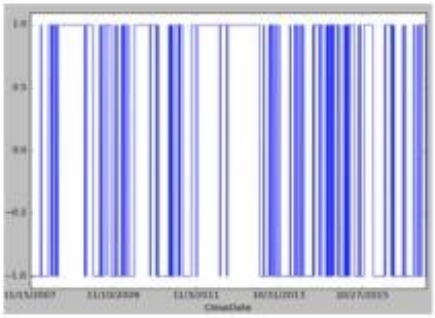
Sample Signals Chosen(0 to 3 factors chosen)

	1M	3M	6M	12M
Bond	×	×	×	×
Gold	×	×	✓	×
Dollar	×	×	×	×
Equity	✓	×	×	×

PnL vs Always-Long



Buy/Sell Indicator



Performance Analytics

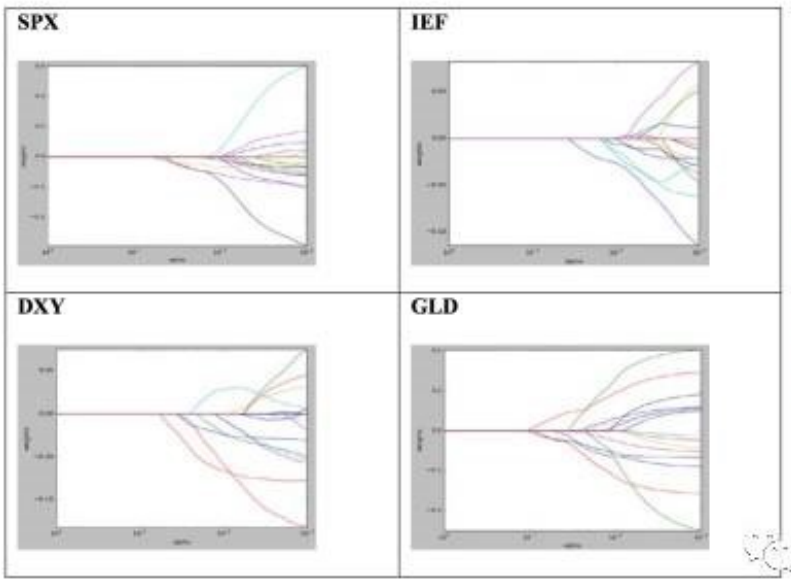
	Annualized Return (%)	Sharpe Ratio
Gold	6.12	0.31
GLD- Lasso	9.49	0.48
Correlation	20.3%	

Sample Signals Chosen(1 to 7 factors chosen)

	1M	3M	6M	12M
Bond	×	✓	×	×
Gold	✓	×	×	✓
Dollar	✓	×	×	×
Equity	×			

然后同样就像上面解释lasso　model时候看到的，我们可以调整其penalty的系数α的值来看对于每个asset，其不同factor系数的变化情况，可以很清楚看到哪些factor是最重要的。

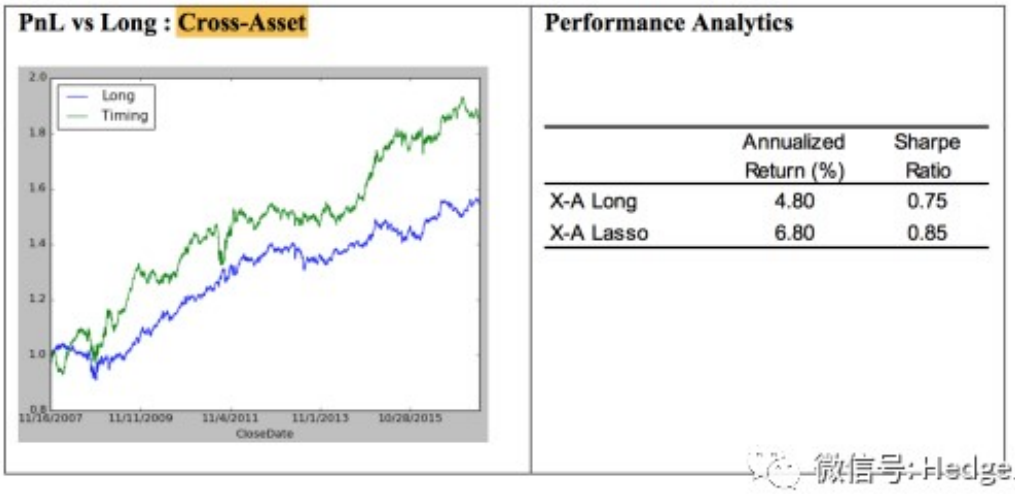
Evolution of betas as function of alpha in Lasso regression for 4 assets



This plot reveals the relation between betas and penalty alpha
From this plot we can know the most principal components of those betas(which means from left to right, along the increasing of penalty, the first non-zero betas are the most important ones to the fitting)

To have the optimal fitting values of betas and alpha, we need to use the CrossValidation method to find the minimum RSE of the lasso fitting.

因为这里我们有4个不同的asset，所以可以将其第二天的预测表现进行排序。前两个做多，后两个做空，也就是所谓的多空策略。



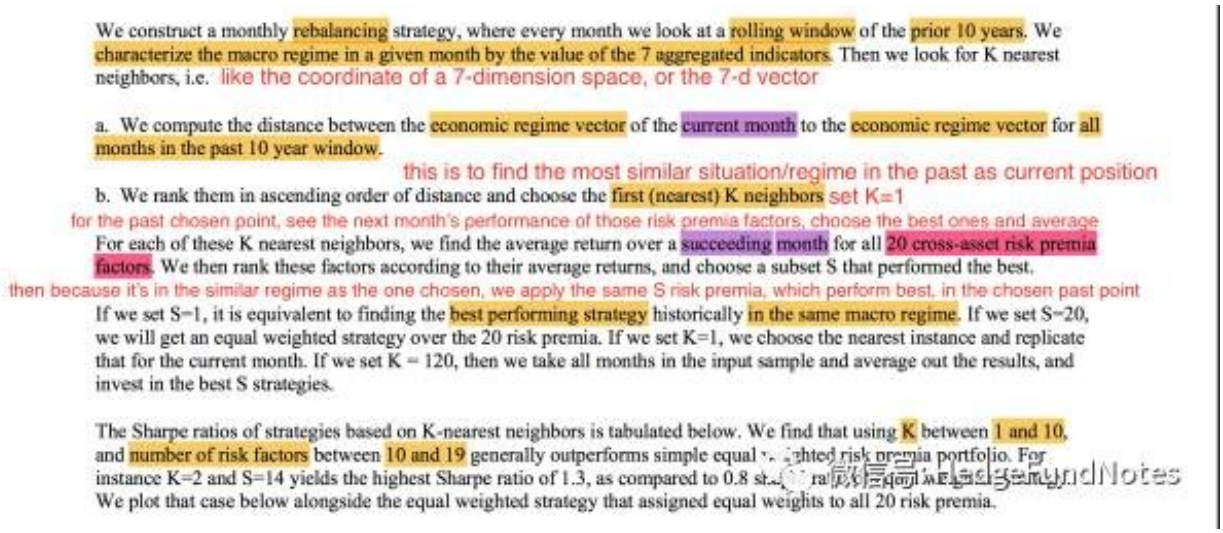
好了，这就是linear里面的regression，也是最经典传统的统计方法。

2.2 Non-Parametric Regression: K-Nearest Neighbor and LOESS

对于处理非线性的问题，一个是可以用高阶的linear regression，另一个就是用非参方法KNN之类

非参方法好处就是如果fundamental的model是非线性，效果会比较好。坏处就是如果model本身是线性的，就容易造成overfit，并且KNN对于outlier的点非常敏感。

这里我们用KNN的方法来择时。(也就是选择历史上类似宏观情况的regime，然后平均一下收益率作为预测值)



具体是这么工作的：

我们有7个类别的indicator，然后用这样一个7-d的vector来表示宏观经济的位置regime(或者说是在7-d空间内的一个点)。
具体怎么构建的这7个indicator可以参照如下slides:
https://www.cmegroup.com/education/files/jpm-systematic-strategies-2013-12-11-1277971.pdf

所以这7个indicator就是我们的7个Xi，构成了一个7维空间，并且由于每个月都有这7个indicator的值，所以都可以用这个空间里的一个点表示。
我们想要预测的Y就是20个risk premia的每一个。

或者可以这么说，每一个月作为一个观察。然后每一个观察包含了这7个Xi和Y(20个risk premia的每一个)的值。并且我们用Xi的7维空间来区分regime。仅仅是画点上去，而不是像之前的regression一样想要用一条线去fit，这里获取Y的方法是将想要预测点周围的K个点对应的Y取平均值。

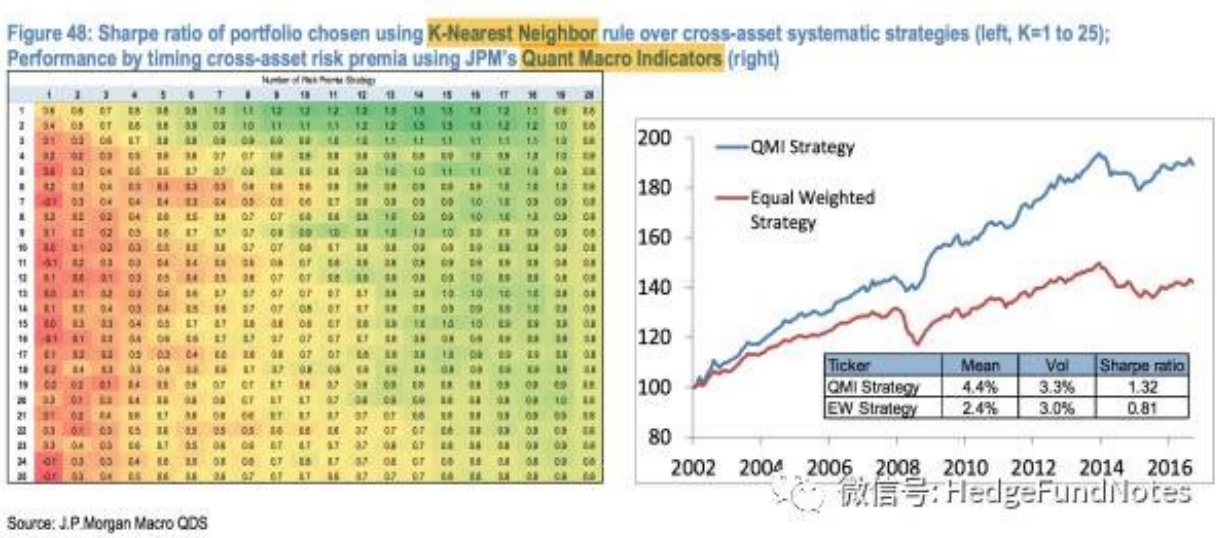
我们将过去的10年每个月都可以画在这7-d indicator的空间中，每一个月都可以被这个空间内的一个点所代表其宏观情况。这样当下这个月也可以画在同样一个空间里，找到其最近的K个点。当然K也可以用其他的值，这样就是average一下附近的K个点的每一个risk premia的return，这样我们可以得到这个点周围附近K个点(也就是历史上类似宏观情况的K个月)这20个risk premia每一个的平均值，(这个地方距离是standardize之后的距离还是直接用原indicator的scale呢？)，这样就找到了历史上最接近现在这个月宏观经济情况的K个月的20个risk premia每一个的平均值。

根据那个月之后的一个月的20个risk premia(此时S=20的情况下)的表现情况(如果20个risk premia是daily的数据，那么就用average来得到一个月每天的平均值，其实也可以直接加起来看每个月的总收益？)，并且将其排序，选择其中一部分然后平均分配资金，来决定当下这个月的下个月投在这20个risk premia上的funding distribution。
这里有一个问题就是，因为不能单纯看20个risk premia的收益大小来选取(不然就全部投给预期收益最高那个risk premia了)。我猜测可能还需要看组合起来看总体的sharp ratio值。所以这样排列下来仅仅只能找到最高收益的那个risk premia，而不是找到最好的sharp ratio。所以需要看这些20个risk premia的subset的组合情况，得到最好的sharp ratio。

但是问题在于，怎么将这些S个risk premia策略组合在一起得到最好的sharp ratio？这个可能就需要看回测的结果了。(但是怎么回测这20个risk premia所有subset的可能性？)

是不是还涉及到给weight的问题呢?
这里似乎就是直接给予这些不同的risk premia策略相同的weights，也就是将funding直接均分给不同的策略。但是其实可以用machine leanring来给不同的策略分配不同的weights来改进。

这些细节的问题我也没有一个确定的答案，可能需要问JPM具体做这个策略的人了...



这就是具体的结果。
左边的图的列向量是选择K从0到25，横向量是选择risk premia的个数。方框中

的数值就是sharp ratio。从这张图看来，确实是需要对所有的组合进行回测。但这里每一行似乎仅仅区分了risk premia的个数，而不是每一个单独不一样的risk premia?
(想要弄清楚，可能需要看看JPM的risk premia是怎么构建的了，应该都是independent，所以并不能单纯以个数来作为loop的条件？)

这里的右图就是单纯给这S个risk premia(S=20所有都用?)平均分配资金，也就是红线。然后就是利用这7个indicator预测的各个risk premia进行排列组合，选出sharp ratio最高的再平均分配资金。

2.3 Tree Based Methods:

首先介绍tree based method的基础，也就是regression tree。

给定很多变量Xi，它的原理就是取遍所有Xi和每一个Xi所有的值，找到一个让全局的RSS(Residual Sum of Squares是一个指标，用来描述预测的Y和实际的Y之间的差距的平方和，也就是预测的效果好坏)最小的某一个Xi的某一个值，这里就算是一个internal node。
这样，Xi的这个值将所有的点分成了两个区域，每一个区域的点所estimate的值就是这整个区域所有点的平均值。
再同样的过程，得到一个让这一步全局RSS最小的Xi的一个值(这里Xi可以跟上一步的Xi是同一个变量)。

同样的过程不断进行下去，一直到某一个条件停止。比如RSS到达某一个值，或者某一个区域的点小于5之后，等等。

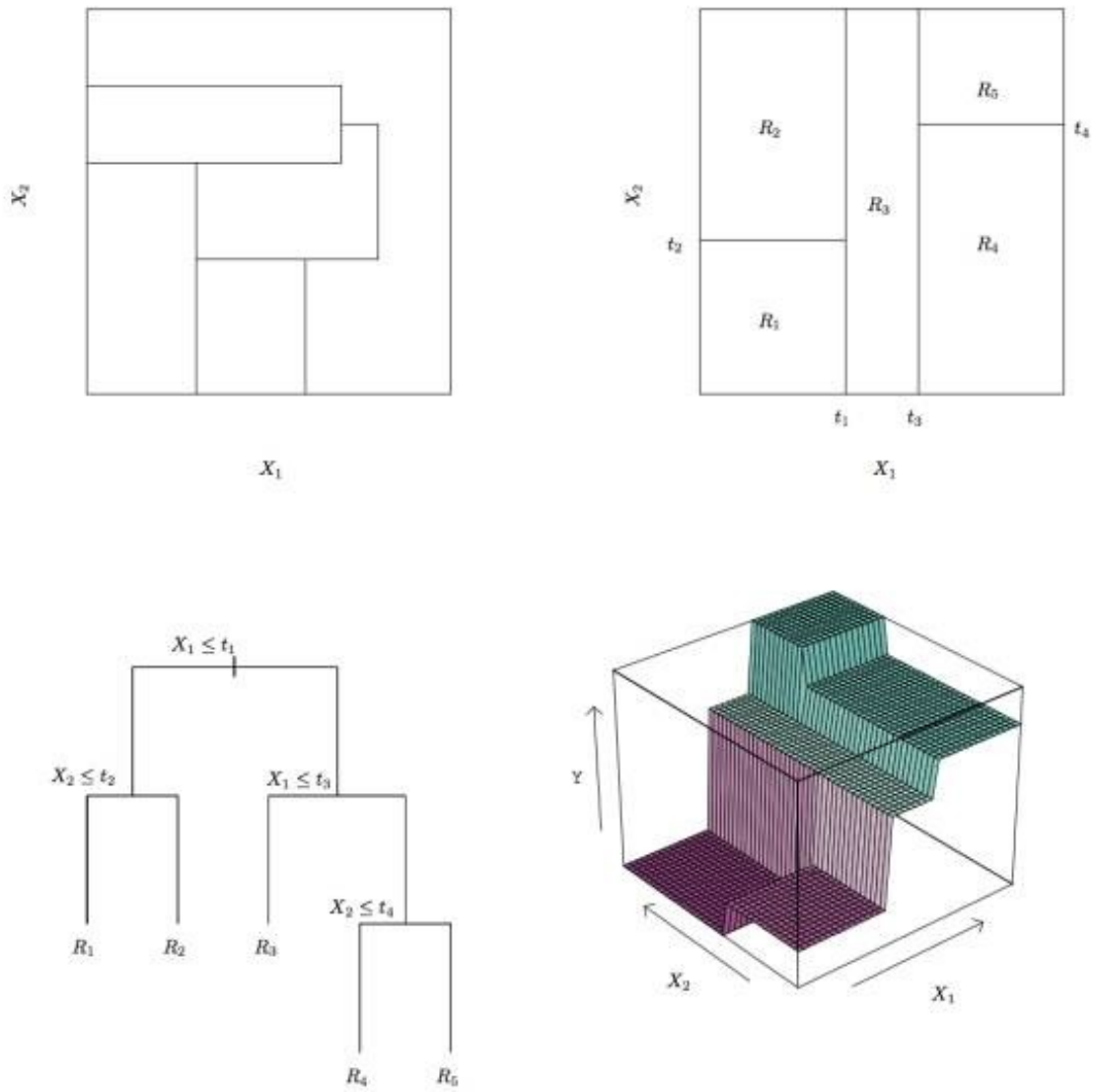


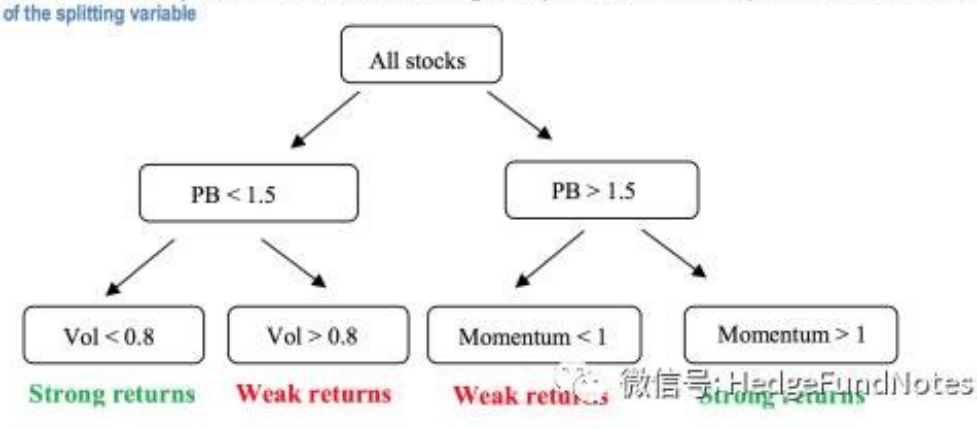
FIGURE 8.3. Top Left: *A partition of two-dimensional feature space that could not result from recursive binary splitting.* Top Right: *The output of recursive binary splitting on a two-dimensional example.* Bottom Left: *A tree corresponding to the partition in the top right panel.* Bottom Right: *A perspective plot of the prediction surface corresponding to that tree.*

记住，这里的RSS是全局的RSS，也就是所有区域的RSS总和，而不仅仅是所分开区域的RSS之和。

同理，这里需要一个penalty函数。因为树越深其实越overfit(可以想象当树深到极致就是每个区内只有一个点)，所以给目标函数RSS加上一个additional的alpha*树的深度，使得其总和达到最小。
合适alpha值的选取就需要通过cross-validaiton来得到，同时可以得到的也有tree的深度。这样一个model就选出来了。

regression tree应用的示意图如下:

Figure 63: Example of decision tree to classify whether future stock returns are good or poor. In each node, we split the data into two subsets depending on the value of the splitting variable



regression tree的好处就是非常容易理解，因为每一个node的Xi取值都一目了然。但是不好的地方就是它对于outliers非常敏感，因为outliers会极大的增加RSS这个指标，所以很容易被影响。

Random Forest:

random forest的意思就是，在上面的regression tree的基础上，我们再利用bootstrap来产生数量为B的test datasets(也就是bagging的model)。对于每一个dataset我们可以利用一个regression tree去fit。所以对于每一个观察，我们都可以得到B个prediction的值。接着我们就把这些值average一下，就得到了我们最终的estimate。这样的好处就是可以降低variance，因为我们用了很多的sample一起求出的平均值，而不是像decision/regression/classification tree那样一次性的estimate。

然后random forest又加了另外一项，也就是限制每一个node可以选取的Xi的数目。比如一共有p个Xi，但是我们可以规定每个node只能randomly选取sqrt(p)的Xi。这样的好处就是为了防止一个factor dominates，不然就会导致这B个tree的correlation非常大，即使求平均值也起不到降低variance的作用。

具体我们可以看如下的例子:

Use of Random Forests in Global stock selection strategy

Let us consider a global stock selection model that predicts 1-month stock returns across the MSCI World universe (over 2400 stocks), using the 14 risk factors shown below. We consider a subset of this universe where we have observations for all 14 factors, which corresponds to about 1400 stocks. The table below shows the 14 risk factors used in this example.

Figure 65: 14 Factors used to predict 1-month ahead stock returns

Factors	
Price-to-book ratio	Earnings Certainty
Gross profit / Total assets	Cash flow yield
ROE	Dividend yield
Net Margin	Realized vol
Asset Turnover	1M momentum
Gearing	12M-1M momentum
Forward earnings yield	Market cap

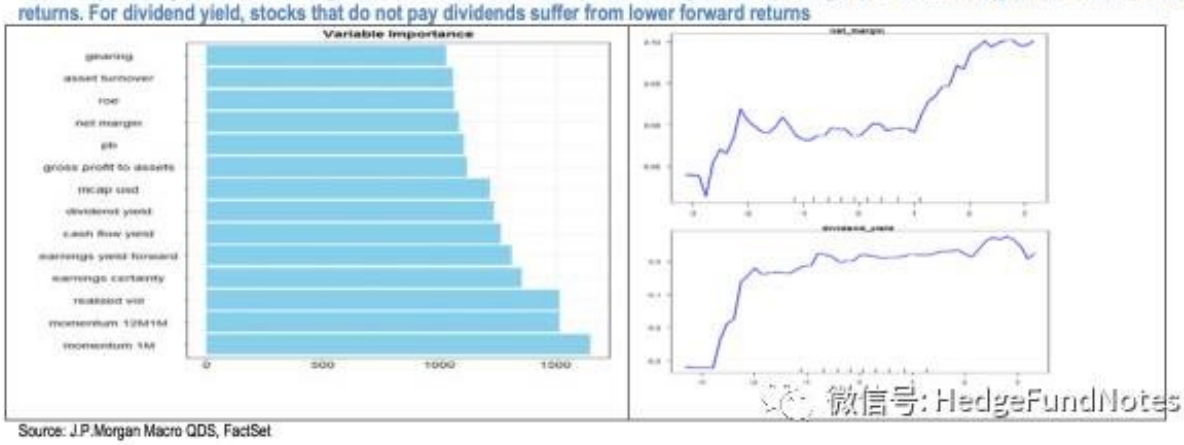
Source: J.P.Morgan Macro QDS

这里要做的事情就是利用random forest的model来选股。我们的股票池里面有1400个股票，然后每个股票我们都有相同的14个factor，也就是有14个Xi(注意这里的factor需要先normalize一下，也就是standardized to mean 0 and variance 1，因为各个factor的scale不一样会导致fitting有bias)，这里的Y就是收益率。

我们利用random forest的model来预测。
首先我们利用bootstrap构造100个test datasets，然后针对每一个dataset去fit一个regression tree。并且每一个tree的每个node处，一般每次只能randomly选择sqrt(14)=3 factors。但是这里我们用OOB这个方式来判断每个node处可以选择Xi的个数(得到的是14个全部可以用)，并且每个tree的深度也是由OOB这个最小值来确定。
接着我们对于每一个股票都有100个prediction的值，然后average，得到我们最终的estimate。

这里的一个问题也是同样，我们是不是需要将股票的收益向右位移一个单位，使得这14个risk factors Xi所对应的Y是下一个月的收益。毕竟如果放在同一个月，那就不存在预测的问题了，因为都是同时发生，而不是用一个去预测接下来即将发生的另一个。

Figure 66: Left: Variable importance plot for the Random Forest model that predicts 1-month ahead stock returns based on 14 factors. Right: Partial dependency plots for Net Margin and Dividend Yield. It shows that the higher the Net Margin of a stock, the higher the 1-month ahead returns. For dividend yield, stocks that do not pay dividends suffer from lower forward returns



Source: J.P.Morgan Macro QDS, FactSet

然后利用每个node对于Xi的选择，我们可以得到上图，可以看出来哪个factor影响最显著。

另一张图就是策略的回测结果，通过对于这1400个股票的预测值进行排序，我们就可以得到每一个quantile(一共分为5个quantiles)的股票basket。

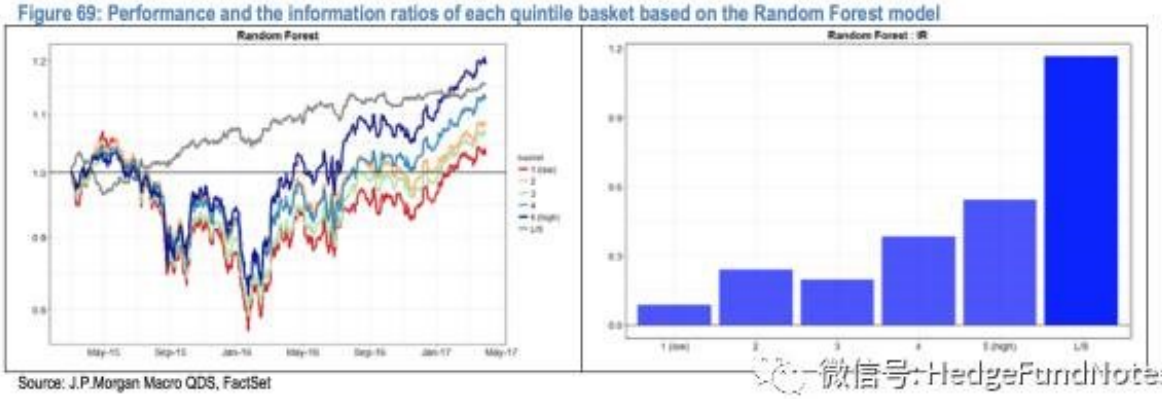
Figure 68: Performance statistics (Feb 2015 – Mar 2017) of the baskets based on the Random Forest model

Basket	Cum. Return	CAGR	Volatility	IR	Max Drawdown	Hit Ratio
1 (low)	3.0%	1.0%	11.3%	0.09	27.7%	37.3%
2	7.9%	2.5%	10.7%	0.24	22.7%	37.8%
3	6.4%	2.1%	10.7%	0.19	23.6%	38.3%
4	12.8%	4.1%	10.6%	0.38	21.9%	37.0%
5 (high)	19.2%	6.0%	11.1%	0.54	20.9%	39.5%
L/S	15.4%	4.8%	4.2%	1.16	6.9%	37.7%

Source: J.P.Morgan Macro QDS, FactSet

The Figure below shows the wealth curves, rolling 12-month returns and the information ratios of each basket.

Figure 69: Performance and the information ratios of each quintile basket based on the Random Forest model



Source: J.P.Morgan Macro QDS, FactSet

图上表示的就是单纯对于每一个quantile的basket做多(5条曲线)，还有long/short的策略(也就是做多第5个最高的quantile basket，同时做空最低的那个quantile basket)，其所得到的收益率是最稳定的。

因为我们可以看到benchmark的曲线在1和5之间，所以做多5，做空1的话就会使得曲线更加平滑。不然会跟大盘的相关性非常大，这也就是一种把大盘波动hedge掉仅仅只剩下1 quantile和5 quantile之间的差值的收益。

Extreme Gradient Boosting:

boosting的原理就是慢慢学习，也就是先给一个不太准确的estimate，然后用真实值减去这个estimate得到residual/error，接着用regression tree去fit这个剩余的residual error得到这个residual的estimate，再用上一层residual减去这个estimate得到下一层的residual，然后进行K次(即一共用K个tree去fit K层的residual)，实质上是从一层一层的residual里面不断缓慢提取信息不断加到之前的estimate上面，使得最终得到一个总的好fitting。boosting指的是一类方法，而不是一个方法。Extreme Gradient boosting是boosting中一个的方法。

XGBoost Algorithm

T is the number of branches +1, m is the number of variables Xi(or the p in the book?)

A regression tree is similar to a decision tree, except that at each leaf, we get a continuous score instead of a class label. If a regression tree has T leaves and accepts a vector of size m as input, then we can define a function $q: \mathbb{R}^m \to \{1, \dots, T\}$ that maps an input to a leaf index. If we denote the score at a leaf by the function w, then we can define the k-th tree (within the ensemble of trees considered) as a function $f_k(\underline{x}) = w_{q(\underline{x})}$, where $w \in \mathbb{R}^T$. For a training set of size n with samples given by $(\underline{x}_i, y_i), \underline{x}_i \in \mathbb{R}^m, y_i \in \mathbb{R}$, a tree ensemble model will use K additive functions to predict the output as follows:

$\hat{y}_i = \phi(\underline{x}_i) = \sum_{k=1}^{K} f_k(\underline{x}_i)$. because fit the residual, so it's additive functions

To learn the set of functions in the model, the regularized objective is defined as follows:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k),$$

penalty

where $\Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2$. here w(x) function is the average of residual/error of x(each x should be inside specific the q region) when each time of retrieving in k-th tree, so put the squared w in to the penalty terms means we want to make the total trees residual /error smaller enough

The tree ensemble model is optimized in an additive manner. If $\hat{y}_i^{(t)}$ is the prediction for the i-th training example at the t-th stage of boosting iteration, then we seek to augment our ensemble collection of trees by a function f_t that minimizes the following objective:

$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(\underline{x}_i)) + \Omega(f_t)$.

The objective is approximated by a second-order Taylor expansion and then optimized. For calculation steps, we refer the reader to Chen and Guestrin (2016) and the expository material on xgboost at link. To prevent overfitting, xgboost uses shrinkage (to allow future trees to influence the final model) and feature sub-sampling (as in random forest). An option for row sub-sampling is also provided.

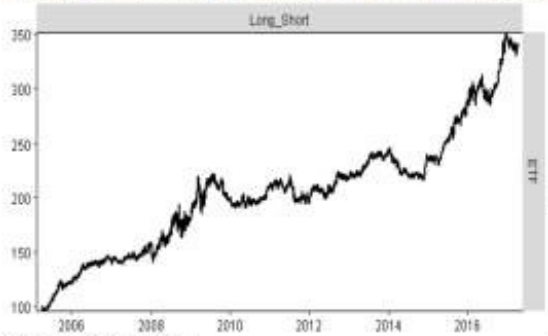
different Ys

We construct a long-short strategy trading nine US sector ETFs: financials, energy, utilities, healthcare, industrials, technology, consumer staples, consumer discretionary and materials. We use XGBoost algorithm to predict next day returns based on 8 macro factors: Oil, Gold, Dollar, Bonds; economic surprise index (CESIUSD), 10Y-2Y spread, IG credit (CDX HG) and HY credit spreads (CDX HY). to each Y, we have 8 Xi, the 8 macro factors

After obtaining prediction returns for US sectors (based on previous day's macro factors), we rank sectors by expected return, and go long top 3 and short bottom 3 sectors. We used the open-source implementation of Xgboost available in R through the 'xgboost' package. For optimizing the parameters and hyper-parameters, we used 5-fold cross-validation, 30 boosting iterations and allowed decision trees to have a maximum depth of 7. The strategy used a rolling window of 252 days and rebalanced daily at the market close. This yielded an annualized return of 10.97% and an annualized volatility of 12.29%, for a Sharpe ratio of 0.89. Correlation of the strategy to the S&P 500 was ~7%. Correlation of the strategy to equity long-short styles, as well as other cross-asset risk premia were also low (see the right figure below).

30 iterations are the number of trees in boosting, which is determined by cross-validation

Figure 54: XGBoost strategy applied to US sectors – performance (left), correlation to cross-asset risk premia (right)



Source: J.P. Morgan Macro QDS

XGBoost	Bond	Commodity	Equity	FX
Carry	-3.9%	3.5%	0.7%	0.4%
Momentum	-3.7%	-1.1%	6.8%	-1.8%
Value	-12.2%	1.9%	-5.3%	-2.4%
Volatility	-6.2%	0.3%	0.8%	-1.2%
Beta	-7.2%	-7.8%	7.0%	-2.1%

这里做的事情是利用XGBoost来预测9个US sector ETFs的表现，financials, energy, utilities, healthcare, industrials,technology, consumer staples, consumer discretionary and materials，对应于Y1，Y2...Y9。

首先我们有8个Xi，也就是8个宏观的factor，
Oil, Gold, Dollar, Bonds; economic surprise index (CESIUSD), 10Y-2Y spread, IG credit (CDXHG) and HY credit spreads (CDX HY)

然后我们要做的事情就是，用这8个Xi，还有extreme gradient boosting的model去预测每一个Yi的表现。
这里extreme gradient boosting model里面的参数是，5-fold-cross-validation来确定tree的数目为30，也就是经过30次的从residual里面extracting information。并且每个regression tree的深度为7，也就是每个tree被分为了7+1=8个区域。
有了这个extreme gradient boosting model不断的iterate(30遍，每一遍的tree的深度是7)来不断extract信息给想要估计的那个estimate函数f^hat之后，f^hat确定了下来。在同样给定了这个想要预测的observation的8个Xi的值(也就是这个8维空间内的一个点)之后，就可以经过这个model得到我们想要的预测值Yi。

这里应该是用的每天的涨跌幅(里面有说rebalance daily at market close)。因

为对于同一个Yi来说，比如我们预测energy这个sector ETF的涨跌幅。我们每一天都有这8个macro的factor的一个值，这样一天在这个8维空间上来说就是一个点，252天的话就是有252个点。这样才能开始利用boosting tree来划分区域进行预测。甚至都不一定是daily的数据，可以是更小级别的数据，这样点(信息)也就更多了，model的预测也更加贴近真实的model。然后利用这252个点来建立model，建立之后预测energy ETF的涨跌幅。

有一个问题就是，同样我们这里可能需要将energy ETF的向右位移一个单位，也就是将第二天的energy ETF作为这一天的Yi，这样8个Xi预测出来的结果也是第二天的energy ETF的涨跌幅。(但这里还有一个问题就是为什么是位移一个单位，也许这些macro factor传导给ETF是有不同时滞的，所以每个Xi甚至都需要不同的位移?)

然后根据long-short策略，每天我们可以预测出这9个ETF的涨跌幅度，将其排序，做多前三个，做空后三个，形成了一个策略。

除此之外，我感觉还可以用另一种方法。那就不用位移，直接将当天的ETF涨跌幅看做是Yi。这里的一个assumption就是市场是有效的，macro factor的变化可以马上传导给ETF的价格。这样的话我们就可以通过预测的值，看如果当天实际并没有达到这个涨跌幅(在快收盘的时候检查condition)，我们可以进去统计套利，也就是赌其一定会往那个涨跌幅移动。也可以将其量化变成相差多少个sigma(假设围绕预测值是一个normal distribution，其mean就是预测的值，width可以取历史上面的error是不是都一样，如果是近似flat可以直接用这个constant作为error，如果不是的话可以继续分析其error的结构)，大于两个sigma的时候进去开仓bet会继续往预测的方向走，不一定是同方向，如果涨跌幅走过了2个sigma那就是bet其一定会reversion。

（未完待续……）