IPCA

1. IPCA 简介

IPCA 是 Instrumented principal component analysis 的简称（Kelly, Pruitt, and Su (KPS, 2019)）。PCA 是指，应用在收益率情况下，假设收益率服从

$$r_{i,t+1}=\beta_{i,1}f_{1,t+1}+,\ldots,+\beta_{i,K}f_{K,t+1}+u_{i,t+1} \tag{1.1}$$

其中，$r_{i,t+1}$ 表示t 到$t+1$ 时刻之间的股票i 的收益率，$f_{k,t+1}$表示t 到$t+1$ 时刻之间的共同因子k 的取值，$\beta_{i,k}$表示股票i 在因子k 上的暴露，$u_{i,t+1}$表示收益率。(1.1)中的暴露和因子值可以用 PCA 的方法来估计。该模型本身可以用作收益模型和风险模型，用作风险模型时就是多因子风险模型的一种。用作收益模型时，对于$r_{i,t+1}$的预测，可以使用截止到t 期的数据估计该模型暴露$\hat{\beta}_{i,k}$ 和因子值 $\hat{f}_{k,t}$，然后将$\hat{\lambda}_k=\frac{1}{t}\sum_{s=1}^{t}\hat{f}_{k,s}$作为对 $f_{k,t+1}$ 的预测，则对$r_{i,t+1}$的预测为$\hat{\beta}_{i,1}\hat{\lambda}_1+,\ldots,+\hat{\beta}_{i,K}\hat{\lambda}_K$。

上述 PCA 的暴露对于不同的股票取值不同，所以假设有N 只股票的话，估计的暴露系数有$N\times K$ 个，假设我们估计模型时数据量有T 期，则估计的因子值有$T\times K$个。IPCA 的思路是，将暴露系数表示成时变的，且是t 时刻已知信息的函数

$$\beta_{i,k,t}=a_{k,0}+z_{i,1,t}a_{k,1}+\ldots+z_{i,P,t}a_{k,P} \tag{1.2}$$

其中，$z_{i,j,t}, j=1,\ldots,P$是t 时刻已知的股票i 的第p 个信息（或特征）。

这样做的好处是，首先，金融理论上没有表明对于共同因子的暴露系数是不时变的。而且，文献上已经发现的好多能够预测股票涨跌的横截面信息，很可能只是因为该信息能够决定风险因子暴露，从而有作用。另外，将暴露系数表示成上述结构，我们只需估计除了$T\times K$个因子值外的$(P+1)\times K$ 个系数，如果P 比N 小很多，那对减小模型的方差来说效果应该不小。即 IPCA 比 PCA 表示的模型更加丰富（通过允许暴露系数时变），类似于减小了模型的偏误，同时又减小了方差（估计系数变少）。这是因为引入了额外的数据$z_{i,j,t}, j=1,\ldots,P$。如果只是允许暴露系数时变而不设定时变的结构，则会增大方差，通过设定时变的结构，则减小了方差。

应用：KPS（2019）将该方法用在美国股票上，使用了 36 个已知信息，每一月根据用前面多月数据估计的模型做均值方差最优化，得到的组合样本外夏普率为 3.89。

2. IPCA 的延伸

上述(1.2)假设暴露系数是特征的线性函数，可以扩展成非线性函数。Gu,Kelly,Xiu（GKX，2019）就做了这样的扩展。

在具体实施的时候，GKX（2019）用到了神经网络的 autoencoder 的概念。标准的 autoencoder 是一种无监督学习，用来描述数据生成过程的。假设该神经网络有L 个隐

层，第l层的神经元记为$r^{(l)}=\left(r_1^{(l)},\dots,r_{K^{(l)}}^{(l)}\right)'$，其中$K^{(l)}$为第$l$层的神经元的个数，原始输入为$N$维（$N$个股票收益率），记为$r^{(0)}=\left(r_1,\dots,r_N\right)'$。则对于$l=1,\dots,L$，

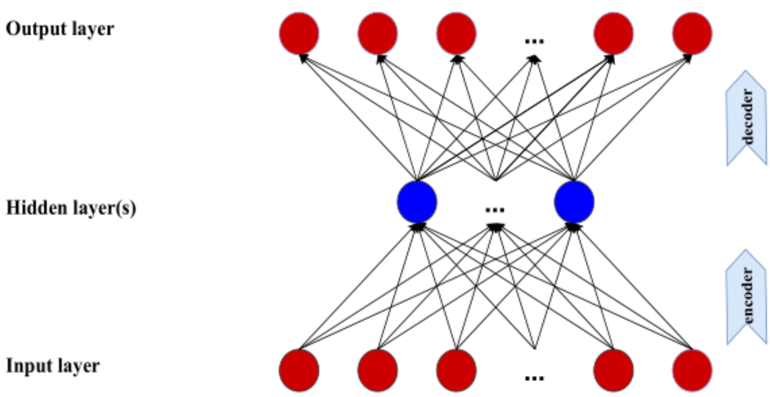
$$r^{(l)}=g\left(b^{(l-1)}+W^{(l-1)}r^{(l-1)}\right) \tag{1.3}$$

其中，g为激活函数，$b^{(l-1)}$和$W^{(l-1)}$为参数，维度分别为，$b^{(l-1)}$为$K^{(l)}\times 1$，$W^{(l-1)}$为$K^{(l)}\times K^{(l-1)}$。该网络的最终输出为

$$G=b^{L}+W^{(L)}r^{(L)} \tag{1.4}$$

该网络如下图：

Figure 1: Standard Autoencoder Model



Note: This figure describes a standard autoencoder with one hidden layer. The output and input layers are identical, while the hidden layer is a low dimensional compression of inputs variables into latent factors, which can be expressed as weighted linear combinations of input variables.

训练该模型的目标函数为

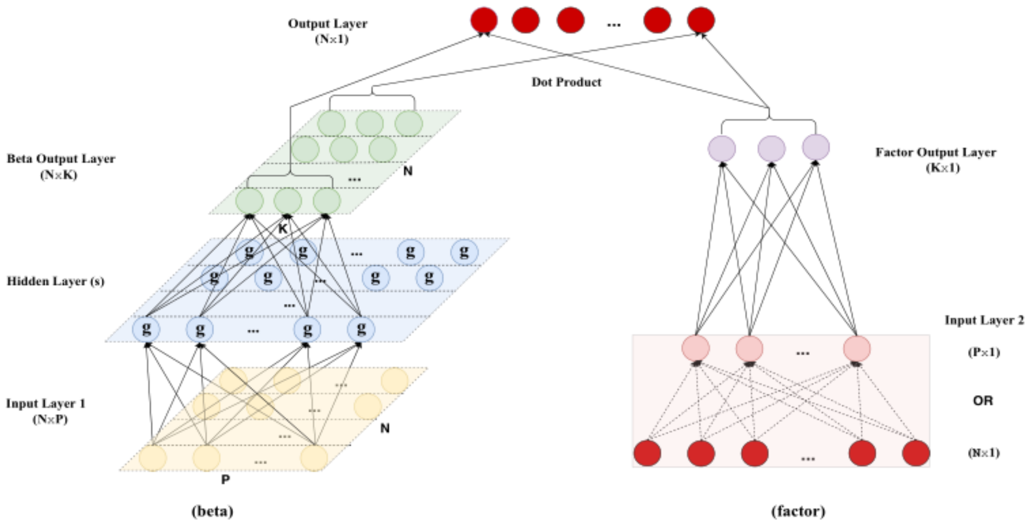
$$\min_{b,W}\sum_{t=1}^{T}\left\|r_t-G\right\|^2 \tag{1.5}$$

可见，该模型就是要找到系数b,W，使得上述网络能够近似输入r_t。

而对于只有 1 个隐层的上述网络，有证明，最小化(1.5)求得的结果等同于 PCA，其隐层$r^{(1)}=\left(r_1^{(1)},\dots,r_{K^{(1)}}^{(1)}\right)'$相当于 PCA 的共同因子$\left(f_1,\dots,f_K\right)$，而$W^{(1)}$相当于 PCA 的暴露系数。

GKX（2019）在标准的 autoencoder 的基础上借鉴 IPCA 的思路，使得暴露是已知特征的函数。即使得上述$W^{(1)}$是特征的函数。所以 GKX（2019）的网络如下：

Figure 2: Conditional Autoencoder Model



Note: This figure presents the diagram of an autoencoder augmented to incorporate covariates in the factor loading specification. The left-hand side describes how factor loadings β_{t-1} at time $t-1$ (in green) depend on firm characteristics Z_{t-1} (in yellow) of the input layer 1 through an activation function g on neurons of the hidden layer. Each row of yellow neurons represents the $P \times 1$ vector of characteristics of one ticker. The right-hand side describes the corresponding factors at time t. f_t nodes (in purple) are weighted combinations of neurons of the input layer 2, which can either be P characteristic-managed portfolios x_t (in pink) or N individual asset returns r_t (in red). In the latter case, the input layer 2 is exactly what the output layer aims to approximate, which is the same as a standard autoencoder.

GKX（2019）将该模型用在美国股票上，使用的特征有 90 几个。同样是月频。使用上述模型的预测，排序法构造十分位减一分位的多空组合（分位里面的股票市值加权），夏普率为 2.6（如果分位里面的股票等权，则夏普率为 1.5）。该作者之前用这 90 几个因子做的机器学习模型，等权的夏普率为 1.35（GKX，2020）。

3. 思路扩展

首先，(1.1)的共同因子模型给了我们启发。之前，我做的一个模型的思路是对于股票 i 的收益率的预测，除了使用股票 i 自己上期的收益率外，还使用其他股票上期的收益率作为自变量。该思路的来源是股票 i 和股票 j 有某种（经济）关联，当股票 j 发生信息是股票 j 股价发生变化时，股票 i 也应该有所反应，但是反应有可能不足。将这种逻辑一般化，当发生某个信息时，股票 i 和股票 j 都对该信息有所反应，但是都有可能反应不足或反应过度，以至于下一期股票 i 和股票 j 对该信息会继续反应。把该信息记为 $f_{k,t}$ ，

则意思是 $r_{i,t}, r_{j,t}$ 对 $f_{k,t}$ 都会有反应，但是由于反应不足或反应过度，$r_{i,t+1}, r_{j,t+1}$ 对 $f_{k,t}$ 仍然会有反应（暴露系数不为 0），即

$$r_{i,t+1} = \beta_{i,1,0} f_{1,t+1} +,\dots,+ \beta_{i,K,0} f_{K,t+1} + \beta_{i,1,1} f_{1,t} +,\dots,+ \beta_{i,K,1} f_{K,t} + u_{i,t+1} \quad (1.6)$$

其中 $\beta_{i,1,s}$ 的第三个下标表示股票对滞后 s 期的共同因子的反应。

对于模型(1.6)，我之前的做法是，先按照模型(1.1)做 PCA，得到暴露的估计暴露 $\hat{\beta}_{i,k}$ 每期共同因子的估计 $\hat{f}_{k,t}$ ，然后用，

$$r_{i,t+1}-\hat{\beta}_{i,1,0}\,\hat{f}_{1,t+1}+,\ldots,+\hat{\beta}_{i,K,0}\,\hat{f}_{K,t+1} \tag{1.7}$$

对

$$\hat{f}_{1,t},\ldots,\hat{f}_{K,t} \tag{1.8}$$

做回归，得到 $\hat{\beta}_{i,k,1}$。然后就可以使用(1.6)做预测了，对 $r_{i,t+1}$ 的预测为

$$\hat{\beta}_{i,1,1}\,\hat{f}_{1,t}+,\ldots,+\hat{\beta}_{i,K,1}\,\hat{f}_{K,t} \tag{1.9}$$

。

上述做法的问题在于我应该多迭代几步，有了 $\hat{\beta}_{i,k,1}$ 后，对于

$$r_{i,t+1}-\hat{\beta}_{i,1,1}\,\hat{f}_{1,t}+,\ldots,+\hat{\beta}_{i,K,1}\,\hat{f}_{K,t} \tag{1.10}$$

再做 PCA，得到暴露的新的估计 $\hat{\beta}_{i,k}$， 每期共同因子的新的估计 $\hat{f}_{k,t}$，再重复上述操作估计，这样多做几步系数的估计应该会收敛。这样估计的系数比我原来的应该更准确。IPCA 的估计用了类似的迭代估计法。

其次，我们也可以借鉴 IPCA，加一些在 t 时刻已知的信息进去，使得暴露是时变的，这样模型更丰富同时估计方差减小。

最后，PCA 的思路在数据量有限的情况下只能得到几个重要的主成分，即对大多数股票都有影响的 f 。如果要得到对少数（比如 10 只）股票有影响的 f ，得寻找新的方法。SPCA（sparse PCA）（Hui Zou）似乎是一个思路。其大致原理是在使用 PCA 的时候，对暴露系数做 l1 正则化。加该正则化的思路是，对于某个共同因子 f ，不是所有的股票都有非 0 的暴露系数，很可能有较多股票的暴露系数为 0。

另外，autoencoder 的思路似乎也有一定价值，该模型的给我的启发是将横截面收益率表示成由几个潜在共同因子决定，这些潜在共同因子决定收益率的多隐层形式似乎有一定价值，虽然上述文章只用了单隐层的形式。

最重要的一个思路是，某个股票在某个因子上的暴露不再是单独一个数，而是将该暴露设定成该股票一些特征的函数，相当于使用股票特征来识别股票，而不是用股票 id 来识别股票，该思路可以大大减少待估计的系数，从而减少模型方差。

4. 应用实践

根据(1.2)和(1.6)，我们的模型（假设滞后阶数为 3）为

$$r_t = Z_{t-1}\Gamma_0 f_t + Z_{t-1}\Gamma_1 f_{t-1} + \ldots + Z_{t-1}\Gamma_3 f_{t-3} + u_t \tag{1.11}$$

，其中，r_t 是 $N_{t-1}\times 1$ 维向量，表示 $t-1$ 时刻符合条件的股票数量，其每个成分为对应股票从 $t-1$ 时刻 到 t 时刻 的收益率。Z_{t-1} 为 $N_{t-1}\times p$ 的，其第 i,j 元表示第 i 个股票的第 j 个特征在 $t-1$ 时刻的取值。Γ_j ，为 $p\times K$ 的，表示滞后 j 阶的系数矩阵，Z_{t-1} 的第 i 行乘以 Γ_j 的第 k 列，得到股票 i 在滞后 j 的第 k 个因子上暴露。 f_t 为 $K\times 1$ 的，

其第k 个成分表示第k 个因子从$t-1$时刻 到t时刻 的因子收益率。

为了估计上述模型，我们的数据是$r_t, t=0,...,T-1$ 和$Z_t, t=-1,...,T-2$ 。为了看得清楚，我们假设$T=6$ ，$lag=3$ ，则(1.11)应用在$t=0,...,5$ 上，具体为

$$\begin{aligned}
r_0 &= Z_{-1}\Gamma_0 f_0 + Z_{-1}\Gamma_1 f_{-1} + Z_{-1}\Gamma_2 f_{-2} + Z_{-1}\Gamma_3 f_{-3} + u_0 \\
r_1 &= Z_0\Gamma_0 f_1 + Z_0\Gamma_1 f_0 + Z_0\Gamma_2 f_{-1} + Z_0\Gamma_3 f_{-2} + u_1 \\
r_2 &= Z_1\Gamma_0 f_2 + Z_1\Gamma_1 f_1 + Z_1\Gamma_2 f_0 + Z_1\Gamma_3 f_{-1} + u_2 \\
r_3 &= Z_2\Gamma_0 f_3 + Z_2\Gamma_1 f_2 + Z_2\Gamma_2 f_1 + Z_2\Gamma_3 f_0 + u_3 \\
r_4 &= Z_3\Gamma_0 f_4 + Z_3\Gamma_1 f_3 + Z_3\Gamma_2 f_2 + Z_3\Gamma_3 f_1 + u_4 \\
r_5 &= Z_4\Gamma_0 f_5 + Z_4\Gamma_1 f_4 + Z_4\Gamma_2 f_3 + Z_4\Gamma_3 f_2 + u_5
\end{aligned} \tag{1.12}$$

对于(1.12)，我们要估计的系数为

$\Gamma_0, \Gamma_1, \Gamma_2, \Gamma_3, f_{-3}, f_{-2}, f_{-1}, f_0, f_1, f_2, f_3, f_4, f_5$ 。

为了估计这些系数，我们最小化目标:

$$L = \sum_{t=0}^{5}\left(r_t - (Z_{t-1}\Gamma_0 f_t + Z_{t-1}\Gamma_1 f_{t-1} + ... + Z_{t-1}\Gamma_3 f_{t-3})\right)'\left(r_t - (Z_{t-1}\Gamma_0 f_t + Z_{t-1}\Gamma_1 f_{t-1} + ... + Z_{t-1}\Gamma_3 f_{t-3})\right)$$

(1.13)

上述损失函数对各个系数求一阶导，并使得导数为 0，则可以解得，

对于 f_{-3} 的估计，相当于对于(1.12)，上述系数除了 f_{-3}，其他都已知，然后估计 f_{-3}，所以，对于 f_{-3} 的估计为，以$r_0 - (Z_{-1}\Gamma_0 f_0 + Z_{-1}\Gamma_1 f_{-1} + Z_{-1}\Gamma_2 f_{-2})$为因变量，以$Z_{-1}\Gamma_3$ 为自变量，做最小二乘回归。

对于 f_{-2} 的估计，相当于对于(1.12)，上述系数除了 f_{-2}，其他都已知，然后估计 f_{-2}，所以，对于 f_{-2} 的估计为，以$\begin{bmatrix} r_0 - (Z_{-1}\Gamma_0 f_0 + Z_{-1}\Gamma_1 f_{-1} + Z_{-1}\Gamma_3 f_{-3}) \\ r_1 - (Z_0\Gamma_0 f_1 + Z_0\Gamma_1 f_0 + Z_0\Gamma_2 f_{-1}) \end{bmatrix}$为因变量，以$\begin{bmatrix} Z_{-1}\Gamma_2 \\ Z_0\Gamma_3 \end{bmatrix}$为自变量，做最小二乘回归。

对于 f_{-1} 的估计，相当于对于(1.12)，上述系数除了 f_{-1}，其他都已知，然后估计 f_{-1}，所以，对于 f_{-1} 的估计为，以$\begin{bmatrix} r_0 - (Z_{-1}\Gamma_0 f_0 + Z_{-1}\Gamma_2 f_{-2} + Z_{-1}\Gamma_3 f_{-3}) \\ r_1 - (Z_0\Gamma_0 f_1 + Z_0\Gamma_1 f_0 + Z_0\Gamma_3 f_{-2}) \\ r_2 - (Z_1\Gamma_0 f_2 + Z_1\Gamma_1 f_1 + Z_1\Gamma_2 f_0) \end{bmatrix}$为因变量，以$\begin{bmatrix} Z_{-1}\Gamma_1 \\ Z_0\Gamma_2 \\ Z_1\Gamma_3 \end{bmatrix}$为自变量，做最小二乘回归。

对于 f_0 的估计，相当于对于(1.12)，上述系数除了 f_0，其他都已知，然后估计 f_0，所

以，对于 f_0 的估计为，以 $\begin{bmatrix} r_0-(Z_{-1}\Gamma_1 f_{-1}+Z_{-1}\Gamma_2 f_{-2}+Z_{-1}\Gamma_3 f_{-3}) \\ r_1-(Z_0\Gamma_0 f_1+Z_0\Gamma_2 f_{-1}+Z_0\Gamma_3 f_{-2}) \\ r_2-(Z_1\Gamma_0 f_2+Z_1\Gamma_1 f_1+Z_1\Gamma_3 f_{-1}) \\ r_3-(Z_2\Gamma_0 f_3+Z_2\Gamma_1 f_2+Z_2\Gamma_2 f_1) \end{bmatrix}$ 为因变量，以 $\begin{bmatrix} Z_{-1}\Gamma_0 \\ Z_0\Gamma_1 \\ Z_1\Gamma_2 \\ Z_2\Gamma_3 \end{bmatrix}$ 为自变量，做最小二乘回归。

对于 f_1, f_2, f_3, f_4, f_5 的估计类似。

对于 $\Gamma_0,\Gamma_1,\Gamma_2,\Gamma_3$ 的估计，相当于对于(1.12)，上述系数除了 $\Gamma_0,\Gamma_1,\Gamma_2,\Gamma_3$，其他都已知，然后估计 $\Gamma_0,\Gamma_1,\Gamma_2,\Gamma_3$，所以，对于 $\Gamma_0,\Gamma_1,\Gamma_2,\Gamma_3$ 的估计 $\begin{bmatrix} vec(\Gamma_0') \\ vec(\Gamma_1') \\ vec(\Gamma_2') \\ vec(\Gamma_3') \end{bmatrix}$ 为，以 $\begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \end{bmatrix}$ 为因变量，以 $\begin{bmatrix} [Z_{-1}\otimes f_0, Z_{-1}\otimes f_{-1}, Z_{-1}\otimes f_{-2}, Z_{-1}\otimes f_{-3}] \\ [Z_0\otimes f_1, Z_0\otimes f_0, Z_0\otimes f_{-1}, Z_0\otimes f_{-2}] \\ \vdots \\ \\ [Z_4\otimes f_5, Z_4\otimes f_4, Z_4\otimes f_3, Z_4\otimes f_2] \end{bmatrix}$ 为自变量，做最小二乘回归。其中，

$vec(A)$ 表示向量化，表示将矩阵 A 的所有列按顺序排列起来组成一个向量。$\otimes$ 表示 kronecker 积。

给定上述参数的初始值，我们按照上述计算过程不断迭代就可以了，直到收敛。收敛可以定义为，相邻两步之间系数估计的差异的绝对值的最大值小于 1e-6。

上述算法，称为算法 1。还有算法 2，算法 2 计算起来比较快。算法 2 与算法 1 的差异在于对 $f_{-3}, f_{-2}, f_{-1}, f_0, f_1, f_2, f_3, f_4, f_5$ 的估计，对 $\Gamma_0,\Gamma_1,\Gamma_2,\Gamma_3$ 的估计，算法 1 和算法 2 相同。算法 2 中，我们认为在 T 比较大时，f_{-3}, f_{-2}, f_{-1} 的影响比较小，所以直接设置 f_{-3}, f_{-2}, f_{-1} 为 0。对于 $f_t, t=0,\ldots,5$ 的估计，我们只使用(1.12)中的第 t 行，即对于 f_t 的估计为，以 $r_t-(Z_{t-1}\Gamma_1 f_{t-1}+\ldots+Z_{t-1}\Gamma_3 f_{t-3})$ 为因变量，以 $Z_{t-1}\Gamma_0$ 为自变量，做最小二乘回归。

算法 2 的估计虽然简单，但是估计出来的系数是有问题的，有偏的。我们发现该问题的途径是：自己根据模型(1.11)，自己给定系数，生成数据，然后分别使用算法 1 和算法 2 去

估计系数，发现算法 2 估计的系数是错误的，算法 1 估计的系数是正确的。

我们事后分析，算法 2 错误的原因是估计 f_t 的时候只使用了(1.12)中的第t 行数据，这样会导致忽略了其他行中的 f_t 跟第t 行中的 f_t 相等（比如第0行的 f_0 跟第$1,2,3$ 行中的 f_0 是相等的 ）这一约束。

另外，模型(1.11)中的系数识别是不唯一的。原因是，如果 $\Gamma_0,\Gamma_1,\Gamma_2,\Gamma_3, f_{-3}, f_{-2}, f_{-1}, f_0, f_1, f_2, f_3, f_4, f_5$ 是一组使得(1.13)最小的系数，那么 $\Gamma_0 R^{-1},\Gamma_1 R^{-1},\Gamma_2 R^{-1},\Gamma_3 R^{-1}, Rf_{-3}, Rf_{-2}, Rf_{-1}, Rf_0, Rf_1, Rf_2, Rf_3, Rf_4, Rf_5$ 同样可以使得(1.13)最小。其中，R 是任意一个 $K\times K$ 的可逆矩阵。虽然系数不唯一，但是 $Z_{t-1}\Gamma_0 f_t + Z_{t-1}\Gamma_1 f_{t-1} + ... + Z_{t-1}\Gamma_3 f_{t-3}$ 或者预测值 $Z_{t-1}\Gamma_1 f_{t-1} + ... + Z_{t-1}\Gamma_3 f_{t-3}$ 都是唯一的。

5. 进一步发现

我们发现算法 1 比算法 2 好，是在数据生成过程（DGP）的残差项为 0 的情况下得到的，但是，将残差项设为非 0，或者实际数据生成过程的 lag 为 1，我们回归的时候按照 lag 为 2 来估计，都会很明显地影响估计收敛。其中原因还不清楚。

参考文献：

Bryan T. Kelly, Seth Pruitt, Yinan Su, Characteristics Are Covariances: A Unified Model of Risk and Return, Journal of Financial Economics (2019)

Gu, Shihao and Kelly, Bryan T. and Xiu, Dacheng, Autoencoder Asset Pricing Models (September 30, 2019). Yale ICF Working Paper No. 2019-04, Chicago Booth Research Paper No. 19-24

Shihao Gu, Bryan Kelly, Dacheng Xiu, Empirical Asset Pricing via Machine Learning, *The Review of Financial Studies*, Volume 33, Issue 5, May 2020, Pages 2223–2273

Hui Zou, Trevor Hastie and Robert Tibshirani (2006): Sparse Principal Component Analysis, Journal of Computational and Graphical Statistics, 15:2, 265-286