Chapter 2
Denoising and Detoning

1

목차

1 Motivation

- **금융 시장에서 공분산 행렬(Covariance matrix) 자주 사용**
  - Regression, estimate risks, optimize portfolios, simulate scenarios, clusters, reduce the dimensionality, etc
- **일반적으로 공분산행렬은 선형적 공행성(linear comovement)을 추정하기 위해 사용되며, 확률변수의 관측값을 통해 계산**
  - 다만 추정에는 노이즈가 포함되어 있음
  - 결함이 있는 데이터에서 산출된 공분산행렬 역시 문제가 있음
- **따라서 노이즈 문제를 해결하지 못하면, 공분산행렬을 활용한 모든 계산에 문제가 발생함**

1 Motivation

- **배경지식(금융)**
  - **포트폴리오**
    - **목적**
      - 리스크 배분 문제
    - **종류**
      - **Markowitz Portfolio**
        - Max return, Given Market Risk / Min risk, Given Expected Return
        - 개별 자산 간의 상관계수가 1이 아닌 경우라면 분산이 감소하여 이득을 얻을 수 있음
      - **Eigen Value Portfolio**
        - 개별자산의 수익률, 혹은 변동성에 대해 고유벡터를 뽑아서 배분
        - 공통요인이 있는 주식들 간의 높은 동행성이 있기 때문에 이를 분산하는 것

1 Motivation

- **배경지식(통계)**
  - **공분산 행렬(Covariance matrix)**
    - **공분산** : 두 변수들의 변동이 얼마나 닮았는가
      - $Cov(x, y) = E\{(x - \mu_x)(y - \mu_y)\}$
    - **공분산 행렬** : 2변량 이상에서 여러 조합의 두 변량 값들의 공분산을 행렬로 표현한 것
      - $Var(X) = \begin{pmatrix} Var(x_1) & \cdots & Cov(x_1, x_n) \\ \vdots & \ddots & \vdots \\ Cov(x_n, x_1) & \cdots & Var(x_n) \end{pmatrix}$
  - **상관 계수 행렬(Correlation Coefficient Matrix)**
    - 공분산의 경우 단위의 크기에 영향을 받음, 이를 보완하기 위해 확률변수의 절대적 크기에 영향을 받지 않도록 단위화
      - $\rho_{XY} = \frac{Cov(X,Y)}{\sqrt{Var(X),Var(Y)}}$
    - **상관관계 (correlation)**: 두 변수 사이에 선형 관계가 어느정도 있는가
    - **상관계수 (correlation coefficient)**: 상관관계에 대한 지표
      - 1이면 선형 관계가 가장 큼, 0이면 선형 관계가 없음
    - **상관계수 행렬(correlation coefficient matrix)** : 상관관계를 행렬로 나타낸 것
      - 변수가 1개일 때는 상관계수가 1개만 나오지만, 변수가 여러 개일 경우에는 가능한 모든 조합으로 나옴(행렬 형식)

1 Motivation

- 배경지식(금융 notation)
  - **수익률** : $(N \times 1)$의 열 벡터
    - N개의 자산에 대한 수익률
    - $\boldsymbol{r} = (r_1, \cdots, r_N)^T$
  - **기대 수익률** : $(N \times 1)$의 열 벡터
    - N개의 자산에 대한 수익률의 기댓값(실제값이 아닌 추정치)
    - $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_N)^T = E(r), \quad where\ E(r_i) = \mu_i$
  - **변동성** : $(N \times 1)$의 열 벡터
    - 각각의 자산에 대한 변동성(위험에 대한 지표)
    - $\boldsymbol{\sigma} = (\sigma_1, \cdots, \sigma_N)^T,\ Var(r_i) = \sigma_i^2$
  - **샤프 비율** : $(N \times 1)$의 열 벡터
    - 자산이나 포트폴리오의 성과를 나타내는 지표, 변동성(위험)을 고려한 수익률
    - $SR = (sr_1, \cdots, sr_N)^T,\ where\ sr_i = \frac{\mu_i - r_f}{\sigma_i}$

1 Motivation

- 배경지식(금융 notation)
  - **변동성 대각행렬(Diagonal Matrix of Volatility)** $(N \times N$ 행렬$)$
    - 주대각선 성분(각 자산의 변동성)을 제외한 다른 모든 성분이 0인 행렬
    - $\Lambda = diag(\sigma_i) = \begin{pmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_N \end{pmatrix}$
    - $SR = \Lambda^{-1}\mu$
  - **상관계수 행렬** : $(N \times N$ 행렬$)$
    - 상관계수 : 어떤 두 자산 간의 상관관계를 나타내는 지표, 상관계수 행렬이란 이를 행렬로 표현한 것, 대각선은 동일 자산의 상관계수이므로 1
    - $C = (\rho_{ik})_{N \times N} = Corr(r, r), \quad \rho_{ii} = 1$
  - **공분산 행렬** : $(N \times N$ 행렬$)$
    - 두 자산간의 공분산의 조합을 행렬로 표시한 것
    - $\Sigma = (\sigma_{ij})_{N \times N} = Cov, \sigma_{ij} = \rho_{Ij}\sigma_i\sigma_j$
  - **가중치** : $(N \times 1)$의 열벡터
    - 포트폴리오에 속하는 각 자산에 대한 가중치(최적화의 목표)
    - $w = (w_1, \cdots, w_N)^T$

1 Motivation

- **배경지식(포트폴리오 notation)**
  - **포트폴리오 수익률** : (스칼라)
    - 각 자산들의 수익률을 가중치별로 가중평균한 값
    - $r_p = \sum w_i r_i = w^T r$
  - **포트폴리오 기대수익률** : (스칼라)
    - 포트폴리오의 기대수익률
    - $\mu_p = E(r_p) = \sum w_i \mu_i = w^T \mu$
  - **포트폴리오 변동성** : (스칼라)
    - 포트폴리오의 분산
    - $\sigma_p^2 = Var(r_p) = \sum\sum w_i w_j \sigma_{ij} = w^T \Sigma w$
  - **베타** : $(N \times 1)$의 열벡터
    - 전체 포트폴리오 대비 각 자산의 베타 계수, 베타란 개별 자산이 전체 포트폴리오 변동에 얼마나 민감하게 반응하는지에 대함
    - $\beta = (\beta_1, \cdots, \beta_N)^T = \frac{\Sigma w}{\sigma_p^2}$

1 Motivation

- 배경지식(선형대수)
  - 대칭행렬(공분산행렬)은 고유값 대각화가 가능하며, 직교행렬로 대각화가 가능함
  - $\boldsymbol{C} = \boldsymbol{W}\boldsymbol{\Lambda}\mathbf{W}^{-1} = \mathbf{W}\boldsymbol{\Lambda}\mathbf{W}'$
    - $\boldsymbol{W}\boldsymbol{W}' = \boldsymbol{I}$
  - $\boldsymbol{C}[\boldsymbol{w}_1 \ \boldsymbol{w}_2 \ \cdots \ \boldsymbol{w}_n] = [\boldsymbol{\lambda}_1\boldsymbol{w}_1 \ \boldsymbol{\lambda}_2\boldsymbol{w} \ \cdots \boldsymbol{\lambda}_n\boldsymbol{w}_n] = [\boldsymbol{w}_1 \ \boldsymbol{w} \cdots \boldsymbol{w}_n]\begin{bmatrix} \boldsymbol{\lambda}_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \boldsymbol{\lambda}_n \end{bmatrix}$
  - $\boldsymbol{C} = [\boldsymbol{w}_1 \ \boldsymbol{w}_2 \cdots \boldsymbol{w}_n]\begin{bmatrix} \boldsymbol{\lambda}_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \boldsymbol{\lambda}_n \end{bmatrix}\begin{bmatrix} \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \\ \cdots \\ \boldsymbol{w}_n \end{bmatrix} = \mathbf{W}\boldsymbol{\Lambda}\mathbf{W}'$

8

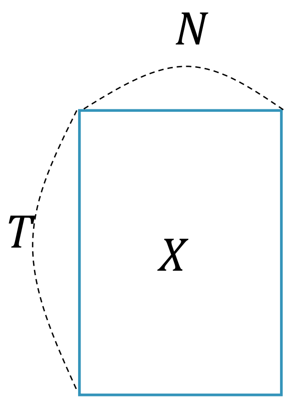
Conclusions

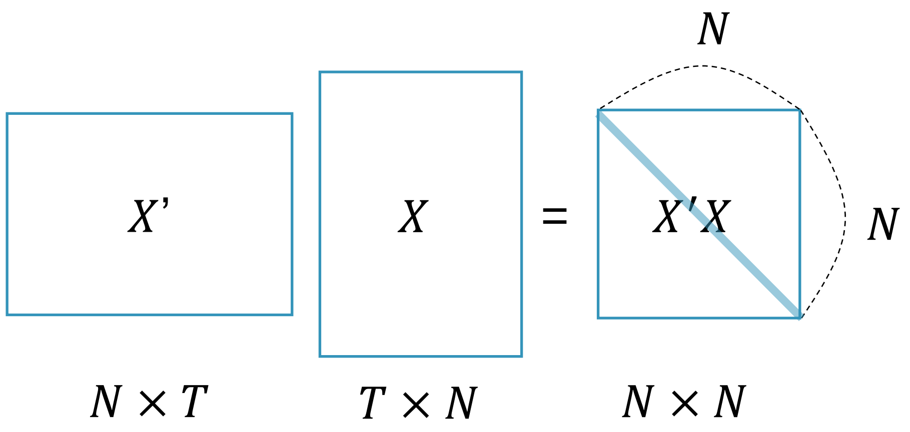
- 일반적으로 금융 시장에서 공분산 행렬은 ill-conditioned함을 보임
  - Parameters의 수에 비해 낮은 개수의 관측값 때문
- 공분산행렬을 다루기 위해서는 이를 고려해야 함
- 공분산행렬이 non-singular이더라도, determinant가 작다면, inversion process 도중에서 error 증가할 수 있음
  - 이런 error는 자산 배분에 문제에 나쁜 영향을 끼칠 수 있음
- Marcenko-Pastur Theorem은 랜덤 행렬의 eigenvalues의 분포 표상
  - 이 분포에 fitting 함으로써, signal 과 연계된 eigenvalues와 noise와 연계된 eigenvalues 분간 가능
  - Noise와 연계된 eigenvalues을 조정함으로써, signal의 희석 없이 ill-conditioning 문제 해결 가능
    - 1) threshold method : 실제 noise로 생성된 varinace와 상관 없이 fixed amount of variance를 결정하는 component 선택
    - 2) shrinkage method : signal을 조금 줄이는 한이 있더라도 noise를 제거
- Denoising은 lowest eigenvalue를 높임으로써 조건수를 낮춤
- Highest eigenvalue를 제거함으로써도 조건수를 낮출 수 있음(market components를 제거하는 효과)

2

The Marchenko-Pastur Theorem

- **목적**
  - **주식수익률간 상관행렬의 고유치분포를 살펴보는 것**
- **정의**
  - **X : $T \times N$ 크기의 I.I.D matrix**(T : time, N : # of assets로 추정)
    - $\boldsymbol{mean : 0, variance : \sigma^2}$
  - **C : $C = T^{-1}X'X$ (μ=0이므로 공분산행렬 표시)**
    - Marcenko-Pastur Probability density function(PDF)에 점근적으로 근사하는 eigenvalues λ 존재
    - $f|\lambda| \begin{cases} \frac{T}{N}\sqrt{\frac{(\lambda_+ - \lambda)(\lambda - \lambda_-)}{2\pi\lambda\sigma^2}} \; if \; \lambda \in [\lambda_-, \lambda_+] \\ 0 \; \; if \; \lambda \notin [\lambda_-, \lambda_+] \end{cases}$
      - $\lambda_+ = \sigma^2\left(1 + \sqrt{\frac{N}{T}}\right)^2, \lambda_- = \sigma^2\left(1 - \sqrt{\frac{N}{T}}\right)^2$
        - 이론적 고유치 분포의 상한과 하한
          - 무작위 상관행렬에 속하는 고유치의 범위
        - 상한과 하한에 속하는 고유치는 확인할 수 있는 무작위 요인들의 속성 표현
          - 무작위 값이 아니므로 경제적 의미 확인 가능
        - 이 범위를 벗어난 고유값은 확률행렬이론으로부터 일탈되어 있다.
    - $\sigma^2 = 1$일 때, C는 correlation matrix와 같다.





2

The Marchenko-Pastur Theorem

```python
from sklearn.neighbors.kde import KernelDensity
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
%matplotlib inline

def getPCA(matrix) :
    # Get eVal, eVec from a Hermitian matrix
    ## np.linalg.eigh : 복잡한 Hermitian 또는 실제 대칭 행렬의 고유 값과 고유 벡터를
    eVal, eVec = np.linalg.eigh(matrix) ## eigenvalue, eigenvector
    indices = eVal.argsort()[::-1] # arguments for sorting eVal desc
    eVal = eVal[indices]
    eVec = eVec[:, indices]
    eVal = np.diagflat(eVal)
    return eVal, eVec

def mpPDF(var, q, pts):
    # Marcenko-Pastur pdf
    # q=T/N
    eMin,eMax=var*(1-np.sqrt(1./q))**2, var*(1+np.sqrt(1./q))**2
    eVal = np.linspace(eMin, eMax, pts).flatten()
    pdf = q/(2*np.pi*var*eVal)*((eMax-eVal)*(eVal-eMin))**.5
    pdf = pd.Series(pdf, index = eVal)
    return pdf
```

```python
def fitKDE(obs, bWidth = 0.25, kernel = 'gaussian', x=None) :
    # Fit kernel to a series of obs, and derive the probability of obs
    # obs : the array of values on which the fit KDE will be evaluated
    if len(obs.shape) == 1 :
        obs = obs.reshape(-1,1)
    if x is None :
        x = np.unique(obs).reshape(-1, 1)
    if len(x.shape) == 1 :
        x = x.reshape(-1, 1)
    kde = KernelDensity(kernel = kernel, bandwidth = bWidth).fit(obs)
    logProb = kde.score_samples(x) #log(density)
    pdf = pd.Series(np.exp(logProb), index = x.flatten())
    return pdf

def draw_mpPDF(pdf0, pdf1):
    fig, ax = plt.subplots(figsize=(10,7))
    ax.plot(pdf0, label = "Marcenko-Pastur")
    ax.plot(pdf1, '--',label = "Empirical:KDE")
    ax.set_title('Figure 2.1 A visualiztion of the Marcenko-Pastur theorem\n', fontsize = 18)
    ax.set_ylabel('prob[$\lambda$]', fontsize = 15)
    ax.set_xlabel('$\lambda$', fontsize = 15)
    leg = ax.legend(fontsize= 15)
```

2

The Marchenko-Pastur Theorem

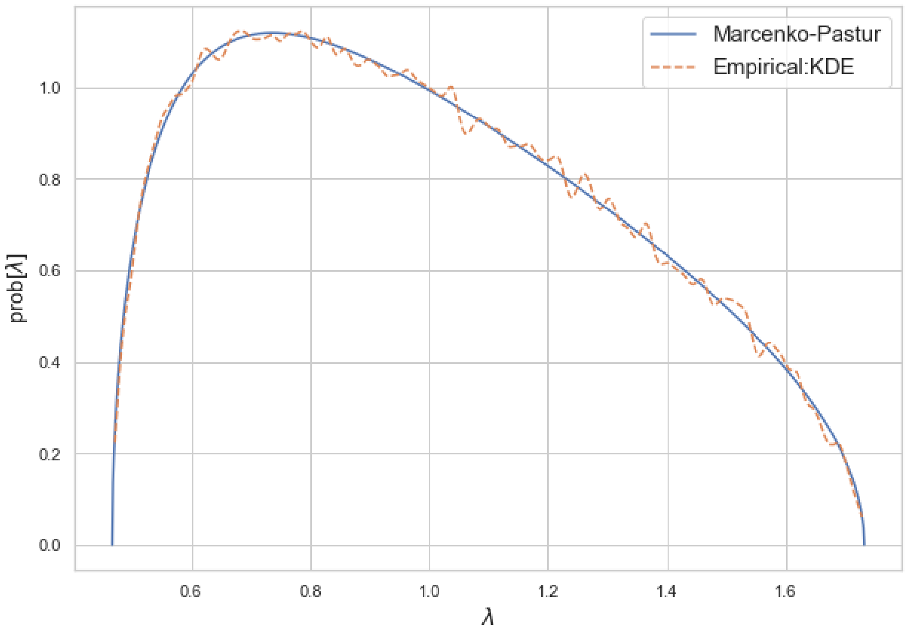
- **모델**
  - **Marcenko-Pastur Theorem**
    - 정사각 Random Matrix의 eigenvalue 분포에 대한 정리
    - 공분산 행렬에서 Noise와 Signal을 어느정도 구분할 수 있음
  - $\lambda \in [\lambda_-, \lambda_+]$: random behavior, $\lambda \notin [\lambda_-, \lambda_+]$: nonrandom behavior
  - $\lambda \in [0, \lambda_+]$ 를 noise와 연결시켜서 생각

2

The Marchenko-Pastur Theorem

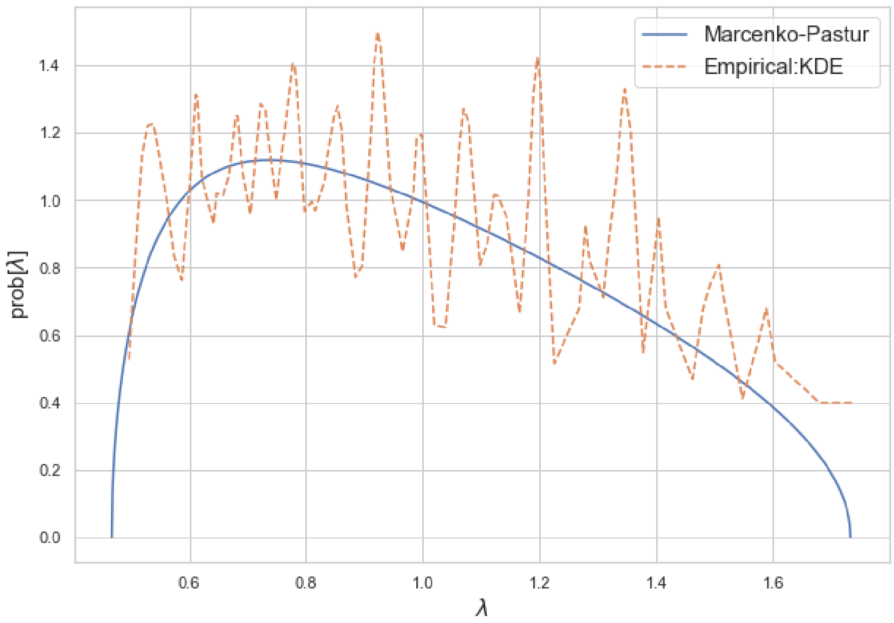
```
### T : time, N : # of assets
T, N = 10000, 1000
## 이론적 random matrix 구현
## 행렬 X는 T X N 크기의 랜덤행렬
X = np.random.normal(size = (T,N))
## C는 1/T x X'X
C = np.corrcoef(X, rowvar=0)
## PCA 진행하여 eigenvalue, eigenvector 추출
eVal0, eVec0 = getPCA(C)
## eVal0 : eigen Values, eVec0 : eigen vectors
### Random Matrix라면 eigen Values가 MPpdf 범위 내에 존재해야 함
q = T/float(N)
# Marcenko-Pastur pdf 구하기
pdf0 = mpPDF(1.0, q = q, pts = 1000)
pdf1 = fitKDE(np.diag(eVal0), bWidth = 0.01)
draw_mpPDF(pdf0, pdf1)
```

Figure 2.1 A visualiztion of the Marcenko-Pastur theorem




```
### T : time, N : # of assets
T, N = 1000, 100
## 이론적 random matrix 구현
## 행렬 X는 T X N 크기의 랜덤행렬
X = np.random.normal(size = (T,N))
## C는 1/T x X'X
C = np.corrcoef(X, rowvar=0)
## PCA 진행하여 eigenvalue, eigenvector 추출
eVal0, eVec0 = getPCA(C)
## eVal0 : eigen Values, eVec0 : eigen vectors
### Random Matrix라면 eigen Values가 MPpdf 범위 내에 존재해야 함
q = T/float(N)
# Marcenko-Pastur pdf 구하기
pdf0 = mpPDF(1.0, q = q, pts = 1000)
pdf1 = fitKDE(np.diag(eVal0), bWidth = 0.01)
draw_mpPDF(pdf0, pdf1)
```

Figure 2.1 A visualiztion of the Marcenko-Pastur theorem



3

Random Matrix with Signal

- 앞서는 random matrix의 경우를 알아봄
- 실제 세계에서 공분산행렬(empirical correlation matrix)의 경우 random이 아닌 eigenvectors 존재

```
def getRndCov(nCOls, nFacts) :
    w = np.random.normal(size = (nCols, nFacts))
    # Random cov matrix, however not full rank
    cov = np.dot(w, w.T)
    # Make full rank cov
    cov += np.diag(np.random.uniform(size=nCols))
    return cov

def cov2corr(cov) :
    # Derive the correlation matrix from a covariance matrix
    std = np.sqrt(np.diag(cov))
    corr = cov / np.outer(std,std)
    # Clipping to prevent numerical errors
    corr[corr <-1] = -1
    corr[corr > 1] = 1
    return corr
```

```
## alpha : random covariance의 비중
alpha = 0.995
## nCols : 전체 asset의 수
nCols = 1000
# nFact : signal을 갖고 있는 # of columns
nFact = 100
q = 10
# cov : covariance matrix
cov = np.cov(np.random.normal(size=(nCols*q, nCols)), rowvar = 0)
# make noise + signal covariance matrix (weights = alpha)
cov = alpha * cov + (1-alpha) * getRndCov(nCols, nFact)
corr0 = cov2corr(cov)
eVal0, eVec0 = getPCA(corr0)
```

4

Fitting the Marcenko-Pastur Distribution

- 몇몇 분산만이 random eigenvectors에 의해 생성되기 때문에, 위 식을 이용하여 σ^2를 조절
  - 가장 높은 값을 갖는 eigenvalue가 λ_+를 넘어설 경우 이는 공분산행렬이 랜덤이 아님 의미
    - 이 경우 σ^2를 $\sigma^2(1-\frac{\lambda_+}{N})$로 변환
- Function $f[\lambda]$를 eigenvalues로부터 추출된 경험적 분포에 적합하여, 내재 변동성(implied σ^2)을 얻음
  - 데이터로부터 σ를 계산하는 것이 아니라, 이론으로부터 σ역산
- 이 경우 상관관계행렬에서부터 얻은 random eigenvectors로부터 설명된 분산 얻을 수 있음
  - 이에 따라 Nonrandom eigenvectors로부터 조정된 cutoff level λ_+를 얻음
- 그림
  - Marcenko-Pastur dist 밖에 있는 Eigenvalues는 Signal에 속함
  - 코드로 구현하였을 때, 초기 세팅 signa의 개수와, dist 밖에 있는 eignevalues의 개수가 같음
    - 이는 시그널과 노이즈를 구분할 수 있음 의미

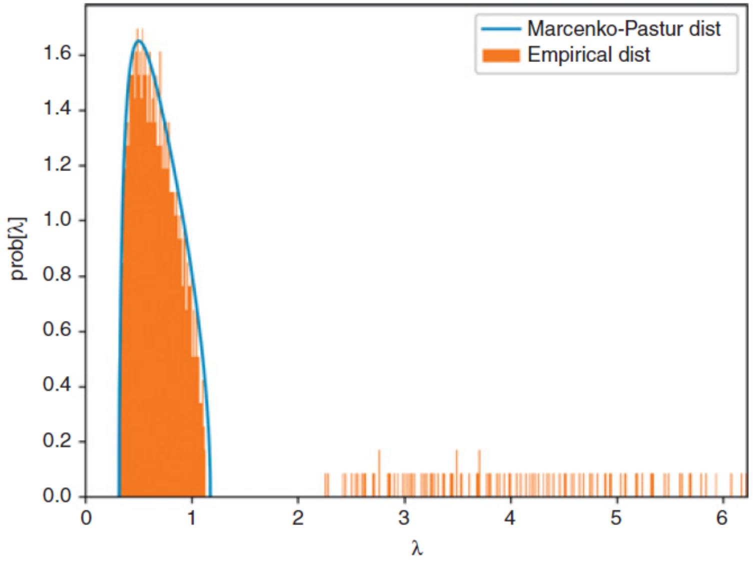


Figure 2.2 Fitting the Marcenko–Pastur PDF on a noisy covariance matrix.

4

Fitting the Marcenko-Pastur Distribution

- Marcenko-Pastur PDF를 시그널은 함유한 랜덤 공분산 행렬에 적합시킴
  - 관측된 eigenvalue의 커널밀도추정(Kernel Density Estimate)과 PDF와의 차이의 제곱합 최소화하는 σ^2 탐색
    - λ_+ : eMax()
    - σ^2 : var()
    - # of factors : nFacts()

```python
from scipy.optimize import minimize
def errPDFs(var, eVal, q, bWidth, pts = 1000) :
    # Fit error
    # Theoretical pdf
    # prevent var values in list format
    if hasattr(var, "__len__") :
        if len(var) == 1 :
            var = var[0]
        else :
            raise ValueErroral ("var  must be scalar")
    pdf0 = mpPDF(var, q, pts)
    # Empirical pdf
    pdf1 = fitKDE(eVal, bWidth, x=pdf0.index.values)
    sse = np.sum((pdf1 - pdf0)**2)
    return sse

def findMaxEval(eVal, q, bWidth) :
    # Find Max random eVal by fitting Marcenko's distribution
    # Scipy.optimize.minimize
    # fun : Theobjective function to be minimized
    # x0 : ndarray, shape(n,), initial guess. array of real emlements of size

    out = minimize(fun = lambda *x : errPDFs(*x), x0 = .5, # first arg : var
                   args = (eVal, q, bWidth), bounds = ((1e-5, 1 - 1e-5),))
    if out['success'] :
        var = out['x'][0]
    else :
        var = 1
    eMax = var * (1 + (1.0 / q) ** 0.5) ** 2
    return eMax, var
```

5 Denoising

- 금융 영역에서 numerically ill-conditioned covariance matrix 자주 축소함
  - Ill-conditioned : 큰 조건수(high condition number)을 가짐
    - Input의 작은 변화만 있어도 결과값에 큰 차이가 발생함
- Covariance matrix를 대각화 하면 할수록 조건수가 줄어드는 효과
- 하지만 noise와 signal을 구분하지 않은 축소는 noise와 함께 signal도 줄이는 효과를 야기할 수 있음
  - 이는 signal이 이미 약한 경우(금융 분야의 낮은 SNR), 이미 없는 signal을 더 제거하는 양상을 보일 수 있음
- 앞서 논증한 eigenvalues를 활용하여 noise와 signal을 구분하는 기법을 활용하여 denoising correlation matrix에 활용

5 Denoising

- **Constant Residual Eigenvalue Method**
  - 모든 랜덤 eigenvectors에 대하여 constant eigenvalue 세팅
  - 수식
    - λ_n : 모든 eigenvalues의 집합, 내림차순으로 정렬
      - $\lambda_i > \lambda_+,\ \lambda_{i+1} \leq \lambda_+$
      - Corrected eigenvalues : $\lambda_j = \left(\frac{1}{N-i}\right)\sum_{k=i+1}^{N} \lambda_k \,, j = i+1, \cdots, N$
    - 고유값분해($VW = W\Lambda$)를 활용
      - C_1 : denoised correlation matrix
      - $\tilde{C_1} = W\tilde{\Lambda}W'$
        - $C_1 = \tilde{C}_1\left[\left(diag[\widetilde{C_1}]\right)^{\frac{1}{2}}\left(diag[\widetilde{C_1}]\right)^{\frac{1}{2}\prime}\right]^{-1}$
        - $\widetilde{\Lambda}$: corrected eigenvalues를 가진 대각행렬

1000개의 columns 중 100개만 signal인 상황

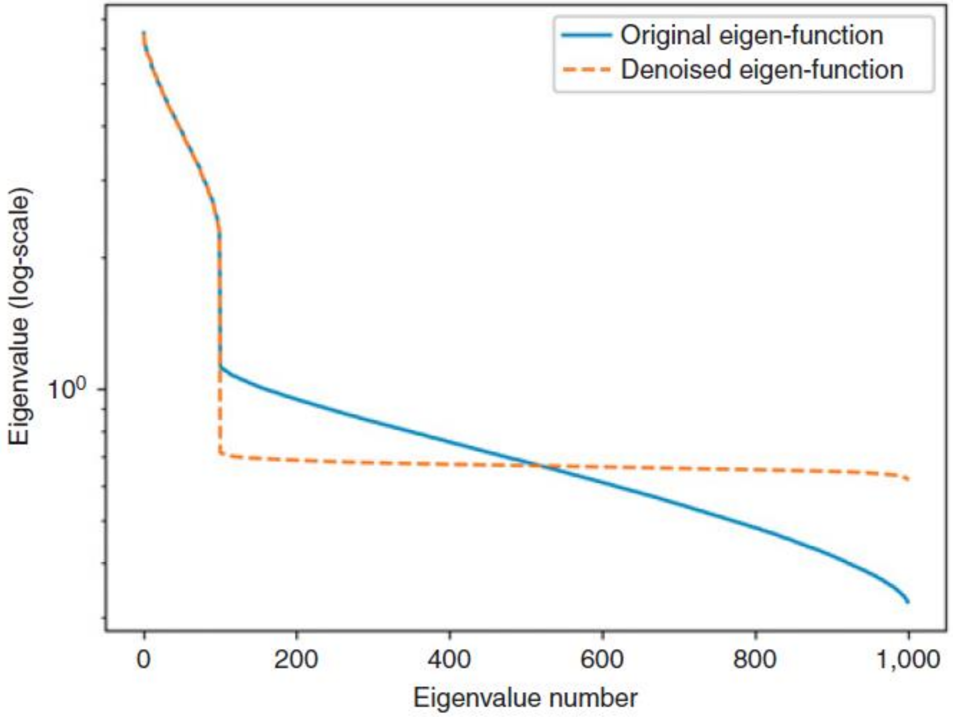
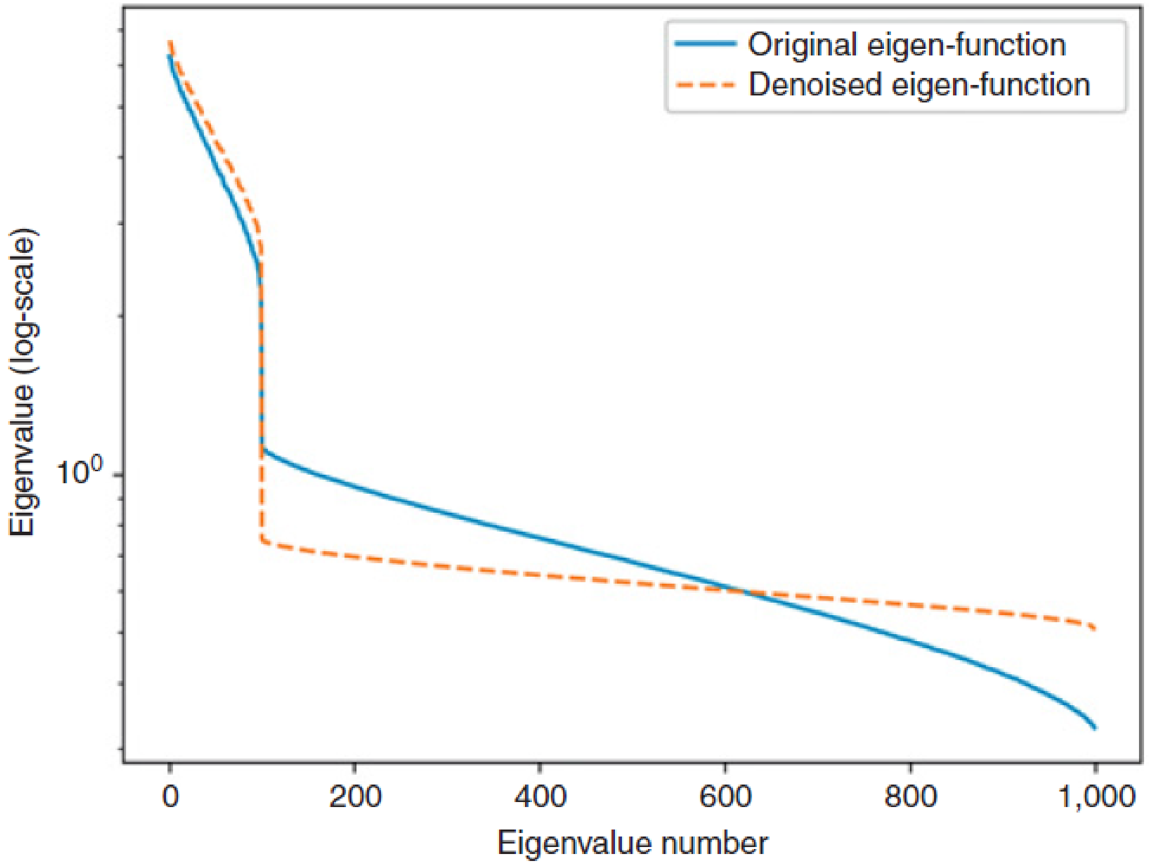


Figure 2.3 A comparison of eigenvalues before and after applying the residual eigenvalue method.

5

Denoising

- Targeted Shrinkage
  - 수식
    - $C_1 = W_L \Lambda_L W_L' + \alpha W_R \Lambda_R W_R' + (1-\alpha) diag[W_R \Lambda_R W_R']$
      - α : noise를 얼마나 줄일 것인지에 대한 계수
      - W_R, Λ_R : $\{n | \lambda_n \leq \lambda_+\}$와 연계된 eigenvectors(eigenvalues)
      - W_L, Λ_L : $\{n | \lambda_n > \lambda_+\}$와 연계된 eigenvectors(eigenvalues)



6 Detoning

- 금융 상관관계 행렬들은 시장구성요소들을 통합함
  - 시장구성요소 : $W_{n,1} \approx N^{-\frac{1}{2}}$일 때 첫번째 eigenvector로 특징
  - 시장구성요소는 공분산행렬에 존재하는 모든 자산에 영향을 끼침
- 클러스터링 관점에서는 시장 구성요소를 제하는 것이 타당함
  - 시장구성요소의 영향이 큰 상황에서는 공분산행렬을 클러스터링하는 것이 어려움
    - 공통요소의 영향이 크기 때문에, 상이함을 포착하기 어려움
  - 시장구성요소를 제거한다면, 각 대상들의 특징을 더 유의미하게 비교 가능
- Detoning : beta-adjusted return을 위한 PCA 방법

6

Detoning

- Denoised 공분산 행렬에 대해서 시장구성요소를 제거할 수 있음
- $\tilde{C}_2 = C_1 - W_M \Lambda_M W_M' = W_D \Lambda_D W_D'$
- $C_2 = \tilde{C}_2 \left[\left(diag[\widetilde{C_2}]\right)^{\frac{1}{2}} \left(diag\ [\widetilde{C_2}]\right)^{\frac{1}{2}\prime} \right]^{-1}$
  - $W_M(\Lambda_M)$: 시장구성요소와 연계된 eigenvectors(eigenvalues)
    - 일반적으로 하나이나, 하나 이상 존재할 수 있음
  - $W_D(\Lambda_D)$: 비시장구성요소와 연계된 eigenvectors(eigenvalues)
- Detoned 공분산 행렬은 singular함
  - Eigen value를 제거했기 때문(full rank 아님)
  - 그렇지만 클러스터링 관점에서 invertible은 중요 관심사가 아니기 때문에 허용 가능
- C_2는 mean-variance portfolio 구축에 직접적으로도 사용은 불가능
  - 다만 선택된 고유값들에 대해서 포트폴리오 최적화 가능

7 Experimental Results

- Minimum Variance Portfolio
  - S&P 500의 detoned된 공분산 행렬을 표상하는 행렬 생성
  - 일반성을 위하여
    - variance는 5% ~ 20% 사이에서 uniform distribution 따르게 설정
    - mean은 normal distribution을 따르게 설정
    - 표본평균의 평균과, 분산이 covariance matrix의 분산을 따르게 설정
      - 경제학에서 efficient market일 경우, 모든 자산이 같은 SR을 갖고 있다는 가정과 부합

```
nBlocks = 10
bSize = 50
bCorr = 0.5
np.random.seed(0)

mu0, cov0 = formTrueMatrix(nBlocks, bSize, bCorr)
```

```
print(mu0.shape, cov0.shape)
```

```
(500, 1) (500, 500)
```

Snippet 2.7 Generating a Block-Diagonal Covariance Matrix and a Vector of Means

```
from scipy.linalg import block_diag
from sklearn.covariance import LedoitWolf

# pre-define functions at Snippet 2.9
def corr2cov(corr, std):
    cov = corr * np.outer(std,std)
    return cov

def formBlockMatrix(nBlocks, bSize, bCorr) :
    # make square matrix (bSize * bSize) (all elements' value is bCorr)
    block = np.ones((bSize, bSize)) * bCorr
    # change diagonal elements to 1
    block[range(bSize), range(bSize)] = 1
    # Make block diagonal matrix with size (bSize*nBlocks by bSize*nBlocks)
    corr = block_diag(*([block]*nBlocks))
    return corr

def formTrueMatrix(nBlocks, bSize, bCorr) :
    # make BlockMatrix in DataFrame Format
    corr0 = formBlockMatrix(nBlocks, bSize, bCorr)
    corr0 = pd.DataFrame(corr0)
    # make columns index to list and shuffle
    cols = corr0.columns.tolist()
    np.random.shuffle(cols)
    # corr matrix shuffled along column & index
    corr0 = corr0[cols].loc[cols].copy(deep=True)
    # final calculations
    std0 = np.random.uniform(low = 0.05, high = 0.2, size = corr0.shape[0])
    cov0 = corr2cov(corr0, std0)
    mu0 = np.random.normal(loc = std0, scale = std0, size = cov0.shape[0]).reshape(-1,1)
    return mu0, cov0
```

7

Experimental Results

- Minimum Variance Portfolio
  - S&P 500의 detoned된 공분산 행렬을 표상하는 행렬 생성
  - 일반성을 위하여
    - variance는 5% ~ 20% 사이에서 uniform distribution 따르게 설정
    - mean은 normal distribution을 따르게 설정
    - 표본평균의 평균과, 분산이 covariance matrix의 분산을 따르게 설정
      - 경제학에서 efficient market일 경우, 모든 자산이 같은 SR을 갖고 있다는 가정과 부합

Snippet 2.8 Generating the Empirical Covariance Matrix

```python
def simCovMu(mu0, cov0, nObs, shrink=False):
    # TODO add comments
    x = np.random.multivariate_normal(mu0.flatten(), cov0, size = nObs)
    mu1 = x.mean(axis = 0).reshape(-1,1)
    if shrink :
        cov1 = LedoitWolf().fit(x).covariance_
    else :
        cov1 = np.cov(x, rowvar = 0)
    return mu1, cov1
```

7

Experimental Results

- Minimum Variance Portfolio
  - $T \times N$ 크기의 random matrix X 의 공분산행렬 추출
- deNoiseCov
  - Constant residual eigenvalue method를 활용하여 desnoise 진행

Snippet 2.9 Denoising of the Empirical Covariance Matrix

```
def corr2cov(corr, std):
    cov = corr * np.outer(std,std)
    return cov

def deNoiseCov(cov0, q, bWidth):

    # TODO add comments
    corr0 = cov2corr(cov0)
    eVal0, eVec0 = getPCA(corr0)
    eMax0, var0 = findMaxEval(np.diag(eVal0), q, bWidth)

    # Finding noise starting point
    nFacts0 = eVal0.shape[0] - np.diag(eVal0)[::-1].searchsorted(eMax0)

    corr1 = denoisedCorr(eVal0, eVec0, nFacts0)

    cov1 = corr2cov(corr1, np.diag(cov0)** 0.5)

    return cov1
```

7 Experimental Results

- Minimum Variance Portfolio
  - 실험 구성
    - 1000번의 Monte Carlo 실험 진행
      - 1. 랜범 공분산 행렬 추출(T = 1000)
        - Shrinkage 적용 0 or 1
      - 2. 공분산 행렬 denoising
        - Denoising 적용 0 or 1
      - 3. Minimum Variance Portfolio 추출(optPort)
  - 결과
    - 평가 기준 : RMSE
    - Denoising이 Shrinkage에 비해 더 효과적임
      - Denoising과 shrinkage를 모두 사용할 경우 65.6%의 RMSE개선 효과 보임

Snippet 2.10 Denoising of the Emprical Covariance Matrix

```
def optPort(cov, mu=None):

    # TODO explain the function

    inv = np.linalg.inv(cov)
    ones = np.ones(shape = (inv.shape[0], 1))
    if mu is None :
        mu = ones

    w = np.dot(inv, mu)
    w /= np.dot(ones.T, w)

    return w
```

	Not denoised	Denoised
Not shrunk	4.95E–03	1.99E–03
Shrunk	3.45E–03	1.70E–03

Figure 2.5 RMSE for combinations of denoising and shrinkage (minimum variance portfolio).

Experimental Results

- Maximum Sharpe Ratio Portfolio
  - SR 기준일 경우에도 shringkage보다 denoising이 더 큰 효과 보임
    - Shrinkage는 70% 감소 효과, denoising은 94.44% 감소 효과
    - Shrinkage와 denoising을 결합했을 때에는 미미한 효과

	Not denoised	Denoised
Not shrunk	9.48E–01	5.27E–02
Shrunk	2.77E–01	5.17E–02

Figure 2.6 RMSE for combinations of denoising and shrinkage (maximum Sharpe ratio portfolio).

8 Conclusions

- 일반적으로 금융 시장에서 공분산 행렬은 ill-conditioned함을 보임
  - Parameters의 수에 비해 낮은 개수의 관측값 때문
- 공분산행렬을 다루기 위해서는 이를 고려해야 함
- 공분산행렬이 non-singular이더라도, determinant가 작다면, inversion process 도중에서 error 증가할 수 있음
  - 이런 error는 자산 배분에 문제에 나쁜 영향을 끼칠 수 있음
- Marcenko-Pastur Theorem은 랜덤 행렬의 eigenvalues의 분포 표상
  - 이 분포에 fitting 함으로써, signal 과 연계된 eigenvalues와 noise와 연계된 eigenvalues 분간 가능
  - Noise와 연계된 eigenvalues을 조정함으로써, signal의 희석 없이 ill-conditioning 문제 해결 가능
    - 1) threshold method : 실제 noise로 생성된 varinace와 상관 없이 fixed amount of variance를 결정하는 component 선택
    - 2) shrinkage method : signal을 조금 줄이는 한이 있더라도 noise를 제거
- Denoising은 lowest eigenvalue를 높임으로써 조건수를 낮춤
- Highest eigenvalue를 제거함으로써도 조건수를 낮출 수 있음(market components를 제거하는 효과)

Thank you