Chapter 21
무차별 대입과 양자 컴퓨터

목차

1 동기

- 이산 수학은 머신러닝 문제에서 자연스럽게 등장
  - 대부분 해석학적 해법을 갖고 있지 않음
    - 무차별 대입법으로만 해결 가능
  - 양자 컴퓨터는 정수 최적화 문제 해결에 적절함

2

조합적 최적화

- **정의**
  - 유한한 가능 해를 가진 문제
  - 유한한 개수의 변수 이산값을 조합한 결과
    - 가능한 조합의 개수가 커질수록 완전 검색 불가능
- **원인**
  - 일반 컴퓨터가 가능한 해를 순차적으로 계산하고 저장하기 때문
- **해결**
  - 모든 가능한 해법을 동시에 계산
    - 양자 컴퓨터의 목표
      - 0, 1논리뿐 아니라 큐비트에 의존
      - 선형 중첩을 저장할 수 있음

3 목적 함수

- **모델**
  - 수익률
    - 다변량 정규분포
    - 시간 독립적
    - 시간에 대해 동일하게 분포
- **거래 궤적**
  - $\boldsymbol{N} \times \boldsymbol{H}$ 행렬 $\boldsymbol{w}$
    - 거래 호라이즌 $\boldsymbol{h}$
      - 평균 예측 : $\boldsymbol{\mu_h}$
      - 평균 분산 : $\boldsymbol{V_h}$
      - 예측 비용 함수 : $\boldsymbol{\tau_h[w]}$
  - 주어진 거래 궤적 $\boldsymbol{w}$에 대한 기대 투자 수익률 $\boldsymbol{r}$
    - $\boldsymbol{r = diag[\mu' w] - \tau[w]}$

3 목적 함수

- 주어진 거래 궤적 $\boldsymbol{w}$에 대한 기대 투자 수익률 $\boldsymbol{r}$
  - $\boldsymbol{r} = \boldsymbol{diag}[\boldsymbol{\mu}'\boldsymbol{w}] - \boldsymbol{\tau}[\boldsymbol{w}]$
    - $\boldsymbol{\tau_1}[\boldsymbol{w}] = \sum \boldsymbol{c_{n,1}}\sqrt{|\boldsymbol{w_{n,1}} - \boldsymbol{w_n^*}|}$
    - $\boldsymbol{h} = \boldsymbol{2}, \cdots, \boldsymbol{H} \rightarrow \boldsymbol{\tau_h}[\boldsymbol{w}] = \sum \boldsymbol{c_{n,h}}\sqrt{|\boldsymbol{w_{n,h}} - \boldsymbol{w_{n,h-1}}|}$
    - $\boldsymbol{w_n^*}$은 상품 $\boldsymbol{n}$에 대한 초기 배분이다.
  - $\boldsymbol{\tau}[\boldsymbol{w}]$는 거래 비용의 $\boldsymbol{H} \times \boldsymbol{1}$벡터
    - 각 자산에 연계된 거래 비용은 자산 배분 변화의 제곱근의 합을 $\boldsymbol{h}$에 대해 변하는 자산 특정 계수 $\boldsymbol{C_h}$로 조정한 것
    - $\boldsymbol{C_h}$는 거래 비용을 결정하는 $\boldsymbol{N} \times \boldsymbol{1}$ 벡터

  - $\boldsymbol{SR}[\boldsymbol{r}] = \frac{\sum \boldsymbol{\mu_h'}\boldsymbol{w_h} - \boldsymbol{\tau_h}[\boldsymbol{w}]}{\sqrt{\sum \boldsymbol{w_h'}\boldsymbol{V_h}\boldsymbol{w_h}}}$

4

문제

- 목적함수
  - $Max\ SR[r]$
  - $s.t.: \sum |w_{i,h}| = 1\ \forall h = 1, \cdots, H$
- 전역 동적 최적
  - 비연속 거래 비용은 r에 내재되어 있음
- 특징
  - 컨벡스 최적화 문제가 아님
    - μ_h와 V_h가 h에 따라 변하므로 수익률은 동일한 분포가 아님
    - $\mu_h[w]$ 는 불연속이고, h에 따라 변함
    - 목적함수 $SR[r]$은 컨벡스가 아님

5 정수 최적화 기법

- 비둘기 집 분할
  - $\boldsymbol{K}$단위의 자본금을 $\boldsymbol{N}$ 자산에 배분하는 가짓수 생각
    - $K > N$
    - $x_1 + \cdots + x_N = K$ 식의 음이 아닌 정수 해의 개수 찾는 것과 동일
- 가능한 정적 해법
  - $\boldsymbol{h}$에 대해 가능한 모든 해법의 집합 계산
    - $\boldsymbol{\Omega}$ 로 표시
  - $\boldsymbol{K}$ 단위의 집합을 분할해 $\boldsymbol{N}$자산 $\boldsymbol{p^{K,N}}$ 으로 분할하는 것을 고려
  - $|\boldsymbol{w_i}| = \frac{1}{K}\boldsymbol{p_i}$가 되도록 하는 절대 가중값 벡터 정의
    - $\sum|\boldsymbol{w_i}| = \mathbf{1}$
      - 전체-투자 조건은 모든 가중값이 양수나 음수일 수 있음
      - 총 개수 : $\mathbf{2^N}$

코드 21.1 *k* 객체를 *n* 칸으로 분할

```
from itertools import combinations_with_replacement
#------------------------------------------
def pigeonHole(k,n):
    # 비둘기 집 칸 문제(k 물체를 n 칸에 정리)
    for j in combinations_with_replacement(xrange(n),k):
        r=[0]*n
        for i in j:
            r[i]+=1
        yield r
```

코드 21.2 모든 분할과 연계된 모든 벡터의 집합 Ω

```
import numpy as np
from itertools import product
#------------------------------------------
def getAllWeights(k,n):
    #1) 분할 생성
    parts,w=pigeonHole(k,n),None
    #2) 분할 수행
    for part_ in parts:
        w_=np.array(part_)/float(k) # abs(가중값) 벡터
        for prod_ in product([-1,1],repeat=n): # 부호 추가
            w_signed_=(w_*prod_).reshape(-1,1)
            if w is None:w=w_signed_.copy()
            else:w=np.append(w,w_signed_,axis=1)
    return w
```

정수 최적화 기법

- 궤적 평가
  - 특징
    - 컨벡스 최적화에 의존하지 않고 전역적인 최적 궤적 선택
    - 공분산 행렬이 불량 조건이나 거래 비용 함수가 불연속적임
  - 단점
    - 극도로 많은 계산량 필요
      - NP-완결 V NP-하드 종류 문제
      - 디지털 컴퓨터에는 부적합
      - 양자 컴퓨팅에는 적합

```
import numpy as np
from itertools import product
#------------------------
def evalTCosts(w, params):
    # 특정 궤적의 거래 비용 계산
    tcost = np.zeros(w.shape[1])
    w_ = np.zeros(shape=w.shape[O])
    for i in range(tcost.shape[O]):
        c_ = params[i] ["c"]
        tcost[i] = (c_*abs(w[:,i] w_)**.5).sum()
        w_=w[:,i].copyO
    return tcost
#'-----------------------
def evalSR(params ,w,tcost):
    # 다수 호라이즌에 대한 SR 계산
    mean, cov = 0,0
    for h in range(w.shape[1]):
        params_=params[h]
        mean += np.dot(w[:, h].T, params_["mean"])[0] - tcost[h]
        cov  += np.dot(w[:, h].T, np.dot(params_["cov"], w[: ,h]))
    sr=mean/cov**.5
    return sr
def dynOptPort(params ,k=NOne):
    # 동적 최적 포트폴리오
    #1) 분할 생성
    if k is None: k=params [0] [' mean ' ].shape[O]
    n=params[O] [' mean ' ].shape[O]
    w_all ,sr=getAllweights(k ,n) ,None
    #2) 궤적을 카티션 곱으로 생성
    for prod_ in product(w_all.T,repeat=len(params)):
        w_=np.array(prod_).T # 상품을 궤적으로 연결
        tcost_=evalTCosts(w_ ,params)
        sr_=evalSR(params ,w_ ,tcost_) # 궤적 평가
        if sr is None or sr<sr_: # 더 나은 궤적은 저장
            sr ,w=sr_ ,w_.copy()
    return w
```

5 수치 예제

- **랜덤 행렬**
  - rndMatwithRank
    - 랭크가 알려진 가우스 값의 랜덤 행렬 반환
- **정적 해법**
  - statOptPortf
    - 지역 최적에서 생성된 궤적의 성능 계산
- **동적 해법**

```python
import numpy as np
# -------------------------------------------------
def rndMatwithRank(nsamples ,ncols , rank ,sigma=0 ,homNoise=True):
    # 주어진 랭크의 랜덤 행렬 x 생성
    rng=np.random.Randomstate()
    U,_,_=np.linalg.svd(rng.randn(nCols, nCols))
    x=np.dot(rng.randn(nsamples, rank), U[:,:rank].T)
    if homNoise:
        x+=sigma*rng.randn(nSamples , nCols) # 등분산 잡음 추가
    else:
        sigmas=sigma*(rng.rand(nCols)+.5)# 이분산 잡음 추가
        x+=rng.randn(nsamples,ncols)*sigmas
    return x
```

```python
def statOptPortf(cov,a):
    # Static optimal porftolio
    # Solution to the "unconstrained" portfolio optimization problem
    cov_inv=np.linalg.inv(cov)
    w=np.dot(cov_inv,a)
    w/=np.dot(np.dot(a.T,cov_inv),a) # np.dot(w.T,a)==1
    w/=abs(w).sum() # re-scale for full investment
    return w
    #———————————————————————————————
#2) Static optimal portfolios
w_stat=None
for params_ in params:
    w_=statOptPortf(cov=params_['cov'],a=params_['mean'])
    if w_stat is None:w_stat=w_.copy()
    else:w_stat=np.append(w_stat,w_,axis=1)
tcost_stat=evalTCosts(w_stat,params)
sr_stat=evalSR(params,w_stat,tcost_stat)
print('static SR:',sr_stat)
```

```python
w_dyn=dynOptPort(params)
tcost_dyn=evalTCosts(w_dyn,params)
sr_dyn=evalSR(params,w_dyn,tcost_dyn)
print 'dynamic SR:',sr_dyn
```

Thank you