Chapter 4
Sample Weights

01.

Motivation

1 동기

- **IID 문제**
  - 대부분의 통계 및 머신러닝 문헌은 **IID 가정에 근거**함
  - 그러나 대부분의 **금융 데이터는 IID하지 않음**
    - 이는 금융 도메인에서 머신러닝 알고리즘의 성능 저하 원인 중 하나
- **목적**
  - 본 장에서는 IID 문제를 해결하고자 함
    - **1) 중첩된 결과를 교정**하는 방법을 통한 샘플링
    - **2) 가중값** 설계
      - 모든 관측이 똑같이 중요한 것은 아니기 때문

1 동기

- **구조**
  - **문제 인식**
    - 금융 시장은 IID하지 않음 **(4.2장)**
  - **해결 방안**
    - **1) 중첩된 결과를 교정**하는 방법을 통한 샘플링 **(4.5장)**
      - **고유성**을 통해 교정
        - 라벨별 고유성 구함**(4.4장)**
          - **공존** 라벨**(4.3장)**
    - **2) 가중값 설계 (4.6-4.8장)**
      - **수익률 기여도**에 따른 가중값 설계(**Return Attribution**)**(4.6장)**
      - **시간**-감쇄(**Time Decay**)**(4.7장)**
      - **부류** 가중값(**Class Weights**)**(4.8장)**

02.

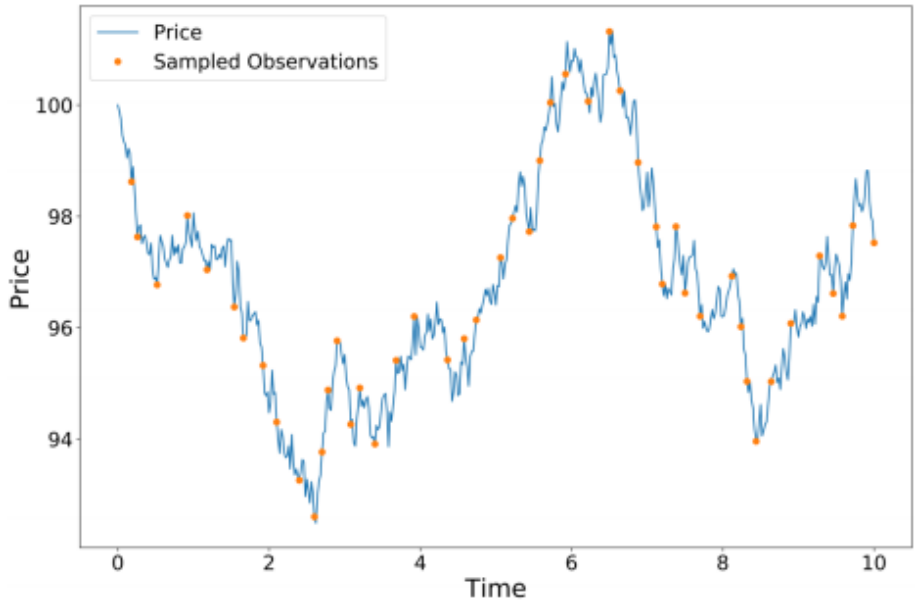
Overlapping Outcomes

2 중첩된 결과

- 구조
  - 문제 인식
    - 금융 시장은 IID하지 않음 **(4.2장)**
  - **해결 방안**
    - **1) 중첩된 결과를 교정**하는 방법을 통한 샘플링 **(4.5장)**
      - **고유성**을 통해 교정
        - 라벨별 고유성 구함**(4.4장)**
          - **공존** 라벨**(4.3장)**
    - **2) 가중값 설계 (4.6-4.8장)**
      - **수익률 기여도**에 따른 가중값 설계**(4.6장)**
      - **시간**-감쇄**(4.7장)**
      - **부류** 가중값**(4.8장)**

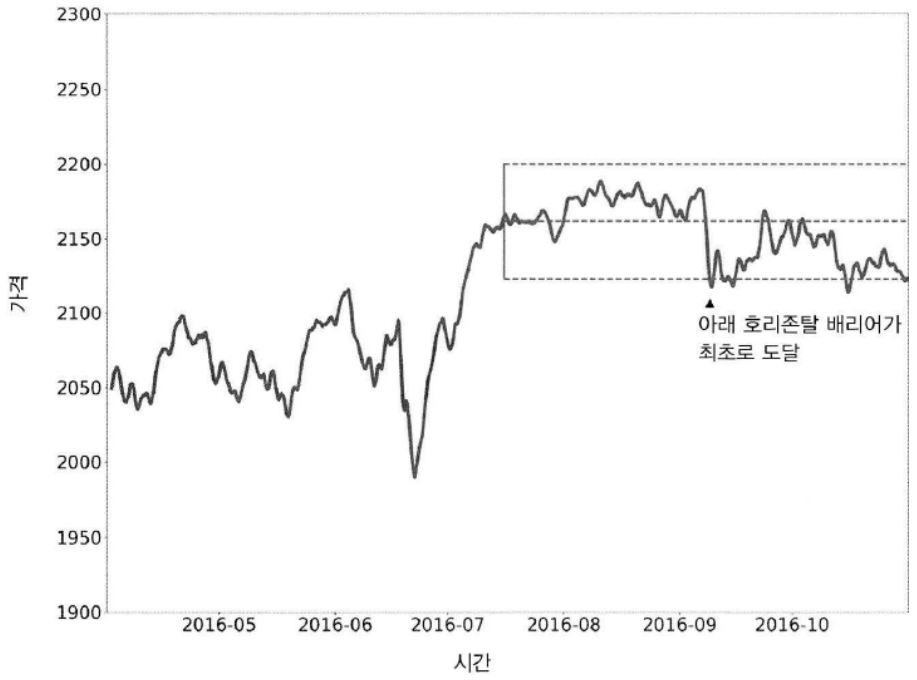
2 중첩된 결과

- 2장 복습
  - 2.5.1. 축소를 위한 표본 추출
    - 구조화된 데이터 세트로부터 특징 샘플링 이유
      - Fitting에 사용될 데이터 양을 줄이기 위함
  - 2.5.2. 이벤트-기반의 표본 추출
    - 베팅은 특정 상황 이후에 하는 것이 일반적
    - 머신러닝을 통해 이런 상황 하에서 표본 추출할 수 있음
- 목표
  - 비균질적 주기를 가진 피쳐 추출
  - 특정 상황의 결과를 예측하는 알고리즘 학습 위함



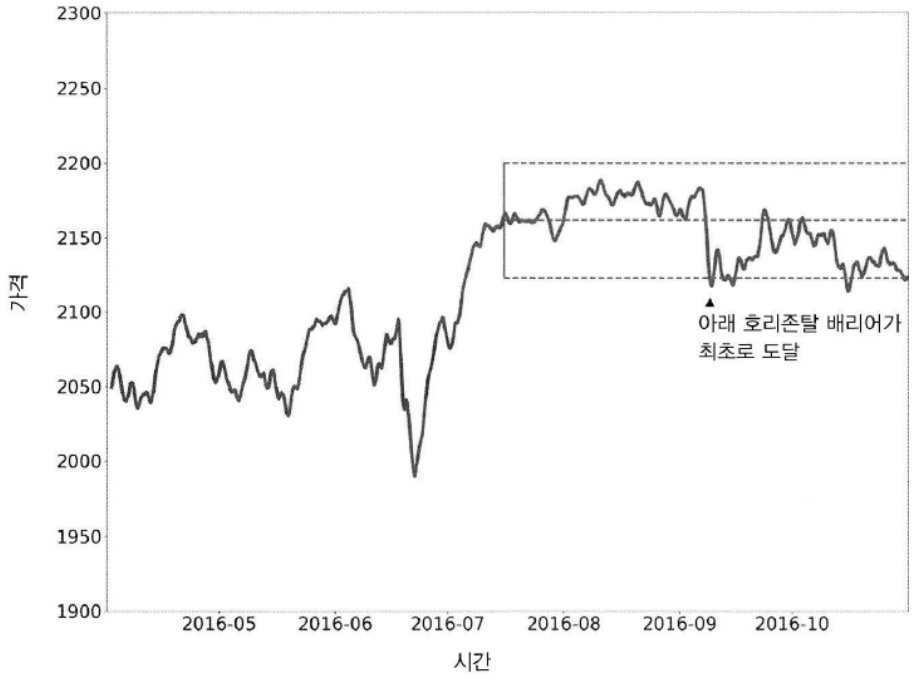
2 중첩된 결과

- 3장 복습
  - 대부분의 금융 ML 논문들에서 fixed-time horizon 방법에 따라 라벨링→X
    - $\begin{cases} -1 \text{ if } r_{t_{i,0},t_{i,0}+h} < -\tau \\ 0 \text{ if } |r_{t_{i,0},t_{i,0}+h}| \leq \tau \\ 1 \text{ if } r_{t_{i,0},t_{i,0}+h} > \tau \end{cases}$
      - X_i : set of features
      - τ : given threshold
      - $t_{i,0}$: index of the bar immediately after X_i takes place
      - $t_{i,0} + h$: index of h bars after $t_{i,0}$
      - $r_{t_{i,0},t_{i,0}+h}$: price return over a bar horizon h
      - 대부분의 논문에 h는 고정된 숫자(fixed-time horizon)



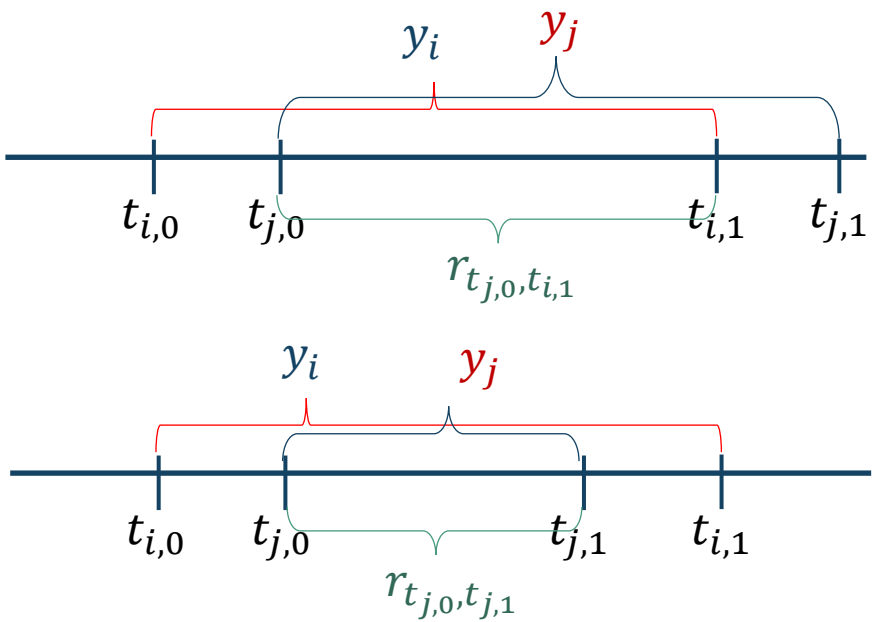
2 중첩된 결과

- **3장 복습**
  - 3장에서는 트리플-배리어 기법 제안
    - **구성**
      - **2개의 가로축(horizontal barrier)**
        - 추정 변동성에 대한 **동적** 상한 및 하한
      - **1개의 세로축(vertical barrier)**
        - 만기 한도
          - h는 만기 한계
    - **용어**
      - $t_{i,1}$: 처음으로 도달한 배리어의 시간
      - $r_{t_{i,0},t_{i,1}}$: 관측된 특징과 연계된 수익률



2 중첩된 결과

- 3장 복습
  - 관측값 X_i에 가격 바 y_i배분
    - y_i : $[t_{i,0}, t_{i,1}]$ 내에서 발생한 가격 바 함수
  - If
    - $t_{i,1} > t_{j,0}$ and $i < j$ → 공통 수익률 $r_{t_{j,0},\min\{t_{i,1},t_{j,1}\}}$발생 → 중첩 → IID 아님
    - 문제 발생



2 중첩된 결과

- 해결 방안
  - 1. 배팅 호라이즌 $t_{i,1} \leq t_{i+1,0}$으로 제한
    - 장점
      - 중첩 제거
    - 단점
      - h에 의해 표본 추출 빈도가 제한되는 조잡한 모형 발생
  - 2. 기간별 표본 추출
    - 표본 추출의 빈도와 호라이즌이 일치하는 문제
    - 첫 배리어에 도달하는 시간에 의해 표본 추출이 제한됨
- 결론
  - 중첩을 없애기 위해 출력의 호라이즌을 제한하면 안 됨
  - $t_{i,1} > t_{i+1,0}$의 경우는 피할 수 없음

사용된 호라이즌에 의해 제한되므로 조잡한 모델이 만들어질 것이다. 한편으로는 한달 동안 지속된 결과를 살펴볼 수도 있는데, 이 경우, 특징들은 월 주기로 표본 추출돼야 한다. 다른 한편으로 표본 추출 빈도를 일 단위로 증가시키면 출력의 호라이즌을 1일로 축소했어야 할 것이다. 게다가 트리플-배리어 기법과 같은 경로-의존 레이블 기술을 사용하려면 표본 추출 주기는 처음 배리어의 도달 시간에 종속될 것이다. 어떤 경우든 중첩을 없애기 위해 출력의 호라이즌을 제한

determine the outcome. On one hand, if we wished to investigate outcomes that lasted a month, features would have to be sampled with a frequency up to monthly. On the other hand, if we increased the sampling frequency to let's say daily, we would be forced to reduce the outcome's horizon to one day. Furthermore, if we wished to apply a path-dependent labeling technique, like the triple-barrier method, the sampling frequency would be subordinated to the first barrier's touch. No matter what

03.

Number of Concurrent Labels

3 공존 레이블의 개수

- **구조**
  - **문제 인식**
    - 금융 시장은 IID하지 않음 **(4.2장)**
  - **해결 방안**
    - **1) 중첩된 결과를 교정**하는 방법을 통한 샘플링 **(4.5장)**
      - **고유성**을 통해 교정
        - 라벨별 고유성 구함**(4.4장)**
          - **공존** 라벨**(4.3장)**
    - **2) 가중값 설계 (4.6-4.8장)**
      - **수익률 기여도**에 따른 가중값 설계**(4.6장)**
      - **시간**-감쇄**(4.7장)**
      - **부류** 가중값**(4.8장)**

3 공존 레이블의 개수

- **정의**
  - 공존하다
    - y_i와 y_j가 하나 이상의 공통 수익률 $r_{t-1,t} = \frac{p_t}{p_{t-1}} - 1$을 가질 경우
- **목표**
  - 레이블의 개수 계산
    - f($r_{t-1,t}$)
- **방법**
  - 1)각 시간 t에서 이진 행렬 $1_{t,i}$구성
    - $1_{t,i} \in \{0,1\}$
      - $1\ if\ [t_{i,0}, t_{i,1}]$과 $[t-1, t]$가 중첩
        - $[t_{i,0}, t_{i,1}]$은 t1 객체에 의해 정의
          - $t1$: 최초로 배리어에 도달했을 때의 타임 스탬프 값(**3.5장**)
      - $0\ O/W$
  - 2) 시간 t에서 공존하는 레이블 개수 계산 $c_t = \sum_{i=1}^{I} 1_{t,i}$

4.1. 레이블의 고유성 계산

```python
def mpNumCoEvents(closeIdx,t1,molecule):
    """
    바별로 공존하는 이벤트 개수 계산
    +molecule[0]은 가중값이 계산될 첫 이벤트 날짜
    +molecule[-1]은 가중값이 계산될 마지막 이벤트 날짜다.
    t1[moldecule].max() 이전에 발생하는 모든 이벤트는 개수에 영향을 미침
    """
    #1) [molecule[0],molecule[-1]] 구간에서 이벤트 탐색
    t1=t1.fillna(closeIdx[-1]) # 드러난 이벤트는 다른 가중값에 영향 미침
    t1=t1[t1>=molecule[0]] # molecule[0] 마지막이나 이후에 발생하는 이벤트
    t1=t1.loc[:t1[molecule].max()] # t1[molecule].max() 이전이나 시작시에 발생하는 이벤트

    #2) 이벤트 바 개수 확인
    iloc=closeIdx.searchsorted(np.array([t1.index[0],t1.max()]))
    count=pd.Series(0,index=closeIdx[iloc[0]:iloc[1]+1])
    for tIn,tOut in t1.iteritems():count.loc[tIn:tOut]+=1.
    return count.loc[molecule[0]:t1[molecule].max()]
```

04.

레이블의 평균 고유성

4 레이블의 평균 고유성

- **구조**
  - **문제 인식**
    - 금융 시장은 IID하지 않음 **(4.2장)**
  - **해결 방안**
    - **1) 중첩된 결과를 교정**하는 방법을 통한 샘플링 **(4.5장)**
      - **고유성**을 통해 교정
        - 라벨별 고유성 구함**(4.4장)**
          - **공존** 라벨**(4.3장)**
    - **2) 가중값 설계 (4.6-4.8장)**
      - **수익률 기여도**에 따른 가중값 설계**(4.6장)**
      - **시간**-감쇄**(4.7장)**
      - **부류** 가중값(**4.8장)**

4 레이블의 평균 고유성

- **목적**
  - 레이블의 고유성(비중첩)을 전체 주기 동안의 평균 고유도를 통해 계산
- **방법**
  - 1) t에서 레이블 i의 고유성은 $u_{t,i} = 1_{t,i}c_t^{-1}$임
  - 2) 레이블 i의 평균 고유성은 레이블의 전체 주기 동안 **평균** $\boldsymbol{u_{t,i}}$임
    - $\bar{u}_i = \left(\sum_{t=1}^{T} u_{t,i}\right)\left(\sum_{t=1}^{T} 1_{t,i}\right)^{-1}$
      - 평균 고유성을 고려할 경우 미래 데이터 사용에 대한 문제제기 가능
        - 다만 train set에서만 진행하기 때문에 허용 범위
        - $\bar{u}_i$ 가 레이블 예측에 사용되지 않기 때문에 정보 누수 없음
- **결과**
  - 각 관측값의 특징마다 비중첩 결과에 따라 **유일성 점수를 0과 1 사이에서 부여** 가능
- **그림**
  - T1에서 도출된 고유성 값을 히스토그램으로 나타낸 것

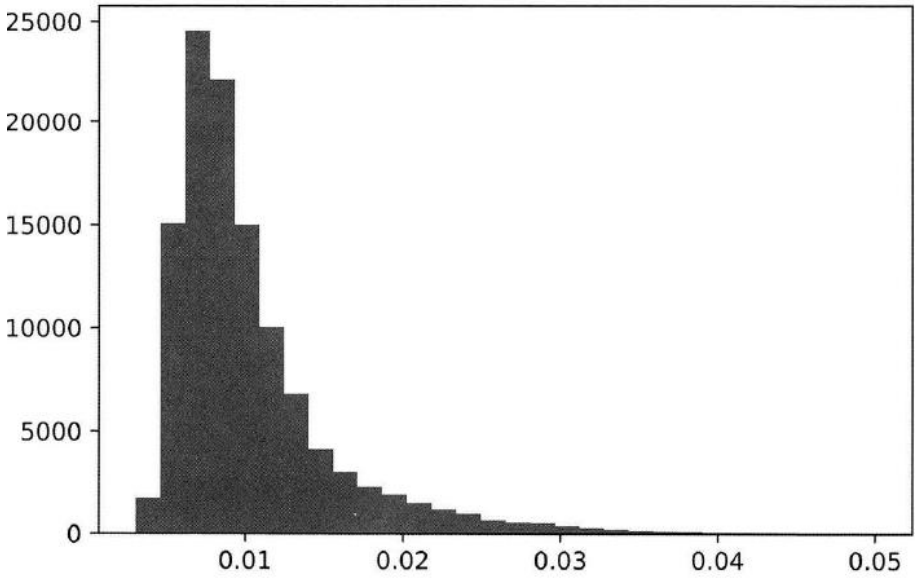


그림 4-1 고유성 값을 히스토그램으로 나타낸 것

4.2. 레이블의 평균 고유성 계산

```
def mpSampleTW(t1, numCoEvents, molecule):
    ## 이벤트 생명 주기 동안의 평균 고유성 도출
    wght = pd.Series(index=molecule)
    for tIn, tOut in t1.loc[wght.index].iteritems():
        wght.loc[tIn] = (1.0 / numCoEvents.loc[tIn:tOut]).mean()
    return wght
```

4 레이블의 평균 고유성

- **종합**
  - **1. 각 관측값 t에 대하여 이진분류행렬 $1_{t,i}$를 구성**
    - 해당 행렬은 결과가 return r_{t-1}로부터 span하는지 여부 표현
  - **2. t 시점에서 공존하는 라벨 확인**
    - $c_t = \sum_{i=1}^{I} 1_{t,i}$
  - **3. t 시점에 고유성 $u_{t,i} = 1_{t,i} c_t^{-1}$**
  - **4. 라벨 i의 평균 고유성은 $u_{t,i}$의 평균값**
    - $\bar{u}_i = \left(\sum_{t=1}^{T} u_{t,i}\right)\left(\sum_{t=1}^{T} 1_{t,i}\right)^{-1}$
  - **5. Sample weights : attributed returns의 합**
    - $[t_{i,0}, t_{i,1}]$, $\tilde{w}_i = \left|\sum \frac{r_{t-1,t}}{c_t}\right|$
    - $w_i = \tilde{w}_i I\left(\sum \tilde{w}_j\right)^{-1}$
    - Weight an observation as a function of the absolute log returns that can be attributed uniquely

05.

배깅 분류기와 고유성

5 배깅 분류기와 고유성

- 구조
  - 문제 인식
    - 금융 시장은 IID하지 않음 **(4.2장)**
  - 해결 방안
    - 1) **중첩된 결과를 교정**하는 방법을 통한 샘플링 **(4.5장)**
      - **고유성**을 통해 교정
        - 라벨별 고유성 구함**(4.4장)**
          - **공존** 라벨**(4.3장)**
    - 2) **가중값 설계 (4.6-4.8장)**
      - **수익률 기여도**에 따른 가중값 설계**(4.6장)**
      - **시간**-감쇄**(4.7장)**
      - **부류** 가중값**(4.8장)**

5 배깅 분류기와 고유성

- **배경**
  - **목적**
    - I 아이템 1개를 가진 집합에서 복원을 동반한 I번 추출시 특정 아이템 i가 선택되지 않을 확률
    - Bagging에 연결(Bootstrap aggregating)
  - **수식**
    - $(1 - I^{-1})^I$
      - $\left(\frac{I-1}{I}\right)^I$
    - $\lim_{I\to\infty}(1 - I^{-1})^I = e^{-1}$
    - 기댓값 : $(1 - e^{-1}) \approx \frac{2}{3}$
- **가정**
  - 비중첩 결과의 최대 개수 : $K \leq I$
    - 특정 i가 선택되지 않을 확률 $(1 - K^{-1})^I$
    - $(1 - I^{-1})^{K\frac{K}{I}} \approx e^{-\frac{I}{K}}$
      - 특정 아이템이 추출될 횟수는 평균 $\frac{I}{K} \geq 1$인 푸아송 분포
        - $f(n;\lambda) = \frac{\lambda^n e^{-\lambda}}{n!}$
      - 부정확한 IID 가정시 과표본 야기

5 배깅 분류기와 고유성

- **문제**
  - $I^{-1}\sum_{i=1}^{I}\overline{u_i} \ll 1$ 일 경우
    - 복원을 동반한 표본 추출(부트스트랩)시 IB 관측값은
      - 1) 서로 중복
      - 2) OOB 관측값과 유사
- **사족**
  - 추출의 비효율성은 부투스트랩 비효율적으로 만듬(6장)
    - Ex) 랜덤 포레스트
      - 모든 트리가 근본적으로 서로 유사한, 과적합된 단일 결정 트리들의 복사품
      - OOB 예제를 IB와 비슷하게 만들어 OOB 정확도 과도하게 부풀림(7장)

5 배깅 분류기와 고유성

- **문제**
  - $I^{-1}\sum_{i=1}^{I}\overline{u_i} \ll 1$ 일 경우
    - 복원을 동반한 표본 추출(부트스트랩)시 IB 관측값은
      - 1) 서로 중복
      - 2) OOB 관측값과 유사
- **해결**
  - **1) 중첩된 출력을 부트스트랩 전에 삭제하는 것 → X**
    - **한계**
      - 대부분의 중첩은 부분적 중첩 → 무분별한 관측값 삭제시 정보 손실
  - **2) 평균 고유성을 활용해 불필요한 정보를 가진 결과가 과도한 영향 미치는 것 감소→ O**
    - 관측값의 일부를 표본추출 V 관측값의 몇 배수 표본추출
    - **구현**
      - sklearn.ensemble.BaggingClassifier의 max_samples parameters
        - Max_smaples=out[“tW”].mean()으로 설정
        - IB 관측값이 이 고유성보다 더 빈번하게 추출되면 안됨

5 배깅 분류기와 고유성

- **해결**
  - **3) 순차적 부트스트랩**
    - **목적**
      - 중복 통제 확률 변화에 따라 추출
    - **정의**
      - 중복이 허용된 추출 시퀀스 φ
    - **방법**
      - 1) (Given)관측값 X_i는 유니폼 분포 $i \sim U[1, I]$에서 추출
        - i 추출 확률 : $\delta_i^{(1)} = I^{-1}$
      - 2) 두 번째 추출의 경우 높은 중첩 결과를 가진 X_j가 추출될 확률 낮춤
        - 현 상황 : $\varphi^{(i)} = \{i\}$
        - j의 고유성 : $u_{t,j}^{(2)} = 1_{t,j}\left(1 + \sum_{k \in \varphi^{(1)}} 1_{t,k}\right)^{-1}$
        - J의고유성 평균 : $u_{t,j}^{(2)}$의 평균 = $\overline{\boldsymbol{u_j}}^{(\mathbf{2})} = \left(\sum_{t=1}^{T} u_{t,j}\right)\left(\sum_{t=1}^{T} 1_{t,j}\right)^{-1} \rightarrow \left\{\delta_j^{(2)}\right\}$
          - $\delta_j^{(2)} = \bar{u}_j^{(2)}\left(\sum_{k=1}^{I} \overline{\boldsymbol{u_k}}^{(\mathbf{2})}\right)^{-1}$, $\sum \delta_j^{(2)} = 1$
    - **결과**
      - 표준 부트스트랩보다 IID에 훨씬 더 가까움
        - $I^{-1}\sum \bar{u}_i$ 증가를 통한 검증

5 배깅 분류기와 고유성

- 해결
  - 3) 순차적 부트스트랩 구현
    - getIndMatrix
      - input
        - barIx : 바의 인덱스
        - t1 : pandas series
          - 관측된 시각을 포함하고 있는 인덱스와 함께 레이블이 결정된 시각을 담고 있는 배열
      - output
        - 어떤 바가 관측값의 레이블에 영향을 미치는지 알려주는 이진행렬
    - getAvgUniqueness
      - input
        - getIndMatrix에 의해 구성된 지표행렬
      - output
        - 각 특징 관측값의 고유성 평균 반환

4.3. 지표 행렬 구축

```
def getIndMatrix(barIx, t1):
    ## 지표 행렬 구하기
    indM = pd.DataFrame(0, index=barIx, columns=range(t1.shape[0]))
    for i, (t0, t1) in enumerate(t1.iteritems()):
        indM.loc[t0:t1, i] = 1.0
    return indM
```

4.4 평균 고유성 계산

```
def getAvgUniqueness(indM):
    # 지표 행렬로부터의 평균 고유성
    c = indM.sum(axis=1)  # 공존
    u = indM.div(c, axis=0)  # 공유성
    avgU = u[u > 0].mean()  # 평균 고유성
    return avgU
```

5 배깅 분류기와 고유성

- 해결
  - 3) 순차적 부트스트랩 구현
    - seqBootstrap
      - input
        - indM : 지표 행렬
        - sLength : 선택적 표본 추출 길이
          - default : indM에 있는 행의 개수
      - output
        - 순차적 부트스트랩에 의해 표본 추출된 특징의 인덱스

4.5. 순차적 부트스트랩으로부터 수익률 표본 추출

```
def seqBootstrap(indM,sLength = None):
    # Generate a sample via sequential bootstrap
    if sLength is None:sLength = indM.shape[1]
    phi = []
    while len(phi) < sLength:
        avgU = pd.Series()
        for i in indM:
            indM_ = indM[phi + [i]] # reduce indM
            avgU.loc[i] = getAvgUniqueness(indM_).iloc[-1]
        prob = avgU/avgU.sum() # draw prob
        phi += [np.random.choice(indM.columns, p=prob)]
    return phi
```

5 배깅 분류기와 고유성

- 해결
  - 3) 순차적 부트스트랩 수치적 예제
    - 정의
      - 레이블 집합 y_i
        - y_1은 수익률 $r_{0,3}$의 함수
        - y_2은 수익률 $r_{2,4}$의 함수
        - y_3은 수익률 $r_{4,6}$의 함수
      - 지표 행렬 $\mathbf{1}_{t,i}$
        - $\{1_{t,i}\} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$

5 배깅 분류기와 고유성

- **해결**
  - **3) 순차적 부트스트랩 수치적 예제**
    - 절차
      - 시작
        - $\varphi^{(0)} = \emptyset$ 과 확률 $\delta_i = \frac{1}{3}, \forall i = 1,2,3$ 유니크 분포
      - 첫번째 랜덤 숫자 선택
        - {1, 2, 3}에서 랜덤으로 숫자 선택 → 2라고 가정
      - 두번째 랜덤 숫자 선택
        - 확률 조정
          - Given : $\varphi^{(1)} = \{2\}$
          - 첫 특징의 평균 고유성 : $\bar{u}_1^{(2)} = \left(\mathbf{1} + \mathbf{1} + \frac{\mathbf{1}}{\mathbf{2}}\right)\frac{\mathbf{1}}{\mathbf{3}} = \frac{\mathbf{5}}{\mathbf{6}} < \mathbf{1}$
          - 두 번째 추출 $\bar{u}_2^{(2)} = \left(\frac{\mathbf{1}}{\mathbf{2}} + \frac{\mathbf{1}}{\mathbf{2}}\right)\frac{\mathbf{1}}{\mathbf{2}} = \frac{\mathbf{1}}{\mathbf{2}} < \mathbf{1}$
            - 가장 낮은 확률은 첫 번째 추출에서 나왔던 특징에 배분
            - $\varphi^{(1)}$ 이외에 두 가지 가능한 추출 중에서 더 높은 확률은 $\delta_3^{(2)}$에 배분, $\varphi^{(1)}$과 중첩이 없는 레이블

$$\{1_{t,i}\} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

5 배깅 분류기와 고유성

- **해결**
  - **3) 순차적 부트스트랩 몬테카를로 실험**
    - **목적**
      - 실험적 기법을 통해 부트스트랩 알고리즘의 효율성 평가
    - 함수
      - getRndT1
        - 다수의 관측값 numObs로부터 랜덤 t1 series 생성
        - 각 관측값은 유니폼 분포에서 번호 추출
        - 경곗값은 0과 numBars -> 바의 개수
      - auxMC
        - 랜덤 t1 series로부터 내재된 지표 행렬 indM을 도출
        - **절차**
          - 표준 부트스트랩으로투버 평균 고유성 도출
          - 순차적 부트스트랩 알고리즘을 적용해 평균 고유성 도출

4.7. 랜덤 T1 Series 생성

```
def getRndT1(numObs, numBars, maxH):
    # Serie t1 aleatoria
    t1 = pd.Series()
    for i in range(numObs):
        ix = np.random.randint(0, numBars)
        val = ix + np.random.randint(1, maxH)
        t1.loc[ix] = val
    return t1.sort_index()
```

4.8. 표준과 순차적 부트스트랩에서의 고유성

```
def auxMC(numObs,numBars,maxH):
    # Parallelized auxiliary function
    t1 = getRndT1(numObs,numBars,maxH)
    barIx = range(t1.max()+1)
    indM = getIndMatrix(barIx,t1)
    phi = np.random.choice(indM.columns,size = indM.shape[1])
    stdU = getAvgUniqueness(indM[phi]).mean()
    phi = seqBootstrap(indM)
    seqU = getAvgUniqueness(indM[phi]).mean()
    return { 'stdU':stdU,'seqU':seqU }
```

5 배깅 분류기와 고유성

- 해결
  - 3) 순차적 부트스트랩 몬테카를로 실험
    - 결과
      - 표준 방법의 고유성 평균 중앙값은 0.6
      - 순차적 방법의 고유성 평균 중앙값은 0.7
      - 단측 표본 t-검정을 평균값의 차이에 대해 수행하면 0에 가까운 값 발생
        - 순차적 부트스트랩 기법이 표준 부트스트랩 기법보다 고유성 기댓값이 적절한 신뢰 구간에 대해

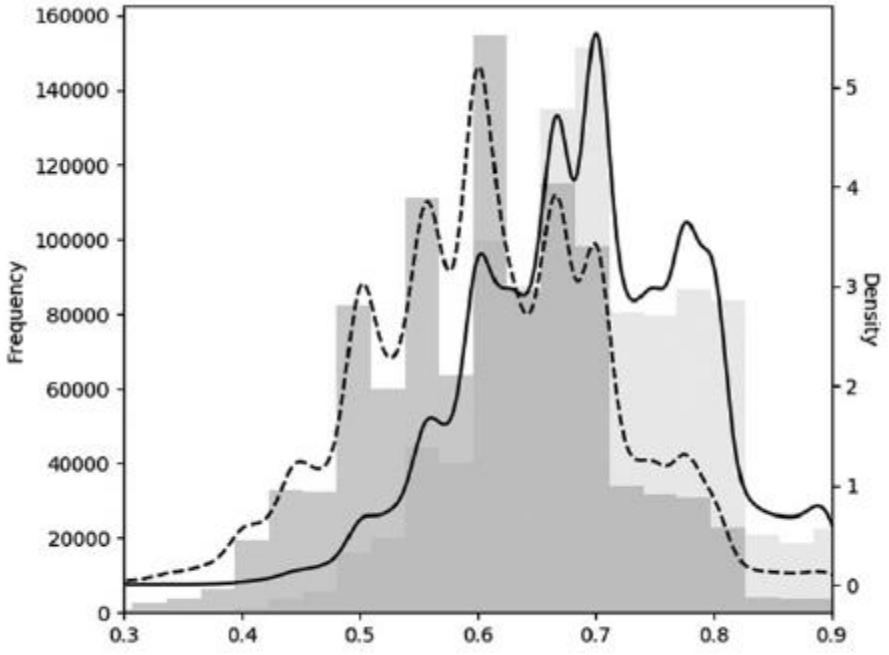


FIGURE 4.2 Monte Carlo experiment of standard vs. sequential bootstraps

06.

수익률 기여도

6 수익률 기여도

- **구조**
  - **문제 인식**
    - 금융 시장은 IID하지 않음 **(4.2장)**
  - **해결 방안**
    - **1) 중첩된 결과를 교정**하는 방법을 통한 샘플링 **(4.5장)**
      - **고유성**을 통해 교정
        - 라벨별 고유성 구함**(4.4장)**
          - **공존** 라벨**(4.3장)**
    - **2) 가중값 설계 (4.6-4.8장)**
      - **수익률 기여도**에 따른 가중값 설계**(4.6장)**
      - **시간**-감쇄**(4.7장)**
      - **부류** 가중값(**4.8장)**

6 수익률 기여도

- **배경**
  - 고도로 중첩되는 결과는 비중첩 출력에 비해불균형한 가중값
  - 절대수익률이 클수록 더 큰 가중치를 둬야 한다
  - → **고유성**과 **절대 수익률**을 고려하는 함수를 사용해 관측값에 가중을 둬야 한다.
- **레이블**
  - **수익률 부호**
    - **표본 가중값**
      - 라벨 : 이벤트의 생명주기 $t_{i,0}, t_{i,1}$동안의 기여 수익율의 합
      - **수식**
        - $\tilde{w}_i = |\frac{\sum r_{t-1,t}}{c_t}|, w_i = \tilde{w}_i I(\sum \tilde{w}_j)^{-1}$
        - $\sum w_i = I$
      - 관측값의 가중값을 관측값에 고유하게 영향을 미칠 수 있는 절대 로그 수익률의 함수로 나타냄
      - 다만 중립적인 경우가 존재할 경우 작동하지 않음
        - 중립적 경우 삭제해야 함

4.10 절대 수익률 기여도에 의한 표본 가중값 결정

```
def mpSampleW(t1, numCoEvents, close, molecule):
    # 수익률 기여에 따른 샘플 가중값 도출
    ret = np.log(close).diff() # 로그 리턴이브로 가산적
    wght = pd.Series(index=molecule)
    for tIn, tOut in t1.loc[wght.index].iteritems():
        wght.loc[tIn] = (ret.loc[tIn:tOut] / numCoEvents.loc[tIn:tOut]).sum()
    return wght.abs()
```

07.

시간-감쇄

7 시간-감쇄

- **구조**
  - **문제 인식**
    - 금융 시장은 IID하지 않음 **(4.2장)**
  - **해결 방안**
    - **1) 중첩된 결과를 교정**하는 방법을 통한 샘플링 **(4.5장)**
      - **고유성**을 통해 교정
        - 라벨별 고유성 구함**(4.4장)**
          - **공존** 라벨**(4.3장)**
    - **2) 가중값 설계 (4.6-4.8장)**
      - **수익률 기여도**에 따른 가중값 설계**(4.6장)**
      - **시간**-감쇄**(4.7장)**
      - **부류** 가중값**(4.8장)**

7 시간-감쇄

- **배경**
  - **시장은 적응적 시스템**
    - 시장이 발달할수록 과거의 예제가 새로운 것보다 더 연관성이 떨어짐
    - 일반적으로 새로운 관측값을 얻게 되면 표본 가중값 감쇄
- **전개**
  - $d[x] \geq 0,\ \forall x \in [0, \sum \bar{u}_i]$
    - $d[\sum \bar{u}_i] = 1$
    - $c \in (-1,1]$
      - $c \in [0,1]$
      - $d[1] = c$
        - 선형 감쇄
    - $c \in (-1, 0)$
      - $d[-c\sum \bar{u}_i] = 0$
      - $[-c\sum \bar{u}_i, \sum \bar{u}_i]$ 사이에서 선형 감쇄, $d[x] = 0\ \forall x \leq -c \sum \bar{u}_i$
      - $d = \max\{0, a + bx\}$

7 시간-감쇄

- **배경**
  - **시장은 적응적 시스템**
    - $d = a + b\sum\bar{u} = 1 \rightarrow a = 1 - b\sum\bar{u}_i$
    - c에 따라
      - $(a)\ d = a + b0 = c \rightarrow b = (1-c)(\sum\bar{u}_i)^{-1}$
      - $(b) d = a - bc\sum\bar{u}_i = 0 \rightarrow b = [(c+1)\sum\bar{u}_i]^{-1}$
  - **감쇄**
    - 누적 고유성 : $x \in [0, \sum\bar{u}_i]$
    - $\boldsymbol{c = 1}$
    - $\boldsymbol{0 < c < 1}$
      - 시간에 대해 가중값 감쇄가 선형이라는 의미
      - 가중값은 오래된 정도에 상관없이 양수의 가중값 부여
    - $\boldsymbol{c = 0}$
      - 시간이 갈수록 가중값이 선형으로 0에 수렴한다는 의미
    - $\boldsymbol{c < 0}$
      - 관측값 중 가장 오래된 부분인 cT는 0의 가중값을 받는다.

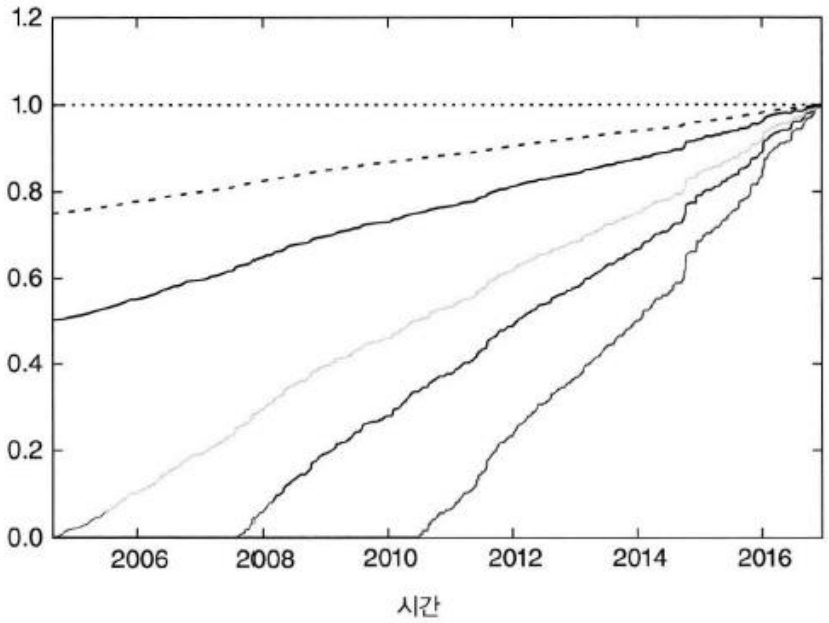


그림 4-3 구간-선형 시간-감쇄 요인

08.

부류 가중값

8 부류 가중값

- 구조
  - 문제 인식
    - 금융 시장은 IID하지 않음 **(4.2장)**
  - 해결 방안
    - 1) **중첩된 결과를 교정**하는 방법을 통한 샘플링 **(4.5장)**
      - **고유성**을 통해 교정
        - 라벨별 고유성 구함**(4.4장)**
          - **공존** 라벨**(4.3장)**
    - 2) **가중값 설계 (4.6-4.8장)**
      - **수익률 기여도**에 따른 가중값 설계**(4.6장)**
      - **시간**-감쇄**(4.7장)**
      - **부류** 가중값**(4.8장)**

8 부류 가중값

- **정의**
  - **얼마 없는 레이블에 가중값을 교정**
  - **중요한 부류의 빈도수가 낮을 경우 특히 중요함**
    - 이런 드문 레이블에 가중값을 더 높게 주지 않으면 흔한 레이블에 대해서만 정확도 극대화
- **방법**
  - Sklearn에 부류 가중값 파라미터
    - Class_weight[j]
- **전개**
  - 금융 응용에 있어서 분류기 알고리즘의 표준 레이블 : $\{-1, 1\}$
    - 0인 경우에는 중립 임계값인 0.5보다 약간 높거나 낮은 확률로 예측
    - Class_weight="balanced"
    - 관측값에 가중값을 재부여해 모든 부류가 동일한 빈도로 나타나도록 해줌
  - 배깅 분류기
    - Class_weight = "balanced_subsample"
    - Class_weight="balanced"가 전체 데이터 세트가 아닌 부트스트랩 표본에 적용


Thank you
YONSEI UNIVERSITY