|  |
| --- |
| * 삽입된 문서의 내용을 요약 정리하자면 LSTM과 transfer learning(incremental learning)를 이용한 모델들의 성과는 기대 이하였다. 많은 항목들의 data를 LSTM으로 학습하여 예측할 때, long-term prediction(20일 이상)은 그런대로 결과가 나왔지만(최고 70%) 1일 또는 2일 후 예측 결과는 실망이라 할 것이다.   그리하여 다른 시도가 필요했다. 단기 예측은 많은 항목들이 필요한 것이 아니지 않을까 하는 생각에 오로지 지수 정보와 지수의 파생 데이터(5, 20, 60, 120, 240일 최저, 최고, 평균가, 수익률), 거래량 정보와 파생 데이터만 포함해서 시도해 보기로 하였다. 그래서 transfer-LSTM-DNN이라는 이름으로 시도하였다.  지수와 거래량 정보 및 파생 정보를 포함하여 총 44개의 항목들로 DNN으로 학습한 결과 1일 예측 1년간 최고 67% 적중 율을 보였고 20일 예측은 최고 80%를 보였다.  학습 방법도 변경하였다. 주어진 train data를 전부 사용하지 않고 random으로 일부만 사용하고 일부는 validation으로 사용하였다. 따라서 validation 결과가 좋게 나올 때까지 계속 반복하여 (100~2500번) 최고의 validation 결과를 보이는 모델을 선택하여 test를 진행한 결과이다.  그리하여 실제 거래에 적용하기 위해서는 데이터 측면에서나 모델 구조 측면에서도 부담스러운 LSTM을 포기하고 가격 + 거래량 + 파생 정보만을 이용하는 DNN 모델을 좀 더 개발하기로 하였다.   * 여기서 사용된 학습 알고리즘의 전략을 상세히 설명해 보자.  우선 주어진 전체 데이터를 다음과 같이 분리한다.   Train test |

Train data 중 random으로 50%만 사용한다.

Real train data

이 중에서 20%를 validation data로 사용하기 위해 random으로 선택한다.  
  
  
 validation data  
  
이렇게 데이터를 구성한 다음 학습에는 real train data를 사용하여 학습하고 학습된 모델을 validation data에서 accuracy를 validate한다. 이러한 과정을 계속 반복하여 validation 결과가 최상인 모델을 선택한다.

이렇게 하는 이유가 무엇인가?

예측과 관련된 serial data에는 많은 noise가 포함되어 있다. 똑 같이 주어진 상황(data)에서도 종종 예기치 못한 변수에 의해 결과가 다르게 나올 수 있다. 이런 현상은 자연스러운 것이지만 데이터 분석 차원에서는 noise로 간주될 수 밖에 없다. 마치 양자 중첩처럼 같이 공존할 수 없는 상황이 동시에 발생하는 것처럼 데이터가 형성되어 버린다. 학습 과정은 완전한 function approach이다. 양자 중첩 같은 현상을 신경망 알고리즘이 해결할 수는 없을 것이다. 따라서 우리가 할 수 있는 최선의 방법은 이렇게 중첩된 데이터 중 어느 하나를 소거하는 것이다. Input data가 완전히 똑 같지 않더라도 90%이상의 유사성을 갖는다 하더라도 같은 target을 갖는다면 이 또한 중첩 현상과 같은 효력을 발휘할 것이다. 중첩된 데이터를 분리하는 깔끔한 알고리즘이 있으면 좋겠지만 (나는 아직 발견하지 못했다.) 그렇지 못한 차 선택으로 위와 같은 방법을 사용하는 것이다. 우연히 선택된 데이터로 학습된 모델이 우수한 validation의 결과를 보인다면 noise가 제거된 결과라고 믿는 것이다.

* 지금 까지는 예측 시스템과 관련된 내용이었고 실제로 거래에 사용하기 위해서는 무엇을 더 해야 하느냐를 고민해야 할 것이다. 실제 거래에서 발생되는 문제점을 열거해 본다.  
    
  - 수동 거래의 문제점: 거래와 예측 시스템이 단절된 상황에서는 제대로 된 거래를 진행하기가 어렵다. 데이터를 다운로드 받는 과정. 학습하고 예측하는 과정. 예측 결과에 의해 거래하는 과정. 이러한 과정의 단계에서 여러 가지 사정으로 인해 단절이 일어날 수 있다. 따라서 이러한 과정을 물 흐르듯이 매끄럽게 자동으로 처리하는 시스템이 필요할 것이다.  
    
  - risk 관리의 필요성: 설사 하루씩 예측하고 거래하는 시스템일지라도 선물의 속성상 하루 동안에도 마진 콜을 당할 수 있다. 즉 다 잃을 수 있다는 것이다. 손실 율 100%가 가능하다. 물론 손절을 정해 놓고 거래를 하면 되지만 적절한 손절 한계치를 파악하기 어렵고 빈번한 손절의 발생은 전체 수익을 손상시키는 악재로 작용할 것이다.  
    
  일 단위로 거래하는 것 보다 시간 단위(60분봉)로 거래하는 것을 생각하였다. 시간 단위로 거래할 때는 매 시간 다음 시간대를 예측하는 것은 예측의 정확도 내지 신빙성의 의혹이 들었다. 그래서, 현재 시간대가 5 ~ 40시간 내에 고점인지, 저점인지를 판단하는 것으로 target을 설정한 고저점 모델을 만들어 target type에 따라 다양한 모델을 생성하였다. 몇 가지 예를 들자면.

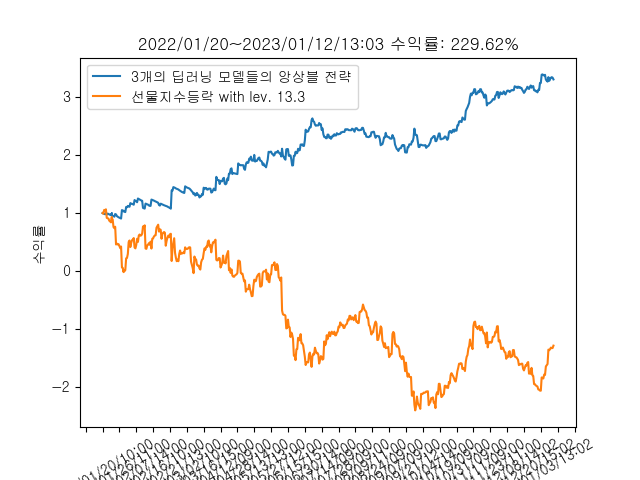
Target 설정 방식

1: 현재 종가가 n시간 내에 (종가, 고가) 대비 고점  
2: 현재 종가가 n시간 내에 (종가, 저가) 대비 저점  
0: 고점도 아니고 저점도 아닌 상태  
  
모델 type 설정 예)  
  
5C – 현재 시간대의 종가를 향후 5시간내에 종가 기준으로 고 저점을 판단하여 target 설정  
  
10HL - 현재 시간대의 종가를 향후 10시간내에 (고가, 저가) 기준으로 고 저점을 판단하여 target 설정  
  
40P – 현재 시간대의 종가와 40시간 뒤의 종가를 단순 비교하여 target 설정

* 키움에서 제공하는 API를 이용하여 60분봉 마다 시그널을 받아 자동 거래하는 시스템 3개를 개발하였다. 참고 자료는 ‘파이썬으로 배우는 알고리즘트레이딩, <http://wikidocs.net/book/110>’을 참고하였다.
* 알고리즘트레이딩2: 앞서 설명한 고 저점 기법을 사용하여 개발된 60분봉 자동 거래 시스템이다. 매시간 키움 API로부터 데이터를 자동으로 다운 받아 (가격과 거래량 정보) 1 ~ 240봉 의 최고, 최저, 평균, 수익률, 변화율 등의 파생 정보들을 덧붙여 전처리하고 미리 학습한 모델들의 앙상블 예측에 따라 자동 거래 한다.  
     
  개별 프로그램들을 열거하고 설명하자면.  
  \* make\_model: 모델을 학습하고 test한다.  
  \* ensemble\_test: n개의 모델들에 대한 앙상블 예측 및 수익을 test한다.  
  \* ensemble: 자동 거래시 사용되는 앙상블 예측 프로그램  
  \* profit: 개별 모델 또는 앙상블 모델의 예측에 따른 거래 수익을 시뮬레이션한다.  
  \* futureTrader60M\_ensemble: kiwoom API를 이용하는 자동 거래 프로그램. PyQt5 library를 이용하여 UI구현. Kiwoom API를 사용하기 위하여 32bit python이 필요함. 1시간 마다 자동으로 ensemble 모델을 실행하여 그 예측 결과에 따라 거래하고 장 종료 5분전에 보유 수량을 전부 청산한다.  
  \* futureTrader60M\_ensemble2: futureTrader60M\_ensemble는 ensemble 예측 version 2, 4, new 중 하나에만 의존하여 거래하는 반면, futureTrader60M\_ensemble2는 2, 4, new 3개 시스템의 앙상블 예측을 모두 사용하여 거래한다. 따라서 각 개별 시스템의 거래량은 총 거래 가능한 양의 1/3씩 할당된다. 즉 3개 시스템의 거래 결과를 평균하려는 의도이다.
* 알고리즘 트레이딩4: 알고리즘트레이딩2와의 차이는 input data구성에 있다. ADX와 DMI정보가 추가 된다.
* 알고리즘 트레이딩new: 알고리즘트레이딩4 input data에 스토캐스틱 슬로우 정보가 첨가된다. 또한 2, 4와는 다르게 모델 type을 C, HL, P로 구분하지 않고 현재 종가가 n봉(1~10) 후 종가와 단순 비교하여 크면 고점, 작으면 저점으로 간주한다. 고점에서는 매도, 저점에서는 매수 시그널을 발생한다.
* cmd에서 실행되는 자동 거래 batch 실행 파일  
  \* futureTrader60M\_ensemble.bat:  
  \* futureTrader60M\_ensemble.bat2
* 거래 프로세스를 요약해서 설명하자면  
  1. 아침 10시 이전에 futureTrader60M\_ensemble.bat2를 실행한다.  
  2. 프로그램 실행 여부 확인을 클릭하면 자동로그인 되면서 프로그램이 명령창에서 실행되고 trading monitor window 화면이 뜬다.  
  3 모니터 화면에는 현재 예탁금 상황, 보유 종목의 손익 상황 등이 display된다.  
  4. 수동 거래도 가능하도록 화면에서 support한다.  
  5. 10시가 되면 자동으로 데이터를 키움 API로부터 다운받아 data file이 업데이트 되고 전처리 되어 저장되고 앙상블 모델이 예측한다.  
  6. 앙상블 예측에 따라 신규 주문, 추가 주문, 청산 후 신규 주문 등의 거래가 자동으로 처리된다.  
  7. 1~6까지의 과정은 매 시간 0분 마다 한 번씩 발생한다.  
  8. 1분마다 손절 여부 체크하여 매입 가격 대비 1%이상 지수 변동이 발생하면 청산.  
  9. 15시 30분에는 달러 선물, 카카오, 삼성전자 주식 거래를 한다. (일 단위 거래). 코스피 선물과 데이터, 예측 및 거래는 유사하지만 일 단위로 한다는 차이가 있다. 거래는 실제 발생하지 않고 거래 장부상으로만 가상으로 이루어진다.  
  10. 장 종료 후 마지막 시간 대의 데이터 다운로드하여 저장.
* 각 시스템의 앙상블 예측 시 self-reflection 기법을 사용한다. Self-reflection이란 자기 반성적 조정 시스템으로서 학습된 모델들의 앙상블 결과를 전면 신봉하지는 않는다는 것이다. 학습된 모델들은 과거 데이터를 기반으로 예측하기 마련이다. 증권 시장은 속성상 패러다임이 수시로 바뀌기 때문에 하나의 모델 또는 모델 그룹의 전략에 의존하는 것은 위험하다. 그렇다고 수시로 모델을 학습하여 패러다임을 반영하고자 하는 시도는 noise를 증가시키는 역효과를 낳을 수 있다.  
    
  간단한 해결책은 예측 모델을 일정 기간 관찰하여 모델의 accuracy를 evaluate하는 것이다. Evaluation의 결과가 기대 이하이면 모델의 예측과는 반대로 예측 값을 발생시킨다.   
    
  축구 같은 스포츠 시합에서도 처음 설정했던 전략이 잘 안 통하면 반대의 전략을 시도하는 것을 종종 볼 수가 있다.  
    
  물론 주기적인 평가에 의해 모델 또는 앙상블 자체를 다른 것으로 switch할 수도 있다. 이것은 인간의 직관으로 수동적으로 행하는 것이고 앞서 설명한 self-reflection은 앙상블 예측 프로세스에 embedding 되어 자동으로 처리된다. 따라서 모델의 장기적 신뢰성을 유지한 채 일시적 환경 변화에 효율적으로 대처할 수 있는 방법이라 할 것이다.  
    
  \* self-reflection은 make\_reinfo라는 모듈에서 구현되어 있다.

\* 모델 예측에 의한 거래 수익을 관찰하여 예측 결과를 reinforcing 하는 make\_reinfo\_profit\_comp, make\_reinfo\_updown 과 같은 모듈들이 있으나 모델의 예측 시그널의 정확도와 거래 수익은 꼭 일치하는 것이 아니기 때문에 일관성이 결여된다. 이러한 모듈들을 적용했을 때 문제점은 이전 수행 결과에 선택된 모델이 향후 비슷한 결과를 보일 확률이 원 모듈 보다도 많이 떨어진다는 것이다. 그래서 이 모듈들은 적용 포기하기로 하였다.

다음은 키움 모의투자 수익률 그래프이다. (2023-01-02부터는 실질 투자)



* 알고리금트레이딩의 기본 모델들  
  \* 알고리즘트레이딩2: [5C, 5HL, 5P, 10C, 10HL, 10P, 5C, 15HL, 15P, 20C, 20HL, 20P, 25C, 25HL, 25P, 30C, 30HL, 30P, 40C, 40HL, 40P,]  
    
  \* 알고리즘트레이딩4: [5C, 5HL, 5P, 10C, 10HL, 10P, 5C, 15HL, 15P, 20C, 20HL, 20P, 25C, 25HL, 25P, 30C, 30HL, 30P, 40C, 40HL, 40P,]  
    
  \* 알고리즘 \트레이딩new: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
* 알고리즘트레이딩에서 거래에 적용하는 모델을 선택하는 방법  
  1. 기본 모델들을 15일 마다 학습시킨다.  
  2. 학습된 모델들의 모든 앙상블(3개 모델)들의 최근 1년간 수익률을 비교해서 best 10 앙상블을 뽑는다. 이 때 앙상블의 설정 변수들은 이전 거래의 적용 앙상블의 설정 변수로 setup (pred\_term, reinfo\_th, losscut, profit\_cut. . .) – create\_all\_termensemble.py 참조  
  3. 2에서 뽑은 10개의 앙상블에 대해 설정 변수값(pred\_term, reinfo\_th) 의 주변의 값들로서 2019~2021, 2022~현재 의 수익률과 표준편차를 뽑아 2019~2021의 년 평균보다 2022~현재의 년 평균 수익률(복리)이 향상한 것들만 추려서 2개의 값을 평균한다..  
  4. 3에서 best 5를 선택하여 최근 수익률(30봉, 7봉)을 평가한다.  
  5. 30봉 수익률이 좋은 것들 중에서 최근 7봉 수익률이 크게 상승되는 것을 선택하여 거래 적용 앙상블 모델로 삼는다.