공학석사 학위논문

경사 하강 유전 프로그래밍을 통한 주식 시장 시뮬레이션

2022년 8월

서울대학교 대학원

컴퓨터공학부

황 순 용

경사 하강 유전 프로그래밍을 통한 주식 시장 시뮬레이션

지도 교수 문 병 로

이 논문을 공학석사 학위논문으로 제출함
2022년 4월

서울대학교 대학원
컴퓨터공학부
황 순 용

황순용의 공학석사 학위논문을 인준함
2022년 6월

위 원 장 _____ 이 광 근 _____ (인)

부위원장 _____ 문 병 로 _____ (인)

위 원 _____ 허 충 길 _____ (인)

초 록

주식 가격 예측 문제는 오랫동안 연구되어 왔으며 최근 기계학습 방법론에 의해 활발히 연구가 진행되고 있다. 일반적인 주식 데이터는 노이즈가 다수 포함되어 있으며 같은 상황이라도 외부의 영향에 의해 다른 주가 방향을 보여주기도 한다. 기계학습 알고리즘은 주식 시장의 노이즈를 줄이고 주가 상승 예측 정확도를 높이는 점에서 크게 각광받았다. 특히 유전 프로그래밍은 다양한 문제 구조에 대해 최적화하기 용이하고 적절한 목적함수를 적용할 수 있다. 이러한 특성 때문에 유전 프로그래밍은 주식 가격 예측 혹은 집행 알고리즘에 자주 활용되었으며 실제 투자에서도 지속적으로 이용되고 있다. 본 연구에서는 이러한 유전 프로그래밍을 개선하여 각국의 주식시장을 분석하고 시뮬레이션을 진행하였다. 유전 프로그래밍은 여러 가지 문제 구조에 대해 탐색할 수 있다는 장점이 있지만 문제 구조가 방대하기 때문에 탐색에 오래 걸린다는 단점이 있다. 이를 보완하기 위해 트리 구조의 유전 프로그래밍 해 집단에서 각각 노드의 gradient를 구하고 이 정보를 이용하여 평가 함수가 빠르게 최대화될 수 있도록 최적화하였다. 이를 효율적으로 계산하고 유지할 수 있도록 LRU(Least Recently Used) 캐시 시스템을 개발하여 빠르게 탐색할 수 있도록 구현하였다. 경사 하강 유전 프로그래밍 방법론을 평가하기 위해 한국, 미국, 일본 3 국가의 주식시장 데이터를 수집하였다. 본 연구에서는 주식시장 데이터를 통해 유전 프로그래밍의 개선점, 시뮬레이션 방법론을 찾을 수 있었다. 각 국가의 주식시장에 대해 기존 유전 프로그래밍 방법과 경사 하강 유전 프로그래밍 방법론을 비교하였으며 같은 세대 수에 비해 개선된 방법이 기존 방법론에 비해 보다 빨리 탐색하는 것을 확인할 수 있었다. 또한 다른 기계학습 알고리즘과 비교하여 전반적으로 높은 정확도를 얻을 수 있었다. 결과적으로 유전 프로그래밍과 쉽게 호환되는 시뮬레이션 방법을 제안하여 지속적으로 평가 및 개발할 수 있도록 하였다.

주요어 : 유전 프로그래밍, 기계학습, 주식 가격 예측, 경사 하강 유전 프로그래밍
학 번 : 2020-23509

목 차

표 목차

그림 목차

제 1 장 서론

주식 가격 예측 문제는 오랫동안 연구되어 왔으며 최근 기계학습 방법론이 크게 발전함에 따라 관련 알고리즘을 적용한 해석도 늘어나고 있다. 다양한 기계학습 알고리즘 중에서도 유전 프로그래밍 방법론은 목적함수를 임의로 구성할 수 있으며 다양한 문제구조에 적합하기 때문에 처음 제안된 이후로도 꾸준히 이용되고 있다. 본 연구에서는 유전 프로그래밍 방법론을 개선하여 목적함수에 가깝도록 보다 빠르게 탐색하는 방법을 제안하고 이를 주식 가격 예측에 활용하고자 한다.

주식시장을 포함하여 선물옵션시장, 해외선물 시장 등 금융시장을 분석하기 위해 여러 연구가 진행되어 왔다. 금융시장에 대한 연구는 가격 예측에 대한 연구(Yu *et al.*, 2020), 거래 집행 알고리즘에 관한 연구(Cui *et al.*, 2009), 포트폴리오 리스크 최적화에 관한 연구(Black *et al.*, 1992) 등 다양한 연구가 이루어지고 있다. 이들의 연구는 공통적으로 금융시장 데이터에 큰 노이즈가 있음에 집중하여 과적합을 피할 수 있는 방법을 제안한다. 본 연구에서는 이러한 연구 중에서도 주식시장의 주식 가격 예측에 관련된 분석을 하고자 한다. 주식 가격이 아니더라도 다양한 자산군에 대해 이러한 시장 데이터가 시간에 따라 지속적으로 변경된다는 특징을 이용하여 시계열 데이터 분석의 확장으로 분석이 이루어졌다(Idrees *et al.*, 2019). Qiu *et al.* (2020)에 따르면 회귀 모델부터 최근에는 딥러닝 모델을 활용하여 각 시점에 적절한 어텐션을 적용하여 예측을 하기도 한다. 하지만 모델이 복잡할수록 주식 데이터는 쉽게 과적합되기 때문에 데이터에 따라서는 오히려 간단한 모델이 좋은 성능을 보여주기도 한다.

유전 프로그래밍은 유전 알고리즘의 여러 연산자를 이용하여 해의 구조가 정해져 있지 않은 최적화 문제를 푸는 기계학습 방법론이다(Koza, John R., 1994). 그에 의해 제안된 방법론은 실생활의 문제를 해결하기 위해 이후에도 꾸준히 개발되었으며 해외 선물, 주식 분야에서도 흔히 쓰이고 있다. 특히 다른 기계학습 방법론의 경우 분류 문제를 풀기 위해 제한된 손실 함수를 사용해야 하지만 유전 프로그래밍의 경우 목적에 맞도록 목적 함수를 설정할 수 있어 다양한 함수 적용이 필요한 금융시장에 적합하다. 본 연구에서는 이를 활용하여 주식 가격 예측을 하고 시뮬레이션을 하여 현실적인 알고리즘 활용 방안을 모색하였다.

본 논문은 2장에서 선행 연구를 통해 유전 프로그래밍과 다른 기계학습 방법론에 대해 알아본다. 3장에서는 기존 유전 프로그래밍을 개선한 빠른 탐색을 가능하게 하는 경사 하강 유전 프로그래밍 방법론에 대해 알아보고 그 구현방법에 대해 설명한다. 4장에서는 주식 데이터 전처리 방법과 실험 결과 및 분석에 대해 다룬다. 5장에서는 연구를 정리하고 향후 연구 방향에 대해 알아본다.

제 2 장 배경 이론

제 1 절 유전 프로그래밍

2.1.1. 유전 알고리즘과 유전 프로그래밍

유전 프로그래밍(Genetic Programming)은 여러 유전적 연산자를 이용하여 프로그램을 탐색하고 해를 찾아가는 과정을 말한다. Koza, John R. (1994)에 의해 제안된 이 방법론은 LISP와 같이 트리 형태의 프로그램을 탐색하는 용도로 개발되었으며 프로그램을 만들어내는 프로그램으로 각광받았다. 이 방법론은 유전 알고리즘과 비교하여 더욱 다양한 해의 표현이 가능하여 현실의 문제에 적용된 여러 알고리즘(Kaboudan *et al.*, 2000; Vasilakis *et al.*, 2013)이 제안되어왔다.

유전 알고리즘은 주어진 해의 표현에 대해 목적에 가까워지도록 해를 탐색해가며 최적화한다. 해는 일반적으로 어떤 변수에 대한 배열이 될 수 있으며 2차원 배열 혹은 특정 문제 공간이 될 수 있다. 문제 공간을 모두 탐색하는 것이 실질적으로 불가능한 경우 유전 알고리즘은 효과적인 대안을 제시한다. 해는 특정 인구 수만큼 초기화하고 이를 여러 세대를 거쳐가며 새로 만들어내고 평가한다. 각 세대 마다 무작위로 여러 해를 추출하여 이를 서로 교차하고 무작위로 변이를 추가하여 해를 생성한다. 결과적으로 각 해는 평가 함수를 통해 목적에 얼마나 가까운지 평가하고 평가 점수가 좋은 해들을 보다 높은 확률로 추출하여 해를 새로 생성한다.

유전 프로그래밍의 여러 연산자는 유전 알고리즘의 개념을 공유한다. 유전 프로그래밍은 해의 공간이 고정적이지 않다는 점에서 유전 알고리즘과 다르다. 유전 알고리즘은 해의 크기가 정해져 있고 문제

성질이 정해져 있는 상태에서 좋은 성능을 보여준다. 유전 프로그래밍은 해의 크기를 동적으로 변경할 수 있다. 예를 들어, 트리의 경우 그 높이를 변경하여 가변적으로 해를 다룰 수 있으며 세대를 반복하며 해가 문제에 적합한 크기로 변형될 수 있다.

일반적인 유전 프로그래밍의 해는 트리로 구성되어 있다. 트리의 말단에 해당하는 노드는 해의 계산에 필요한 데이터 혹은 상수가 될 수 있다. 반면에 트리의 말단이 아닌 노드의 경우 사칙연산 혹은 논리 연산자가 될 수 있으며 때로는 조건문, 반복문 또한 노드로 사용하기도 한다. 데이터와 연산자로 노드가 이루어지는 경우 계산은 말단에서 루트 노드까지 데이터가 계산되며 하나의 값이 되고 이 값이 평가에 활용된다. 노드에 설정 가능한 데이터와 연산자의 범위에 따라 트리의 문제 공간이 정해진다.

2.1.2. 유전 프로그래밍의 연산자

해를 생성하고 탐색하기 위해 유전 프로그래밍에서는 여러 연산자가 필요하다. 연산자는 유전 알고리즘에서 사용하는 것과 비슷하며 해의 구조에 따라 그 과정이 달라진다. 트리 구조의 유전 프로그래밍에서 교차는 여러 방법을 통해 이루어진다. 두 트리가 있을 때 교차는 두 트리의 노드를 임의로 선택하여 노드를 포함하는 하위 리까지 모두 바꾼다. 변이는 주어진 트리의 노드를 임의로 바꾸는 식으로 진행된다. 노드의 연산자를 바꾸거나 트리 말단 데이터를 바꾸거나 노드의 하위 트리를 전부 바꾸거나 트리를 성장시킬 수 있다. 본 연구에서는 유전 프로그래밍에 같은 확률로 위와 같은 변이를 진행하도록 하였다. 생성된 새로운 해는 각각 평가를 하여 해집단에 평가 지표가 좋아지도록 대체한다.

유전 프로그래밍의 연산자가 아닌 각 노드의 연산자의 경우 보다 간단한 연산을 선택하여 문제 구조의 공간을 줄이는 편이다. 본 연구에서는 일반적인 경우를 다루기 위해 반복문 등을 사용하지 않고 사칙연산을 포함한 기본적인 연산에 집중하였다. 또한 사칙연산은 각 노드의 gradient를 구하기 용이하다는 장점이 있다. 표 1에서 본 연구에 활용한 7가지 연산을 확인할 수 있다.

구분	예시
+	a + b
−	a - b
*	a * b
~	−a
log	log(1 + abs(a))
min	minimum(a, b)
max	maximum(a, b)

표 1. 각 노드의 연산 후보군

제 2 절 기계학습 알고리즘

2.2.1. 의사결정나무

전통적인 분류 문제를 풀기 위해 유전 프로그래밍을 제외하고도 수많은 머신러닝 알고리즘들이 개발되어 왔다. 의사결정나무(Cramer *et al.*, 1976)는 트리 구조의 해가 주어진 모든 데이터를 구분 기준을 통해 점진적으로 분류한다. 트리의 끝까지 데이터가 분류될수록 보다 자세한 분류가 가능해진다. 각 노드에서 분류할 때 최대한 레이블 수가 잘 나누어지도록 분류를 반복한다. 일반적인 분류 체계에서는 나무가 급격히 커지고 잘못 학습되는 것을 방지하기 위해 가지치기(Mingers *et al.*, 1989) 등 보완 알고리즘을 적용하여 학습한다. 경우에 따라서는 분류 모델이 아닌 값을 예측하는 모형으로 활용하기도 한다.

2.2.2. Bagging

Bagging이란 Bootstrap aggregating의 약어로 여러 분류기를 합쳐 성능을 올리는 앙상블 방법 중 하나이다(Breiman, 1996). 그가 제안한 Bagging 기법은 주어진 데이터를 여러번 샘플링하고 각각 분류기를 만들어 분류기들의 평균을 통해 최종 분류기를 구성한다. 의사결정나무와 같이 불안정한 분류기에 대해 Bagging은 좋은 성능을 발표했지만 안정적으로 알려진 선형 모델과 같은 분류기에 비해 성능이 좋지 않았다. 이후 Friedman and Hall, (2007)은 보다 안정적인 분류기를 제안했으며 Bühlmann and Yu, (2002)는 Bagging의 안정성을

수학적으로 증명하려는 시도를 하였다. Bagging 방법론은 주어진 모델에 대해 쉽게 성능을 개선할 수 있는 앙상블 기법으로 자주 이용되는 기법이다.

2.2.3. Boosting

Boosting 기법은 분류기의 앙상블 기법 중 하나로 약한 분류기로 불리는 여러 분류기의 결과를 종합하여 강한 분류기의 최종 분류를 하는 모델이다(Freund and Schapire, 1999). 그들은 처음으로 실질적 사용가능한 Boosting 계열의 AdaBoost 방법론을 제안하였다. 이는 모든 약한 분류기를 초기화하고 각 분류기의 정확도를 계산하여 각 분류기에 데이터에 따라 가중치를 주고 틀린 데이터에 큰 가중치를 주어 보완하는 역할을 한다. 이후 Schapire and Singer, (1999)는 AdaBoost 방법론이 steepest descent 알고리즘임을 증명하였으며 이를 활용하여 실수값을 예측하는 Real Boost 방법론을 제안하였다. Friedman, (2001)은 AdaBoost를 확장하여 각 분류기의 가중치를 구할 때 gradient를 구하여 각 분류기를 결합하는 방법을 제안하였다. 보통 자주 활용하는 로그 손실 함수와 분류기를 의사결정나무로 활용하는 Gradient Boost를 Logit Boost로 불리며 널리 쓰이는 알고리즘이다.

제 3 절 Computational Graphs

최근 딥러닝의 발전으로 자연어 처리, 이미지 인식 분야에서 괄목할 만한 성능을 보여주고 있다. 전통적인 기계학습 알고리즘의 경우 고차원의 데이터를 처리하기 쉽지 않았지만 신경망의 경우 이를 반복적으로 처리하여 큰 차원의 데이터에서도 과적합 없이 문제를 학습할 수 있었다. 여러 층으로 이루어져 있는 네트워크는 다양한 연산자를 통해 결과를 도출한다. 각각의 연산자와 그 사이의 feature map을 모두 그래프의 노드와 간선으로 표현할 수 있다.

흔히 이용되는 딥러닝 프레임워크에서는 딥러닝 모델을 computational graph로 관리하여 학습과 추론을 지원한다. 기존의 TensorFlow(Abadi *et al.*, 2016)에서는 모델을 구성할 때 데이터를 Tensor로 표현하고 이를 연산하는 것을 Function이라 칭한다. 각각 Function을 노드로 하고 Function에 쓰이는 텐서를 자식 노드로 하여

트리를 구성할 수 있다. 학습 혹은 추론을 하기 전 트리를 생성해 활용하는데 이를 정적 그래프로 명명한다. 다른 딥러닝 프레임워크 중 하나인 PyTorch(Paszke *et al.*, 2019)는 이와 반대로 동적 그래프를 생성한다. 어떤 연산이 호출될 때 필요한 텐서와 그 결과 텐서를 그래프로 만들어 나간다. 이런 경우 매 연산마다 그래프를 생성해야 한다는 단점은 있지만 유동적으로 그래프를 만들 수 있다는 장점이 있다. 이러한 딥러닝 프레임워크에서는 각각 그래프마다 연산의 gradient 계산 과정이 정의되어 있다. 역전파 알고리즘에 의해 파라미터가 학습될 때 각 gradient또한 계산되며 경사하강법을 통해 학습된다.

Tree-based Genetic Programming에서 활용되는 트리 또한 일종의 computational graph라고 볼 수 있다. 각각의 연산이 반복문 등 미분이 불가능한 경우가 아닌 경우 각 연산 마다 gradient 계산이 가능하여 딥러닝의 경우와 같다고 볼 수 있다. 딥러닝 모델의 경우와는 달리 유전 프로그래밍에서는 각 노드값이 탐색을 반복하며 변경된다. 본 연구에서는 어떤 트리가 평가될 때 재귀적으로 말단에서 루트까지 모든 계산이 수행되고 루트로부터 다시 말단까지 gradient 계산을 수행하도록 구현하였다.

제 3 장 경사 하강 유전 프로그래밍

제 1 절 경사 하강 유전 프로그래밍 알고리즘

유전 프로그래밍은 제한되지 않은 해의 구조를 탐색하여 해를 찾아가는 장점이 있지만 문제 공간이 매우 넓어 빠르게 최적화가 어렵다. 본 연구에서는 이를 해결하기 위해 트리 구조의 해를 역전파 알고리즘을 활용하여 gradient를 구하고 이를 통해 local optimization에 활용하고자 한다. 역전파 알고리즘과 유전 알고리즘을 결합하여 알고리즘 성능을 개선하려는 여러 연구가 있었다. 분류 문제에서 최종 평가 성능을 개선하는 연구(Topchy and William, 2001)에서는 분류 모델의 성능을 개선하기 위해 데이터에 해당하는 말단의 데이터를 바꾸는 과정을 개발하였다. 본 연구에서는 트리에 있는 모든 노드에 대해 gradient를 구할 수 있도록 구성하여 보다 넓은 범위의 해를 탐색하고자 한다.

각 노드의 gradient를 알게 되면 목적 함수에 보다 가까워지는

방향에 대해 알 수 있으며 어떤 노드로 바꾸었을 때 보다 좋은 성능을 가져올지 알 수 있다. 각 노드의 gradient를 알고 있을 때 gradient의 방향의 값에 적합한 하위 트리로 노드를 변경한다면 성능의 이득을 기대할 수 있다. 유전 프로그래밍의 탐색 과정 중 각 노드의 gradient 값을 계산하고 여러 하위 트리 중 그 계산값이 gradient의 값에 가장 가까운 값으로 대체한다. 하위 트리의 후보군은 트리의 높이에 따라 그 갯수가 매우 많아지며 모든 하위 트리의 정보를 유지하며 개선하기는 현실적으로 불가능하다. 이를 효율적으로 개선하기 위해 탐색과정 중 관찰 가능한 subtree에 대해서 그 정보를 저장하기로 하였다. 유전 프로그래밍의 특성상 각 하위 트리의 값은 루트까지 값을 계산할 때 항상 계산해야 하는 값이다. 이를 이용하여 탐색 과정 중 하위 트리의 값이 계산될 때 하위 트리의 postfix 값과 그 계산 값을 데이터베이스에 저장하도록 하였다. 데이터베이스의 최대 크기를 지정하고 LRU 형식으로 캐시를 설정하여 데이터가 저장되도록 하였으며 만약 검색하여 해당 데이터가 검색되면 다시 캐시의 가장 앞쪽에 위치하도록 하였다.

본 연구에서는 각 해를 평가하기 전 임의로 하나의 노드를 선택하여 그 노드의 계산 값과 gradient 값을 저장하도록 했다. 재귀적으로 트리의 값이 계산될 때 해당 노드의 값을 계산하고 이후 역전파 알고리즘을 수행한다. 역전파 알고리즘은 모든 노드를 대상을 수행할 필요 없이 해당 노드로 이어지는 부분만 수행하면 된다. 이미 저장한 캐시로부터 그 방향으로 얼마나 일치하는지 값을 검색하여 속도면에서 이득을 취할 수 있었다. 본 유전 프로그래밍 알고리즘은 표 2에 소개되어 있다.

```
Input: populations, generations, cache
for step in 1...generations:
    s1, s2 = selection(populations)          // 두 해를 추출
    s = crossover(s1, s2)                    // 교차 연산
    s = mutation(s)                          // 변이 연산
    node, value, gradient = evaluate(s)
    new_node = search(cache, value, gradient)
    swap(s, node, new_node)  // 추출한 노드로 바꾼다.
    evaluate(s)
    replacement()
```

```
Function search(cache, value, gradient)
    max_fit, max_node = 0, None
    for node, item in 1...items:
        // sign은 양수면 1, 음수면 -1을 반환하는 함수이다. cache에는 subtree와 그 계산값이 있으며 각각 계산값이 gradient 방향으로 얼마나 이루어져 있는지 계산한다.
        fit = sum(sign(gradient) * sign(item - value))
        if fit > max_fit:
            Max_fit = fit, max_node = node
    return node
```

표 2. 경사 하강 유전 프로그래밍 방법론과 캐시 탐색 방법

제 2 절 알고리즘 구현 방법

실험적으로 본 알고리즘을 구현했을 때 기존 유전 프로그래밍으로부터 큰 속도 저하가 있었다. 이를 보완하기 위해 빠르게 탐색하기 위한 여러 시도를 적용하였다. 첫번째로 학습 혹은 테스트를 할 때 하나의 샘플에 대해서 재귀적으로 계산하지 않고 모든 샘플을 하나의 벡터로 만들어 계산하도록 하였다. 또한 계산을 GPU 상에서 하도록 하여 한 트리를 평가할 때 최대한 재귀적인 연산을 줄이고 분산처리를 할 수 있도록 구현하였다.

한편 어떤 노드와 두 자식 노드가 모두 '+' 연산일 경우 이 자식 노드들을 재배치해도 같은 결과를 가져온다. 이를 활용하여 트리의 높이에 균형을 맞추기 위해 'max', 'min' 연산 등 연산 순서를 바꿔도 같은 결과를 가져오는 성질을 가지고 있으면 트리 높이에 균형을 줄 수 있도록 재배치하는 과정을 추가하였다.

캐시에서 gradient 값과 노드 값을 통해 다른 하위 트리를 검색하는 과정 또한 시간이 오래 걸리는 과정 중 하나이다. 모든 아이템을 검색하여 가장 좋은 값을 추출하는 경우 모든 값을 차례대로 탐색하기 보다 모든 값을 텐서로 합치고 탐색 계산을 GPU 상에서 하여 계산 시간을 최대한 줄였다. 계산을 통해 가장 큰 값의 위치를 찾아내고 이 값에 대응되는 postfix 값은 새로 트리로 생성되어 다른 노드에 대체될 수 있도록 한다.

제 4 장 실험 결과

제 1 절 실험 데이터셋

본 연구에서는 금융시장에서 유전 프로그래밍의 효과를 확인하기 위해 한국, 미국, 일본 3개국의 주식시장 데이터를 추출하여 실험하였다. 한국 주식시장의 경우 코스피, 코스닥에 상장된 주식 약 3000 종목을 대상으로 일별 주가와 거래량 데이터를 수집하였다. 한국 주식의 데이터는 대신증권㈜에서 제공하는 API 서비스를 이용하여 각 종목에 대한 정보를 정리하였다. 미국 주식시장의 경우 Compustat 데이터 제공자를 통해 북미 전역의 주식 종목 약 2만개를 수집할 수 있었다(Boritz *et al.*, 2013). 일본 주식시장 또한 마찬가지로 Compustat 데이터 제공자를 통해 일본 전역의 주식 종목 약 3천개의 데이터 샘플을 받을 수 있었다. 주식 시장의 데이터는 일별로 2010년부터 2021년까지의 모든 데이터를 수집하였다.

주식시장은 최근 급격한 변동을 겪었다. 특히 2020년의 경우 연초에는 크게 하락했으며 연말에는 급등 시장을 볼 수 있었다. 보다 객관적인 평가를 위해 모델을 학습하고 평가할 때는 2020년의 데이터를 쓰지 않고 2019년과 2021년 데이터를 사용했다. 만약 학습, 검증, 평가가 필요한 경우 2018년 데이터로 학습하고 2019년 데이터로 검증하여 2021년 데이터로 평가하였다.

주식 예측에서 쓰이는 주가 데이터는 보통 있는 그대로 쓰지 않고 시장의 흐름을 표현할 수 있는 다른 전처리를 추가로 수행하여 분석에 활용한다. 주가와 거래량 데이터를 활용하기 위해 본 논문에서는 주가, 거래량 각각의 모멘텀과 변동성을 분석하기 위해 표 3와 같은 데이터를 사용하였다. 알고리즘을 평가할 때는 각 연도 별로 2만 개의 시점을 임의로 선택하여 각 지점에 대해 데이터 전처리를 수행하였다.

구분	설명
ma5	(price − mean(price, 5)) / std(price, 5)
ma10	(price − mean(price, 10)) / std(price, 10)
ma20	(price − mean(price, 20)) / std(price, 20)
vol5	(volume − mean(volume, 5)) / std(volume, 5)
vol10	(volume − mean(volume, 10)) / std(volume, 10)
vol20	(volume − mean(volume, 20)) / std(volume, 20)
stk5	(price - min(price, 5)) / (max(price, 5) − min(price, 5))
stk10	(price - min(price, 10)) / (max(price, 10) − min(price, 10))
stk20	(price - min(price, 20)) / (max(price, 20) − min(price, 20))
macd	mean(price, 9) / mean(price, 26) - 1.0

표 3. 각 전처리 값의 계산 방법. price는 예측 시점의 주가를 의미하며 volume은 log(거래량 + 1) 을 의미한다. mean(, k), std(, k), min(, k), max(, k) 함수는 각각 k일간의 평균, 표준편차, 최소값, 최대값을 의미한다.

제 2 절 실험 과정 및 분석

4.2.1. 구현 알고리즘의 탐색 속도

구현한 경사 하강 유전 프로그래밍과 일반적인 유전 프로그래밍 방법론의 탐색 속도를 비교하기 위해 3개국의 주식 시장 별로 실험을 진행하였다. 각 예측 시점을 기준으로 데이터는 26일간의 주가 정보를 가지고 있으며 예측은 5, 10, 20일 후를 예측하는 것을 목표로 하였다. 유전 프로그래밍에서 각 해의 평가는 예측 정확도가 아닌 상승으로 예측한 시점들의 기하평균 수익률으로 하도록 하였다. 실험을 진행했을 때 각 예측 시점에 주가 흐름은 다르지만 이를 일률적으로 0 또는 1의 예측을 할 때 분류기의 성능이 좋지 않았다. 이를 보완하기 위해 예측한 종목에 대해 실제 수익률을 대상으로 평가하게 되었다.

각 프로그래밍을 대상으로 인구 수는 1000개, 세대 수는 1000번, 한 세대에 새로 생성하는 해를 40개로 설정하여 유전 프로그래밍을 실행하였다. gradient를 구할 때 손실 함수는 binary cross entropy를 활용하여 루트 노드에 대해 계산하였고 이를 통해 gradient를 구하였다. 각 시장에 대한 비교는 그림 1에서 확인할 수 있다.

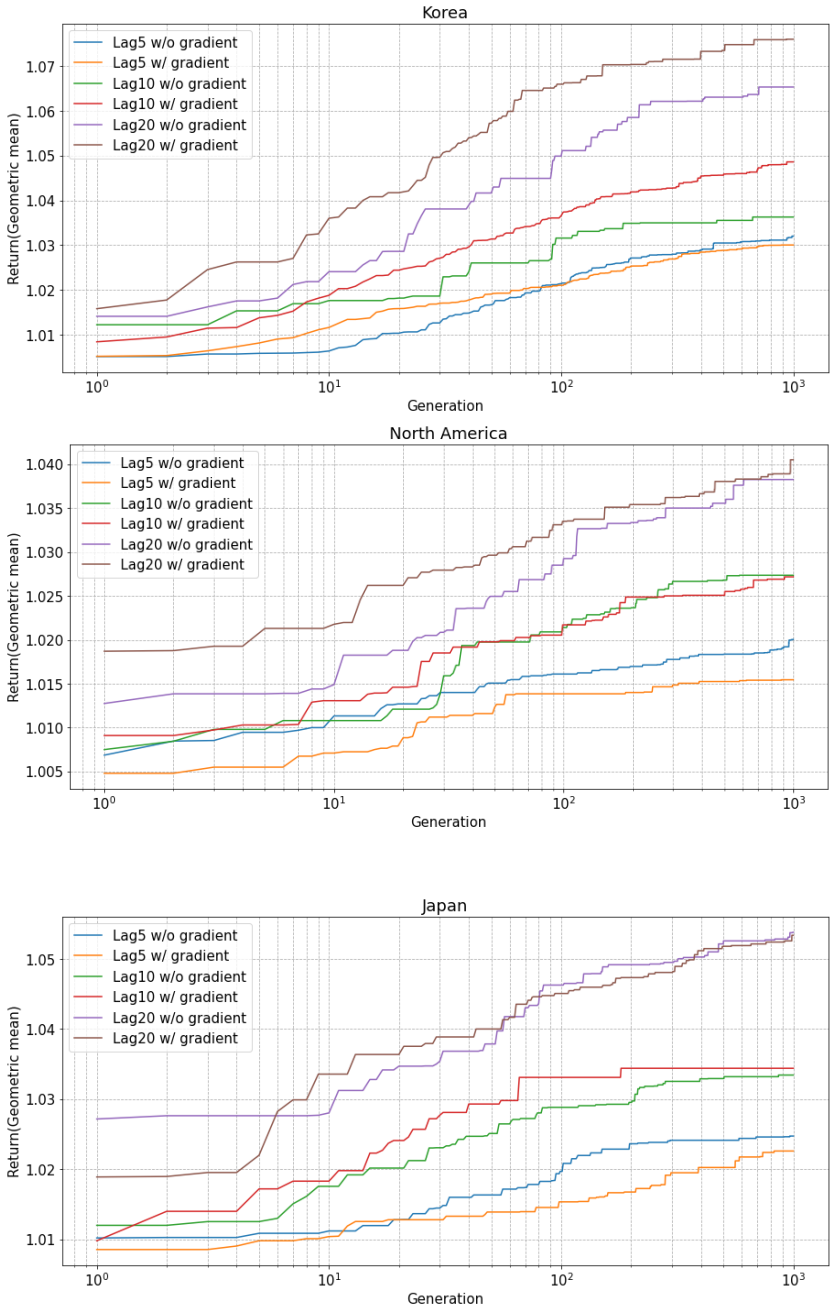

그림 1. 각 시장의 학습 속도 비교. 5, 10, 20일 후 상승률에 대해 유전 프로그래밍과 비교하였다.

4.2.2. 기계학습 알고리즘과 비교

본 연구에서 제안한 유전 프로그래밍이 다른 기계학습 분류 알고리즘에 비해 어느정도 효과가 있는지 확인하기 위해 앞 절과 같은

데이터로 학습 및 평가를 수행하였다. 본 주식 시장 분류 문제는 각 예측 시점에 대해 데이터 차원이 크지 않은 편이며 각 데이터가 각기 다른 값과 의미를 가지고 있기 때문에 의사결정나무 계열의 모델이 잘 작동할 것으로 예상하였다. 본 연구에서는 의사결정나무, AdaBoost, Gradient Boosting, Random Forest 총 4개의 모델을 유전 프로그래밍과 같은 데이터로 학습하고 평가하였다. 보다 공정한 평가를 하기 위해 2018년 데이터를 학습 데이터, 2019년 데이터를 검증 데이터 2021년 데이터를 평가 데이터로 활용하였다. 각각 모델마다 모델 파라미터를 여러번 다르게 학습하며 가장 검증 결과가 좋은 모델 파라미터를 추출하였고 이 모델로 최종 모델 정확도를 계산하였다. 모델 파라미터의 경우 의사결정나무는 최대 깊이를 3, 5, 7과 최대 feature 수를 3가지로 변경해가며 학습하였다. AdaBoost와 Gradient Boosting은 학습률을 0.1, 0.5, 1.0 3가지 값과 약한 분류기 수를 10, 20, 50, 100, 200개까지 값을 바꾸어가며 학습하였다. random forest는 트리의 최대 깊이를 3, 5, 7로 바꾸어 가며 학습하고 트리의 수는 10, 20, 50, 100, 200개로 바꾸어가며 학습하였다.

	의사결정 나무	Random Forest	AdaBoost	Gradient Boosting	Grandient GP
한국 시장	52.55%	52.69%	53.20%	52.71%	**53.26%**
미국 시장	51.38%	51.64%	52.35%	**53.31%**	52.04%
일본 시장	50.07%	50.01%	50.30%	50.69%	**51.42%**

표 4. 각 알고리즘의 평가 정확도(%). 경사 하강 유전 프로그래밍은 Gradient GP로 표현하였다.

실험 결과(표 4) 각 알고리즘의 정확도는 50%에 가까운 값을 확인할 수 있다. 일반적인 데이터보다 주식 데이터의 경우 노이즈가 많아 정확도가 높지 않다. 각 시장 마다 각기 다른 모델의 성능이 좋은 것을 볼 수 있었으며 시장에 특성과 각 알고리즘의 특성에 따라 다른 알고리즘을 적용해야 할 것이다.

4.2.3. 테스트 시뮬레이션

보다 실질적인 알고리즘 활용을 위해 이를 적용할 수 있는

시뮬레이션 방법을 제안한다. 유전 프로그래밍 학습 후 데이터에 대해 예측 성능이 좋은 트리를 얻을 수 있을 것이다. 이를 테스트 기간에 적용하여 마찬가지로 모든 종목의 각 지점마다 가격 예측치를 계산한다. 계산 결과 값에 sigmoid 함수를 적용하여 0.5 이상인 경우 매수한다고 가정하고 같은 시점에 매수하는 종목에 대해서는 동일 비중으로 매수한다고 보았다. 예를 들어 10일 후 예측을 한다면 각 종목은 10일 후 종가에 매도하는 것으로 보았으며 주어진 매수 금액의 1/10을 각 날짜에 매수하는 것으로 생각하였다. 실제 투자의 경우 보다 예측 점수가 높은 시점에 투자를 할 때 적은 경우에 대해 큰 수익률을 기대할 수 있다. 시뮬레이션을 할 때 sigmoid 함수가 0.5 이상이 아닌 0.7 혹은 0.9 이상인 경우에 매수를 하면 실험적으로 높은 수익률을 기대할 수 있었다. 그림 2는 2019년 데이터로 유전 프로그래밍 학습을 하고 2020년 데이터로 시뮬레이션을 진행한 결과이다. 거래비용은 매수, 매도 각각 0.15% 소모된다 가정하고 테스트하였다. 예측된 점수에 대해 0.9 이상의 경우 매수하면 기대수익률이 좋아지는 것을 볼 수 있다.

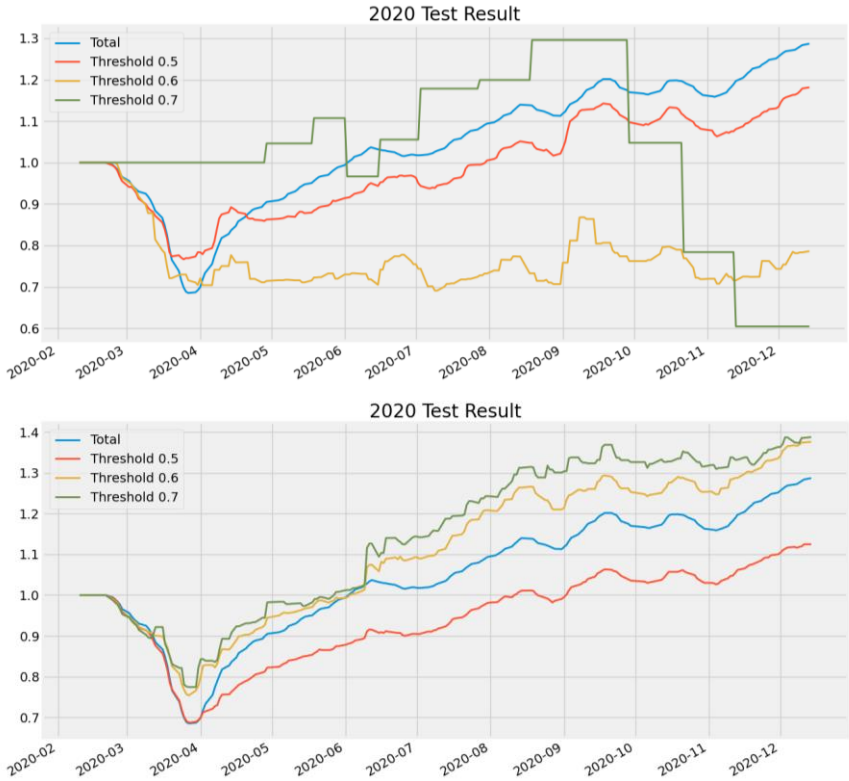

그림 2. 한국 시장의 테스트 기간 시뮬레이션. 위 그래프는 일반적인 유전 프로그래밍 방법론이며 아래 그래프는 경사하강을 통한 유전 프로그래밍 방법론으로 시뮬레이션 한 결과이다. 임계값에 대해 보다 부드럽게 상승한다.

제 5 장 결론 및 제언

본 연구에서는 한국, 미국, 일본 3개국 주식 시장의 주가 예측을 하기 위해 유전 프로그래밍 학습 방법론에서 gradient를 이용하여 성능을 개선하고 테스트 시뮬레이션을 수행하였다. 주식 시장의 데이터는 다른 데이터에 비해 노이즈가 심한 편이며 외적인 영향을 받아 같은 경우에 대해서도 다른 경향성을 보이기도 한다. 유전 프로그래밍은 다양한 문제 구조에서 확장적으로 사용되기 때문에 본 연구는 주식 가격 예측 문제를 풀기 위해 유전 프로그래밍 방법을 개선하여 분석하였다. 특히 주식 데이터는 각 예측 시점에 대해 실수의 여러 값들이 있기 때문에 트리 구조로 분석하기 적합하며 이 구조를 gradient를 구하기 쉽도록 변경하여 각 트리를 변경하도록 하였다.

실험 결과 본 연구에서 제안한 유전 프로그래밍 구조는 기존의 유전 프로그래밍 구조보다 빠르게 해를 탐색할 수 있었으며 다른 기계학습 알고리즘과 비교해서도 일부 시장에서 좋은 탐색 결과를 확인할 수 있었다. 유전 프로그래밍 특성상 한번 학습할 때 수만에서 수백만 해의 경우를 탐색하고 평가하게 되는데 기계학습 알고리즘에서 탐색하는 해의 개수와 차이가 있어 유전 프로그래밍의 결과가 좋게 나온 것으로 확인된다. 트리의 문제 공간을 보다 효율적으로 탐색하기 위해 gradient를 이용하여 탐색하였으며 이를 캐시를 통해 해결할 수 있었다.

추후 연구로 유전 프로그래밍 방법을 개선하기 위해 gradient을 통한 해의 탐색을 시도해볼 수 있을 것이다. 현 시스템에서는 주어진 캐시에서 가장 gradient에 적합한 아이템을 반환하는데 이때 gradient의 모멘텀을 활용하거나 여러 번 gradient를 구하여 local optimization을 보다 깊이 적용해볼 수 있을 것이다. 혹은 노드를 대체하는 방식으로 gradient를 활용하지 않고 해당 노드에 어떤 값을 더하거나 곱하여 노드를 개선할 수 있을 것이다.

참고 문헌

Cui, Wei, Anthony Brabazon, and Michael O'Neill. "Efficient trade execution using a genetic algorithm in an order book based artificial stock market." Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers. (2009).

Yu, Pengfei, and Xuesong Yan. "Stock price prediction based on deep neural networks." Neural Computing and Applications 32.6 (2020): 1609-1628.

Black, Fischer, and Robert Litterman. "Global portfolio optimization." Financial analysts journal 48.5 (1992): 28-43.

Idrees, Sheikh Mohammad, M. Afshar Alam, and Parul Agarwal. "A prediction approach for stock market volatility based on time series data." IEEE Access 7 (2019): 17287-17298.

Qiu, Jiayu, Bin Wang, and Changjun Zhou. "Forecasting stock prices with long-short term memory neural network based on attention mechanism." PloS one 15.1 (2020): e0227222.

Koza, John R. Genetic programming II. Vol. 17. Cambridge: MIT press, 1994.

Kaboudan, Mark A. "Genetic programming prediction of stock prices." Computational Economics 16.3 (2000): 207-236.

Vasilakis, Georgios A., et al. "A genetic programming approach for EUR/USD exchange rate forecasting and trading." Computational economics 42.4 (2013): 415-431.

Cramer, G. M., R. A. Ford, and R. L. Hall. "Estimation of toxic hazard—a decision tree approach." Food and cosmetics toxicology 16.3 (1976): 255-276.

Mingers, John. "An empirical comparison of pruning methods for decision tree induction." Machine learning 4.2 (1989): 227-243.

Breiman, Leo. "Bagging predictors." Machine learning 24.2 (1996): 123-140.

Friedman, Jerome H., and Peter Hall. "On bagging and nonlinear estimation." Journal of statistical planning and inference 137.3

(2007): 669-683.
Bühlmann, Peter, and Bin Yu. "Analyzing bagging." The annals of Statistics 30.4 (2002): 927-961.
Freund, Yoav, Robert Schapire, and Naoki Abe. "A short introduction to boosting." Journal-Japanese Society For Artificial Intelligence 14.771-780 (1999): 1612.
Schapire, Robert E., and Yoram Singer. "Improved boosting algorithms using confidence-rated predictions." Machine learning 37.3 (1999): 297-336.
Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine." Annals of statistics (2001): 1189-1232.
Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M. and Ghemawat, S. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." arXiv preprint arXiv:1603.04467 (2016).
Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. and Desmaison, A. "Pytorch: An imperative style, high-performance deep learning library." Advances in neural information processing systems 32 (2019).
Topchy, Alexander, and William F. Punch. "Faster genetic programming based on local gradient search of numeric leaf values." Proceedings of the genetic and evolutionary computation conference (GECCO-2001). Vol. 155162. San Francisco, CA: Morgan Kaufmann, (2001).
Boritz, J. Efrim, and Won Gyun No. "The quality of interactive data: XBRL versus Compustat, Yahoo Finance, and Google Finance." Yahoo Finance, and Google Finance (April 18, 2013) (2013).

Abstract

Stock Market Prediction via Gradient Descent Genetic Programming

HWANG Soonyong
Computer Science and Engineering
The Graduate School
Seoul National University

Stock market prediction problems have been widely studied for some decades and have been recently researched by machine learning algorithms. Stock data can be easily included in noise data, and even in the same situation, it may cause different results due to external influences. Machine learning algorithms have been in the spotlight for reducing noise in the stock market and improving the accuracy of the stock market prediction. In particular, the genetic programming was proposed to apply various objectives. For this reason, genetic programming has been frequently used in stock market prediction or execution algorithms. This paper analyzed the genetic programming for simulating stock markets in each country. Genetic programming has the advantage of being able to explore various problem structures, but it has the disadvantage of taking a long time to explore large problem space. To alleviate this disadvantage, this paper calculates gradient of each node in the tree and replaces the node to another subtree by using the gradient. In addition, LRU cache system was proposed to maintain fast

searching algorithm. Stock market data from Korea, North America, and Japan were collected to evaluate the gradient descent genetic programming methodology. In this paper, the experiments between the gradient descent genetic programming and naïve genetic programming for each stock market showed that the gradient descent genetic programming methodology explores faster than in the most cases. Compared to the other machine learning algorithms, different aspects could be seen depending on the stock market, and high accuracy could be obtained overall. In addition, this paper proposes back-testing procedure that is easily compatible with genetic programming to enable continuous evaluation and development.

Keywords : Genetic programming, Machine learning, Stock market prediction, Gradient descent genetic programming
Student Number : 2020-23509