QUANTITATIVE ISSUE

2026. 2. 2

MACRO팀

김동영, CFA Quant Analyst
dy76.kim@samsung.com

리서치센터 리포트
바로가기

퀀트 모델링 A to Z

(9) LSTM 모델의 이해

- LSTM의 원리에 대한 상세한 설명과, 투자 관점에서의 모델 활용법 등을 정리함

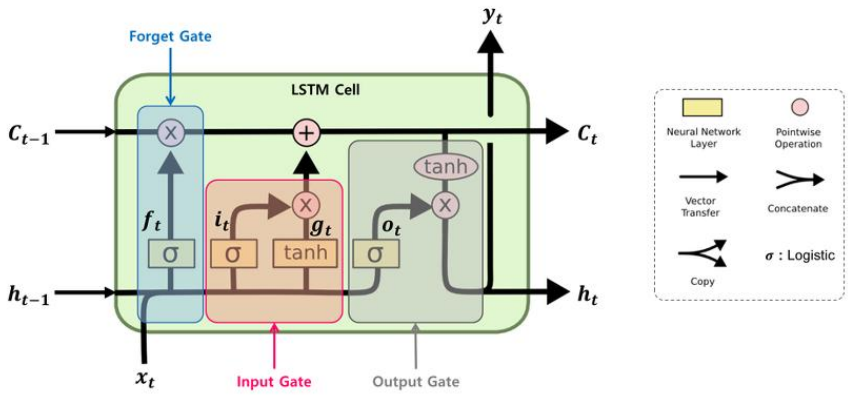
WHAT'S THE STORY?

LSTM(Long Short-Term Memory)은 전통적인 RNN 모델이 지닌 기울기 소멸 문제를 구조적으로 해결한 순환 신경망으로, 장기 기억과 시간 의존성이 중요한 시계열 데이터 분석에서 핵심적인 역할을 수행해 왔다. 특히 금융 시장과 같이 국면 전환과 비선형 패턴이 빈번하게 나타나는 환경에서 LSTM은 과거 정보를 선택적으로 기억하고, 필요에 따라 잊을 수 있는 유연한 구조를 제공한다.

본 리포트는 LSTM의 내부 동작 원리를 수식과 구조 중심으로 체계적으로 정리하고, 각 게이트와 셀 상태가 어떤 의미를 가지는지를 직관적으로 설명한다.

또한, LSTM을 주식 투자 모델에 적용할 때의 활용 가능성과 한계를 투자 관점에서 분석한다.

LSTM(Long Short-Term Memory) 모델



자료: https://velog.io/@yuns_u

삼성증권

I. LSTM 모델

본 리포트는, 순환/재귀 신경망에서 가장 중요한 모델 중 하나인 LSTM(Long Short-Term Memory)에 대해서, 정확한 이해 및 원리적인 구조를 설명하고자 한다. 이를 통해 금융 투자 모델에 LSTM을 접목하는 아이디어에 대해서도 논의할 계획이다.

딥러닝 모델의 기초 이해 및 공통 사용법에 대해서는, 26/1/2일 자 "퀀트 모델링 A to Z - (6) MLP(Multi-Layer Perceptron) AI 투자 모델" 리포트의 선행 확인을 추천한다. RNN 계열 모델에 대한 기본적인 이해는, 26/1/21일 자 "퀀트 모델링 A to Z - (8) RNN(순환 신경망) AI 투자 모델" 리포트에 정리되어 있다.

여기서 사용 및 설명하는 코드의 프로그래밍 환경은 다음과 같다.

```
Python 3.12.7
tensorflow 2.20.0
tensorflow.keras 3.12.0
```

LSTM 모델의 이해

LSTM(Long Short-Term Memory)은 기본적인 RNN 모델이 보이는 "과거 데이터를 잘 기억하지 못하는 문제(vanishing gradient problem, 기울기 소멸 문제)"를 해결할 수 있는 특별한 순환 신경망 모델이다. LSTM은 Hochreiter & Schmidhuber의 1997년 논문에서 처음 소개되었으며, 뛰어난 장기 기억 능력으로 인해 아직도 많이 사용되는 신경망 모델이다.

vanishing gradient problem은, 전통 RNN에서 순차 데이터가 길어질수록 초기 시점의 옛 정보가 사라지는 문제를 말한다. 직전 리포트에 나온 Simple RNN의 수식을 보면, $\boldsymbol{h_t}$를 계산할 때, $\boldsymbol{h_{t-1}}$에 가중치 행렬 $\boldsymbol{U}$를 곱하고 여기에 활성화 함수도 또 적용된다. timestep이 계속 올라가게 되면, 초기의 $\boldsymbol{h}(\boldsymbol{h_0}, \boldsymbol{h_1}$ 같은) 정보들은 가중치 행렬 $\boldsymbol{U}$가 계속 누적해서 곱해지면서 해당 정보(수치)가 0으로 축소되는 현상을 겪는다(아니면, 폭주하는 경우도 발생). 즉, 순차 데이터가 길어질수록 초기의 정보가 거의 사라지고 기억하지 못하는 문제가 생기게 된다.

이의 해결을 위해, LSTM은 장기 기억을 담당하는 별도의 셀 상태(cell state)를 만들고 활용하여, 유연하게 장기 기억이 가능하도록 네트워크를 구성했다.

LSTM을 설명하는 데 도움을 주기 위해, 직전 리포트에 있었던 SimpleRNN 모델의 수식 표현을 다시 한번 가져왔다.

참고) SimpleRNN 계산의 수식 (수학적 표기법)

[가정]

입력 데이터가 d 차원의 열 벡터임. $\boldsymbol{x} = [x_1\ x_2\ \dots\ x_d]^T$.

이때 t 시점별로 데이터가 존재함. $\boldsymbol{x_t} = [x_{1t}\ x_{2t}\ \dots\ x_{dt}]^T$

혹은, "입력 벡터 $\boldsymbol{x_t} \in \mathbb{R}^{\boldsymbol{d}}$"로 표기 ($\boldsymbol{x_t}$는 실수 d개로 이루어진 벡터라는 뜻)

은닉층 출력 데이터가 m 차원의 열 벡터임. $\boldsymbol{h} = [h_1\ h_2\ \dots\ h_m]^T$

이때 t 시점별로 데이터가 존재함. $\boldsymbol{h_t} = [h_{1,t}\ h_{2,t}\ \dots\ h_{m,t}]^T$

혹은, "RNN 유닛 출력 벡터 $\boldsymbol{h_t} \in \mathbb{R}^m$"로 표기 ($\boldsymbol{h_t}$는 실수 m개로 이루어진 벡터라는 뜻)

시점별 데이터를 전부 가지고 있는 "하나의 샘플"을, 하나의 시퀀스(sequence)라고 말함.

RNN은 입력과 출력을 시퀀스 단위로 처리함. 은닉층 출력과 입력 데이터 간의 관계식은 다음과 같음.

[SimpleRNN의 수식]

$$\boldsymbol{h_t} = f(\boldsymbol{W} \cdot \boldsymbol{x_t} + \boldsymbol{U} \cdot \boldsymbol{h_{t-1}} + \boldsymbol{b})$$

이때, $\boldsymbol{W}$는 $\boldsymbol{x}$와 $\boldsymbol{h}$의 관계를 나타내는 시불변 가중치 행렬.

$\boldsymbol{U}$는 $\boldsymbol{h_{t-1}}$과 $\boldsymbol{h_t}$의 관계를 나타내는 시불변 가중치 행렬.

$\boldsymbol{b}$는 편향 벡터.

f는 활성화 함수(tanh 함수, 시그모이드 함수 같은 미리 정한 임의의 비선형 함수).

자세히 보면, 여기서는 $\boldsymbol{h_t}$의 수식이 $\boldsymbol{h_t} = f(\boldsymbol{W} \cdot \boldsymbol{x_t} + \boldsymbol{U} \cdot \boldsymbol{h_{t-1}} + \boldsymbol{b})$로 되어 있다. 직전 리포트 상에서는 $\boldsymbol{h_t}$의 수식 표현이 $\boldsymbol{h_t} = f(\boldsymbol{x_t} \cdot \boldsymbol{W} + \boldsymbol{h_{t-1}} \cdot \boldsymbol{U} + \boldsymbol{b})$로 되어 있었다($\boldsymbol{W}$ 와 $\boldsymbol{U}$ 의 곱해지는 위치가 바뀜).

이 둘의 차이는, 수식 표기 시 "수학적 표기법"을 쓸 건지 "컴퓨팅 연산 기준 표기법"을 쓸 건지에 따라서 변환 가능한 부분이다(아래 표 참고). 본 자료에서는 앞으로 모델 설명 시 수학적 표기법을 기준으로 서술하겠다.

수학적 표기법:

전통적으로 수학에서는 벡터를 표기할 때 열 벡터(column 방식)로 표기함. 수학적 표기법에서는. 데이터에 대해서 "한 개 샘플" 상에서의 $\boldsymbol{x}$와 $\boldsymbol{h}$의 계산만을 보여줌. 따라서, 입출력 데이터가 열 벡터로 표기되고, 가중치 행렬과의 곱은, $\boldsymbol{U} \cdot \boldsymbol{h_{t-1}}$과 같이 가중치가 앞에 오는 모습을 띰.

컴퓨팅 연산 기준 표기법:

실제 딥러닝의 컴퓨팅에서는 입출력 데이터를 사용할 때 행 벡터(row 방식)와 batch 처리(여러 샘플 데이터의 처리)를 우선으로 하는 구조임. 일괄적인 계산을 위해서, 입출력 데이터를 행 벡터로 처리함. 실제 연산 시에는, 여러 개의 샘플 데이터를 같이 넣어서, 입력 데이터를 shape=(n_samples, d) 형태의 행렬로 처리함.

따라서, 행 벡터 표기 시에는, 수식에서 $\boldsymbol{h_{t-1}} \cdot \boldsymbol{U}$와 같이 가중치가 뒤에 오는 모습을 띰.

→ 어떤 표기법을 쓸지에 따라서, 수식에서의 위치만 살짝 바뀌는 차이가 존재함.

여기서 하나 기억할 것은, 딥러닝 수식에서 $\boldsymbol{W}$(weight의 약자)와 $\boldsymbol{U}$는 대부분 "훈련 중에 학습해야 하는 가중치 행렬 파라미터"를 의미한다. MLP에서의 $\boldsymbol{W}$, CNN에서의 $\boldsymbol{K}$, SimpleRNN에서의 $\boldsymbol{W}$와 $\boldsymbol{U}$가 모두 가중치 행렬에 해당한다. 이 가중치 행렬은, 값이 미리 정해지는 것이 아니고 X와 y의 데이터를 가지고 계속 학습하면서 데이터에 따라서 수치가 정해지는 방식이다. 쉽게 말해, 이는 "1차 단순 회귀 분석에서의 $\boldsymbol{x}$의 기울기 계수 $\boldsymbol{a}$"와 같은 역할이다. 또한 딥러닝 수식에서 $\boldsymbol{b}$(bias의 약자)도 항상 "훈련 중에 학습해야 하는 바이어스 파라미터"를 의미한다. 쉽게 말해, "1차 단순 회귀 분석에서의 $\boldsymbol{x}$의 절편 계수 $\boldsymbol{b}$"와 완전히 같은 역할이다.

앞으로 나올 수식에서도 $\boldsymbol{W}, \boldsymbol{U}, \boldsymbol{b}$는 모두 학습되는(trainable) 가중치 행렬 및 바이어스의 파라미터들을 의미한다. 이 변수들은 모두, 훈련 데이터를 통해 학습되면서 데이터에서 나타나는 패턴을 저장하는 용도의 변수다.

이제 본격적으로 LSTM 모델을 설명하려고 하는데, 이때 회로도 같은 그래픽적 표현보다는 수식을 통해 개념을 이해하는 것이 훨씬 편리하다. 우선 LSTM 모델 전체의 원론적인 수식 표현은 다음 페이지의 표와 같다.

LSTM 모델의 수식 (수학적 표기법): 원론적인 표기법

[변수 및 설명]

아래첨자 t는 timestep 상에서 t 시점을 의미함.

위첨자 d는 입력 벡터의 차원을 의미함. 위첨자 m은 LSTM 유닛 출력 벡터의 차원을 각각 의미함.

$\boldsymbol{x_t} \in \mathbb{R}^{\boldsymbol{d}}$: 입력 벡터. d차원

$\boldsymbol{h_t} \in \mathbb{R}^{m}$: LSTM 유닛 출력 벡터. m차원

$\boldsymbol{f_t} \in (0,1)^{m}$: forget gate. ($(0,1)$의 수치 범위를 가지는 m차원 벡터)

$\boldsymbol{i_t} \in (0,1)^{m}$: input gate.

$\boldsymbol{o_t} \in (0,1)^{m}$: output gate.

$\boldsymbol{c_t} \in \mathbb{R}^{m}$: cell state.

$\tilde{\boldsymbol{c}}_{\boldsymbol{t}} \in (-1,1)^{m}$: cell candidate. ($(-1,1)$의 수치 범위를 가지는 m차원 벡터) (c 틸드로 발음)

$\boldsymbol{W} \in \mathbb{R}^{\boldsymbol{m}\times\boldsymbol{d}}, \boldsymbol{U} \in \mathbb{R}^{\boldsymbol{m}\times\boldsymbol{m}}$ and $\boldsymbol{b} \in \mathbb{R}^{\boldsymbol{m}}$: 훈련 중 학습해야 하는 가중치 행렬 및 바이어스 벡터 파라미터들

$\sigma(x)$: 시그모이드 함수, $tanh(x)$: tanh(하이퍼볼릭 탄젠트) 함수, $\odot$: 원소별(element-wise) 곱 연산

[모델 수식]

$$\boldsymbol{f_t} = \sigma(\boldsymbol{W_f x_t} + \boldsymbol{U_f h_{t-1}} + \boldsymbol{b_f})$$

$$\boldsymbol{i_t} = \sigma(\boldsymbol{W_i x_t} + \boldsymbol{U_i h_{t-1}} + \boldsymbol{b_i})$$

$$\boldsymbol{o_t} = \sigma(\boldsymbol{W_o x_t} + \boldsymbol{U_o h_{t-1}} + \boldsymbol{b_o})$$

$$\tilde{\boldsymbol{c}}_{\boldsymbol{t}} = tanh(\boldsymbol{W_c x_t} + \boldsymbol{U_c h_{t-1}} + \boldsymbol{b_c})$$

$$\boldsymbol{c_t} = \boldsymbol{f_t} \odot \boldsymbol{c_{t-1}} + \boldsymbol{i_t} \odot \tilde{\boldsymbol{c}}_{\boldsymbol{t}}$$

$$\boldsymbol{h_t} = \boldsymbol{o_t} \odot tanh\ (\boldsymbol{c_t})$$

여기서, 초기 값은 다음과 같음. $\boldsymbol{c_0} = 0, \boldsymbol{h_0} = 0$

위의 수식에는 수학적인 간결성과 엄밀성이 있다. 그러나, 처음 이해하기에는 설명이 매우 부족한 편이다.

따라서, 다음에 나올 "설명 버전"으로 서술하는 것이 유용할 것으로 판단한다.

LSTM 모델에 대한 상세한 설명 버전:

우선, 각각의 사용 변수에 대해 좀 더 설명적인 이름을 사용하겠다. forget gate라고 하는 것은, 직관적으로 말하면 "forget weight"를 의미한다. 이때의 weight(가중치)는 말 그대로, 원 수치에 곱해서 얼마만큼 반영할지/남길지를 정하는 수치 하나를 말한다. 이는, 주식에 있어서의 '시가총액 가중 배당 수익률' 계산, 확률의 경우의 수에서의 케이스별 가중치 등에 쓰이는 "가중치"와 완벽히 같은 의미에 해당한다. 마찬가지로 input gate, output gate도 실제로는 input weight, output weight의 역할을 하게 된다. 따라서, 각각의 변수명을 다음과 같이 바꿔보겠다.

(쉬운 설명 #1)

[변수 및 설명]

아래첨자 t는 timestep 상에서 t 시점을 의미함.

위첨자 d는 입력 벡터의 차원을 의미함. 위첨자 m은 LSTM 유닛 출력 벡터의 차원을 각각 의미함.

$\boldsymbol{x_t}$: 입력 벡터(d차원)

$\boldsymbol{h_t}$: LSTM 유닛 출력 벡터(m차원)

$\boldsymbol{forget_wgt_t}$: forget weight. $(0, 1)$의 수치 범위를 가지는 m차원 벡터)

$\boldsymbol{input_wgt_t}$: input weight. $(0, 1)$의 수치 범위를 가지는 m차원 벡터)

$\boldsymbol{output_wgt_t}$: output weight. $(0, 1)$의 수치 범위를 가지는 m차원 벡터)

$\boldsymbol{cell_state_t}$: "장기 기억을 저장하는" cell state(m차원 벡터).

$\boldsymbol{temp_cell_state_t}$: 신규 입력용의, 임시 cell state(m차원 벡터). (cell candidate)

$\boldsymbol{W} \in \mathbb{R}^{\boldsymbol{m}\times\boldsymbol{d}}, \boldsymbol{U} \in \mathbb{R}^{\boldsymbol{m}\times\boldsymbol{m}}$ and $\boldsymbol{b} \in \mathbb{R}^{\boldsymbol{m}}$: 훈련 중 학습해야 하는 가중치 행렬 및 바이어스 벡터 파라미터들

$\sigma(x)$: 시그모이드 함수, $tanh(x)$: tanh(하이퍼볼릭 탄젠트) 함수, $\odot$: 원소별(element-wise) 곱 연산

LSTM에서 가장 큰 특징은, 순환/재귀 계산을 반복할 때 $\boldsymbol{h_t}$만 쓰는 것이 아니고, 장기 기억을 담당하는 별도의 내부 $\boldsymbol{cell_state_t}$를 만든다는 것이다. 약간 비슷한 예시를 든다면, Hidden Markov Model(은닉 마코프 모델, HMM)에서 1) 눈에 보이는 정보와 2) 관측할 수 없는 내부의 "은닉 상태" 정보를 따로 관리하는 것과 비슷하다고 할 수 있다. HMM에서의 은닉 상태(hidden state)의 역할을, LSTM 모델에서는 cell state가 담당한다.

비교해서 보자면, SimpleRNN에서는 RNN 유닛 상의 $\boldsymbol{h_t}$만 관리하고 있기 때문에, 이 변수는 timestep이 흘러갈수록, 계속 연산이 곱해지며 초기 정보가 계속 흐려진다. 그러나, LSTM에서는 별도의 장기 기억용 cell state 변수를 따로 준비한다. 그리고, 매 timestep 상의 계산상에서는 새로운 정보를 기존 cell state에 일부만 "더하는 식"으로 업데이트가 진행된다. 따라서, "timestep의 초기에 저장된 cell state"라 하더라도, timestep 별 순환/반복 계산이 계속되더라도 계속해서 정보의 유지가 가능하다.

(#2)

[수식]

파란색 변수는 실제 "정보"가 담긴 변수를 의미함. (개, 고양이, 주가 수익률 등의 진짜 정보들)

빨간색 변수는, "정보"를 얼마만큼 반영할지를 결정하는 "가중치" 수치를 의미함. 0~1의 값을 나타냄.

$$cell_state_t = forget_wgt_t \odot cell_state_{t-1} + input_wgt_t \odot temp_cell_state_t$$

이때, ⊙는 원소별(element-wise) 곱 연산

※ data shape: (m,) = (m,)⊙(m,) + (m,)⊙(m,)

우선, 여기서 ⊙ 기호는 원소별(element-wise) 곱 연산자로, 각 벡터/행렬의 원소끼리 곱하는 연산이다. 전문적인 용어로 Hadamard product(하다마르 곱)라고 한다. Python에서 np.array의 A, B 행렬이 있다고 할 때, ⊙는 "A * B" 혹은 "np.multiply(A, B)" 코드의 의미다. 반면, 행렬 곱은 Python에서 "A @ B" 혹은 "np.dot(A, B)" 코드로 표현된다.

위 수식은, timestep t 시점에서 장기 기억 정보인 cell state를 업데이트하는 방법을 나타낸 것이다.

$cell_state_t$ 계산 시에는, 1) 우선 $cell_state_{t-1}$정보에 "망각 가중치"만큼 비중을 적용하고, 2) t 시점의 신규 셀용 정보를 의미하는 $temp_cell_state_t$정보(물론 x_t와 h_{t-1}에 의해서 만들어짐)에 "입력 가중치"만큼 비중을 적용하여, 3)이 두 수치를 더한 값이 입력된다. 즉, t 시점의 장기 기억 $cell_state_t$는 t-1 시점의 장기 기억 $cell_state_{t-1}$과 t 시점의 신규 정보 $temp_cell_state_t$의 가중 평균한(정확히는 가중 합의) 정보에 해당한다.

여기서, 모델상의 "실질적인 정보"가 담기는 변수는, $cell_state$ 와 $temp_cell_state$ 뿐이다. 반면, $forget_wgt$ 와 $input_wgt$는, 정보를 얼마만큼 반영할지를 결정하는, 단순한 수치 계수에 불과하다. 이 수치가 1로 가면 정보를 그대로 다 반영하는 것이고, 이 수치가 0으로 가면 정보를 거의 반영하지 않아서 지운다는 의미다.

한편, 최초 설정 시에 LSTM 유닛 출력 벡터의 차원을 m차원으로 정했기 때문에, $cell_state_t$ 와 $temp_cell_state_t$가 모두 m차원 벡터임을 기억할 필요가 있다. 예를 든다면 이 m차원 벡터는, m개의 서로 다른 의미/정보를 가지는 수치를 담는 벡터가 될 수도 있고, 자연어 처리에 있어서는 m차원으로 표현되는 문맥 벡터(혹은, 단어 비슷한) 정보가 될 수도 있다.

앞으로 보겠지만, $temp_cell_state_t$를 계산할 때는, 당연히 x_t와 h_{t-1}가 사용된다. 이때 신경망 모델 컨셉에 맞게 모두 가중치 행렬이 포함되어서 수치가 만들어지고, 이 가중치 행렬은 데이터 학습을 통해서 사후적으로 업데이트된다. 더 중요한 것은 $forget_wgt_t$ 등의 경우에도, 고정된 수치 계수가 아니라 입력 데이터값에 따라서 그에 맞춰서 수치 계수가 자동으로 결정되는, 데이터 학습을 반영하는 방식으로 만들어진다. 이에 대해서는 뒤에서 다시 다룬다.

(#3)

$$\boldsymbol{temp_cell_state_t} = tanh(\boldsymbol{W_c x_t} + \boldsymbol{U_c h_{t-1}} + \boldsymbol{b_c})$$

※ data shape: (m,) = (m, d) × (d,) + (m, m) × (m,) + (m,)

위 식은, t 시점의 최종 장기 기억인 $\boldsymbol{cell_state_t}$를 만들 때 일부가 반영되는, "신규 입력용" 임시 cell state를 만드는 수식을 설명한 것이다.

이 수식은 SimpleRNN 모델의 전체 수식 구조인 [$\boldsymbol{h_t} = f(\boldsymbol{W} \cdot \boldsymbol{x_t} + \boldsymbol{U} \cdot \boldsymbol{h_{t-1}} + \boldsymbol{b})$]와 상당히 비슷함을 알 수 있다. 이 수식은, 장기 기억인 cell state에 반영되는 "신규 입력" 정보는, t 시점의 입력 벡터 $\boldsymbol{x_t}$와 t-1 시점의 LSTM 출력 벡터 $\boldsymbol{h_{t-1}}$을 조합해서 만들어진다는 것을 뜻한다. MLP 방식이 아닌 순환 신경망이기 때문에, 직전 시점의 출력 벡터 정보 $\boldsymbol{h_{t-1}}$ 이 사용되는 것이 당연하다. 또 당연히 신경망 모델이므로, 이 벡터들 앞에 가중치 행렬이 사용되고, 데이터의 학습 및 역전파를 통해서 이 가중치 행렬이 데이터의 패턴을 익히게 되는 방식이다.

수식에서 $tanh()$ 부분은, 활성화 함수로 임의의 f 함수가 아닌 tanh 함수를 썼음을 명시적으로 표시한 것이다. 딥러닝의 가장 중요한 특징인 "비선형성"을 만들기 위해서, 신경망의 모든 계산에는 항상 활성화 함수가 붙게 된다. 순환 신경망 계열에서는, 출력 값이 0을 기준으로 대칭이고 출력 범위가 (-1, 1)로 제한되는 tanh 함수를 활성화 함수로 쓰는 것이 일반적이다. 따라서 여기의 tanh 적용도 상식적인 접근이다.

(#4)

$$\boldsymbol{forget_wgt_t} = \sigma(\boldsymbol{W_f x_t} + \boldsymbol{U_f h_{t-1}} + \boldsymbol{b_f})$$

이때, $\sigma()$는 시그모이드 함수.

※ data shape: (m,) = (m, d) × (d,) + (m, m) × (m,) + (m,)

위 식은, forget_weight 혹은 forget gate를 계산하는 산식이다. forget gate의 역할은, 장기 기억인 cell state에서 "t-1 시점의 정보를, t 시점의 정보에 얼마만큼 반영할지"를 결정하는 단순한 0~1 범위 수치 계수에 불과하다. forget_weight가 거의 1이면, t-1 시점의 cell state 정보를 t 시점의 cell state에도 거의 1만큼 다 반영하겠다는 뜻이다. forget_weight가 거의 0이면, t-1 시점의 cell state 정보를 거의 기억하지 않겠다는 뜻이다. (정확히는 그 의미가 dont_forget or forget의 weight라고 할 수 있음.)

중요한 것은 이 forget_weight가 고정된 값이 아니고, $\boldsymbol{x_t}$와 $\boldsymbol{h_{t-1}}$에 따라서 실시간으로 조절된다는 것이다. 신경망 모델의 일반적인 구성처럼 $\boldsymbol{W_f}$와 $\boldsymbol{U_f}$ 가중치 행렬을 붙임으로써, 과거 데이터 패턴에 따라 가중치 행렬이 학습을 하게 되고, $\boldsymbol{forget_wgt_t}$는 $\boldsymbol{x_t}$와 $\boldsymbol{h_{t-1}}$에 따라서 가변적으로 변화한다. 순환 신경망이므로, 모든 의사 결정에는 $\boldsymbol{h_{t-1}}$정보도 항상 같이 들어간다.

예를 들어보자. 자연어 처리에서, $\boldsymbol{x_2}$에 "나는 서울에 계속 살고 있었다"란 정보가 있고, cell state가 내가 살고 있는 도시의 정보를 기억한다고 하자. 이 경우, $\boldsymbol{forget_wgt_2}$는, $\boldsymbol{x_2}$의 "계속 살고 있었다" 정보를 고려해서 1에 가까운 값으로 계산되고, $\boldsymbol{cell_state_2}$는 직전 t=1 시점의 정보를 그대로 유지하는 것이 가능하다. 만약, $\boldsymbol{x_3}$에 "부산으로 이사했다"란 정보가 들어온다고 하면, $\boldsymbol{forget_wgt_3}$는, $\boldsymbol{x_2}$의 "이사했다" 정보를 고려해서 0에 가까운 값으로 계산될 수 있다. 이때는 $\boldsymbol{cell_state_3}$는 직전 t=2 시점의 정보를 거의 버리고, "부산"이란 정보가 담긴 신규 입력 데이터 위주로 교체되는 변화가 발생한다.

혹은, 금융 시계열 정보를 사용하는 모델을 가정해 보자. 예를 들어 $\boldsymbol{x_t}$가 환율 변화율을 의미하고, $\boldsymbol{x_t}$의 급격한 스파이크가 있을 때, $\boldsymbol{forget_wgt_t}$가 0에 근접하는 식으로 모델이 학습될 수 있다. 이는, 환율의 급격한 스파이크(급등, 급락)가 있을 때, 장기 기억 정보(cell state)에 현재의 환율 움직임만 기억하고 스파이크 이전의 상황은 잊어버리라는 뜻이 된다. 즉, 금융 시장의 regime change 상황을, 신경망 모델이 자연스럽게 학습하는 것이 가능하다.

즉, LSTM에서 장기 기억 정보를 얼마나 기억할지 혹은 잊어버릴지는, 과거 패턴과 입력 데이터에 따라서 항상 가변되는 방식으로 작동한다.

(#5)

$$\boldsymbol{input_wgt_t} = \sigma(\boldsymbol{W_i x_t} + \boldsymbol{U_i h_{t-1}} + \boldsymbol{b_i})$$

※ data shape: (m,) = (m, d) × (d,) + (m, m) × (m,) + (m,)

위 식은, input_weight 혹은 input gate를 계산하는 산식이다. input gate의 역할은, 장기 기억인 cell_state의 t 시점 정보 생성 시 "신규 입력" 정보를 얼마만큼 반영할지를 결정하는 단순한 0~1 범위 수치 계수다.

$\boldsymbol{cell_state_t}$ 의 계산 시 사용되는 "신규 입력" 정보는, 앞에서 봤듯이 ($\boldsymbol{x_t, h_{t-1}}$ 로 정해지는) $\boldsymbol{temp_cell_state_t}$를 말한다. input_weight는 0~1 범위를 가지고, 이 "신규 입력" 정보의 반영 규모를 결정한다.

즉, 개념상으로 $\boldsymbol{cell_state_t}$는 $\boldsymbol{cell_state_{t-1}}$와 $\boldsymbol{temp_cell_state_t}$의 가중 평균 혹은 가중 합의 정보다(아래 식 참고).

$$\boldsymbol{cell_state_t} = f(0\sim1\ 범위) \cdot \boldsymbol{cell_state_{t-1}} + i(0\sim1\ 범위) \cdot \boldsymbol{temp_cell_state_t}$$

좀 더 절약적인 모델이라면 가중치를 각각 f와 $1-f$로 표현하는 방식을 썼을 수도 있겠지만, 일단 LSTM에서는 f와 i 계수를 각각 따로 관리하는 방식을 썼다.

중요한 것은 이 input_weight도 고정된 값이 아니고, $\boldsymbol{x_t}$와 $\boldsymbol{h_{t-1}}$에 따라서 실시간으로 조절된다는 것이다. input_weight는 역전파에 의해 데이터 패턴이 학습되고, 입력 데이터 등에 따라서 가변적으로 결정된다.

(#6)

$$\boldsymbol{h_t} = \boldsymbol{output_wgt_t} \odot tanh\ (\boldsymbol{cell_state_t})$$

※ data shape: (m,) = (m,) ⊙ (m,)

위 수식은, timestep t 시점의 LSTM 유닛 출력 벡터($\boldsymbol{h_t}$)와 t 시점의 장기 기억 정보 $\boldsymbol{cell_state_t}$간의 관계를 나타낸 식이다.

LSTM 모델은 1) 출력 벡터와 2) 장기 기억용 cell state를 따로 가지는 것이 가장 특징이라고 언급했었다. 즉, $\boldsymbol{h_t}$와 정보 $\boldsymbol{cell_state_t}$는 같은 정보가 아니고, $\boldsymbol{h_t}$를 계산할 때는 $\boldsymbol{cell_state_t}$에다가 변환 과정을 한 번 더 거치게 된다. 위 산식은 그 변환 과정을 의미한다.

우선, $\boldsymbol{h_t}$계산 과정의 처음에, $tanh\ (\boldsymbol{cell_state_t})$를 먼저 계산한다. $\boldsymbol{cell_state_t}$와 $\boldsymbol{h_t}$가 각각의 정보(노드 개념)를 의미하므로 정보 이동 시 활성화 함수를 적용하는 것은 일반적인 접근법이다.

그리고 나서, $\boldsymbol{output_wgt_t}$ 수치가 내부적으로 계산된 다음에, 이 둘이 원소 기준의 곱으로 다시 계산된다. $\boldsymbol{output_wgt_t}$는, $tanh\ (\boldsymbol{cell_state_t})$로 1차 계산된 정보를 얼마만큼 최종 출력 벡터에 반영할지를 결정하는, 0~1의 수치 계수를 의미한다. 앞서 본 forget_weight, input_weight와 같은 역할을 하는 것이다. $\boldsymbol{output_wgt_t}$가 1에 가까우면 $\boldsymbol{h_t}$에 "$tanh\ (\boldsymbol{cell_state_t})$" 수치가 거의 그대로 들어가고, $\boldsymbol{output_wgt_t}$가 0에 가까우면 $\boldsymbol{h_t}$에는 0에 가깝게 흐려진 정보가 들어간다.

$\boldsymbol{output_wgt_t}$도, 고정된 수치 계수가 아니라, 입력 데이터값에 따라서 그에 맞춰서 수치 계수가 자동으로 결정되는 데이터 학습을 반영하는 방식으로 만들어진다.

(#7)

$$\boldsymbol{output_wgt_t} = \sigma(\boldsymbol{W_o x_t} + \boldsymbol{U_o h_{t-1}} + \boldsymbol{b_o})$$

※ data shape: (m,) = (m, d) × (d,) + (m, m) × (m,) + (m,)

위 식은, output_weight 혹은 output gate를 계산하는 산식이다. 다른 게이트들과 마찬가지로, output_weight도 고정된 값이 아니고, $\boldsymbol{x_t}$와 $\boldsymbol{h_{t-1}}$에 따라서 실시간으로 조절된다. output_weight는 역전파에 의해 데이터 패턴이 학습되고, 입력 데이터 등에 따라서 가변적으로 결정된다.

LSTM 모델은, 장기 기억을 유지 및 출력함에 있어서 regime change와 같이 특정 이벤트 후의 급변동 같은 부분을 모델이 잘 학습하도록, 패턴을 학습하는 구조가 뛰어나다.

이상으로 LSTM 계산 구조의 모든 수식이 다 정리되었다. 산식 #1~#7을 모두 모아서 하나의 표로 정리하면 다음과 같다.

LSTM 모델의 구조: 설명 버전의 표기법

[변수 및 설명]

아래첨자 t는 timestep 상에서 t 시점을 의미함.
위첨자 d는 입력 벡터의 차원을 의미함. 위첨자 m은 LSTM 유닛 출력 벡터의 차원을 각각 의미함.
$\boldsymbol{x_t}$: 입력 벡터(d차원)
$\boldsymbol{h_t}$: LSTM 유닛 출력 벡터(m차원)
$\boldsymbol{forget_wgt_t}$: forget weight. ($(0,1)$의 수치 범위를 가지는 m차원 벡터)
$\boldsymbol{input_wgt_t}$: input weight. ($(0,1)$의 수치 범위를 가지는 m차원 벡터)
$\boldsymbol{output_wgt_t}$: output weight. ($(0,1)$의 수치 범위를 가지는 m차원 벡터)
$\boldsymbol{cell_state_t}$: "장기 기억을 저장하는" cell state(m차원 벡터).
$\boldsymbol{temp_cell_state_t}$: 신규 입력용의, 임시 cell state(m차원 벡터). (cell candidate)
$\boldsymbol{W} \in \mathbb{R}^{m \times d}, \boldsymbol{U} \in \mathbb{R}^{m \times m}$ and $\boldsymbol{b} \in \mathbb{R}^{m}$: 훈련 중 학습해야 하는 가중치 행렬 및 바이어스 벡터 파라미터들
$\sigma(x)$: 시그모이드 함수, $tanh(x)$: tanh(하이퍼볼릭 탄젠트) 함수, $\odot$: 원소별(element-wise) 곱 연산

[모델 수식]

(파란색 변수는 실제 "정보"가 담긴 변수, 빨간색 변수는 정보를 얼마만큼 반영할지 결정하는 "가중치" 수치)

$$\boldsymbol{cell_state_t} = \boldsymbol{forget_wgt_t} \odot \boldsymbol{cell_state_{t-1}} + \boldsymbol{input_wgt_t} \odot \boldsymbol{temp_cell_state_t}$$

$$\boldsymbol{temp_cell_state_t} = tanh(\boldsymbol{W_c x_t} + \boldsymbol{U_c h_{t-1}} + \boldsymbol{b_c})$$

$$\boldsymbol{forget_wgt_t} = \sigma(\boldsymbol{W_f x_t} + \boldsymbol{U_f h_{t-1}} + \boldsymbol{b_f})$$

$$\boldsymbol{input_wgt_t} = \sigma(\boldsymbol{W_i x_t} + \boldsymbol{U_i h_{t-1}} + \boldsymbol{b_i})$$

$$\boldsymbol{h_t} = \boldsymbol{output_wgt_t} \odot tanh\ (\boldsymbol{cell_state_t})$$

$$\boldsymbol{output_wgt_t} = \sigma(\boldsymbol{W_o x_t} + \boldsymbol{U_o h_{t-1}} + \boldsymbol{b_o})$$

여기서, 초기 값은 다음과 같음. $\boldsymbol{cell_state_0} = 0, \boldsymbol{h_0} = 0$

여기서, $\boldsymbol{temp_cell_state_t}$뿐 아니라, 3개의 weight 변수(gate 변수)도 $\boldsymbol{x_t}$와 $\boldsymbol{h_{t-1}}$에 따라서 수치가 동적으로 결정되는 구조다.

LSTM의 파라미터 수

입력이 d차원 벡터고, 출력 벡터가 m차원 벡터일 때, SimpleRNN에서는 파라미터 수가 $(d + m + 1) \cdot m$였다. 이는, SimpleRNN에서는 $\boldsymbol{W}$($m \times d$ 행렬)와 $\boldsymbol{U}$($m \times m$ 행렬)와 $\boldsymbol{b}$가 쓰었기 때문이다.

LSTM의 구조를 잘 보면, SimpleRNN 같은 구조가 4번 사용된다. $\boldsymbol{temp_cell_state_t}$와 $\boldsymbol{forget_wgt_t}$, $\boldsymbol{input_wgt_t}$, $\boldsymbol{output_wgt_t}$가 모두 SimpleRNN과 비슷한 구조다.

따라서, LSTM에는 파라미터 수가 $4 \cdot (d + m + 1) \cdot m$이 된다. 1) $\boldsymbol{W_c}, \boldsymbol{W_f}, \boldsymbol{W_i}, \boldsymbol{W_o}$와 2) $\boldsymbol{U_c}, \boldsymbol{U_f}, \boldsymbol{U_i}, \boldsymbol{U_o}$와 3) $\boldsymbol{b_c}, \boldsymbol{b_f}, \boldsymbol{b_i}, \boldsymbol{b_o}$가 모두 학습 파라미터들이다.

(참고로, cell state는 기억하는 메모리 공간이지, 학습의 대상이 되는 파라미터가 아님.)

II. LSTM 사용법

keras에서는 LSTM 계층에 대해서 keras.layers.LSTM 클래스를 제공한다.

LSTM의 선언 문법은 다음과 같다.

```
keras.layers.LSTM(
    units,
    return_sequences=False,
    return_state=False,
    ***
)
```

우선 LSTM 계층의 입력 데이터 구조는, SimpleRNN에서와 마찬가지로 (n_samples, timesteps, n_features)의 3차원 텐서 형태다. LSTM 같은 순환 신경망 모델은, 하나의 시퀀스(한 개 샘플) 내에 timestep별로 순환 계산을 하기 때문에 이런 3차원 배열 구조가 필요하다.

LSTM 선언 문법에서 units는 LSTM의 노드 수, 즉 출력 벡터의 크기(차원 수)를 지정하는 변수다. units를 늘릴수록 복잡한 패턴을 탐지할 수도 있으나, 과적합의 위험도 늘어날 수 있다.

LSTM 선언문의 return_sequences 옵션은 RNN 레이어의 출력을 1) 마지막 timestep의 은닉층 노드 값($\boldsymbol{h_T}$)만 할지, 2) 모든 timestep의 은닉층 노드 값($\boldsymbol{h_t}$)을 모두 출력할지를 정하는 옵션이다. 이 또한 SimpleRNN 때의 설명과 동일하다. return_sequences=False의 디폴트 옵션을 사용하면, 마지막에 생성된 $\boldsymbol{h_T}$만 다음 층으로 출력하는 "many-to-one" 구조가 된다. 이 경우, 출력 데이터의 shape는 (n_samples, units)가 된다. return_sequences=True의 옵션을 사용하면, 매번 timestep의 $\boldsymbol{h_i}$를 다음 층으로 계속 출력하는 "many-to-many" 구조가 된다. 이 경우, 출력 데이터의 shape는 (n_samples, timesteps, units)가 된다.

return_state 옵션은 LSTM의 리턴 값을 단순 출력만 할지, 아니면 cell_state 정보까지 모두 출력할지 결정하는 옵션이다.

예를 들어, return_sequences=True와 return_state =True 옵션을 다음처럼 사용했다고 하자.

```
outputs, h_last, c_last = LSTM(units,
                               return_sequences=True,
                               return_state=True,
                               ) (X)
```

이때의 출력 변수들은 다음의 의미를 가진다.

변수명	의미	shape
outputs	모든 시점의 hidden state	(n_samples, timesteps, units)
h_last	마지막 hidden state	(n_samples, units)
c_last	마지막 cell state	(n_samples, units)

그리고, LSTM layer에서 가중치(파라미터) 정보를 실제 보는 방법은, 다른 layer 클래스와 비슷하게 get_weights() 메서드를 사용한다. 이때 lstm.get_weights()에서 리턴하는 데이터는 weights = [W, U, b]의 구조로 되어 있다. 여기서 W는 각 게이트에서의 W 행렬 정보가 모아져 있고, U는 각 게이트에서의 U 행렬 정보가 순서대로 들어있다. b에는 각 게이트에서의 bias 정보가 모여 있다.

즉 W, U, b의 shape는 각각 (input_dim, 4 * units), (units, 4 * units), (4 * units)가 된다. 이때 gate의 순서는 항상 다음의 순서로 저장된다.

[input, forget, cell state, output]

한편, LSTM보다 gate 수를 더 줄인 간략한 모델로는 GRU(Gated Recurrent Unit) 모델이 대표적이다. GRU 모델은 LSTM과 비슷하나 더 간단한 구조를 띤다.

III. LSTM의 금융 투자 모델 활용법

LSTM은 장기 기억 특성과 regime change의 반영 가능 등 장점으로 인해서 많이 활용되며, 금융 시장에서의 접목도 많이 시도되고 있는 상황이다.

여기서는 LSTM을 주식 투자 등에 활용한 학술 논문을 피인용 수(Google Scholar 기준)가 높은 주요 논문 위주로 정리하여, 내용 요약 및 평가를 진행하였다.

Chen, Kai, Yi Zhou, and Fangyan Dai. "A LSTM-based method for stock returns prediction: A case study of China stock market." 2015 IEEE international conference on big data (big data). IEEE, 2015.

(Cited by 1030)

이 논문은 중국 A주 시장에서 주식들의 과거 30일 주가 정보를 넣고, 주식의 사후 수익률을 구간별로 나눠 classification 방식의 주가 수익률 예측 모델을 만들었다. 이때, 입력 데이터에는 High / Low / Open / Close / Volume 및 이의 부분 집합을 사용했다. 논문은 수익률 예측에서 정확도가 존재한다고 결론 내린다. 평가를 해보면, 모델의 구성법은 적절하나, 입력 데이터로 과거 주가 및 거래량 정보만 사용한 점은 추후 개선의 필요성이 있어 보인다.

Moghar, Adil, and Mhamed Hamiche. "Stock market prediction using LSTM recurrent neural network." Procedia computer science 170 (2020): 1168-1173.

(Cited by 807)

이 논문은 GOOGL, NKE 종목을 대상으로, 과거 opening prices를 입력 데이터로 넣고, 다음 날의 주가를 예측하는 모델을 제시하였다. 이 논문 또한 단편적인 가격 정보만을 썼다는 점이 단점이라고 할 수 있다.

Yadav, Anita, C. K. Jha, and Aditi Sharan. "Optimizing LSTM for time series prediction in Indian stock market." Procedia Computer Science 167 (2020): 2091-2100.

(Cited by 485)

이 논문은 인도 주식 시장의 NIFTY 50 내 대표 4개 주식에 대해서, 다음 날의 주가를 예측하는 모델을 연구하였다. 논문은 입력 데이터로 주식의 과거 주가(정확히는 차분한 주가 변화량)를 사용했으며, Stateless와 Stateful LSTM 모델을 비교한 것이 또한 특징이다. 여기서, Stateless LSTM은 각 배치(샘플)가 입력될 때 hidden 및 cell state가 초기화되는 기본적인 LSTM 구조를 말한다. Stateful LSTM은 시간순의 배치(샘플)가 입력되더라도 state 초기화를 하지 않고 계속 누적해 가는 방식을 말한다(keras에서 구현되어 있음). 논문의 결론은 Stateful보다 Stateless LSTM이 우수하고, LSTM 은닉층 개수도 여러 개보다 1개가 더 효율적이라고 주장한다.

그러나 논문에서는 LSTM의 timestep 수를 1로 설정한 것으로 보여서, 너무 단순화된 모델을 사용한다는 단점 또한 가진다.

Bhandari, Hum Nath, et al. "Predicting stock market index using LSTM." Machine Learning with Applications 9 (2022): 100320.

(Cited by 444)

이 논문은 S&P 500 지수의 다음 날 종가를 예측하는 모델을 제시했다. 여기서 입력 변수는 1) 지수의 과거 종가와 2) VIX, 연방기금 실효 금리, 실업률, 소비자 심리 지수, 달러 인덱스와 같은 경제 지표들, 3) MACD, ATR, RSI 같은 기술적 지표 등 여러 변수를 사용한 것이 특징이다(월별 데이터 존재 지표는 forward filling 방식으로 일별 데이터로 변환함). 논문은 단일 LSTM과 다층 LSTM의 성능을 비교하고, 최종적으로는"단일 LSTM + 적절한 데이터 사용"이 가장 효과적이라고 결론 내린다. 전체적으로 보면, 주가지수의 변화에 영향을 줄 수 있는 여러 경제 지표 등의 유의미한 정보를 적극 활용한 점이 긍정적이다.

이상으로 LSTM을 활용한 투자법에 대한 여러 논문을 점검해 보았다. LSTM 모델의 실제 투자로의 접목에 있어서는, 아직도 많은 개선과 발전이 나타날 가능성이 높다고 생각한다.

정리

금융 시계열 예측에 있어 데이터의 망각과 기억을 동적으로 조절하는 LSTM의 설계 철학은 시장의 비선형적 및 레짐 체인지 패턴을 해석하는 데 강력한 이점을 제공한다. 본 자료가 다양한 순환 신경망 모델을 실제 금융 투자 모델에 활용하는 데 있어서, 참고 자료로 활용되기를 기대한다.

삼성 All That Quant

[팩터 모델 실전 전략 및 이론]

25/10/27 "영업이익 증가율 모멘텀 팩터 - 신규 추천 전략" - High return, Low correlation의 고효율 팩터전략 신규 개발 및 제시
25/8/19 "Long-only 전략 상세 분석법 (롱 팩터 전략 시리즈 #2)" - 효율적 Long-biased 전략을 위한 프로세스와 추천 전략 제시
25/7/21 "Long-only 전략과 비대칭 알파 ((롱 팩터 전략 시리즈 #1)" - 코스피 5000 시대를 대비하는, long-biased 전략 집중 분석
24/11/20 "연구개발 팩터 투자전략" - R&D에 기반한 신규 팩터 전략을 개발하고, 이의 특징과 활용 방법 등을 제시함
24/8/8 "코스닥 투자자를 위한 퀀트 전략 Redux" - 코스닥 시장에서 변화하는 팩터 및 스타일 전략의 특징을 상세 재정리함
24/7/2 "연기금 투자자를 위한 장기 퀀트 전략" - 투자 horizon을 장기로 가져갔을 때의 팩터 투자 전략의 변화를 상세하게 비교 정리함
23/11/30 "퀀트 전략에서의 업종 특성 고려, 2편 - 업종 대비 상대 밸류에이션 팩터의 개선 전략" - 상대 밸류 팩터에서의 업종 특성 고려 전략 제시
23/9/4 "퀀트 전략에서의 업종 특성 고려, 1편 - 저P/B 팩터의 업종 조정 전략" - 한국 업종의 특성을 감안한 수정된 밸류에이션 전략 제시
23/7/26 "Data-driven Quant Model 2 - #연속 국면 #머신러닝 #부스팅 기법 #동적 팩터 모델" - 부스팅 알고리즘 기반의 팩터 로테이션 모델 제시
23/5/17 "퀄리티 팩터는 정말 퀄리티 있는 전략인가" - 퀄리티 팩터가 가지는 고유 특성과 세부 전략 간의 차이점을 비교 정리
23/1/4 "포트 전략 - 롱/숏 시그널로 롱 포트폴리오 구성하기" - 롱/숏 전략을 Long-only 포트로 전환하는 효율적인 알고리즘을 신규 제시
22/11/21 "Data-driven Quant Model - 머신러닝 기법으로 시장의 파도에 올라타기" - 머신러닝 기법에 의한 데이터 기반의 팩터 로테이션 모델 제시
22/8/17 "변동성 구간에서의 주식 스타일 특징 - 낙폭 과대주 팩터 활용법" - 팩터 투자 틀 안에서 낙폭 과대주 팩터의 필요성 입증과 활용법 제시
22/5/25 "경기 국면에 따른 리스크 팩터 배분 전략" - 리스크 팩터 전반에 대한 설명과 함께 팩터와 경기 국면을 결합한 투자 전략을 제시
21/2/24 "이익 모멘텀의 차별화" - 종목 특성을 감안한, 변동성 조정 이익 모멘텀 팩터 전략을 제시
21/1/11 "주식 스타일 Fact Book - 주식 스타일 및 매크로 변수 상관성 팩트 체크" - 주식 스타일 분석 체계의 정리 및 매크로 변수 연결
19/4/15 "팩터 모델의 구조와 해설 - Factor Models for Asset Returns" - 팩터 모델 이론에 대한 상세 설명 자료
18/7/3 "Fama-French 3 팩터 모델 - 후편" - Fama & French의 3팩터 모델 1993년 논문에 대한 전문 번역과 한국 시장 분석 추가 자료
18/5/8 "Fama-French 3 팩터 모델 - 전편" - Fama & French의 3팩터 모델 1992년 논문에 대한 전문 번역과 한국 시장 분석 추가 자료
15/10/20 "가치주 투자와 금리의 연관성" - 금리와 가치/성장 스타일의 인과 분석을, Equity Duration 개념을 활용하여 국내 최초로 제시
14/11/24 "투자 Horizon에 따른 퀀트전략 변화 - Advanced Factor Book" - 1M ~ 6M의 투자 기간별로 팩터 모델 백테스팅 분석
13/4/23 "매크로 기반 Dynamic Quant 모델" - 매크로 국면 판단에 기반한 동적 팩터 투자 모델 정립
12/2/27 "스타일 로테이션의 허와 실" - 스타일 로테이션 전략의 형태로서 모멘텀 방식, 리버전(컨트래리안) 방식, Asness 방식을 각기 비교 분석

[퀀트 전략 일반]

26/1/21 "퀀트 모델링 A to Z - (8) RNN(순환 신경망) AI 투자 모델" - 시계열 및 순차 데이터의 모델링이 뛰어난 RNN 모델의 원리와 활용법을 정리함
26/1/8 "퀀트 모델링 A to Z - (7) CNN AI 투자 모델 (기술적 분석 기반)" - CNN 모델의 전체적인 이해와 사용 방법 등을 설명
26/1/2 "퀀트 모델링 A to Z - (6) MLP(Multi-Layer Perceptron) AI 투자 모델" - 딥러닝을 활용한 퀀트 투자 모델 개발법을 상세히 설명함
25/11/14 "자산배분 전략의 체계" - 자산배분에 대한 기본적인 개념과 접근법을 체계적으로 정리함
25/9/2 "머신 러닝 애널리스트, 파트 1" - 주식 시장의 기업 펀더멘털 분석에서, 머신러닝 및 AI를 활용하는 접근법 제시
25/7/29 " KR-FinBERT의 활용법" - 센티먼트 분석 AI를 손쉽게 사용하는 방법을 정리함
25/5/20 "Corporate Action Book ver 2.0" - 여러 corporate action 이벤트에 대한 메커니즘 설명과 시장 반응의 최근 패턴을 정리함
25/3/20 "MSCI 지수 업데이트 ver 5.0" - MSCI 지수의 최근 이슈과 방법론 상의 변화, 정기 리뷰 로직의 요약 정리 제공
25/1/23 "MSCI 미국 지수 2월 리뷰 편출입 최종 전망" - 24년말부터 MSCI USA 종목 편출입 전망 리포트를 최초 개시함
25/1/14 "TAX 전략 2. 소득세뿐 아니라 건강보험료도 존재한다" - 개인 투자자 관점에서의 건강보험료 및 광의의 세금에 대한 설명 및 전략 자료
24/11/7 "AI와 퀀트 투자" - AI에 의한 세상의 변화와 알고리즘에 대한 이해, 퀀트 투자와의 접목법을 정리함
24/5/27 "KOSPI 지수 장기 수익률 성과 분해" - KOSPI 장기 수익률을 기업이익 증가, 주식수 변동, 밸류에이션으로 분해하여 부진 원인을 정리
24/5/8 "주당 지표(per share) 기준의 펀더멘털 접근법" - 지수 차원에서의 주당 지표 산출법과 이를 통한 시장 분석, 관련 팩터 전략 등을 정리
24/1/29 "이전 상장 이벤트 및 활용 전략" - 코스피 이전 상장 종목에 대한 주가 변화 결정 요인 상세 분석
23/7/11 "한미 간 자사주 매입의 구조적 차이" - 자사주 매입에 대한 한국과 미국 제도를 비교하고, 시가총액 정의 차이에 따른 문제점을 정리
23/4/17 "NSS(Nelson Siegel Svensson) 모델을 활용한 채권 퀀트 전략, 파트2" - NS 및 NSS 모델을 활용한 실전적인 채권 퀀트 전략 제시
22/8/1 "주식과 채권의 상관성, 파트 2 - 주식과 채권 간 상관 관계의 다이내믹스" - 주식/채권 상관관계의 변화를 경기과 물가 요인으로 설명 및 전망
20/10/28 "한국이 MSCI 선진시장에 들어가야 하는 이유" - MSCI 선진시장 승격의 당위성과 승격 시의 변화 등에 대해 총정리
19/11/18 "코스피 단기예측 AI 모델 - 랜덤 포레스트 기법을 활용한 머신러닝 기반 모델" - 랜덤 포레스트 기반의 시장 예측 모델 및 소스 코드 제시
17/3/8 "Volatility Drag - 레버리지 투자 수익과 변동성 손실의 관계" - 변동성 손실 개념과 레버리지 상품의 위험성에 대한 상세 설명
16/1/4 "기계학습(Machine Learning)과 투자전략 - 빅 데이터 기법: Naive Bayes Classifier의 활용" - 머신 러닝에 기반한 퀀트 전략 개발
15/7/20 "퀀트 Investment Clock, 한국편" - 성장과 물가 지표를 조합한 4개 국면 기반의 자산배분 모델 제시
14/4/24 "CAPE in Everywhere - Cyclically Adjusted P/E의 다용도 활용 분석" - 한국 증시의 CAPE 지표에 대한 본격 분석
11/5/17 "MSCI 종목편입 효과 실증분석 - 종목 편출입에 따른 최적 투자전략" - MSCI 이벤트 활용 투자 전략의 시작

Appendix. Samsung Q.Pack 메일링 자료

글로벌 주식시장 동향부터, FICC, 경제지표, 국내외 종목 및 ETF 정보까지 수록함. 이메일을 통해 정기적으로 엑셀 자료 전송

Index	Region /Country	Current Index (p)	Index change (%) (a=b+c)								Fwd EPS change (%) (b)				Fwd PER change (%) (c)				EPS Growth (% y-y)			PER (x)			
			1D	2D	1W	MTD	1M	3M	YTD	1Y	1M	3M	YTD	1Y	1M	3M	YTD	1Y	Fwd	21E	22E	Fwd	21E	22E	4yAvg
MSCI	AC World	742.26	0.0	0.0	1.3	4.6	2.5	2.7	14.9	28.0	0.2	4.1	24.6	32.9	2.3	-1.3	-7.9	-3.7	12.2	49.3	7.7	**18.5**	19.7	18.3	16.6
Index	World (DM)	3,151.16	0.0	0.1	1.3	4.8	2.5	3.4	17.1	30.2	0.6	4.7	26.0	34.1	1.9	-1.2	-7.1	-2.9	12.4	48.2	7.9	**19.5**	20.9	19.3	17.3
	N. America	4,627.53	-0.2	0.1	1.6	5.8	3.5	4.1	20.8	32.4	1.0	4.7	28.8	35.6	2.5	-0.6	-6.2	-2.3	12.3	48.0	8.4	**21.5**	23.1	21.4	18.7
	Europ																								
	EMU																								
	Pacifi																								
	Emerg																								
	EM A																								
	EM L																								
	EM E																								
	USA																								
	Fran																								
	Gerr																								
	UK																								
	Aust																								
	Japa																								
	Hon																								
	Chin																								
	India																								
	Kore																								
	Taiw																								
	Thai																								
	Indo																								
	Braz																								

Code	Name	연간실적																매출액(십억)				매출액y-y(%)		
		매출액(십억)		매출액y-y(%)		영업이익(십억)		영업이익y-y(%)		순이익(십억)		순이익y-y(%)		지배순이익(십억)		지배순이익y-y(%)								
		2021E	2022E	2021E	2022E	2021E	2022E	2021E	2022E	2021E	2022E	2021E	2022E	2021E	2022E	2021E	2022E	1Q21	2Q21	3Q21	4Q21	1Q21	2Q21	3Q21
A005930	삼성전자	276,340	297,907	16.7	7.8	52,686	56,360	46.4	7.0	39,936	43,047	51.2	7.8	38,912	42,428	49.1	9.0	65,389	63,672	73,000	74,295	18.2	20.2	9.0
A000660	SK하이닉스	42,530	49,328	33.3	16.0	12,035	13,823	140.1	14.9	9,128	10,460	91.8	14.6	9,113	10,425	91.7	14.4	8,494	10,322	11,814	11,902	18.0	19.9	45.3
A035420	NAVER	6,784	8,312	27.9	22.5	1,360	1,756	11.9	29.2	15,824	1,825	1,772.6	-88.5	16,582	1,854	1,554.8	-88.8	1,499	1,664	1,727	1,900	-13.5	-12.6	1,000.3
A207940	삼성바이오로직스	1,491	1,690	28.0	13.4	490	531	67.4	8.4	396	435	64.3	9.8	396	435	64.2	9.8	261	412	411	421	25.9	34.0	49.7
A051910	LG화학	44,085	49,876	46.6	13.1	5,849	4,968	225.3	-15.1	4,439	3,476	550.5	-21.7	4,077	3,173	695.2	-22.2	9,650	11,456	11,285	11,733	35.6	74.9	50.3
A035720	카카오	5,987	7,607	44.0	27.1	786	1,147	72.3	46.0	951	1,097	448.6	15.3	902	1,029	479.4	14.1	1,258	1,352	1,644	1,733	44.9	41.9	49.4
A006400	삼성SDI	13,871	16,666	22.8	20.2	1,170	1,577	74.3	34.8	1,194	1,498	89.2	25.5	1,119	1,410	94.6	26.0	2,963	3,334	3,617	3,957	23.6	30.3	17.2
A005380	현대차	116,993	125,659	12.5	7.4	7,136	8,018	198.0	12.4	6,635	7,393	244.7	11.4	6,020	6,795	322.7	12.9	27,391	30,326	28,132	31,054	8.2	38.7	2.0
A068270	셀트리온	2,047	2,419	10.7	18.2	820	963	15.2	17.4	654	762	26.0	16.4	641	746	25.5	16.4	457	432	486	584	22.6	0.7	-11.4
A000270	기아	70,551	75,507	19.2	7.0	5,320	5,854	157.5	10.0	4,733	5,190	218.1	9.7	4,730	5,181	217.9	9.5	16,582	18,339	17,125	18,478	13.8	61.3	4.9
A323410	카카오뱅크					296	521	141.8	75.9	244	399	114.3	64.0	244	399	114.3	64.0	225	254					
A005490	POSCO	72,925	74,829	26.2	2.6	8,884	8,158	269.7	-8.2	6,603	5,877	269.3	-11.0	6,094	5,425	280.4	-11.0	16,069	18,292	20,610	19,280	10.5	33.3	44.5
A012330	현대모비스	41,047	45,462	12.1	10.8	2,228	2,706	21.8	21.4	2,616	3,102	71.3	18.6	2,599	3,081	70.0	18.5	9,816	10,285	9,900	11,004	16.5	36.5	-0.9
A259960	크래프톤	2,128	3,306	27.4	55.4	907	1,531	17.2	68.9	722	1,197	29.8	65.9	722	1,197	29.8	65.9	461	459	460	757	-9.3	21.2	31.4
A028260	삼성물산	32,740	32,926	8.4	0.6	1,407	1,448	64.2	2.9	1,923	1,796	65.7	-6.6	1,730	1,563	67.1	-9.6	7,840	8,548	8,034	8,377	12.6	18.3	2.3
A066570	LG전자	71,956	74,202	13.7	3.1	4,288	4,993	34.2	16.4	2,109	3,675	2.2	74.3	1,821	3,615	-7.5	98.5	18,809	16,117	18,785	17,986	27.7	25.6	11.0
A096770	SK이노베이션	45,612	52,196	33.5	14.4	1,938	2,132	175.5	10.0	901	1,430	142.0	58.8	878	1,401	140.4	59.5	9,240	11,120	12,270	13,120	-17.2	54.5	45.7
A017670	SK텔레콤	19,538	20,303	4.9	3.9	1,526	1,678	13.1	10.0	2,655	2,905	77.0	9.4	2,634	2,834	75.1	7.6	4,780	4,818	4,914	5,036	7.4	4.7	3.9
A105560	KB금융					6,172	6,351	33.7	2.9	4,497	4,688	28.4	4.2	4,463	4,646	29.2	4.1	17,584	12,242	16,473		-8.8	14.2	56.4
A051900	LG생활건강	8,401	9,115	7.1	8.5	1,313	1,463	7.5	11.5	892	1,005	9.7	12.7	874	984	9.6	12.6	2,037	2,021	2,139	2,203	7.4	13.4	3.3
A055550	신한지주					5,993	6,188	21.6	3.3	4,334	4,484	23.9	3.5	4,292	4,414	25.7	2.8	6,895	6,813			-15.3	11.6	
A302440	SK바이오사이언스	976	3,572	332.4	266.2	438	1,578	1,060.7	260.5	355	1,227	980.7	245.2	355	1,227	980.4	245.3	113	145	173	499	397.4	266.8	78.8
A034730	SK	94,268	100,607	15.2	6.7	4,642	5,335	2,922.4	14.9	3,106	3,609	2,964.8	16.2	1,377	1,642	627.2	19.3	22,250	23,406	23,724	25,547	-6.2	24.7	18.9
A091990	셀트리온헬스케어	1,838	2,179	12.9	18.5	331	438	-8.6	32.5	257	336	7.0	30.8	257	336	7.0	30.8	356	433	488	569	-0.2	3.1	5.2
A361610	SK아이이테크놀로	676	1,076	44.1	59.2	189	333	50.7	76.3	148	255	68.1	72.0	149	246	69.3	65.1	139	155	178	227			
A015760	한국전력	59,706	62,797	1.9	5.2	-1,631	-951	-139.9	41.7	-2,165	-1,710	-203.5	21.0	-2,230	-1,755	-212.0	21.3	15,075	13,519	16,039	15,091	-0.1	3.4	2.1
A011200	HMM	12,561	12,447	95.9	-0.9	6,315	5,669	543.9	-10.2	3,837	4,905	2,995.0	27.8	3,866	4,994	3,020.8	29.2	2,428	2,907	3,592	3,489	84.9	111.4	109.0
A003550	LG	8,303	8,612	25.2	3.7	2,797	2,957	64.3	5.7	3,054	2,606	98.3	-14.7	2,958	2,507	101.8	-15.2	2,180	1,924	2,011	2,214	37.5	28.7	2.8
A032830	삼성생명	20,270	20,905	-41.3	3.1	1,182	689	-34.0	-41.7	1,755	1,399	28.1	-20.3	1,653	1,359	30.6	-17.8	10,008	8,195	4,701	4,759	-3.5	7.3	-39.5
A086790	하나금융지주					4,393	4,649	14.5	5.8	3,259	3,416	21.4	4.8	3,200	3,357	21.4	4.9	14,011	6,908	13,195		-32.0	10.8	102.5

Compliance notice

- 본 조사자료는 당사의 저작물로서 모든 저작권은 당사에게 있습니다.
- 본 조사자료는 당사의 동의 없이 어떠한 경우에도 어떠한 형태로든 복제, 배포, 전송, 변형, 대여할 수 없습니다.
- 본 조사자료에 수록된 내용은 당사 리서치센터가 신뢰할 만한 자료 및 정보로부터 얻어진 것이나, 당사는 그 정확성이나 완전성을 보장할 수 없습니다. 따라서 어떠한 경우에도 본 자료는 고객의 주식투자의 결과에 대한 법적 책임소재에 대한 증빙자료로 사용될 수 없습니다.
- 본 자료에는 외부의 부당한 압력이나 간섭 없이 애널리스트의 의견이 정확히 반영되었습니다.

삼성증권

삼성증권주식회사
서울특별시 서초구 서초대로74길 11(삼성전자빌딩)
Tel: 02 2020 8000 / www.samsungpop.com

삼성증권 Family Center: 1588 2323

고객 불편사항 접수: 080 911 0900

Member of
Dow Jones
Sustainability Indices
Powered by the S&P Global CSA