컴퓨터공학부 이찬하

프로젝트 개요

Fire together wire together이라는 심리학 이론에 기반하여 만들어본  
10세 아동이 폭력을 학습하는 과정

폭력에 대한 빈번한 노출이  
뇌를 변화시키는 과정

Fire together? Wire together!!

목차

1. 주제 선정 배경
2. 문제에 대한 정의
3. 회로 구현 방법 및 기획 의도
4. 주요 함수 및 변수 설명
5. 프로젝트를 마치며

주제 선정 배경

2020년 1학기가 끝나갈 즈음에 윤경로 교수님의 “컴퓨터공학 개론”수업에서 인공신경망에 대해 개략적으로 배울 기회가 있었다. 노드와 가중치 등 처음 접하는 단어들 속에서 핵심이 무엇인지 고민하다가 문득 같은 학기에 배우고 있었던 “현대인과 정신건강”이라는 교양 수업의 “fire together wire together”라는 문구가 기억났다. 하나의 자극에서 시작된 흥분의 파도에 같이 몸을 실은 뉴런은 더 가까워지고, 같이 흥분하지 못한 뉴런은 멀어진다는 내용이었는데 인간의 뇌를 모방해서 만든 인공신경망이라면, 이런 알고리즘을 추가한다면 더 인간다워 지지 않을까? 라는 생각이 들었다. 흘려 보내기 너무 아까운 아이디어라는 생각에 강의를 중단하고 대충 정리를 해놓았고, 언제 교수님께 질문을 드려 봐야겠다고 생각했는데 정신없이 1학기가 마무리 되고, 잊혀졌다. 시간이 지나고, 개인적으로 프로젝트를 만들 필요가 생겨 프로젝트의 아이디어를 찾던 중 전에 정리했던 ‘인공 신경망 아이디어’(첨부파일)을 발견하게 되었다. 2020년 1학기가 끝나고 한학기의 시간 동안 이산 수학도 배우고 c언어에 좀더 익숙해졌지만, 컴퓨터 공학 개론 수준의 지식만으로 신경망회로에 대한 알고리즘을 짜야 하는 것이 부담되었고, 구현에 실패했을 경우 너무나도 큰 리스크가 있었지만, 개인적 흥미와 별세하신 정주영 회장님의 “해봤어?”의 마음으로 건곤일척을 시작했다.

주제를 어떻게 확장하고 세부 목표를 정했는지 설명하기에 앞서서 프로젝트의 한계에 대해 먼저 짚고 넘어가겠다. 이 프로젝트를 집어 들고 나서 변수명과 함수명을 무엇으로 할까 고민하며 인터넷 사전을 뒤지다가 내 머릿속에서 시작된 아이디어라고 생각했던 것이 수 십년 전에 연구되어 논문으로 나오기까지 한 이론이라는 것을 알게 되었고, 요즘 핫하다는 기계 학습의 골자가 되는 내용임을 알게 되었다. 다른 아이디어로 새로 시작할까 고민하다가 더이상 검색 없이 내가 알고 있는 개론 수준의 지식과 심리학과 생물학, 이산 수학을 총동원해서 외부 간섭없이 지금 당장 알고 있는 지식으로만 프로젝트를 진행하는 방향으로 정했다. 원래의 프로젝트가 심리학에 기초한 가설에서 시작된 것인 만큼 인공신경망에 대한 선행 지식이 있는 사람이 보기엔 “부두술이야 말로 진정한 과학이다!” 라고 외치는 것처럼 느껴질 만큼 황당한 비약이 있을 수 있다. 해당 프로젝트를 지금까지 연구되어 발전해온 인공신경망에 비추어 비교하기보단, 강화 학습이라는 인지심리학 분야에 기초적인 컴퓨터 공학이 더해진 컴퓨터 공학부의 커리큘럼을 아직 1년도 배우지 않은 전과생의 창작으로 보아 주었으면 한다.

문제에 대한 정의

신경망을 구현하기로 정했으나, 단순히 아무런 정보나 받아서 소화하고 아무런 결과를 내어 놓으며 무의미하고 뻔한 가중치 변화 추이를 보여주기보다는 1학기에 배웠던 인지심리학의 단편적인 지식을 응용해 각각의 노드에 역할을 부여해 노드가 유기적으로 연결되고, 각 단계별로 가늘지만 인과관계를 부여해서 유의미한 인풋을 받아 유의미한 아웃풋을 내놓는 식으로 프로그램을 구현해 보기로 했다. 몇 가지 주제를 검토하다가 전에 관심을 가졌었던 폭력의 습관화 과정을 테마로 잡았다.

자신이 어떻게 상황을 인식하고 생각하는지를 기준으로 프로그램을 구현하는 것이 가장 정확하고 설득력 있겠으나 대상이 나이가 많을 수록 상황을 그대로 받아들이는게 아니라 이것저것 재는 것이 많아지기 때문에 프로그램이 너무 복잡해질 것을 고려해서 10세 아동의 생각 회로를 모방하는 것으로 세부 목표를 정했다.

회로 구현 방법 및 기획 의도

시스템 환경 및 사용 언어

하드웨어: Apple silicon M1, 8Gb RAM을 탑재한 Mac mini

소프트웨어: macOS Big Sur version 11.1

사용 언어: C (c11)

사용 IDE: JetBrain CLion

Windows 10환경에선 텍스트가 깨지는 현상이 발생한다. 따라서 윈도우에서 실행하기 위해선 텍스트 편집기로 텍스트의 인코딩 방식을 바꿔주는 중간 절차가 필요하다.

개요

회로는 4개의 Level을 가진 방향 그래프의 형식으로 만들어졌다. 각 Level 산하 여러 개의 노드가 존재하고, 같은 Level에 속하는 노드 사이엔 연결선이 존재하지 않으나 인접한 상위 Level에 속하는 노드와는 연결선이 존재한다. 연결은 하위 레벨 노드에서 상위 레벨 노드로만 연결되어 있고, 그 반대의 경우는 연결되어 있지 않는다. 다음 장의 도해를 참고하자.

Diagram

Description automatically generated

Level 0 – 상황인식 레벨

시뮬레이션이 시작되면 가장 하위 레벨인 레벨 0에서는 input 파일로부터 부여된 상황에 대한 정보를 가져와 각 노드에 마련된 흥분 여부를 기록하는 공간에 저장한다. 이후 각 노드 산하의 엣지들은 해당 정보와 미리 입력된 가중치를 곱하여 상위 레벨에 속한 노드로 값을 전송한다.

Level 1 – 감정, 합리 레벨

Level 1의 산하의 각 노드들은 Level 0에서 엣지 들로부터 넘겨받은 값들을 취합해 자신의 흥분 역치와 비교하고, 취합한 값이 역치보다 클 경우 흥분 여부를 기록하는 공간에 1을 저장한다. 이후 마찬가지로 각 노드 산하의 엣지들은 해당 기록과 자신의 가중치를 곱한 값을 다음 레벨의 노드로 전송한다.

Level 2 – 선거인단 레벨

Level 1과 마찬가지로 Level 2의 각 노드는 이전 레벨의 노드에서 출발한 엣지로부터 넘겨받은 값들을 취합해서 흥분 여부를 저장하고, 해당 노드 산하의 엣지는 다음 레벨의 노드로 가중치를 곱한 값을 전달한다.

Level 3 – 행동 레벨

Level 3는 단일 노드로 구성되어 있고, 해당 노드는 이전의 레벨의 엣지로부터 받은 값을 취합해 흥분 여부를 결정하고, 결과를 저장한다. 상위 레벨이 없으므로 엣지 또한 존재하지 않는다.

회기 마무리 – 가중치 수정

회로가 돌아가며 저장했던 흥분 여부를 중심으로 각 엣지의 가중치를 조정하고, 이후 사용자가 진행 상황을 확인할 수 있도록 표준 출력 창에 결과를 내보낸다. 시뮬레이션의 회기는 여기서 끝나고 다음 input값이 존재할 경우 다음 회기를 시작하기 위해 Level 0단계로 복귀한다.

상세1 – 준비단계

회로가 돌아가기 전, 각 노드에 대한 초기 정보를 입력 파일로부터 읽어와야 한다. 초기 세팅 값은 총 00개로 각 레벨 별 노드의 개수, 노드의 이름, 노드의 흥분 역치, 엣지의 초기 가중치, 엣지의 가중치 변화량 등을 미리 입력된 파일로부터 입력 받는다. 총 5개의 파일로 나뉘어 있으며, 노드의 개수 정보와 노드의 이름이 담겨있는 graphInfo.ignit파일을 제외하곤 소스코드 상단에서 MAKE\_SCRIPT를 define함으로 사용자가 원하는 파일을 프로그램 실행에 맞는 형식으로 작성할 수 있다. 각 파일은 폴더 내의 다른 .txt파일과 구별하기 위해 확장자를 .ignite로 변경하였고, 텍스트파일의 형식으로 작성되었기 때문에 메모장과 같은 텍스트 읽기 프로그램에 확장자를 등록하거나 .txt로 확장자를 변경한다면 읽거나 변경할 수 있다.

상세2 – 각 Level의미

각 레벨은 사고의 단계를 의미하며 산하에 MAX\_SIZE로 정의한 수 이하의 노드를 가질 수 있다. 해당 프로그램 에선 MAX\_SIZE를 20으로 설정하였지만, 변경 가능하고, MAX\_SIZE에 맞추어 작동하도록 설계했다. 각 레벨은 코어 노드를 하나 또는 여러 개 가질 수 있으며, 코어 노드는 동 레벨 다른 노드보다 조금 더 큰 가중치를 갖는다. 또한 코어 노드는 노드의 이름 옆에 \* 기호로 인덱스를 표시할 수 있다.

Level 0는 상황인식단계로 사용자가 직접 세팅한 input을 가지고 가중치를 적용해 다음 레벨로 전달하는 역할을 한다. 기본으로 제공된 basic preset에는 13개의 노드가 있으며 각각 한 사건에 있을 수 있는 다양한 조건들을 나타낸다. 해당 preset에서 코어 노드는 다수와 함께 행동과 오늘의 컨디션 상태이다. (preset은 평범한 환경의 10세 남아를 기준으로 작성되었다.)

Level 1은 감정과 합리 노드로 상황 인식을 기반으로 각 감정이 촉발되는 과정을 표현했다. 10세 남아를 기준으로 작성된 preset은 합리는 단일 노드로 구성되고 6개의 감정 노드로 구성된다. 코어 노드는 즐거움, 분노, 합리이다. 세부사항으로 합리, 무기력에 해당하는 노드는 다음 레벨의 노드들의 흥분을 막는 역할을 부여해 산하 엣지들이 대체로 음의 가중치를 갖고, 모든 감정과 합리 노드들은 각각의 입체성을 고려해 음의 가중치와 양의 가중치를 갖는 엣지들을 모두 가지도록 설정했다.

Level 2는 선거인단 노드이다. 실제 우리 뇌에선 상황에 대한 분석과 그로 인한 감정과 합리적 판단으로 부터 촉발한 스파크는 여러과정들을 거치며 다른 성질의 스파크로 변화한다. 이러한 과정들은 같은 수의 level을 거치지 않으며 같은 회로를 맴돌며 가중치가 증폭되기도 하는 등 개인간의 편차가 매우 커서 모델링하기 적합하지 않다고 판단하여, 상술한 복잡한 과정과 그밖의 여러 사잇단계들을 압축하여 5그룹의 선거인단이 최종결과를 결정하는 방식으로 모델링 하였다. 각 선거인단은 최종결과 하나로만 연결되는 엣지를 가지고 있고, 각 엣지가 갖는 가중치의 변동성은 나이가 어릴수록 증가하고, 일정 이상의 나이를 넘어서면 급격히 줄어들지만, preset의 대상이 10세 아동이므로 가중치의 변동폭이 크다. 다른 레벨과 다르게 상황의 결과로부터 가중치를 변동시킬 소스를 가져오는데 무작위의 결과를 만들기 위해 srand함수를 이용해 random값을 생성하고, 행동을 했을 경우 -10에서부터 10사이의 random수치를 이용해 가중치를 보정하고 행동하지 않고 상황을 넘겼을 경우 -5에서부터 5까지의 random수치를 이용해 가중치를 보정한다.

마지막 level 3는 행동 노드이다. 행동 여부를 출력하는 단 하나의 노드만으로 구성되며 연결된 다른 노드가 없기 때문에 엣지는 존재하지 않는다. 역치를 넘는 자극이 들어오면 행동을 하고, 넘지 않는 자극이 들어올 경우 행동을 하지 않는다. 엣지가 없으므로 가중치와 가중치 변화량이 없다.

상세 3 – 데이터 분석

Calendar

Description automatically generated 각 회기가 끝날 때마다 노드의 흥분 여부를 표준출력창을 통해 출력한다. 이 전 장에 나온 도해와 같은 형식으로 조잡하게나마 데이터를 배치했고, 다음과 같은 보고서를 받을 수 있다.

해당 보고서의 수치를 토대로 가중치 흥분 역치와 가중치, 가중치 변화량을 조정하는데 도움을 받을 수 있다. 주의 할 점은 level2에 속한 노드들이 random값으로 피드백을 받으며, 가중치의 조정은 인접한 노드의 흥분 결과에 따라 유기적으로 이루어지므로 같은 input을 입력 하더라도 운이 나쁘면 정규분포 끄트머리에나 있을 법한 결과를 보고받을 수도 있다. 따라서 같은 데이터도 여러차례 시뮬레이션 해서 수치를 보정하는 것이 좋다.

주요 함수 및 변수 설명

주요 변수

1. MAX\_SIZE 매크로: 각 Level이 가질 수 있는 node의 최대 개수
2. GRAPH 구조체:

Text

Description automatically generated회로에 대한 정보를 담는 구조체. 산하에 Level들을 가지고 있으며, 각 Lev에 속한 노드의 숫자에 대한 정보에 접근하는 int형 포인터 num을 가지고 있다.

1. NODE 구조체:

Text

Description automatically generated노드에 대한 정보를 담는 구조체. 산하에 다음 노드로 향하는 엣지를 가지고 있으며, 노드의 이름을 저장하는 저장공간 name과 흥분 여부를 저장하는 공간 activate, 흥분 역치를 저장하는 공간 criticalPoint를 가지고 있다.

1. EDGE 구조체:

Graphical user interface, text

Description automatically generated엣지에 대한 정보를 담는 구조체 산하에 엣지의 가중치를 저장하는 변수 weigh와 엣지의 가중치 변화량을 저장하는 변수 dWeigh를 가지고 있다.

주요 함수

Text

Description automatically generated전체 함수 목록

Util Function

1. memFree함수: 프로그램에서 사용된 동적 메모리들을 해방하는 함수
2. fileFre함수: 프로그램에서 사용되었던 파일들을 다시 닫는 함수 파일 포인터를 묶은 배열과 배열의 정보를 인수로 받아 작동한다.
3. selectLevel 함수: 그래프의 포인터와 접근을 원하는 Level의 정보를 인수로 받아 해당 Level의 포인터를 넘긴다. 반복문에서 레벨에 접근하기 위해 프로그램 전체에서 자주 사용된다.

MAKE\_SCRIPT Function

1. askMakeScript함수: MAKE\_SCRIPT가 선행처리지시문으로 정의되었을 때 사용된다. 사용자에게 어떤 파일을 작성할지 물으며, 작성할 파일을 생성하고, 해당 파일을 작성하는 함수로 안내한다.
2. makeScript함수: 엣지들의 가중치와 가중치 변화량을 작성하는 함수. 사용자로부터 값을 입력 받기 위해 recordDefaultWeigh함수를 호출한다.
   1. recordDefaultWeigh함수: makeScript함수의 호출을 받고 사용자로부터 표준 입력창으로 가중치 또는 가중치 변화량을 입력 받고, 해당 입력을 atof함수를 이용해 변환한 후 정보의 위치에 저장한다.
3. recordCriticalPoint함수: askMakeScript에서 호출을 받고 작동한다. 노드의 역치 값을 기록하는 함수이다.
4. recordSimulationInput함수: askMakeScript에서 호출을 받고 작동한다. 시뮬레이션에 대입할 input값을 사용자로부터 입력 받아 작성한다.

grab a value from preset Function

1. grabAName함수, setName함수: 외부 파일로부터 각 노드의 이름을 읽어온 후 각 노드에 입력한다.
2. grabAWeigh함수: 외부 파일로부터 각 엣지의 가중치 정보를 읽어와 엣지의 weigh에 입력하고, 가중치의 변화량에 대한 정보도 읽어와 dWeigh에 입력한다.
3. grabACritical함수: 외부 파일로부터 각 노드의 역치 값을 읽어와 각 노드의 criticalPoint에 저장한다.

simulate Function

1. simulate 함수: 시뮬레이션을 총괄 지휘하는 함수로 시뮬레이션에 필요한 변수들을 선언하고 관리하며, 각 단계별로 하위 함수들을 불러와 사용한다. 두번째 회기부터 각 회기의 시작 전에 시뮬레이션을 계속할 것인지, 중단할 것인지, auto모드로 끝까지 진행할 것인지 묻고, 그에 맞추어 중단 명령을 받거나 입력 파일의 EOF를 만날 때까지 시뮬레이션을 진행한다.
   1. beginWeighSum함수: 초기 그래프에 저장된 각 Level의 가중치와 전체 가중치 합을 구해서 저장한다. 이 데이터들은 후에 나올 보고서를 작성하는데 사용될 것이다.
   2. readInput함수: 파일로부터 시뮬레이션 입력 값을 읽어와 Level0의 흥분 여부를 결정하는 공간에 저장한다.
   3. inspectInput 함수: DEBUG가 선행 처리 지시문으로 define되었을 경우 실행되는 함수이다. 함수명에서 알 수 있듯이 Input으로 들어온 값들을 감시하는 역할로 표준 출력 스트림에 Level 0에 속하는 각 노드의 입력 값을 출력한다.
   4. transmitSpark 함수: Level 0에 입력된 값을 바탕으로 상위 Level로 흥분을 전달하는 함수이다. 흥분 여부와 가중치를 곱한 값들의 합을 노드의 역치 값과 비교하여 해당 노드의 흥분 여부를 결정하고 값을 저장한다.
   5. updateWeigh함수: 회로의 핵심 함수이다. Level 0 부터 Level 3까지 모든 노드가 흥분 여부를 결정하였으니 흥분 여부를 가지고 엣지들의 가중치를 조정한다. 엣지의 가중치를 조정하는 방식은 엣지가 속한 Level에 따라 다르다. 먼저 Level 0와 Level 1에 속한 노드의 엣지의 경우는 다음과 같다.

1. 출발 노드와 도착노드 모두 흥분했을 경우 가중치의 변화량만큼 변화시킨다. (두 노드의 관계에 따라 음의 가중치, 가중치 변화량을 갖는 엣지도 존재한다.)

2. 출발 노드가 흥분했으나 도착 노드가 흥분하지 않았을 경우 가중치 변화량만큼 가중치를 깎는다. (양의 변화량을 갖는 엣지는 상향되고, 반대로 음의 변화량을 갖는 엣지는 하향된다.)

3. 나머지의 경우엔 상향 또는 하향하지 않는다.

Level 2에 속하는 노드의 엣지들에 대한 가중치 계산은 상위항목에서 설명한 바 있으므로 생략하겠다.

* 1. visualiseCircuit함수: 회로의 모든 노드들의 흥분 여부를 시각화 하는 함수이다. 도해와 유사한 배치로 출력되어 노드들의 흥분 여부를 확인할 수 있다. 윈도우 Visual Studio기준 콘솔 화면이 가로로 충분히 넓지 않다면 제대로 출력되지 않을 수 있다.
  2. inspectCircuitStatus함수: 보고서를 작성하는 함수이다. 해당 회기까지 시행 중 최종 Level의 행동 노드가 몇번의 흥분을 보였는지, 그리고 흥분의 백분율에 대한 정보를 출력한다. 또한 이번 시행에서 몇개의 노드가 흥분했는지, 흥분한 노드의 비율에 대한 백분율 값을 보고하고 마지막으로 이번 회기 시행 후 각 Level의 가중치 합과 전체 가중치 합이 어떻게 되었는지, 시행 전과 후를 비교하고 얼마나 증가했는 지 백분율로 나타낸다.

**실행 모드에 따라 선택적으로 실행되는 함수들**

텍스트이(가) 표시된 사진

자동 생성된 설명

실행 모드는 총 5가지 모드가 있다. 모든 노드의 현재 가중치, 가중치 변화량, 입력 값을 모두 확인할 수 있는 DEBUG모드, 인풋되는 파일들을 형식에 맞게 작성하도록 도와주는 MAKE\_SCRIPT모드, 그리고 랜덤하게 노드의 가중치와 가중치 변화량을 생성하고, simulationinput까지 실행해주는 MAKE\_SKRIPT\_AND\_SIMULATE\_FULL\_AUTO모드, 그리고 가중치를 변화시키지 않으면서 simulationinput파일의 내용 중 내가 최종적으로 행동을 하도록 자극한 인풋들에 대해 따로 외부 파일에 모아주는 FIND\_STIMULANT\_AND\_RECORD모드가 있다.

사진에 나온 함수들은 각 모드의 목적에 맞게 기존의 함수들을 #ifndef 구문으로 비활성화하고 몇가지 변경점이 적용된 함수들의 선언부이다. 각 함수들의 역할은 변경되지 않았으므로 중복하여 설명하지 않겠다.

프로젝트를 마치며

건곤일척의 마음으로 저돌적으로 시작한 프로젝트가 기획 초기에 이미 누군가 그것도 수십년 전에 구현한 것이라는 것을 알게 되고 중단을 고민할 만큼 큰 위기가 있었지만, 나만의 방식으로 구현해보겠다는 생각으로 끝까지 밀고 나가 수많은 크고 작은 난관을 해치고 여기까지 왔다. 이 전에 이산수학 과제로 작성했던 탐색 프로그램이나 C프로그래밍 수업에서 구현했던 프로그램들과는 다르게 기획부터 구현까지 혼자 모두 하는 것이 쉽지만은 않았지만 매 순간 즐거웠다. 이 코드를 어떻게 간단하게 정리할까? 시뮬레이션 시각화는 어떤 방법으로 구현하는 게 보기 좋을까? 등의 고민으로 머릿속이 가득 찼고, 그때마다 최선은 아닐 수도 있지만, 비 전공자의 관점에선 괜찮은 해법이 떠오르는 행운이 잘 따라 주어 다행이라고 생각한다. 결과적으로 기획 단계에서 구상한 아이디어들을 상당히 많이 구현했고, 중간중간 생각났던 디테일 또한 코드에 상당부분 반영했지만, 그래도 시간이나 실력의 한계로 구현하지 못했던 몇 가지 아쉬운 점들을 이 자리를 빌어 소개하고자 한다.

1. 어른의 뇌를 구현하지 못했다. 전에 언급한 바와 같이 나이가 찰 수록 이것 저것 재는 것이 많아진다. 역치를 넘었다고 다음 노드로 단순히 가중치를 곱한 값을 내보내는 것이 아니라 후행 노드의 어떤 것이 흥분하고 동일한 레벨에 속한 노드중 어떤 것이 흥분할 경우 역치를 넘지 않았음에도 흥분한다거나 그 반대의 경우가 나타나는 등 복잡한 알고리즘을 구현하는데 시간적, 기술적 한계가 있었다. (조건문을 잘 사용하면 될 법도 하지만 코드가 너무 복잡해지고, 라이브러리 함수들을 잘 이용하면 좀더 간단해 질 것 같은데 지식이 없었다.)
2. 노드 간의 연결이 지금 구현한 알고리즘과 같이 수직적인 관계가 아니고, 경계가 모호하면 좋을 것 같다. 상술한 아쉬운 점의 해법이 될 것으로 예상하는데 예를 들어 감정들 간에 서로를 부추기는 일이 일어나기도 하고, 서로 떨어진 노드가 공명하며 증폭하기도 하는 등 레벨의 수직성 문제로 인해 구현하지 못한 아쉬움이 있다.
3. 출력되는 회로의 모습을 좀더 매끄럽게 구현하지 못했다. 물론 실제 상용 프로그램의 그래픽 처리는 별도의 알고리즘이 담당해서 노드와 엣지들을 3D로 멋지게 구현할 수 있겠지만, 그래도 원주율을 이용해서 좀더 각 Level을 둥그스름하게 만들었으면 좋았을 것이다. 그리고 지금의 2D상의 배치를 구현하는 것도 좀 더 다듬었으면 코드가 좀더 정갈해질 수 있었을 것이라는 아쉬움도 남는다.
4. 이번에 새로이 배우고 있는 c++으로 해당 코드들을 이식하여 30개 가량의 객체를 만들고, 객체간 상호작용을 하는 함수들을 만들어 여러 개체들이 주거니 받거니 의견을 교환하고, 대립하는 시뮬레이션을 만들려고 시도했다. 보시다시피 완성하지 못했다. 일단 c++은 물론이고 java도 제대로 배우지 못했기 때문에 객체지향 언어에 대한 이해도가 부족했고, 시도의 초기에 배운 것들을 모두 응용해보겠다고 c의 헤더를 사용하지 않고 모두 c++으로 전환하려 하여 수십 시간을 리펙터링에 쏟아 붓고도 몇가지 난관에 봉착하여 결국 c++의 기초 문법을 거의 배웠을 다음 방학으로 계획을 미루었다.
5. 가중치를 업데이트 하는 수식을 설계하는 과정에서 내의 의도대로 완벽하게 구현하기엔 이런 수식 설계 경험이 부족하다는 것을 뼈저리게 느꼈다. 로그 함수로 변화 량이 완만하게 줄어들게 하고싶고, 거기에 사칙연산을 집어넣어 보정하고 싶었으나 단순히 정형화된 수능 문제를 푸는 방법에 익숙해져 있던 내 수학적 지식들을 이러한 수식 설계에 응용하기엔 이러한 수식들을 어떻게 설계하는지, 결과를 통해 어떻게 보정하는지 등의 경험의 공백이 너무 컸다. 그래도 이런 프로젝트를 시도하지 않았으면 전혀 이런 부족함을 인지하지 못하고 넘어갔을 것이다.
6. 내가 설계했지만, 아직 이런 비선형적 모델에 익숙하지 않아 수식을 어떻게 보정해야 하는지, 입력값을 어떻게 변경해야 할지, 각 레벨 역치의 평균은 어느 수준으로 정해야 할지 등에서 한계를 마주했다. 이는2가지의 추가 실험 중 한가지의 실패와 한가지의 불완전한 성공의 원인이 되었다.