퍼셉트론과 진화 알고리즘을 이용한 게임 인공지능

한남대학교 수학과

20172581 김남훈

1. 퍼셉트론 소개

퍼셉트론은 인공지능을 구현하기 위한 고전적인 방법 중 하나로, 입력받은 벡터에 선형변환과 활성화 함수라고 부르는 Component-Wise 한 비선형 함수를 번갈아 적용하며 출력 벡터를 만드는 알고리즘이다.

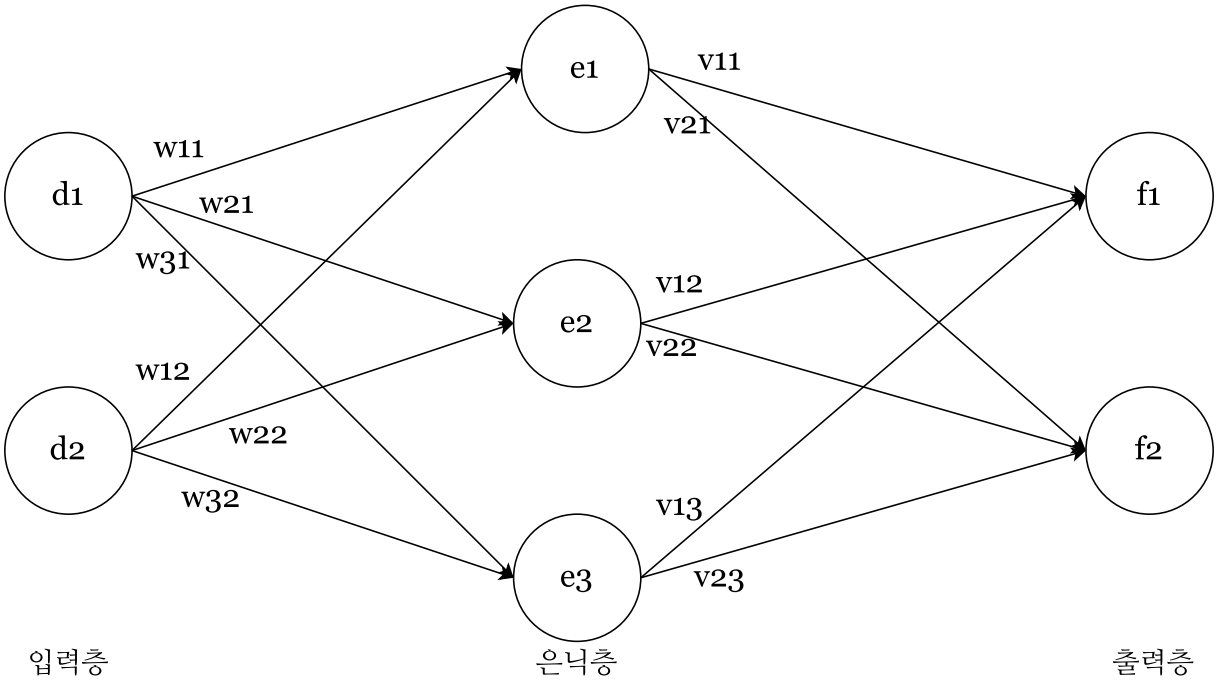

Figure 1: 하나의 은닉층을 갖는 퍼셉트론을 나타낸 그림

방향 그래프의 각 변은 선형변환, 즉 행렬의 각 성분에 대응한다. 또한 각 정점은 벡터의 성분에 대응한다. 방향 그래프의 각 변에 대응하는 실수를 가중치, 정점에 대응하는 실수를 신호라고 한다.

$$D = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix}, E = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix}, F = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$$

$$W = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{pmatrix}, V = \begin{pmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \end{pmatrix}$$

로 놓고 Component-Wise 한 임의의 함수 Activ 를 활성화 함수라고 하면

$$E = \text{Activ}(WD)$$
$$F = \text{Activ}(VE)$$

가 된다. 다음은 활성화 함수로 이용되는 대표적인 세 함수이다.

$$\mathrm{RELU}(d) = \max(0, d)$$

$$\mathrm{Sigmoid}(d) = \frac{e^d}{e^d + 1}$$

$$\mathrm{Sign}(d) = \begin{cases} -1 \text{ if } d \leq 0 \\ 1 \quad \text{if } d > 0 \end{cases}$$

활성화 함수가 필요한 이유는, 선형변환의 합성은 반드시 선형변환이므로 활성화함수 없이는 선형적이지 않은 문제를 풀 수 없기 때문이다.

2. 진화 알고리즘과 퍼셉트론

퍼셉트론이 주어진 문제를 풀 수 있는지는 가중치가 잘 설정되었느냐에 따라 갈린다는 것을 위의 내용으로부터 예상할 수 있다. 가중치는 사람이 직접 설정해줄 수도 있지만 각 층의 차원에 커지면 가중치의 수가 차원의 제곱으로 증가하기 때문에 사실상 불가능하다. 실제로 쓰이는 방법은 적당한 알고리즘을 이용해 랜덤한 초기값을 설정한 뒤 기계학습 알고리즘에 의해 가중치가 자동으로 조정되도록 하는 것이다.

알파고, 챗GPT, 스테이블 디퓨전 등 유명한 인공지능은 역전파 알고리즘을 학습에 이용하지만, 우리는 진화 알고리즘에 대해 알아볼 것이다. 정답이 존재하여 그 정답에 가능한 한 가깝게 접근하는 것이 목적인 상용 인공지능과 달리 게임 속 인공지능의 목적은 게이머에게 재미를 선사하는 것이므로 보다 예측 불가능하고 다양한 행동 패턴을 갖기를 원하기 때문이다.

가장 단순한 형태의 진화 알고리즘은 다음과 같이 구현할 수 있다.

```
def generate_child(self):
    weights = [w_11, w_12, ..., w_32, v_11, ..., v_23] // 기존 퍼셉트론의 가중치를 성분으로 갖는 벡터
    n = randint(1, 12) // 1 에서 12(벡터 weights 의 길이) 사이의 임의의 정수
    for i in range(n): // n 개의 정수를 임의로 선택해
```

이러한 방법으로 생성된 퍼셉트론을 기존 퍼셉트론의 자손이라고 한다. 다수(보통 10개 이상)의 자손을 생성한 뒤, 그 중 가장 정답에 가까운 자손을 부모로 삼아 다시 자손을 생성한다. 진화 알고리즘은 이러한 과정을 개발자가 원하는 만큼 정답에 가까워질 때까지 반복한다.