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.

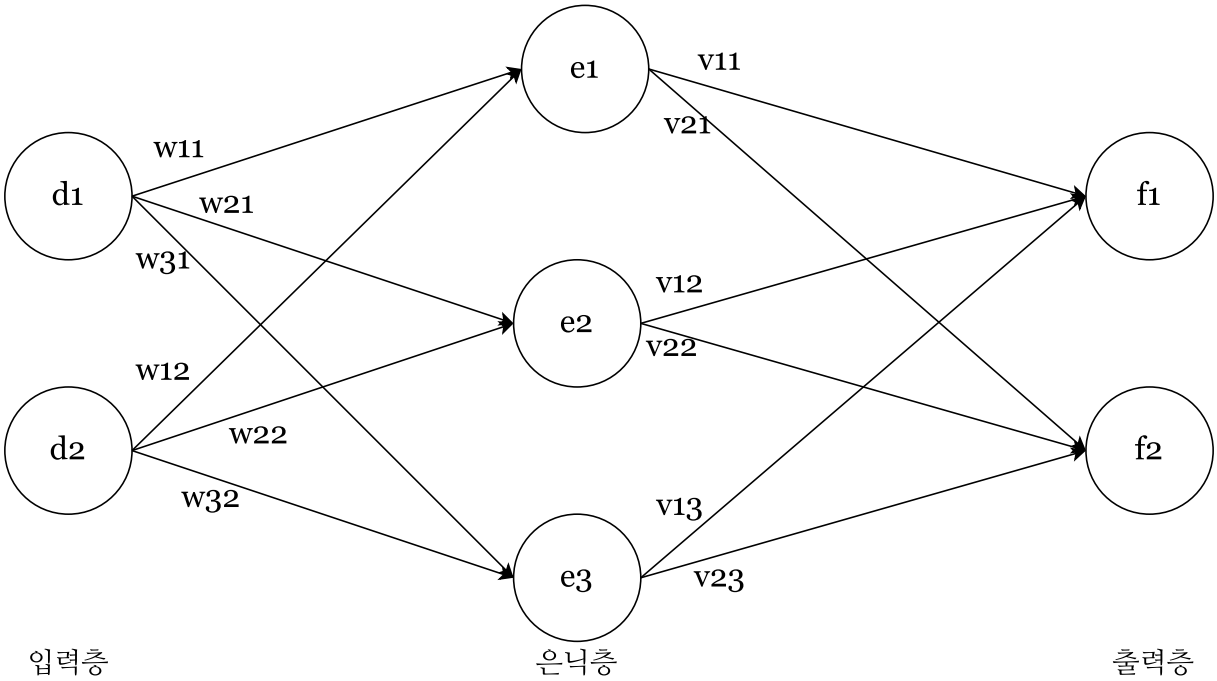

Figure 1: 하나의 은닉층을 갖는 퍼셉트론을 나타낸 그림

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, 이 퍼셉트론의 활성화 함수를 Activ 라고 하면

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

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.

퍼셉트론의 가중치를 설정하는 대표적인 방법으로는 역전파 알고리즘과 진화 알고리즘이 있는데, 여기서는 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.

단순한 형태의 진화 알고리즘은 다음과 같이 구현할 수 있다. 다음은 주어진 퍼셉트론의 가중치 중 일부를 임의로 선택해 -0.1 에서 0.1 사이의 임의의 실수를 더해 얻어진 퍼셉트론을 반환하는 함수이다.

```
def generate_child(self):
    weights = [w11, w12, ..., w32, v11, ..., v23] // 기존 퍼셉트론의 가중치를 성분으로 갖는 벡터
    n = randint(1, 12) // 1 에서 12(벡터 weights 의 길이) 사이의 임의의 정수
    selected = weights.select(n) // 1 에서 12 사이의 정수 n 개를 임의로 선택

    for i in selected:
        multiplier = randomfloat(-0.1, 0.1) // −0.1 에서 0.1 사이의 임의의 실수
        weights[i] = weights[i] + multiplier

    return Perceptron(weights) // weights 의 각 성분을 가중치로 갖는 퍼셉트론을 반환
```

위 코드를 통해 기존의 퍼셉트론에서 약간 변형된 퍼셉트론을 얻을 수 있다. 이것을 기존 퍼셉트론의 **자손** 이라고 부르는데, 적게는 수십, 많게는 수백개의 자손을 생성한 뒤 그 중 정답에 가장 근접한 자손을 선택하여 다음 세대의 부모로 삼아 앞의 과정을 다시 수행한다. 이 때, n 번째 반복에서 생성된 자손들을 n **세대** 라고 한다. 진화 알고리즘은 이러한 과정을 사용자가 원하는 만큼 정답에 근접한 퍼셉트론이 등장할 때까지, 적게는 10 세대, 많게는 수백 세대에 걸쳐 위의 과정을 반복한다.

3. 진화 알고리즘과 게임

선형대수학을 수강한 학생들은 주어진 문제의 정답이 속한 공간의 차원이 증가할수록 각 세대가 지날 때마다 정답에 더 근접하기 위해 생성해야 하는 자손의 수가 지수적으로 증가함을 짐작할 수 있을 것이다. 그러므로, 진화 알고리즘은 비교적 단순한 문제에 유효하다. 위에서 언급한 **역전파**

알고리즘 은 그보다 복잡한, 차원이 높은 문제에 유효하다. 게임 NPC 의 행동은 차원이 낮은 문제에 속하므로, 진화 알고리즘을 통한 학습이 적절하다.

진화 알고리즘을 이용해 인공지능에게 운전을 가르치는 예시
https://youtu.be/Aut32pR5PQA
여기에서는 두 개의 은닉층을 갖는 퍼셉트론을 사용하였다. 매 순간의 행동은 2 차원 공간의 벡터로 주어진다.

진화 알고리즘을 이용해 인공지능에게 뱀 게임을 가르치는 예시
https://youtu.be/C4WH5b-EidU
https://github.com/kairess/genetic_snake
소스 코드가 공개되어 있다. 매 순간의 행동은 3 차원 공간의 벡터로 주어진다.

참고문헌

Negnevitsky, M. (2005). Artificial intelligence: a guide to intelligent systems. Pearson education.
Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.