퍼지 및 지능시스템학회 논문지 2004, Vol. 14, No. 4, pp. 383-389

대전 액션 게임을 위한 신경망 지능 캐릭터의 구현

An Implementation of Neural Networks Intelligent Characters for Fighting Action Games

조병헌* · 정성훈** · 성영락* · 오하령*

Byeong-Heon Cho, Sung-Hoon Jung, Yeong-Rak Seong, Ha-Ryoung Oh

* 국민대학교 전자정보통신공학부

** 한성대학교 정보공학부

요 약

본 논문에서는 신경망을 이용하여 대전 액션 게임의 캐릭터들을 지능화하는 방법을 제안한다. 일반적인 대전 액션 게임에서 어떤 행동은 여러 개의 시간 단위에 걸쳐 이루어진다. 그러므로 캐릭터의 어떤 행동에 대한 결과는 곧바로 나타나지 않고 몇 개의 시간 단위가 지난 후에 나타난다. 이러한 캐릭터들에 적합한 신경망을 설계하기 위해서는 신경망을 학습시키는 시점을 결정하는 것과 더불어 학습시에 사용되는 입력과 출력 값을 선정하는 것이 매우 중요하다. 본 논문에서는 캐릭터의 행동의 적합도를 게임 점수의 변화로 평가한다. 그러므로 게임 점수의 변화가 생길 때마다 신경망은 학습된다. 학습을 위해서는 우선 그 변화를 야기한 이전의 결정을 파악하고, 그 당시의 입력값, 출력값, 그리고 현재의 점수의 변화를 이용하여 신경망을 학습시킨다. 제안된 알고리즘의 성능을 평가하기 위하여 여러 실험을 간단한 (하지만 실제 게임과 매우 유사한) 게임 환경에서 수행하였다. 실험 결과 학습 초기에는 무작위의 캐릭터에 대해 점수를 획득하지 못하던 지능 캐릭터가 제안된 알고리즘으로 학습하면 최대 3.6 배의 점수를 획득하는 성능을 보였다. 그러므로 제안된 지능 캐릭터가 게임의 규칙과 기술을 학습하는 능력이 있는 것으로 결론지을 수 있다. 제안된 알고리즘은 온라인 게임과 같이 캐릭터들이 서로 대결하는 게임들에 적용할 수 있다.

Abstract

This paper proposes a method to provide intelligence for characters in fighting action games by using a neural network. Each action takes several time units in general fighting action games. Thus the results of a character 's action are not exposed immediately but some time units later. To design a suitable neural network for such characters, it is very important to decide when the neural network is taught and which values are used to teach the neural network. The fitness of a character 's action is determined according to the scores. For learning, the decision causing the score is identified, and then the neural network is taught by using the score change, the previous input and output values which were applied when the decision was fixed. To evaluate the performance of the proposed algorithm, many experiments are executed on a simple action game (but very similar to the actual fighting action games) environment. The results show that the intelligent character trained by the proposed algorithm outperforms random characters by 3.6 times at most. Thus we can conclude that the intelligent character properly reacts against the action of the opponent. The proposed method can be applied to various games in which characters confront each other, e.g. massively multiple online games.

Key words : 게임, 인공지능, 신경망, 강화 학습

1. 서 론

컴퓨터 게임에서 사용되는 인공지능 기법으로서 유한 상태 기계(Finite State Machine:FSM)와 같은 전통적인 기법에서부터 퍼지 상태 기계(Fuzzy State Machine:FuSM), 인공생명, 신경망 등과 같은 다양한 기법들이 연구되고 있다[1,2,3]. 이러한 기존의 인공지능 기법들은 대부분 바둑이나 체스, 오목 등과 같이 개개의 돌과 말의 행동을 결정할 때, 게임의 전체적인 상황을 고려하는 보드 게임을 위한 것이다. 최근 들어서는 신경망 또는 인공생명 등을 이용하여 군집을 이루는 캐릭터들의 전체적인 전략을 조율하는 연구가 시도되고 있다[4,5,6]. 그러나 전체적인 전략 수립을 목적으로 하는 기존의 방법들은 대전 액션 게임이나 온라인 게임 내의 캐릭터들에게 적용하기 어려운 문제가 있다. 대전 액션 게임 등과 같은 장르에서는 게임 전체의 전략을 조율하는 것도 중요하지만, 그 보다는 게임 내의 캐릭터가 주변의 적들의 위치나 행동에 대하여 적절한 행동을 하도록 지능화 시키는 것이 더욱 중요하다.

본 논문에서는 보다 일반화된 게임으로 확장하기 위하여 다음과 같은 세 가지의 가정을 둔다. 첫째, 게임에서의 각 행동이 하나의 시간 단위에서 완료하는 것이 아니고, 행동별로 여러 단계가 필요하다. 둘째, 캐릭터가 어떤 행동을 수행하고 있을 때, 상대방 캐릭터의 공격으로 인하여 타격을 받으면 그 캐릭터의 행동은 무효화된다. 셋째, 캐릭터가 수행 중이던

접수일자 : 2004년 5월 22일
완료일자 : 2004년 6월 21일

행동을 임의로 취소하거나 다른 행동으로 바꿀 수 없다. 이와 같은 가정이 적용된 대전 액션 게임에서는 캐릭터의 행동의 시작과 그 결과가 나타나는 시점이 달라지기 때문에 학습시킬 때 사용되는 학습 데이터를 적절히 선정하는 방법을 고안하여 적용해야만 한다. 제안한 알고리즘에서는 게임 규칙의 학습을 위하여 지능 캐릭터가 행동을 결정하는 시점의 입출력 값과 상대방 캐릭터와 지능 캐릭터 사이의 점수의 차를 이용하여 강화 학습하였다.

본 논문에서 제안한 방법을 테스트 해보기 위하여 일반적인 대전 액션 게임과 유사한 게임을 구현하고 실험했다. 지능 캐릭터가 구현한 게임의 규칙을 학습하도록 하기 위해서 무작위로 행동하는 상대방 캐릭터와 대결시켜 실험한 결과, 학습 초기에는 점수를 획득하지 못하던 지능 캐릭터가 학습 후에는 상대방 캐릭터보다 최대 3.6배의 점수를 획득하는 것을 보았다. 이러한 실험 결과는 제안된 지능 캐릭터가 게임의 규칙과 기술을 학습하는 능력이 있어서 본 논문에서 제안한 방법이 일반적인 대전 액션 게임에도 적용될 수 있음을 보여준다. 본 논문에서 제안한 방법은 대전 액션 게임뿐만 아니라, 여러 개체가 군집을 이루어 대전하는 게임이나 온라인 게임에서의 특정 종족에도 활용할 수 있는 등 실용성이 매우 높은 방법이다.

본 논문의 구성은 다음과 같다. 2절에서는 게임에 적용된 기존 알고리즘을, 3절에서는 본 논문에서 제안하는 지능 캐릭터에 대하여 알아본다. 그리고 4절에서는 지능 캐릭터를 검증하기 위해 구현한 대전 액션 게임과 실험 결과를 알아보고, 마지막으로 5절에서 결론을 맺는다.

2. 게임에 적용된 기존의 인공지능 기법

현재 게임 분야에서 가장 널리 사용되는 인공지능 기법인 FSM은 상태들 간의 전이에 의해 통제되는 그래프 내에 유한한 개수의 상태들이 연결되어 있는 규칙 기반 시스템이다[7,8]. FSM은 구현이 쉽고 행동이 정확히 정의되는 장점이 있으나 게임의 진행이 미리 정의된 방식으로만 동작하기 때문에 게임에 어느 정도 숙달되면 캐릭터의 행동을 쉽게 예측될 수 있어서 게임의 흥미를 반감시키는 요인으로 작용한다. 이러한 단점을 보완하기 위해 FuSM이 사용되고 있다[7]. FSM에 퍼지 이론을 접목한 FuSM의 경우에는 입력과 출력에 퍼지 함수를 적용하여 어느 정도의 무작위 요소가 포함된다. 따라서 동일한 상황에서 다른 행동을 할 가능성이 있어서 상대방의 행동을 예측하기 어렵다. 그러나, FSM과 FuSM은 캐릭터의 상태의 수가 많아지게 되면, 상태도를 정리하기도 어렵고, 프로그램이 급격하게 복잡해지는 단점이 있다. 또한 FSM과 FuSM 모두 새로운 행동 양식을 추가하기 위해서는 새롭게 프로그램을 해야만 하는 단점이 있다.

인공생명은 생명체가 나타내는 현상을 컴퓨터, 로봇 등과 같은 인공 매체 상에 재현함으로써 생명의 일반적인 특성에 대해 연구하는 학문이다. 게임 개발자들은 인공생명 기법을 게임에 적용하려고 오래 전부터 시도를 해왔다. 그러나 지금까지 연구의 초점은 주로 캐릭터들이 군집을 형성하는 게임에서 전체적인 전략을 결정하는 것에 중점을 둔 연구로서 아직 기초적인 단계에 머물고 있다[6].

마지막으로, 신경망은 인간의 신경 체제를 모사한 것으로서, 수많은 간단한 콤포넌트들이 연결된 구조를 가지고, 입력되는 데이터의 패턴 인식에 기반하여 출력을 산출한다[10,11]. 신경망이 게임에 적용되었을 때의 가장 큰 장점으로는, 신경망이 학습 능력을 가지고 있기 때문에 게임의 규칙을 스스로 학습하고 또한 게임을 진행하면서 계속적으로 지능이 향상될 수 있다는 점이다. 그래서 지금까지 신경망을 게임에 적용하려는 연구가 많이 진행되고 있는데, 대부분이 오목, Tic-Tac-Toe 등 보드 게임이 주류를 이룬다[5]. 보드 게임은 보드 상에서 움직이는 말들의 전체적인 상황을 인지하여 개개의 말들의 움직임을 결정하는 특성을 가지고 있다. 반면에 본 논문에서 다루는 대전 액션 게임은 하나의 캐릭터가 외부 환경에 적응하여 자신의 행동을 결정해야 하는 특성을 가지고 있다.

우리는 [9]에서 신경망을 이용하여 대전 액션 게임을 위한 지능 캐릭터를 구현하는 방법을 제안하였다. 이 논문에서 우리는 신경망이 지능 캐릭터를 구현하는 데에 있어서 매우 우수하고 가능성이 높은 방식임을 보였다. 그런데 이 논문에서는 문제를 단순화시키기 위해서 게임에서의 모든 행동이 하나의 시간 단위에서 완료한다는 가정을 두고 있다. 반면에 현재 존재하는 대부분의 게임에서는 캐릭터들의 행동이 여러 시간 단위에 걸쳐서 이루어진다. 이러한 차이점으로 인해서 [9]의 방법은 실제의 게임에 바로 적용하는 데에 한계가 있다. 본 논문에서는, 현재의 대전 액션 게임에 바로 적용할 수 있도록, 여러 시간 단위에 걸쳐서 행동이 일어나는 게임들을 위한 신경망을 이용한 지능 캐릭터의 학습 방법을 제안한다.

3. 지능 캐릭터의 구현

지능 캐릭터를 위한 신경망을 구성하기 위해서는 (i) 지능 캐릭터가 어떤 입력을 받아서 어떠한 과정을 거쳐서 출력을 발생시킬 것인가 하는 것과 (ii) 그 출력에 대해 어떻게 학습할 것인가를 고려해야 한다. 전자는 신경망의 구조에 대한 것이고, 후자는 학습 방법에 대한 것이다.

3.1 신경망의 구조

본 논문에서 적용한 대전 액션 게임의 모델과 이에 따른 신경망의 구조는 그림 1과 같다.

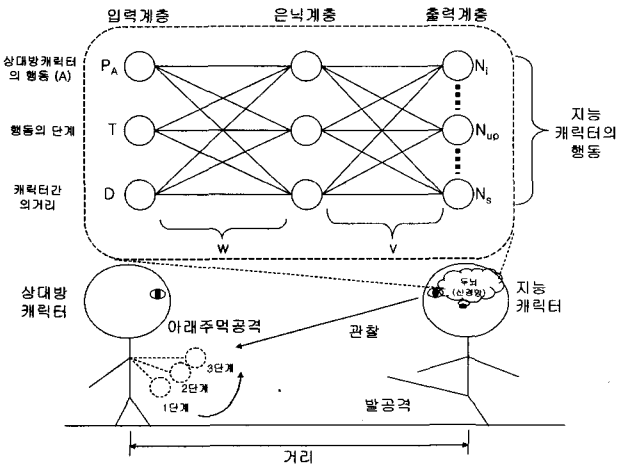

그림 1. 신경망의 구성

Fig 1. The Structure of the neural network

그림 1에서 보듯이 지능 캐릭터는 상대방 캐릭터와 대결을 통하여 정보를 획득하고, 획득한 정보를 이용하여 학습한

다. 상대방 캐릭터는 기존의 방법으로 구현된 캐릭터와 온라인 게임에서 사람이 조작하는 캐릭터도 가능하다. 상대방 캐릭터는 자신만의 전략과 공격 및 방어법을 가지고 지능 캐릭터와 대결을 한다. 게임 규칙을 학습하지 않은 초기의 지능 캐릭터는 상대방 행동에 대하여 임의의 행동을 수행하고, 그 결과를 관찰하여 스스로 게임 규칙을 익힌다.

실제 상황에서 어떤 사람 N이 어떤 상대와 대전해야 하는 경우를 생각해 보자. N의 행동은 무엇에 의해서 결정되어야 할까? 만약 N의 대전 능력이 상대에 비해서 월등한 경우라면, 물론 상대의 행동에 상관없이 공격해도 무관할 것이다. 그러나 만약 그렇지 못하다면 상대방이 어떤 행동(A)을 하는 지를 유심히 살펴서 그에 따라서 자신의 행동을 결정해야 한다. 또한 N이 엄청나게 민첩하지는 못하므로 이런 저런 행동을 하다가 보면 상대방의 행동이 어느 정도 지난 다음에야 상대방의 행동을 인식하게 될 것이다. 그러므로 같은 상대방의 행동이라도 그 행동이 어느 정도 진행(T)되었는가에 따라 다르게 대처해야 한다. 마지막으로 상대방의 같은 행동이라도 상대방과의 거리(D)를 고려해서 대처해야 한다. 상대방이 멀리서 주먹을 휘두르는 것과 가까이에서 주먹을 휘두르는 것은 분명히 다른 상황이다.

본 논문에서 제안하는 지능 캐릭터도 위의 실제 대전에서와 같이 상대방 캐릭터의 행동'(A) 및 그 행동이 현재 몇 단계 (T)에 있는지 또한 상대방 캐릭터와의 거리 (D)가 어느 정도인지를 관찰하고, 이를 신경망에 입력한다. 신경망에 입력하기 위해서는 각 입력값이 숫자로 매핑되어야 한다. 상대방 캐릭터의 행동은 게임에서 주먹, 발차기 등으로서, 행동의 종류는 유한하기 때문에 정수로 매핑한다. 행동의 단계란 상대방의 특정 행동을 여러 시간 단위로 나눈 것으로 본 논문에서는 정수값으로 매핑한다. 예를 들어 주먹동작이 시작부터 완료까지 3단계로 이루어진다면 행동의 단계는 1부터 3까지의 정수로 입력된다. 상대방 캐릭터와의 거리는 지능 캐릭터가 관찰한 거리로서 이 또한 정수로 처리한다. 행동의 단계와 거리 모두 실세계에서는 연속적인 값이지만 컴퓨터에서는 이산적인 값으로 처리할 수밖에 없고, 따라서 정수로 매핑해도 문제가 없다. 결국 어떤 순간에 거리 2에서 상대방 캐릭터의 주먹공격이 단계 3을 수행 중이라면, 그 순간의 신경망의 입력은 $P_A=7$, $T=3$, $D=2$ 가 된다[1]. 이와 같이 결정된 입력값을 신경망에 적용하면 출력이 계산된다. 출력의 개수는 지능 캐릭터가 취할 수 있는 행동의 개수(11)와 동일하며 0에서 1 사이의 실수값으로 출력된다. 지능 캐릭터는 이 중 가장 큰 수로 출력된 행동을 하게 된다. 예를 들어 11개의 출력 중에서 N_s(그림 1 참조)가 가장 크다면 지능 캐릭터는 행동 s(예를 들어, 필살기)를 수행하게 된다.

신경망은 전방향 신경망 (feedforward neural networks)을 사용하며 일반적인 전방향 신경망과 동일하게 출력을 계산한다. 다음은 전방향 신경망에서 입력값을 이용하여 은닉 계층과 출력 계층의 노드들의 출력을 계산하는 식이다.

$$h_j = f\left(\sum_{i=0}^{N_i}(x_i \cdot w_{ij})\right) \quad (1)$$

$$z_j = f\left(\sum_{i=0}^{N_h}(h_i \cdot v_{ij})\right) \quad (2)$$

여기서, $f(x)$는 시그모이드(sigmoid) 함수, h_j는 j번째 은닉 노드의 출력, N_i는 입력 노드의 개수, x_i는 i번째 입력 노드, w_{ij}는 i번째 입력 노드와 j번째 은닉 노드간의 링크 가중치, z_j는 j번째 출력 노드, N_h는 은닉 노드의 개수, v_{ij}는 i번째 은닉 노드와 j번째 출력 노드간의 링크 가중치를 나타낸다. 신경망은 식 (1), (2)를 이용하여 출력을 계산한다. 신경망의 초기 가중치는 무작위적 값으로서, 초기에 신경망의 출력도 무작위 값이 된다. 그러므로 신경망을 이용하여 지능 캐릭터의 행동을 결정하기 전에 먼저 게임 규칙에 대하여 학습을 해야 한다.

3.2 학습 방법

지능 캐릭터를 구현하는 신경망을 학습시키기 위해서는 먼저 어떤 방법을 이용하여 학습 시킬 것인지 결정해야 하고, 학습 방법이 결정되면 학습시킬 때 사용되는 학습 데이터를 어떻게 선택할 것인지를 고려해야 한다. 이 절에서는 이 두 가지 사항에 대하여 차례대로 알아본다.

일반적으로 신경망을 학습하는데는 교사 학습(supervised learning), 비교사 학습(unsupervised learning) 그리고 강화 학습(reinforcement learning) 등이 있다. 교사 학습은 상태 공간의 일부분에서 또는 전역적인 최적의 해가 알려진 상태에서 훈련 예제에 함축되어 있는 매핑을 오프라인으로 학습하는 방법으로서, 본 논문에서의 신경망처럼 어떤 행동이 최적의 행동인지는 알 수 없는 경우에는 사용할 수 없다. 그러나 어떤 행동이 최적의 행동인지는 몰라도 현재 행동에 대한 적절성(본 논문의 경우 획득 점수의 차)이 주어진 경우 이를 이용하여 학습하는 방법으로 강화 학습이 사용될 수 있다. 강화 학습은 사전 지식이 없어도 학습이 가능하고, 환경과의 반복적인 시도와 오류를 통해서 최적에 가까운 매핑을 학습한다. 강화 학습은 속도는 느리지만 훈련 예제와 사전 지식이 충분히 주어지지 않은 경우에도 온라인 학습이 가능한 방법이다. 그림 2는 본 논문에서 사용한 강화 학습 방법을 나타낸다.

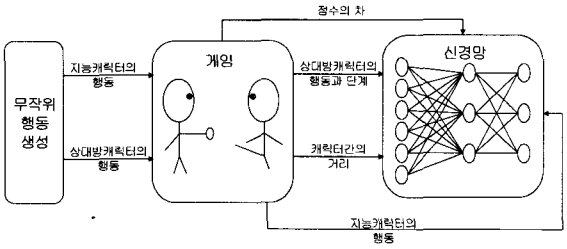

그림 2. 강화 학습 방법
Fig 2. Reinforcement learning

지능 캐릭터와 대결하는 상대방 캐릭터는 어떠한 행동 방식을 가진 것이라도 무방하지만, 본 논문에서는 학습을 위하여 무작위 행동을 하는 캐릭터를 이용하였다. 학습하기 전에는 지능 캐릭터의 어떤 공격이 좋은지 알 수 없으므로 지능 캐릭터의 행동도 무작위로 선택을 하게 한다. 그림 2에서 보듯이 무작위로 선택된 상대방 캐릭터의 행동과 지능 캐릭터의 행동이 결정되면 이를 게임에 적용한다. 그런 후 상대방 캐릭터의 행동과 단계 및 캐릭터간의 거리를 신경망의 입력으로 사용하고 두 캐릭터의 행동의 결과로 나타난 점수의 차를 강화값으로 사용하여 현재 무작위로 선택된 지능 캐릭터의 행동이 적당한지를 학습시킨다.

논문 [9]에서는 모든 행동이 하나의 시간 단위에서 동시에 시작해서 동시에 끝나기 때문에 신경망의 입력을 결정하는 것이 매우 단순하였다. 그러나 본 논문에서는 각 행동별로 행동

1) 주먹 공격이 정수 7로 매핑된다고 가정한다.

의 단계를 두었기 때문에 신경망의 입력이 단순하게 결정되지 않는다[2]. 이를 포함하여 본 논문에서의 가정은 다음과 같다.

가정 1 : 캐릭터의 행동이 1 클럭에 완료하는 것이 아니고, 행동별로 여러 단계가 필요하다.

가정 2 : 캐릭터가 어떤 행동을 수행하고 있을 때, 상대방 캐릭터의 공격으로 인하여 타격을 받으면 그 캐릭터의 행동은 무효화된다.

가정 3 : 캐릭터가 수행 중이던 행동을 임의로 취소하거나 다른 행동으로 바꿀 수 없다. 즉, 어떤 행동을 수행 중이었다면 그 행동을 수행하는데 필요한 시간이 경과하거나 상대방 캐릭터의 공격을 받아 무효화되지 않으면 다른 행동을 시작할 수 없다.

세 가지 가정 모두 일반적인 대전 액션 게임에서의 캐릭터의 행동을 분석하여 얻은 결과로서, 지능 캐릭터의 학습 방법을 어렵게 만드는 요인이 된다. 이와 같은 가정을 바탕으로, 학습할 때 신경망에 입력되는 파라미터의 값을 어느 시점의 값으로 할지를 적절히 선택해야한다. 그림 3은 각 캐릭터가 행동하는데 5 클럭의 시간을 필요로 하는 행동 s를 시차를 두고 행동했을 때, 공격 행동이 완료되어 점수가 발생하는 두 가지 대표적인 상황을 나타낸다. (a) 그림은 지능 캐릭터 행동의 완료로 점수가 발생하는 것이고, (b) 그림은 상대방 캐릭터 행동의 완료로 점수가 발생하는 경우를 각각 보여준다.

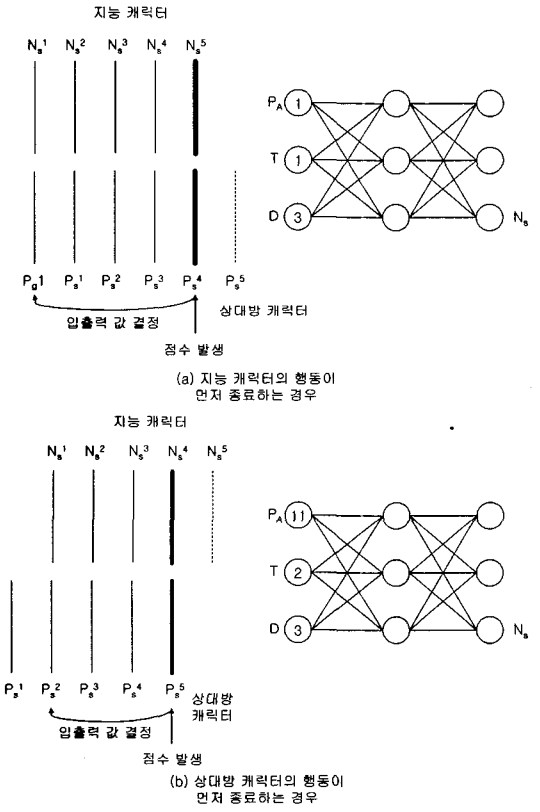

그림 3. 학습할 때의 입출력 값의 결정

Fig 3. Deciding Input/Output values for learning

그림에서 N은 지능 캐릭터, P는 상대방 캐릭터를 나타내며, N_s^2는 지능 캐릭터가 s라는 행동의 2단계를 수행하고 있는 것을 나타낸다. 그림 3(a)는 상대방 캐릭터가 행동하는 도중에 지능 캐릭터의 행동이 완료되었기 때문에, 만약 지능 캐릭터의 행동이 상대방에게 타격을 주는 유효한 행동이었다면 그 시점에 점수가 발생하고, 상대방 캐릭터의 행동은 중단되어 P_s^5 행동은 무효화된다. 이와 비슷하게 그림 3(b)는 지능 캐릭터가 행동하는 도중에 상대방 캐릭터의 행동이 완료되었기 때문에, 만약 상대방의 행동이 지능 캐릭터에게 타격을 주는 유효한 행동이었다면 그 시점에 점수가 발생하고, 지능 캐릭터의 행동은 중단되어 N_s^5 행동은 무효화된다. 이와 같이 점수가 발생하면 신경망의 입출력 값을 결정하고 강화 학습한다.

학습할 때의 입출력 값을 결정하는 방법은 다음과 같다. 예를 들어 두 캐릭터 간의 거리가 3일 때를 가정한다. 그림 3(a)의 경우에는, 점수는 지능 캐릭터의 행동 s가 끝나는 시점에 발생하지만, 그 점수는 상대방 캐릭터가 행동 g의 단계 1이 된 시점에서 지능 캐릭터가 행동 s를 결정했기 때문에 발생한 것이다. 그렇기 때문에, 이 경우에는 상대방 캐릭터의 행동은 1, 단계 1, 그리고 두 캐릭터 사이의 거리 3이 입력되고[3], 출력은 지능 캐릭터의 행동 s로 결정한다. 그리고 이 행동의 결과로 얻은 점수의 차('지능 캐릭터의 점수-상대방 캐릭터의 점수)를 이용하여 학습을 하게 된다[4]. 이와 비슷하게 그림 3(b)에서는 점수는 상대방 캐릭터의 행동 s가 끝나는 시점에 발생하지만, 그 점수는 상대방 캐릭터가 행동 s의 단계 2가 된 시점에서 지능 캐릭터가 행동 s를 결정했기 때문에 발생한 것이기 때문에, 상대방의 행동은 11, 단계 2, 그리고 두 캐릭터 사이의 거리 3이 입력되고[5], 출력은 지능 캐릭터의 행동 s로 결정한다. 이렇게 학습을 하면 후에 유사한 상황에서 유사한 결정을 하게 되고 이 결정은 지능 캐릭터가 점수를 획득하는 쪽으로 반영된다.

이와 같이 입출력 값이 결정되면, 이를 이용하여 오류 역전파 알고리즘으로 학습을 수행한다. 오류 역전파 알고리즘을 이용한 학습은 신경망을 구성하는 각 링크의 가중치를 각 계층별로 수정한다. 먼저 은닉 계층과 출력 계층 사이의 링크의 가중치는 식 (3), (4)에 의해 수정된다.

$$\delta_j = z_j \cdot (1-z_j) \cdot (d_j - z_j) \quad (3)$$

$$v_{ij} \mathrel{+}= \alpha(t) \cdot \delta_j \cdot h_i \quad (4)$$

여기서, d_j는 목표값(desired output)을 의미하며, z_j는 j번째 출력 노드, v_{ij}는 i번째 은닉 노드와 j번째 출력 노드 간의 링크 가중치, $\alpha(t)$는 시간에 따른 학습률의 함수로서, 본 논문에서는 학습률 $\cdot e^{-t/\Delta t}$를 사용한다. Δt는 $t_{end}-t_{start}$이고, t_{start}는 학습 시작 시간, t_{end}는 학습 종료 시간이다.

식 (3)에서 d_j는 출력 노드별로 다르게 주어진다. 지능 캐릭터의 행동으로 결정된 출력 노드의 d_j는 점수의 차가 0 이상일 경우에는 점수의 차의 크기에 비례해서 0에서 1사이의 실수로 변환하고, 음수일 경우에는 0으로 정한다. 그 외의 출력 노드들은 학습할 필요가 없으므로 학습을 시키지 않기 위해서 d_j값을 신경망의 출력값 z_j와 같게 한다. 이렇게 하면

2) 대부분의 대전 액션 게임에서는 행동이 몇 개의 단계로 구성된다.

3) 행동 g는 정수 1로 매핑된다고 가정한다.

4) 여기서, 행동 g는 행동하는데 필요한 시간을 1 클럭으로 가정하며, 그렇기 때문에 상대방 캐릭터가 행동 g를 한 후, 바로 다른 행동(s)을 수행할 수 있다.

5) 행동 s는 정수 11로 매핑된다고 가정한다.

δ_j가 0이 되어서 해당 출력의 가중치 변화가 일어나지 않는다. 예를 들어 게임에서 발생할 수 있는 최대 점수차가 5점이고, 그림 3(a)에서 +3점의 점수차가 발생했다면, d_j는 0.6(=3/5)으로 설정된다. 이렇게 함으로써 점수의 차가 양수이면 해당 출력 노드는 다음에도 동일한 상황에 대해 같은 점수의 차가 나오도록 수정하고, 반면에 다른 노드들은 가중치를 수정하지 않는다. 또한 그림 3(b)에서 −3점의 점수차가 발생했다면, 점수차가 음수이므로 목표값을 0으로 설정하여 동일한 상황에서 해당 출력의 행동을 하지 않도록 링크의 가중치가 수정된다. 위의 과정을 반복하여 학습이 완료되면, 실제 게임 중에는 특정 상황에서 출력 노드들 중에서 가장 큰 값, 즉 점수의 차가 가장 큰 노드를 지능 캐릭터에게 가장 유리한 행동으로 결정할 수 있다.

식 (3), (4)를 이용하여 은닉 계층과 출력 계층 사이의 링크의 가중치를 수정한 다음, 입력 계층과 은닉 계층 사이의 링크의 가중치는 식 (5), (6)을 이용하여 수정된다.

$$\delta_j = h_j \cdot (1-h_j) \cdot \left(\sum_{k=0}^{N_k} w_{jk}\delta_k\right) \quad (5)$$

$$w_{ij} \mathrel{+}= \alpha(t) \cdot \delta_j \cdot x_i \quad (6)$$

여기서 x_i는 i번째 입력 노드, h_j는 j번째 은닉 노드, N_h는 은닉 노드의 개수, w_{ij}는 i번째 입력 노드와 j번째 은닉 노드간의 링크 가중치, $\alpha(t)$는 학습률 함수를 나타낸다.

학습이 진행됨에 따라서 지능 캐릭터의 신경망은 특정한 상황 (즉, 상대방 캐릭터의 행동과 그 단계, 거리)에 따른 자신의 최적 행동이 무엇인지를 학습하게 된다. 지능 캐릭터가 최적의 행동을 학습하기 위해서 경험해야 하는 입출력 패턴의 종류는 총 2376가지(상대방 캐릭터의 행동×행동의 단계×상대방 캐릭터와의 거리×지능 캐릭터의 행동)로서 상당히 많다고 할 수 있다. 그러나 그림 2에서 무작위 행동 생성이 모든 상황에 대한 것을 생성하지 않기 때문에 많은 학습을 하더라도 최적의 행동을 찾는다는 것을 보장하지 않지만, 학습 기간이 길어질수록 최적의 행동을 찾을 가능성은 높아진다.

4. 실험 및 결과

본 논문에서 제안한 지능 캐릭터를 검증하기 위하여 간단한 대전 액션 게임 모델을 개발했다. 게임은 두 캐릭터가 제한된 1차원 좌표 위를 이동하면서 공격과 방어를 하며, 그 결과에 따라 점수를 획득한다. 캐릭터가 할 수 있는 행동은 표 1과 같다. 캐릭터의 모든 행동은 클릭에 동기가 맞춰져 있고, 각각의 공격 행동을 하는데 필요한 시간은 표 2와 같으며, 그 외 행동은 모두 1 클릭이다.

표 1. 상대방 캐릭터의 행동별 정수 매핑 값
Table 1. The integer mapping value according to opponent character's actions

캐릭터의 행동 A	정지 i	전진 f	후진 b	점프 j	앉기 d	막기 g	아래주먹 dp	위주먹 up	아래발 dk	위발 uk	필살기 s
신경망의 입력 PA	1	2	3	4	5	6	7	8	9	10	11

캐릭터가 할 수 있는 공격은 크게 주먹공격, 발공격 그리고 먼 거리에서 행할 수 있는 필살기의 세 종류로 구분되며, 주먹공격과 발공격은 각각 상대방 캐릭터의 하반신을 공격하는 아래공격과 상반신을 공격하는 위공격으로 세분화된다. 실제 게임처럼 각각의 공격에 따라 공격의 효과를 얻을 수 있는 유효 거리와 공격점수를 설정하였다. 표 2는 각 공격의 속성을 표시한 것이다.

표 2. 각 공격의 필요시간/공격점수/유효거리
Table 2. Necessary time/Attack score/Effective range of each attack

공격행동	필요시간(클릭)	공격점수	유효거리
아래주먹	2	1	0~2
위주먹	4	2	
아래발	6	3	2~3
위발	8	4	
필살기	10	5	3~5

그리고, 아래주먹과 아래발공격은 상대방이 막거나 점프를 하면 점수를 얻지 못하고, 위주먹과 위발공격은 상대방이 막거나 앉으면 점수를 얻지 못한다. 필살기 공격은 상대방이 막으면 50%의 점수만을 획득한다.

지능 캐릭터가 게임의 규칙을 학습했는지 여부를 판단하기 위해서는 판단의 기준이 필요하다. 본 논문에서는 게임을 일정 시간 수행한 후 상대방 캐릭터와 지능 캐릭터가 얻은 획득 점수의 비를 비교한다.

신경망의 은닉 계층 노드의 수는 몇 차례의 사전 실험 중 가장 좋은 결과를 보였던 15로 하여 실험을 하였다. 학습률은 0.01~1.0, 학습 시간은 1000~8192000, 그리고 각 경우에 대하여 10개의 무작위 초기 값에 대하여 수행한 후 획득 점수의 비의 평균값과 표준편차를 조사하였다. 표 3은 상대방 캐릭터의 행동과 지능 캐릭터의 행동을 모두 무작위로 선택하여 일정 시간 학습시킨 후, 무작위로 행동하는 상대방 캐릭터와 일정 시간 대결시켜 지능 캐릭터와 상대방 캐릭터가 획득한 점수의 비를 나타낸 것이며 그림 4는 이를 그래프로 표현한 것이다. 표 3에서 "학습전"은 지능 캐릭터가 학습되지 않았을 때 (즉 지능 캐릭터 신경망의 링크가 무작위로 주어진

표 3. 획득 점수의 비 (평균/표준편차)
Table 3. The ratio of the acquired score (average/standard deviation)

학습률 / 학습시간	0.01	0.1	1.0	학습전
1000	0.2/0.1	0.2/0.1	0.1/0.0	0.0
2000	0.2/0.1	0.1/0.0	0.0/0.0	
4000	0.2/0.1	0.1/0.0	0.0/0.0	
8000	0.2/0.1	0.1/0.0	0.8/0.3	
16000	0.2/0.0	0.1/0.0	1.8/0.2	
32000	0.1/0.0	0.6/0.2	1.9/0.1	
64000	0.1/0.0	0.5/0.1	2.2/0.1	
128000	0.1/0.0	1.5/0.0	2.3/0.0	
256000	0.6/0.2	2.1/0.4	2.4/0.0	
512000	0.6/0.2	2.3/0.1	2.8/0.2	
1024000	1.3/0.3	2.4/0.1	3.1/0.2	
2048000	1.7/0.0	2.5/0.0	3.4/0.2	
4096000	2.1/0.5	2.9/0.0	3.6/0.1	
8192000	2.9/0.1	3.4/0.1	3.6/0.3	

상태) 무작위로 행동하는 캐릭터와 대표적으로 10000 클럭 동안 행동했을 경우의 결과이다. 표 3에서 보듯이 "학습전"의 지능 캐릭터가 획득하는 점수는 0점이다. 지능 캐릭터가 0점을 획득하는 이유는 신경망의 링크가 무작위 값으로 초기화되어, 현재 신경망에 입력되는 값의 범위에서는 대부분의 경우에서 전진 또는 막기에 해당하는 출력 노드가 가장 큰 값을 갖게 되어 공격을 못하기 때문이다.

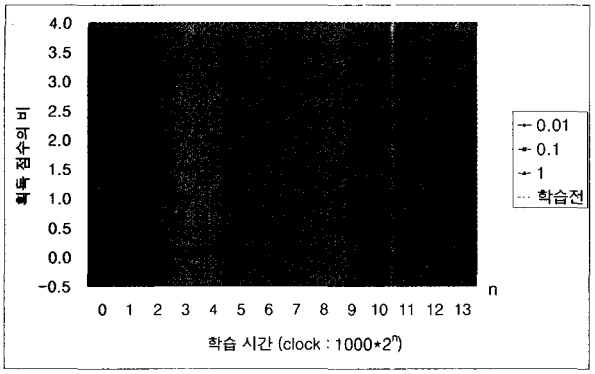

그림 4. 획득 점수의 비
Fig 4. The ratio of the acquired score

실험 결과들을 보면, 학습 시간이 증가하면 모든 경우에서 "학습전"의 지능 캐릭터보다는 학습 후에 획득 점수의 비가 증가했고, 또한 같은 학습률에 대해서는 학습 시간이 커질수록 획득 점수의 비가 증가하여 최대 3.6배까지 획득하는 것을 알 수 있다. 이와 같은 결과는 지능 캐릭터가 학습을 거듭할수록 많은 게임의 규칙을 학습하여 상대방 캐릭터보다 큰 점수를 획득하는 기술을 학습했음을 보여준다. 그리고 그림 4에서 학습 시간이 증가함에도 불구하고 획득 점수의 비가 감소하는 경우가 있다. 이와 같은 결과가 나타나는 이유는 복잡한 상태 공간에서 더 좋은 해를 찾아가기 위해서는 그 도중에 있는 나쁜 해를 거쳐야하는 경우가 있기 때문이다. 즉 신경망이 지능 캐릭터에게 더 유리한 행동을 찾아가는 과정에서 일시적으로 불리한 행동을 하도록 학습해서 실험 결과와 같이 획득 점수의 비가 감소하게 된다. 그러나 학습을 계속 진행하면 더 나은 행동을 학습하여 결과적으로는 획득 점수의 비가 증가하는 추세를 보이게 된다. 그리고 신경망이 8192000 클럭 동안 학습하여 게임 규칙을 학습하는데 소요되는 시간은 펜티엄 IV 2.8 GHz에서 평균 2분 정도여서 학습 속도가 상당히 빠르다는 것을 알 수 있다.

표 4는 학습률이 1일 때 약 800만 클럭 동안 학습한 후의 지능 캐릭터의 행동을 나타낸 것이다. 두 캐릭터간의 거리가 0, 1, 4, 5일 때에는 대부분 해당 거리에서 수행 할 수 있는 공격 중 점수를 많이 획득할 수 있는 행동으로 매핑되었다(표1 참조). 그러나 거리가 2, 3일 때에는 점수를 얻을 수 있는 공격의 종류가 상대적으로 많고, 또한 지능 캐릭터 같은 행동을 하더라도 다른 결과를 얻을 수 있기 때문에 다양한 공격으로 매핑되었다. 예를 들어 표 4의 결과에서는 상대방 캐릭터와의 거리가 2이고, 상대방 캐릭터가 전진 1단계를 행동할 때, 지능 캐릭터는 위발공격을 수행한다. 그 다음에 상대방 캐릭터가 위발공격을 수행하면 지능 캐릭터의 행동인 위발공격이 먼저 완료하여 지능 캐릭터가 점수를 획득한다. 그러나 상대방 캐릭터가 전진 후에 아래발공격을 수행한다면 상대방 캐릭터의 아래발공격이 먼저 완료하기 때문에 상대방 캐릭터가 점수를 획득한다. 만약 상대방 캐릭터가 어떤 특이한 공격 성향을 가져서 특정 행동들을 반복적으로 행동한다면 지능 캐릭터는 상대방 캐릭터의 행동 양식을 학습하여 보다 나은 행동을 할 수 있을 것이나, 이에 대한 연구는 추후에 수행할 예정이다.

표 4. 학습 후의 지능 캐릭터의 행동
Table 4. The intelligent character's action after learning

두 캐릭터 간의 거리																	
0			1			2			3			4			5		
P_A	T	N_A	P_A	T	N_A	P_A	T	N_A	P_A	T	N_A	P_A	T	N_A	P_A	T	N_A
1	1	8	1	1	8	1	1	10	1	1	10	1	1	11	1	1	11
2	1	8	2	1	8	2	1	10	2	1	10	2	1	11	2	1	11
3	1	8	3	1	8	3	1	10	3	1	10	3	1	11	3	1	11
4	1	8	4	1	8	4	1	10	4	1	10	4	1	11	4	1	11
5	1	8	5	1	8	5	1	10	5	1	11	5	1	11	5	1	11
6	1	8	6	1	8	6	1	10	6	1	11	6	1	11	6	1	11
7	1	8	7	1	8	7	1	9	7	1	11	7	1	11	7	1	11
7	2	8	7	2	8	7	2	7	7	2	11	7	2	11	7	2	11
8	1	8	8	1	8	8	1	9	8	1	10	8	1	11	8	1	11
8	2	8	8	2	8	8	2	8	8	2	11	8	2	11	8	2	11
8	3	8	8	3	8	8	3	7	8	3	11	8	3	11	8	3	11
8	4	8	8	4	8	8	4	7	8	4	11	8	4	11	8	4	11
9	1	8	9	1	8	9	1	9	9	1	9	9	1	11	9	1	11
9	2	8	9	2	8	9	2	8	9	2	9	9	2	11	9	2	11
9	3	8	9	3	8	9	3	8	9	3	7	9	3	11	9	3	11
9	4	8	9	4	8	9	4	8	9	4	7	9	4	11	9	4	11
9	5	8	9	5	8	9	5	8	9	5	7	9	5	11	9	5	11
9	6	8	9	6	8	9	6	8	9	6	7	9	6	11	9	6	11
10	1	8	10	1	8	10	1	9	10	1	9	10	1	11	10	1	11
10	2	8	10	2	8	10	2	8	10	2	9	10	2	11	10	2	11
10	3	8	10	3	8	10	3	8	10	3	7	10	3	11	10	3	11
10	4	8	10	4	8	10	4	8	10	4	7	10	4	11	10	4	11
10	5	8	10	5	8	10	5	8	10	5	7	10	5	11	10	5	11
10	6	8	10	6	8	10	6	8	10	6	7	10	6	11	10	6	11
10	7	8	10	7	8	10	7	8	10	7	7	10	7	11	10	7	11
10	8	8	10	8	8	10	8	8	10	8	7	10	8	11	10	8	11
11	1	8	11	1	8	11	1	9	11	1	9	11	1	9	11	1	11
11	2	8	11	2	8	11	2	8	11	2	9	11	2	9	11	2	11
11	3	8	11	3	8	11	3	8	11	3	9	11	3	9	11	3	11
11	4	8	11	4	8	11	4	8	11	4	9	11	4	9	11	4	11
11	5	8	11	5	8	11	5	8	11	5	9	11	5	9	11	5	11
11	6	8	11	6	8	11	6	8	11	6	9	11	6	9	11	6	11
11	7	8	11	7	8	11	7	8	11	7	7	11	7	7	11	7	11
11	8	8	11	8	8	11	8	8	11	8	7	11	8	7	11	8	11
11	9	8	11	9	8	11	9	8	11	9	7	11	9	7	11	9	11

5. 결 론

본 논문에서는 신경망을 이용하여 일반적인 대전 액션 게임을 위한 지능 캐릭터를 구현하는 방법을 제안하였다. 제안한 알고리즘은 게임 규칙의 학습을 위하여 지능 캐릭터가 행동을 결정하는 시점의 입출력 값과 상대방 캐릭터와 지능 캐릭터 사이의 점수의 차를 이용하여 강화 학습하였다. 제안한 방법을 테스트해 보기 위하여 실제와 유사한 대전 액션 게임을 개발 적용하여 실험하였다. 대전 액션 게임에 적용해 본 결과 무작위로 행동하는 캐릭터보다 최대 3.6배의 점수를 획득함을 보았다. 이와 같은 결과로써 지능 캐릭터가 학습을 거듭할수록 많은 게임의 규칙을 학습하여 상대방 캐릭터보다 큰 점수를 획득하는 기술을 학습했음을 보여준다. 본 논문에서 제안한 지능 캐릭터는 대전 액션 게임을 위한 것이기 때문에, 캐릭터들이 행동을 함으로써 결과가 나타나는 게임들, 예를 들어 전략 게임이나 최근 많이 개발되어 서비스되고 있는 온라인 게임 등에 적용될 수 있어서 활용할 수 있는 곳이 매우 많다. 지능 캐릭터가 외부 환경의 변화에 적응하는 방법

과 상대방 캐릭터가 어떤 특이한 공격 성향을 보일 때 이에 대처하는 방법 등에 대한 연구를 추후에 수행할 예정이다.

참 고 문 헌

[1] Daniel Fu, Ryan Houlette, Stottler Henke, "Putting AI In Entertainment: An AI Authoring Tool for Simulation and Games", IEEE Intelligent and Systems July/August, Vol.17, No.4, 2002.

[2] Daniel Johnson, Janet Wiles, "Computer Games With Intelligence", IEEE International Fuzzy Systems Conference, 2001.

[3] Mark DeLoura, Game Programming Gems 2, Charles River Media, 2001.

[4] Bernd Freisleben, "A Neural Network that Learns to Play Five-in-a-Row", 2nd New Zealand Two-Stream International Conference on Artificial Neural Networks and Expert Systems, 1995.

[5] David B. Fogel, "Using Evolutionary Programming to Create Neural Networks that are Capable of Playing Tic-Tac-Toe", Intl. Joint Confrence Neural Networks, New York, pp.875-880, 1993.

[6] 조남덕, 성백균, 김기태, "인공생명 시뮬레이션을 통한 게임 캐릭터의 전략 구현", 정보과학회 2000년 춘계학술대회, VOL.27, NO.01, pp.0241~0243, April, 2000.

[7] 이만재, 게임에서의 인공지능 기술, 한국정보처리학회지, Vol. 9, No. 3, pp.69-76, May, 2002.

[8] Steve Rabin, AI Game Programming Wisdom, Charles Rivers Media, 2002.

[9] 조병헌, 정성훈, 성영락, 오하령, "신경망을 이용한 지능형 게임 캐릭터의 구현", 정보과학회, submitted for publication 2003.

[10] Chin-Teng Lin, C. S. George Lee, Neural Fuzzy Systems, Prentice Hall, 1996.

[11] Richard. P. Lippman, An Introduction to Computing with Neural Nets, IEEE ASSP Magazine, pp.4-22, April, 1987.

저 자 소 개

조병헌(Cho, Byeong-heon)

1997년 : 국민대학교 전자공학과(공학사)
1999년 : 국민대학교 전자공학과(공학석사)
1999년~현재 : 국민대학교 전자정보통신공학부 박사과정

관심분야 : 유전자 알고리즘, 시뮬레이션, 신경망

Phone : 02-910-5063
FAX : 02-912-7586
E-mail : d995552@hanmail.net

정성훈(Jung, Sung-hoon)

1988년 : 한양대학교 전자공학과(공학사)
1991년 : 한국과학기술원 전기및전자공학과(공학석사)
1995년 : 한국과학기술원 전기및전자공학과(공학박사)
1995년~1996년 : 한국과학기술원 전기 및 전자공학과(위촉연구원)
1996년~1998년 : 한성대학교 정보전산학부 정보통신공학전공 전임강사
1998년~2002년 : 한성대학교 정보전산학부 정보통신공학전공 조교수
2002년~현재 : 한성대학교 정보공학부 정보통신공학전공 부교수

관심분야 : 지능 시스템, 유전자 알고리즘, 신경망, 뉴로퍼지
Phone : 02-760-4344
FAX : 02-760-4217
E-mail : shjung@hansung.ac.kr

성영락(Seong, Yeong-rak)

1989년 : 한양대학교 전자공학과(공학사)
1991년 : 한국과학기술원 전기 및 전자공학과(공학석사)
1995년 : 한국과학기술원 전기 및 전자공학과(공학박사)
1995년~1996년 : 한국과학기술원 위촉연구원
1996년~1998년 : 국민대학교 전자공학부 전임강사
1998년~2002년 : 국민대학교 전자공학부 조교수
2002년~현재 : 국민대학교 전자정보통신공학부 부교수

관심분야 : 시뮬레이션, 고장감내, 내장형 시스템
Phone : 02-910-5063
FAX : 02-912-7586
E-mail : yeong@kookmin.ac.kr

오하령(Oh, Ha-ryoung)

1983년 : 서울대학교 전기공학과(공학사)
1983년~1986년 : 삼성전자 종합연구소
1988년 : 한국과학기술원 전기전자과 컴퓨터공학전공(공학석사)
1992년 : 한국과학기술원 전기전자과 컴퓨터공학전공(공학박사)
1992년~1996년 : 국민대학교 전자공학부 조교수
1996년~2001년 : 국민대학교 전자공학부 부교수
2001년~현재 : 국민대학교 전자정보통신공학부 교수

관심분야 : 병렬처리, 내장형 시스템, 고장감내
Phone : 02-910-4708
FAX : 02-912-7586
E-mail : hroh@kookmin.ac.kr