신경망을 이용한 지능형 게임 캐릭터의 구현

조 병 헌[†] · 정 성 훈[††] · 성 영 락[†††] · 오 하 령[††††]

요 약

본 논문에서는 신경망 기반의 지능형 게임 캐릭터를 구현하는 방법을 제안한다. 지능 캐릭터를 구현하는 신경망은 상대방 캐릭터의 행동과 상대방 캐릭터와의 거리를 입력받아 지능 캐릭터의 행동을 결정하여 출력한다. 신경망은 두 캐릭터들의 행동으로 인한 점수를 강화 값으로 사용하여 강화 학습된다. 제안한 방법의 효용성을 보이기 위해서 간단한 대전 액션 게임을 구현하고 그 환경에서 여러 가지 실험을 수행하였다. 실험 결과 제안한 지능형 캐릭터가 게임의 규칙을 잘 학습할 수 있음을 보였다. 제안된 방법은 대전 게임뿐만 아니라 대규모 온라인 게임상의 캐릭터 구현에도 적용될 수 있다.

An Implementation of Intelligent Game Characters using Neural Networks

Byeong-heon Cho[†] · Sung-hoon Jung[††] · Yeong-rak Seong[†††] · Ha-ryoung Oh[††††]

ABSTRACT

In this paper, we propose a scheme to implement intelligent game characters based on neural networks. Neural networks that implement intelligent game character receive the action of an opponent character and the distance between them, decide intelligent character's action, and output the decision. The neural networks are trained by reinforcement learning using the scores acquired by the actions of two characters as reinforcement values. To show the usefulness of the proposed scheme, a simple fighting action game is implemented and various experiments are performed. Experimental results show that proposed intelligent characters can learn the rule of the game. The proposed scheme can be applied to massively multiple online games as well as fighting action games.

키워드 : 게임(Game), 인공지능(Artificial Intelligent), 신경망(Neural Network), 강화 학습(Reinforcement Learning)

1. 서 론

지금까지 여러 연구에서 컴퓨터 게임에 인공지능 기법의 접목이 시도되었다[1-3]. 그 동안의 연구에서는 인공지능 기법으로서 주로 유한 상태 기계(Finite State Machine : FSM), 퍼지 상태 기계(Fuzzy State Machine : FuSM), 인공생명 등을 사용하였다[3-5].

이러한 기존의 연구는 대부분이 보드 게임 등과 같이 게임 전체의 상황을 인식하여 대처하는 게임을 위한 것으로 게임 설계자가 모든 상황을 미리 규정해 설계하고 일일이 코딩해야 하는 단점이 있다. 또한 이렇게 제작된 게임이라고 할지라도 스스로 환경을 인식하여 적응하는 능력이 없어 상황이나 게임의 규칙이 변화하면 다시 코딩을 해야 한다.

최근 들어 신경망 또는 인공생명 등을 이용하여 게임에 지능을 부여하려는 연구가 시도되었다[6-8]. 그러나 이러한 연구는 주로 캐릭터들이 군집을 형성하는 분야에서 전체적인 전략을 조율하는 인공생명에 근간을 둔 연구로서 아직 기초적인 연구에 머물고 있다.

본 논문에서는 게임 캐릭터에 지능을 부여하여 스스로 게임의 규칙을 학습할 수 있도록 하는 방법을 제안한다. 제안된 방법에서는 지능 캐릭터를 구현하기 위하여 학습 능력이 있는 신경망을 이용하며, 게임 규칙의 학습을 위해서는 강화 학습을 사용한다. 지능 캐릭터는 초기에는 게임의 규칙을 전혀 모르기 때문에 무작위로 행동하는 캐릭터와의 대전을 통하여 스스로 게임 규칙을 학습하도록 한다. 이러한 방법은 특히 대전 액션 게임이나 전략 게임 등에서 지능을 갖는 캐릭터를 구현하는데 유용하며, 또한 온라인 게임 등 다양한 종류의 게임으로도 확장할 수 있는 장점을 갖고 있다.

제안한 지능 캐릭터를 실제로 구현하고 테스트해 보기 위하여 간단한 대전 액션 게임을 구현하여 실험하였다. 실험에 적용한 게임은 두 캐릭터가 1차원 게임 공간에서 이동하면서 11가지 행동을 한다. 11가지 행동은 이동/공격/수비 등으로 이루어져 있고, 두 캐릭터의 행동 결과에 따라 점수를

† 준 회 원 : 국민대학교 대학원 전자정보통신공학부
†† 정 회 원 : 한성대학교 정보공학부 교수
††† 종신회원 : 국민대학교 전자정보통신공학부 교수
†††† 정 회 원 : 국민대학교 전자정보통신공학부 교수
논문접수 : 2004년 8월 4일, 심사완료 : 2004년 10월 1일

부여한다. 제안된 방법으로 학습한 캐릭터를 무작위로 행동하는 캐릭터와 대결시킨 결과, 무작위로 행동하는 캐릭터보다 최대 5.1배의 점수를 획득했다. 이러한 실험결과는 본 논문에서 제안한 방법으로 학습한 지능 캐릭터가 실제로 게임 규칙을 학습하여 대처함을 보여준다.

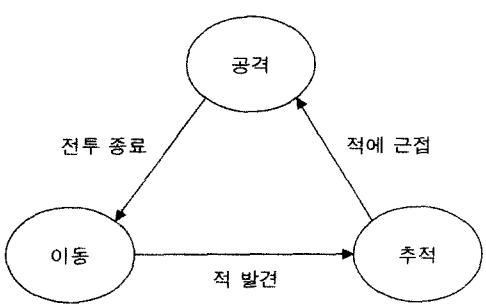
본 논문의 구성은 다음과 같다. 2장에서는 게임에 적용된 기존 알고리즘을, 3장에서는 본 논문에서 제안하는 지능 캐릭터에 대하여 알아본다. 그리고 4장에서는 지능 캐릭터를 검증하기 위해 구현한 대전 액션 게임과 실험결과를 알아보고, 마지막으로 5장에서 결론을 맺는다.

2. 기존 알고리즘

게임에서 사용되는 인공 지능 기법은 FSM(Finite State Machine), FuSM(Fuzzy State Machine)과 같은 전통적인 기법에서부터, 인공생명, 신경망 등에 이르기까지 다양한 기법이 있다. 본 장에서는 이러한 인공 지능 기법들을 FSM과 FuSM, 인공생명, 신경망의 세 부류로 나누어 설명한다.

2.1 FSM과 FuSM

FSM은 상태들 간의 전이에 의해 통제되는 그래프 내에 유한 개의 상태들이 연결되어 있는 규칙 기반 시스템으로서, 현재 가장 널리 사용되는 인공 지능 기법이다[9, 10]. FSM은 단순히 if-else나 switch-case 문장만으로 구현할 수 있기 때문에 이해하기 쉽고 프로그램으로 구현하기 쉬워 널리 사용된다. (그림 1)은 FSM을 이용하여 게임을 구현하는 간단한 예를 보여준다. 그림에서 표시한 바와 같이 캐릭터의 상태를 여러 가지로 나누고, 환경에 따라 상태가 변화하게 된다.



(그림 1) FSM의 예제

예를 들어 캐릭터가 이동 중에 적을 발견하게 되면, 이동 상태에서 추적 상태로 바뀌게 되어 발견한 적을 쫓아가게 된다. 그리고 적이 일정한 거리 내에 들어오면, 공격 상태로 바뀌어 적을 공격하게 된다. 이와 같이 FSM을 이용하면 구현이 쉽고 행동이 정확히 정의되는 장점이 있으나 게임의 진행이 미리 정의된 방식으로만 동작하는 단점이 있다. 즉 게임의 상대방이 FSM으로 구현된 경우 상대방의 행동 양식이 일정하기 때문에 일정시간 게임을 한 후에는 쉽게 예측될 수 있다는 것이다. 상대방의 행동을 예측할 수 있다는 것은 게임의 흥미를 반감시키는 요인으로 작용한다. 이러한 단점을 보완하기 위해 FuSM이 사용되고 있다[9].

FSM에 퍼지 이론을 접목한 FuSM의 경우에는 입력과 출력에 퍼지 함수를 적용하여 어느 정도 무작위적으로 동작할 수 있도록 한다. 무작위 요소가 포함되어 있기 때문에 게임에서의 상대방이 동일한 상황에서 다른 행동을 할 가능성이 있어서 상대방의 행동을 예측하기가 어렵게 된다. 그러나, FSM과 FuSM은 캐릭터의 상태의 수가 적을 때는 간단하게 구현할 수 있지만, 상태의 수가 많아지게 되면, 상태 다이어그램을 정리하기도 어렵고, 프로그램이 급격하게 복잡해지는 단점이 있다. 또한 FSM 과 FuSM 모두 새로운 행동 양식을 추가하기 위해서는 새롭게 프로그램을 해야만 하는 단점이 있다.

2.2 인공생명

인공생명이란 생명체가 나타내는 현상을 컴퓨터, 로봇 등과 같은 인공 매체 상에 재현함으로써 생명의 일반적인 특성에 대해 연구하는 학문이다[11]. 생물학이 생물 현상에 대해 분석적인 방법으로 접근하였다면, 인공생명은 종합적인 방법으로 접근한다. 게임 개발자들은 인공생명 기법을 게임에 적용하려고 오래 전부터 시도를 해왔다. 그러나 인공생명의 예측 불가능한 특성때문에, 제한적으로 사용되어 왔다[4]. 최근 들어 다시 인공생명을 게임에 적용하려는 시도가 되고 있다. 인공생명의 기본적인 특성인 적응성, 창발성 등을 게임에 적용하게 되면, 복잡한 환경이나 사용자의 조작에 적응하거나 전혀 예상치 못한 창발적인 행동으로 인하여 게임의 흥미를 높일 수 있기 때문이다. 그러나 지금까지 연구의 초점은 주로 캐릭터들이 군집을 형성하는 게임에서 전체적인 전략을 결정하는 것에 중점을 둔 연구로서 아직 기초적인 연구에 머물고 있다[8].

2.3 신경망

신경망은 인간의 신경 체제를 모사한 것으로서, 수많은 간단한 콤포넌트들이 연결된 구조를 가지고, 입력되는 데이터의 패턴 인식에 기반하여 출력을 산출한다[12, 13]. 신경망이 게임에 적용되었을 때의 가장 큰 장점으로는, 신경망은 학습 능력을 가지고 있기 때문에 게임을 진행하면서 계속적으로 지능이 향상될 수 있다는 점이다. 그래서 지금까지 신경망

을 게임에 적용하려는 연구가 많이 진행되고 있는데, 대부분이 오목, Tic-Tac-Toe 등 보드 게임이 주류를 이룬다 [7, 14]. 보드 게임은 보드 상에서 움직이는 말들의 전체적인 상황을 인지하여 개개의 말들의 움직임을 결정하는 특성을 가지고 있다. 반면에 대전 액션 게임은 전체적인 상황이 아닌 상대방 캐릭터의 행동에 대하여 적절한 자신의 행동을 결정해야 하는 특성이 있다. 대전 게임에서의 캐릭터를 지능화하기 위한 방법으로는 현재까지 주로 FSM이나 FuSM을 사용하였으며 본 저자들이 지금까지 조사한 바에 의하면 신경망을 적용한 예는 없다.

위에서 살펴본 것과 같이 지금까지 몇몇 연구에서 게임에 인공지능 기법을 적용하려는 시도가 있었다. 그 중 FSM나 FuSM의 방법들은 몇 개의 정해진 행동 패턴만을 가지며, 외부 환경의 변화에 적응하지 못하는 등 지극히 낮은 수준의 지능만을 구현 할 수 있다. 본 연구에서는 이러한 FSM과 FuSM의 한계를 넘어 높은 수준의 지능을 구현할 수 있도록 신경망의 기법을 이용한 지능 캐릭터를 구현한다.

일반적으로 신경망은 학습에 긴 시간이 걸리는 약점을 가진다. 그러나 보통 게임을 서비스하기 전에 한 번만 학습하면 되기 때문에 이 문제는 큰 문제가 되지 않는다. 또한 최근에 개발된 프로세서의 컴퓨팅 파워가 매우 좋아서 학습에 오랜 시간이 소요되지 않는다. 실제로 실험해본 결과, 신경망이 상대방 캐릭터의 행동에 대하여 적절히 행동하는데 필요한 학습 시간은 10분에 불과하였다. 이와 더불어 신경망으로 구현할 경우에는 게임 규칙이나 게임 환경의 변화가 일어난 경우에도 쉽게 적용이 가능하다는 장점을 갖는다. 이러한 점들을 고려하여 우리는 대전 액션 게임의 캐릭터를 지능화하기 위한 방법으로 신경망을 사용하였다.

3. 지능 캐릭터의 구현

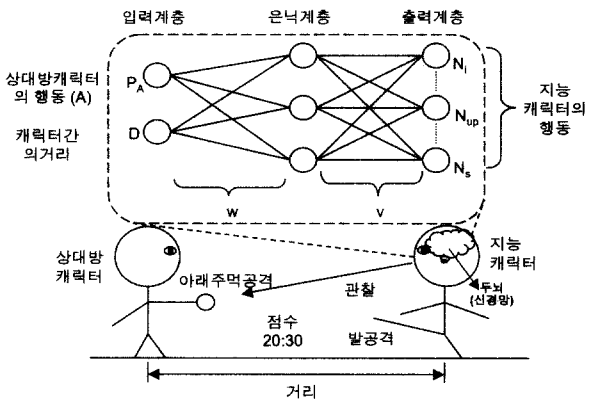
본 논문에서는 게임의 규칙을 스스로 학습할 수 있는 지능 캐릭터를 신경망을 이용하여 구현한다. 이 절에서는 먼저 지능 캐릭터를 구현하기 위한 신경망의 구성에 대하여 설명하고 이어서 신경망을 학습하기 위한 강화 학습 방법을 설명한다.

3.1 신경망의 구성

본 논문에서 적용한 대전 액션 게임의 모델과 이에 따른 신경망의 구조는 (그림 2)와 같다.

(그림 2)에서 보듯이 지능 캐릭터는 상대방 캐릭터와 대결을 통하여 정보를 획득하고, 획득한 정보를 이용하여 학습한다. 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.

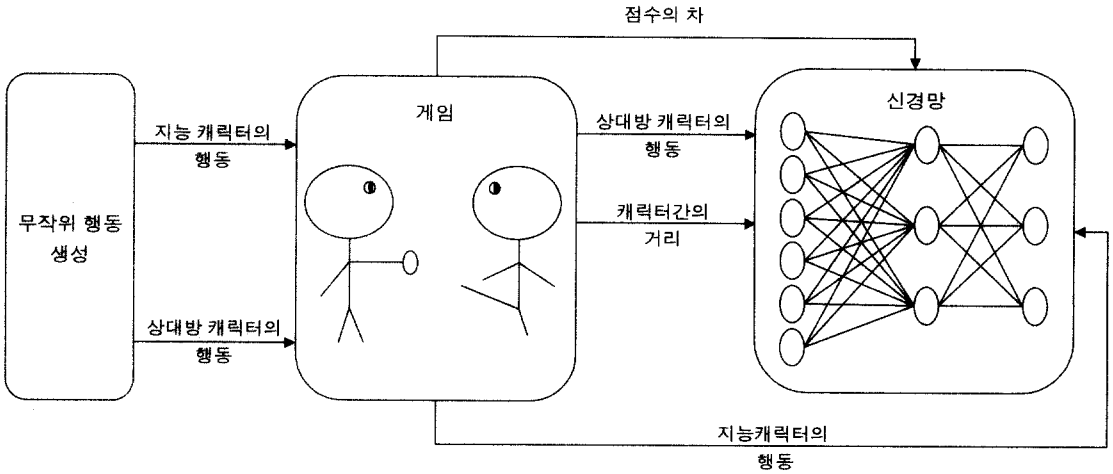
실제 상황에서 어떤 사람 N이 어떤 상대와 대전해야 하는 경우를 생각해 보자. N의 행동은 무엇에 의해서 결정되어야 할까? 만약 N의 대전 능력이 상대에 비해서 월등한 경우라면, 물론 상대의 행동에 상관없이 공격해도 무관할 것이다. 그러나 만약 그렇지 못하다면 상대방이 어떤 행동(A)을 하는 지를 유심히 살펴서 그에 따라서 자신의 행동을 결정해야 한다. 또한 상대방의 같은 행동이라도 상대방과의 거리(D)를 고려해서 대처해야 한다. 상대방이 멀리서 주먹을 휘두르는 것과 가까이에서 주먹을 휘두르는 것은 분명히 다른 상황이다.



(그림 2) 신경망의 구성

본 논문에서 제안하는 지능 캐릭터도 위의 실제 대전에서와 같이 상대방 캐릭터의 행동 (A) 및 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. 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 캐릭터가 주먹공격을 수행 중이라면, 그 순간의 신경망의 입력은 $P_A = 7$, $D = 2$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

1) 주먹 공격이 정수 7로 매핑된다고 가정한다.



(그림 3) 기초 학습 단계

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.

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

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

여기서, $f(x)$는 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.

3.2 오류 역전파 알고리즘을 이용한 강화 학습

(그림 2)와 같이 구성된 신경망을 이용하여 지능 캐릭터를 구현하기 위해서는 게임의 규칙에 따른 학습을 수행하여야 한다. 일반적으로 신경망을 학습하는데는 교사학습(supervised learning)을 사용한다. 교사학습의 경우는 최적의 원하는 출력 값을 알고 있을 때 적용할 수 있는 것으로서 본 논문에서의 경우처럼 어떤 행동이 최적의 행동인지는 알 수 없는 경우에는 사용할 수 없다. 그러나 어떤 행동이 최적의 행동인지는 몰라도 현재 행동에 대한 적절성이 판단될 수 있는 경우에는 그것을 이용한 강화 학습 방법을 사용할 수 있다. 강화 학습이란 에이전트가 환경으로부터 받는 누적 보상 값을 최대화 할 수 있는 최적의 행동 전략을 학습하는 것으로서 본 논문에서는 행동에 의해 발생하는 점수의 차이를 이용하여 신경망을 학습시킨다. 즉 지능 캐릭터의 획득 점수가 상대방 캐릭터가 획득한 점수보다 큰 경우에는, 같은 상황에서 계속 그 행동을 하도록 장려하고, 반대의 경우에는 그 행동을 하지 않도록 학습시킨다.

본 논문에서 제안한 지능 캐릭터는 기초 학습과 실전 학습의 두 단계로 학습한다. 지능 캐릭터는 처음에는 게임에 대한 어떠한 지식도 없기 때문에, 기초 학습 단계에서는 게임의 규칙을 학습하며, 실전 학습 단계에서는 게임 중에 환경이나 상대방의 전략이 변경되었을 경우에 대처하기 위한 학습을 한다.

3.2.1 기초 학습 단계

기초 학습 단계는 게임의 규칙에 대한 사전 지식이 없는 지능 캐릭터가 게임의 규칙을 학습하는 단계이다. (그림 3)은 기초 학습 단계를 도식화한 것이다.

〈표 1〉 기초 학습 단계에서 신경망의 입출력값

상대방 캐릭터의 행동	P_A	지능캐릭터의 행동	두 캐릭터간의 거리 D	지능캐릭터의점수 - 상대방캐릭터의점수
전진	1	주먹	2	1

지능 캐릭터의 훈련 상대인 상대방 캐릭터는 어떠한 행동 방식을 가진 것이라도 무방하지만, 본 논문에서는 학습을 위하여 무작위 행동을 하는 캐릭터를 이용하였다. 학습이 되기 전의 지능 캐릭터의 신경망의 링크 가중치는 무작위 값을 갖고 있다. 즉 지능 캐릭터는 게임의 규칙을 모르는 상태이며 상대방의 공격에 대하여 적절한 행동을 출력

하지 못한다. 그러므로 지능 캐릭터의 행동도 무작위적으로 선택을 하게 한다. (그림 3)에서 보듯이 무작위적으로 선택된 상대방 캐릭터의 행동과 지능 캐릭터의 행동을 게임에 적용한 다음, 게임 상에서 각 캐릭터가 얻은 점수를 강화값으로 이용하여 학습한다. 예를 들어, <표 1>에서 상대방 캐릭터의 행동과 지능 캐릭터의 행동이 무작위로 각각 전진과 주먹으로 결정되었다면, 신경망의 입력 P_A는 전진 행동에 해당하는 정수값 1이 입력되고, 거리 D는 2가 입력된다. 그리고 지능 캐릭터의 행동으로 결정된 주먹에 해당하는 출력 노드를 그 때 발생한 점수의 차 1점으로 강화학습 한다. 즉, 주어진 거리 상에서 상대방 캐릭터의 행동에 대한 지능 캐릭터의 행동이 잘한 것인지 못한 것인지를 지능 캐릭터가 획득한 점수와 상대방 캐릭터가 획득한 점수의 차로써 판단하고, 이를 이용하여 강화학습을 한다. 만약 이 값이 양의 값이면 동일한 상황(상대방 캐릭터의 행동, 상대방 캐릭터와의 거리)에서 해당 행동을 권장하고, 음의 값이면 해당 행동을 선택하지 않게 학습한다. 물론, 점수차의 절대값이 클수록 크게 강화된다. 무작위 행동 생성이 모든 상황(상대방 캐릭터의 행동×상대방 캐릭터와의 거리×지능 캐릭터의 행동)에 대한 것을 생성하지 않기 때문에 많은 학습을 하더라도 최적의 행동을 찾는다는 것을 보장하지는 못한다. 상대방 캐릭터의 행동, 지능 캐릭터의 행동, 두 캐릭터 간의 거리 및 행동의 결과로 나온 점수차(강화값)가 모두 계산되면, 이를 이용하여 오류 역전파 알고리즘으로 학습을 수행한다. 오류 역전파 알고리즘을 이용한 학습은 신경망을 구성하는 각 링크의 가중치를 각 계층별로 수정한다. 먼저 출력 계층과 은닉 계층 사이의 링크의 가중치는 식 (3), 식 (4)에 의해 수정된다.

$$d_j = z_j \cdot (1 - z_j) \cdot score \tag{3}$$

$$v_{ij} += \alpha(t) \cdot d_j \cdot h_i \tag{4}$$

여기서, *score*는 점수의 차를 의미하며, z_j는 j번째 출력 노드, v_{ij}는 i번째 은닉 노드와 j번째 출력 노드간의 링크 가중치, $\alpha(t)$는 학습률 함수를 나타낸다.

식 (3)에서 *score*는 출력 노드별로 다르게 주어진다. 즉, 지능 캐릭터의 행동으로 결정된 출력 노드의 *score*는 점수의 차로, 그 외의 출력 노드들은 그 점수의 차의 부호를 반대로 하여 적용한다. 이렇게 함으로써 점수의 차가 양수이면 해당 출력 노드는 다음에도 동일한 상황에 대해 같은 출력이 나오도록 수정하고, 반면에 다른 노드들은 그 노드들이 출력되지 않는 방향으로 링크의 가중치를 수정한다. 점수의 차가 음수이면 위와는 반대로 링크의 가중치가 수정된다.

식 (4)에서 $\alpha(t)$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.

다음으로 입력 계층과 은닉 계층 사이의 링크의 가중치는 식 (5), 식 (6)을 이용하여 수정된다.

$$d_j = h_j \cdot (1 - h_j) \cdot \left(\sum_{k=0}^{Nh} w_{jk} d_k\right) \tag{5}$$

$$w_{ij} += \alpha(t) \cdot d_j \cdot x_i \tag{6}$$

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.

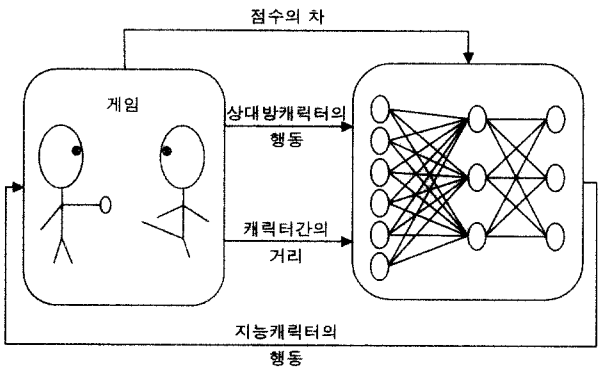
강화 학습이 진행됨에 따라서 지능 캐릭터의 신경망은 특정한 상황(즉, 상대방 캐릭터의 행동과 거리)에 따라서 어떤 행동이 적절한 행동인지를 학습하게 된다. 물론 모든 경우에 대한 최적의 행동을 찾지는 못하지만 학습기간이 길어질수록 최적의 행동을 찾을 가능성은 높아진다.

3.2.2 실전 학습 단계

본 논문에서는 기초 학습 단계가 완료된 후에도, 필요에 따라 실제 게임을 진행하면서도 계속해서 학습할 수 있도록 실전 학습 단계를 둔다. 온라인 게임의 경우, 게임이 서비스되고 있는 중에도 게임이 변경될 수 있다. 만약 게임의 규칙이 소폭 변경되었다면, 기초 학습 단계를 거쳐 대부분의 게임 규칙을 학습한 지능 캐릭터를 실전 학습 단계를 이용하여 게임 서비스를 중단시키지 않고도 변경된 게임 규칙을 학습시킬 수 있다. 여기서, 지능캐릭터가 게임의 규칙이 변경되었다는 것을 알아내는 방법은 본 논문의 주제와 거리가 있으므로 다루지 않는다.

(그림 4)는 실제 게임 중에 신경망이 학습되는 상황을 보여준다. (그림 3)의 기초 학습 단계와는 달리 실제 게임 중이기 때문에 지능 캐릭터의 행동이 무작위로 생성되는 것이 아니라 신경망의 출력값에 의해 결정된다. 신경망에 상대방 캐릭터의 행동과 현재 두 캐릭터 간의 거리가 입력되면 신경망은 현재 학습된 상황에서 지능 캐릭터에게 유리하다고 판단되는 행동을 수행하고, 그 결과 점수 차이를 이용하여 다시 강화 학습을 하게 된다. 이외의 신경망 학습의 구체적인 수식이나 방법은 기초 학습 단계와 같다. 이러한 실전 학습은 게임의 규칙이 일부 변경되더라도 지능 캐릭터가 이를 적응해 적절히 반응하는데 도움이 된다. 예를 들어, 점수의 규칙이 바뀐 경우 기초 학습 단계에서와 동일한 상황(상대방 캐릭터의 행동, 상대방 캐릭터와의 거리, 지능 캐릭터의 행동)에서 점수의 차가 바뀌게 되고 새로운 점수의 차로 학

습하게 되므로 서서히 새로운 점수 체계에 따른 적절한 행동을 하게끔 신경망이 학습되게 된다. 예를 들어 점수의 규칙이 바뀌기 전 점수의 차가 +10이었던 것이 점수의 규칙이 바뀐 후 동일한 상황에서 -20이 되었다고 하자. 그러면 점차적으로 해당 행동을 선택하지 않는 쪽으로 학습하게 되어 일정 기간 동안 동일한 상황에 대해 학습이 이루어진다면 신경망은 점수의 규칙이 바뀌기 전과는 다른 선택을 하게 될 것이다. 점수의 규칙이 바뀌지 않은 경우에는 계속적으로 기초 학습 단계에서와 동일한 점수의 차가 나오고 이것이 계속 학습된다. 이 경우 기초 학습 단계에서 최적을 찾지 못했다면 최적의 행동을 찾기가 매우 어렵다. 이를 위해 가끔 신경망을 적용한 선택이 아닌 다른 방법으로 선택을 하는 새로운 학습 방법이 필요하다. 새로운 학습 방법에 대한 연구는 이 논문의 초점을 벗어나는 것으로 추후 연구에서 수행할 예정이다.



(그림 4) 실전 학습 단계

〈표 2〉 실전 학습 단계에서 신경망의 입출력값

상대방 캐릭터의 행동	P_A	지능캐릭터의 행동	두 캐릭터간의 거리 D	지능캐릭터의점수 - 상대방캐릭터의점수
주먹	7	전진	2	-1

실전 학습 단계에서 신경망의 입출력 값은 <표 2>와 같이 결정된다. 상대방 캐릭터의 행동이 주먹이고 두 캐릭터간의 거리가 2라면, 신경망의 입력 P_A는 주먹 공격에 해당하는 7이 입력되고, D는 2로 입력된다. 신경망은 이 입력값을 이용하여 출력 노드들을 계산하고, 그 결과 전진에 해당하는 노드가 가장 큰 값을 출력하면 지능 캐릭터의 행동은 전진으로 결정된다. 두 캐릭터의 행동과 거리가 결정되면, 이 내용을 게임에 반영한다. 그 결과 점수의 차가 -1점이라면, 즉 지능 캐릭터의 행동이 좋지 않은 행동으로 판단되면, 지능 캐릭터는 다음에는 이 행동을 하지 않도록 강화학습 한다.

4. 실험 및 결과

4.1 실험 환경

본 논문에서 제안한 지능 캐릭터를 검증하기 위하여 간단한 대전 액션 게임 모델을 개발했다. 게임은 두 캐릭터가 제한된 1차원 게임 공간에서 이동하면서 공격과 방어를 하며, 그 결과에 따라 점수를 획득한다. 캐릭터가 할 수 있는 행동은 11가지로 다음과 같다.

정지, 전진, 후진, 막기, 점프, 앉기, 아래/위 주먹공격, 아래/위 발공격, 필살기

게임의 구성을 간단하게 하기 위해서, 두 캐릭터의 모든 행동은 동시에 시작해서 동시에 끝나는 것으로 가정한다. 그리고 행동하는데 소요되는 시간은 논리적으로 1이라고 가정한다. 이는 게임이 클럭 틱에 의해 관리되며, 캐릭터의 행동은 클럭에 동기가 맞춰져 있고, 한 클럭에 모든 행동이 끝난다는 의미이다.

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 3>은 각 공격의 속성을 표시한 것이다.

〈표 3〉 각 공격의 공격점수/유효거리

공격행동	공격점수	유효거리
아래주먹	1	0~2
위주먹	2	
아래발	3	2~3
위 발	4	
필살기	5	3~5

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

4.2 평가 기준

지능 캐릭터가 게임의 규칙을 학습했는지 여부를 판단하기 위해서는 판단의 기준이 필요하다. 본 논문에서는 두 가지 기준을 이용하여 학습 여부를 판단한다. 첫 번째는 상대방 캐릭터의 행동에 따라 지능 캐릭터가 어느 정도 적절한 행동을 했는지를 판단하는 것이다. 즉, 상대방 캐릭터가 특정한 행동을 수행했을 때 지능 캐릭터가 취할 수 있는 모든 행동에 대하여 획득 가능한 점수와 지능 캐릭터가 실제로 획득한 점수를 비교 평가하는 것이다. 두 번째 기준은 게임을 일정 시간 수행한 후 상대방 캐릭터와 지능 캐릭터가 얻은 획득 점수의 비를 비교한다.

첫 번째 평가 기준인 행동의 적절성을 판단하기 위해서

거리와 공격 방법에 따른 최대 점수와 최소 점수가 계산되었다. 예를 들어, 두 캐릭터 간의 거리가 2에 해당하고, 상대방 캐릭터가 아래주먹 공격을 해서 1점을 획득했다고 가정하자. 이 때, 지능 캐릭터가 얻을 수 있는 가장 큰 점수는 위발 공격을 했을 때인 4점이므로, 획득 점수의 차는 최대 3이 된다. 반면에 지능 캐릭터가 전진을 하게 되면, 지능 캐릭터는 점수를 얻지 못하여 최저 획득 점수의 차는 -1이 된다. 각각의 경우에 대하여 획득 점수의 차의 범위를 3등분하여 상, 중, 하로 정했다. 예를 들어 거리 2에서 상대방 캐릭터가 아래주먹 공격을 했을 경우 나타날 수 있는 획득 점수의 차의 범위는 -1에서 3 사이이다. 지능 캐릭터가 어떤 행동을 하여 발생한 획득 점수의 차가 1점이라면, "중"으로 판단한다.

4.3 실험 결과

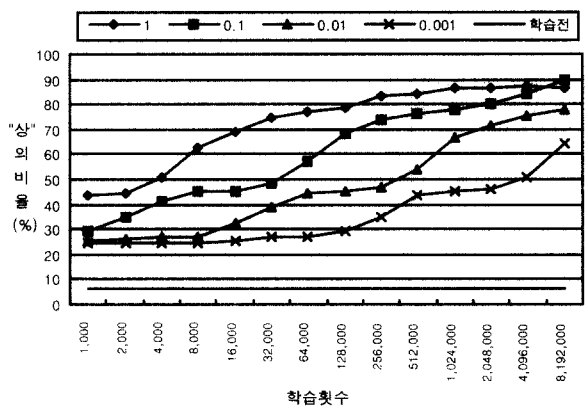
신경망의 은닉 계층 노드의 수는 15로 하여 기초 학습 단계에 대한 실험을 하였다. 상대방 캐릭터의 행동과 지능 캐릭터의 행동을 모두 무작위로 선택하여 일정 횟수 학습시킨 후, 무작위로 행동하는 상대방 캐릭터와 일정 횟수 대결시켜 지능 캐릭터의 행동 중 "상"이 차지하는 비율(%)을 조사했다. 학습률은 0.001~1.0, 학습 횟수는 1000~8192000, 그리고 각 경우에 대하여 10개의 무작위 초기 값에 대하여 수행한 후 "상"의 비율의 평균값과 표준편차를 조사하였다. <표 4>에서 "학습전"은 지능 캐릭터가 학습되지 않았을 때 (즉 지능 캐릭터 신경망의 링크가 무작위적으로 주어진 상태) 무작위로 행동하는 캐릭터와 대표적으로 10000번 행동했을 경우의 결과이다.

〈표 4〉 "상"의 비율

(% : 평균/표준편차)

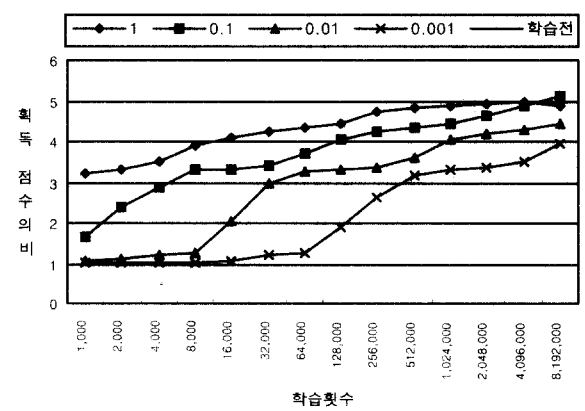
학습률 / 학습횟수	1.0	0.1	0.01	0.001	학습전
1000	43.4/0.8	29.6/5.8	25.1/4.4	24.4/4.2	6.0
2000	44.4/0.1	35.1/1.0	26.1/4.7	24.4/4.2	
4000	51.1/4.5	41.5/2.6	27.0/5.0	24.6/4.2	
8000	62.5/1.0	45.2/3.4	27.0/5.0	24.7/4.3	
16000	68.8/1.1	45.5/3.3	32.2/1.5	25.5/4.5	
32000	75.0/0.2	48.3/4.7	39.0/1.7	26.8/5.0	
64000	77.1/0.4	56.9/3.3	44.8/2.7	27.1/5.0	
128000	78.6/1.1	67.9/1.4	45.1/3.4	29.7/2.0	
256000	83.4/0.7	74.2/1.1	47.1/5.1	35.3/2.0	
512000	84.1/0.3	75.9/0.5	53.7/4.3	43.6/2.1	
1024000	86.3/1.0	77.8/0.0	67.0/1.7	45.3/3.3	
2048000	86.6/0.5	79.8/0.7	71.7/1.3	45.8/4.5	
4096000	86.9/0.2	84.3/0.4	75.5/0.6	51.0/5.1	
8192000	86.5/1.1	89.6/0.9	77.4/0.0	64.0/1.8	

<표 4>는 학습률 및 학습 횟수에 따른 "상"의 비율을 보여주며 (그림 5)는 표를 그래프로 나타낸 것이다. (그림 5)를 보면 모든 경우에서 학습 전의 지능 캐릭터보다는 학습 후에 "상"의 비율이 증가했고, 또한 같은 학습률에 대해서는 학습 횟수가 많아질수록 "상"의 비율이 증가하는 것을 볼 수 있다. 이와 같은 실험결과로 볼 때 본 논문에서 제안한 지능 캐릭터가 학습할수록 게임의 규칙을 많이 학습하여 자신에게 유리한 행동을 많이 한다는 사실을 알 수 있다. (그림 5)에서 학습률이 1일 때는 학습 횟수가 증가할수록 "상"의 비율이 감소하는 경우도 있다. 이런 현상이 나타나는 이유는 학습률이 높은 경우 학습 횟수가 더 많아지면 경우에 따라서 학습 결과가 나빠지는 경우가 발생하기 때문이다. 그렇지만 전체적으로는 증가하는 경향을 보이고 있다.



(그림 5) "상"의 비율

<표 5>는 본 논문의 두 번째 평가 기준인 지능 캐릭터와 상대방 캐릭터가 획득한 점수의 비를 나타낸 것이며 (그림 6)은 이를 그래프로 표현한 것이다.



(그림 6) 획득 점수의 비

〈표 5〉 획득 점수의 비

(평균/표준편차)

학습률 / 학습횟수	1.0	0.1	0.01	0.001	학습전
1000	3.2/0.1	1.7/0.3	1.1/0.1	1.0/0.1	0.0
2000	3.3/0.0	2.4/0.1	1.1/0.1	1.0/0.1	
4000	3.5/0.1	2.9/0.1	1.2/0.2	1.0/0.1	
8000	3.9/0.0	3.3/0.1	1.3/0.2	1.0/0.1	
16000	4.1/0.0	3.3/0.1	2.1/0.2	1.1/0.1	
32000	4.3/0.0	3.4/0.1	3.0/0.0	1.2/0.2	
64000	4.3/0.1	3.7/0.1	3.3/0.1	1.3/0.2	
128000	4.5/0.0	4.1/0.0	3.3/0.1	1.9/0.2	
256000	4.7/0.1	4.3/0.0	3.4/0.1	2.7/0.1	
512000	4.8/0.1	4.3/0.0	3.6/0.1	3.2/0.0	
1024000	4.9/0.1	4.5/0.0	4.0/0.0	3.3/0.1	
2048000	4.9/0.1	4.6/0.0	4.2/0.0	3.3/0.1	
4096000	5.0/0.0	4.9/0.1	4.3/0.0	3.5/0.1	
8192000	4.9/0.0	5.1/0.0	4.4/0.0	3.9/0.0	

〈표 6〉 최적 행동

거리 \ 상대방의 행동	0	1	2	3	4	5	6	7	8	9	10
0	7	7	7	Any	7	6	7	7	7	7	7
1	7	7	7	Any	7	6	7	7	7	7	7
2	9	9	9	Any	9	8	9	9	9	9	9
3	10	10	10	10	10	10	10	10	10	10	10
4	10	10	10	10	10	10	10	10	10	10	10
5	10	10	10	10	10	10	10	10	10	10	10
6	Any										{1}c

정지 0
전진 1
후진 2
막기 3
점프 4
앉기 5
아래주먹 6
위주먹 7
아래발 8
위발 9
필살기 10

실험 결과들을 보면, 첫 번째 기준에 대한 실험과 유사하게 모든 경우에서 학습 전의 지능 캐릭터보다는 학습 후에 획득 점수의 비가 증가했고, 또한 같은 학습률에 대해서는 학습 횟수가 많아질수록 획득 점수의 비가 증가하는 것을 알 수 있다. 이와 같은 결과는 지능 캐릭터가 학습을 거듭할수록 많은 게임의 규칙을 학습하여 상대방 캐릭터보다 많은 점수를 획득할 수 있음을 보여준다. 그리고 신경망이 8192000번 학습하여 게임 규칙을 학습하는데 소요되는 시간은 펜티엄 IV 2.8 GHz에서 평균 10분 정도여서 학습 속도가 상당히 빠르다는 것을 알 수 있다.

지능 캐릭터가 학습이 제대로 되는지를 보다 자세히 알아보기 위하여 특정한 상황에서 지능 캐릭터가 취하는 행동을 최적 행동과 비교해 보았다. <표 6>은 게임 규칙에 지능 캐릭터의 최적 행동이고, <표 7>은 학습률 1.0에서 약 250만번 학습시켰을 경우의 상대방 캐릭터의 행동과 거리에 대한 지능 캐릭터의 행동을 나타낸 것이다.

〈표 7〉 학습 후의 행동

거리 \ 상대방의 행동	0	1	2	3	4	5	6	7	8	9	10
0	7	7	7	7	7	7	7	7	7	7	7
1	7	7	7	9	9	9	7	7	7	7	7
2	9	9	9	10	10	8	9	9	9	9	9
3	10	10	10	10	10	10	10	10	10	10	9
4	10	10	10	10	10	10	10	10	10	10	10
5	10	10	10	10	10	10	10	10	10	10	10
6	10	10	10	10	10	10	10	10	10	10	10

정지 0
전진 1
후진 2
막기 3
점프 4
앉기 5
아래주먹 6
위주먹 7
아래발 8
위발 9
필살기 10

앞서 설명한 본 대전 액션 게임의 규칙을 분석해보면 두 캐릭터 간의 거리가 0, 1일 때에는 위주먹(7)을, 2일 때에는 위발(9)을 행동하는 것이 가장 좋다. 그러나 상대방 캐릭터가 막기(3)를 했을 때는 지능 캐릭터가 어떤 공격을 하더라도 점수를 획득할 수 없기 때문에 특정한 최적 행동이 존재하지 않고, 상대방 캐릭터가 앉기(5)를 했을 때는 아래 공격만이 유효하기 때문에 각각 아래주먹(6), 아래발(8) 공격이 최적 행동이다. 또한, 두 캐릭터 간의 거리가 3~5일 때에는 필살기(10)을 행동하는 것이 가장 좋다. 그리고 6 이상의 거리에서는 어떤 행동을 하든 점수가 발생하지 않기 때문에 학습 여부를 판단할 때에 영향을 주지 않는다. 그러나 거리가 6이고 상대방 캐릭터가 필살기를 했을 경우, 지능 캐릭터가 전진을 하면 필살기 공격의 유효거리에 들어가기 때문에 전진이 아닌 어떤 행동이라도 최적 행동으로 볼 수 있다. <표 7>의 결과를 보면 대부분의 경우 이러한 분석과 일치되게 학습되었음을 알 수 있다. 그러나 다섯 가지 경우에서는 최적의 행동을 학습하지 못한 것이 나타나는데 이 곳은 표에서 음영으로 나타낸 부분이다. 이러한 부분도 학습이 더욱 진행되면 최적의 행동을 학습할 수 있다.

5. 결 론

본 논문에서는 신경망을 이용하여 게임 내에 존재하는 게임 캐릭터에 지능을 부여하는 방법을 제안하였다. 제안한 알고리즘은 게임 규칙의 학습을 위하여 상대방 캐릭터와 지능 캐릭터 사이의 점수의 차를 이용하여 강화 학습하였다. 제안한 방법을 테스트해 보기 위하여 간단한 대전 액션 게임을 개발 적용하여 실험하였다. 대전 액션 게임에 적용해 본 결과 무작위로 행동하는 캐릭터보다 최대 5.1배의 점수를 획득함을 보았다. 구현을 간단히 하게 하기 위하여 본 논문에서 사용한 대전 액션 게임은 한 클럭에 모든 행동이 끝나는 것으로 가정하였다. 이러한 가정은 실제게임과는 약간 차이가 있는 것으로 차후 연구에서는 보다 실제적인 게임 상황에서의 지능 캐릭터를 구현할 예정이다. 또한 게임 중의 학습을 위한 방법 그리고 상대방이 어떤 특이한 공격 성향을 보일 때 이에 대처하는 방법 등에 대한 연구를 추후에 수행할 예정이다.

참 고 문 헌

[1] Darrin C. Bentivegna, Ales Ude, Christopher G. Atkeson, Gordon Cheng, "Humanoid Robot Learning and Game Playing Using PC-Based Vision," IEEE/RSJ Intl. Conference on Intelligent Robots and Systems EPFL, Lausanne, Switzerland, October, 2002.

[2] Stephen Cass, "Mind Games : to Beat The Competition Video Games are Getting Smarter," IEEE Spectrum Online, Dec., 2002.

[3] 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.

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

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

[6] 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.

[7] 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.

[8] 조남덕, 성백균, 김기태, "인공생명 시뮬레이션을 통한 게임 캐릭터의 전략 구현", 정보과학회 2000년 춘계학술대회, Vol. 27, No.01, pp.0241-0243, April, 2000.

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

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

[11] Langton, C., Studying artificial life with cellular automata Physica D, Vol.22, pp.120-149, 1986.

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

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

[14] David B. Fogel, Using Evolutionary Programming to Create Neural Networks that are Capable of Playing Tic-Tac-Toe, in Proc. 1993 Int. Joint Conf. Neural Networks(IJCNN'93), pp.875-880, 1993.

조 병 헌

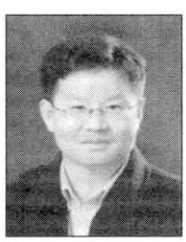
e-mail : d995552@hanmail.net

1997년 국민대학교 전자공학과(공학사)

1999년 국민대학교 전자공학과(공학석사)

1999년~현재 국민대학교 전자정보통신 공학부 박사과정

관심분야 : 유전자 알고리즘, 시뮬레이션, 신경망



정 성 훈

e-mail : shjung@hansung.ac.kr

1988년 한양대학교 전자공학과(공학사)

1991년 한국과학기술원 전기및전자공학과 (공학석사)

1995년 한국과학기술원 전기및전자공학과 (공학박사)

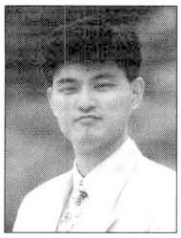
1995년~1996년 한국과학기술원 전기및전자공학과(위촉연구원)

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



성 영 락

e-mail : yeong@kookmin.ac.kr

1989년 한양대학교 전자공학과(공학사)

1991년 한국과학기술원 전기 및 전자공학과 (공학석사)

1995년 한국과학기술원 전기 및 전자공학과 (공학박사)

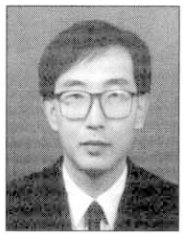
1995년~1996년 한국과학기술원 위촉연구원

1996년~1998년 국민대학교 전자공학부 전임강사

1998년~2002년 국민대학교 전자공학부 조교수

2002년~현재 국민대학교 전자정보통신공학부 부교수

관심분야 : 시뮬레이션, 고장감내, 내장형 시스템



오 하 령

e-mail : hroh@kookmin.ac.kr

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