Journal of The Korean Association
of Information Education Vol. 21, No. 1,
February 2017, pp. 139-149

http://dx.doi.org/10.14352/jkaie.2017.21.1.139

초등학생의 인공지능 교육을 위한 교수 학습 모델 개발 및 적용

김갑수* · 박영기**

서울교육대학교 컴퓨터교육과* · 춘천교육대학교 컴퓨터교육과**

요 약

21세기 지식 정보 사회에 인공지능 교육이 매우 중요하다. 4차 산업혁명 시대에 컴퓨터 교육에서 인공지능을 이해하고 컴퓨터 프로그래밍 교육을 해 보는 것이 매우 중요하지만 인공지능에 대해서 이해하고, 컴퓨터 프로그래밍 교육을 하는 교수 학습 모델이 없다. 본 연구에서는 제안하는 모델은 문제 이해 단계, 데이터 정리하기 단계, 인공지능 모델 정하기 단계, 프로그래밍하기 단계, 보고서 작성하기 단계로 구성된다. 프로그래밍하기 단계에서는 학생들의 수준에 적합하게 복사하기, 변형하기, 창조하기, 도전하기로 나눌 수 있다. 본 연구에서는 초등학교 교사들의 델파이 평가로 모델의 타당도를 입증하였고, 그에 따라 초등학생들이 쉽게 이해할 수 있는 사례를 만들었다. 본 연구의 결과는 초등학생들에서 인공지능 프로그램을 실습해 볼 수 있는 좋은 기회를 제공한다.

키워드 : 인공지능, 인공신경망, 초등학생, 프로그래밍 교육, 교수학습 모델

A Development and Application of the Teaching and Learning Model of Artificial Intelligence Education for Elementary Students

Kapsu Kim* · Youngki Park**

Dept. of Computer Education, Seoul National University of Education* ·
Dept. of Computer Education, Chuncheon National University of Education**

ABSTRACT

Artificial intelligence education is very important in the 21st century knowledge information society. Even if it is very important to understand artificial intelligence and practice computer programming in computer education in the fourth industrial revolution, but there is no teaching and learning model to understand artificial intelligence and computer programming education. In this paper, the proposed model consists of problem understanding step, data organizing step, artificial intelligence model setting step, programming step, and report writing step. At the program step, students can choose to copy, transform, create, and challenge steps to their level. In this study, the validity of the model was proved by the Delphi evaluation of elementary school teachers. The results of this study provide a good opportunity for elementary school students to practice artificial intelligence programs.

Keywords : Artificial intelligence, artificial neural network, elementary school students, programming education, teaching and learning model

이 논문은 2016년도 교내연구비에 의하여 연구되었음.
교신저자 : 박영기(춘천교육대학교 컴퓨터교육과)
논문투고 : 2016-12-00
논문심사 : 2017-00-00
심사완료 : 2017-00-00

1. 서론

현재 지식 정보 사회를 넘어 인공지능 사회로 접어들고 있고 우리 교육에도 새로운 변화가 있어야 한다. 2016년 3월 알파고 이후 국내에서 인공지능을 활용한 산업은 크게 변화하고 있지만, 교육의 변화는 상대적으로 미미한 수준이다.

우리 사회는 제4차 산업의 진입으로 초등학생들에게도 인공지능 교육 경험이 필요하다. 그러나 아직도 우리나라 교육은 인공지능이나 4차 산업을 이야기하면서 초등학교 교육과정에서 인공지능 교육 프로그램을 경험할 수 없을 뿐만 아니라 소프트웨어 교육도 2015년 개정 교육과정에서 초등학교 6년 동안 17시간을 제시하고 있다. 그래서 소프트웨어 교육과 더불어 21세기에 주도적인 삶을 살아가기 위해서 초등학생들이 인공지능 교육 프로그램을 직접 경험해 보는 것이 필요하다.

엘런 튜링은 1950년에 발표한 computing machinery and Intelligence[7,8,20]에서 컴퓨터는 생각할 수 있을까? 라는 질문을 던지며 이미테이션 게임을 제안한다. 이 게임에는 심판관과 두 명의 참가자가 있다. 이때 두 명의 참가자는 컴퓨터와 사람이다. 심판관이 질문하면 컴퓨터와 사람이 응답하는데, 어떤 것이 사람인지 구별하는 것이다. 심판관이 컴퓨터와 사람의 답변을 일관성 있게 구별하지 못하면 컴퓨터가 이긴 것으로 간주한다. 1966년 미국의 조셉 와이젠바움은 인공지능 시스템인 엘리자를 만들어 이와 같은 실험을 수행했다. 환자와 상담가가 대화하는 모습을 정한 후에 엘리자가 상담자 역할을 하는 것이다. 환자는 엘리자가 기계인지 사람인지 구별하는 것이고 많은 환자들은 엘리자를 사람으로 생각하였다고 한다. 미국인 언어 철학자 존 설은 컴퓨터가 튜링 테스트에 합격해도 생각한다고 할 수 없다고 하였다. 예를 들어, 다음과 같은 중국어 방의 사고 실험을 생각해 보자. 영어를 모르는 중국인이 방에 있고, 그 방의 창문으로 심판관이 영어로 된 문장을 전달하면, 그 중국인이 어떤 규칙에 따라 영어로 된 답변을 작성하여 다시 창문으로 심판관에게 전달하는 것이다. 존 설에 따르면 만약 심판관이 이 사람이 영어를 아는 사람이라고 생각했더라도, 우리는 이 중국인이 영어를 아는 사람이라고 볼 수 없다는 것이다. 이 주장에 대해서는 많은 반론이 존재한다. 1991년부터는 뢰브너상이 주어지는 채터봇대회가 열리고 있다.

인공지능 프로그램과 컴퓨터 프로그램의 근본적인 차이점을 초등학생들이 이해하고 실제 인지해야 한다. 기존의 컴퓨터 프로그램은 사람이 설계한 알고리즘대로 수행하는 것이고, 사람의 규칙과 능력을 뛰어 넘을 수 없는 것이다. 그러나 인공지능 프로그램은 스스로 학습하여 판단하는 것이다. 따라서 초등학생들이 인공지능이 어떤 것인지를 이해하는 것이 4차 산업 사회에서 매우 중요하고 필요하다.

2015년 개정 교육과정[14]에서 소프트웨어 교육을 실시하고 있기 때문에서 인공지능 이론을 컴퓨터 프로그램으로 해결하는 것이 매우 중요하다. 김갑수[10]는 Miltiadou와 Yu[13]가 개발한 것을 변형하여 컴퓨터 프로그램 교육을 받을수록 자기 효능감이 좋게 때문에 단순한 인공지능을 개념을 학습하게 하는 것보다 실제 컴퓨터 프로그램으로 해결하게 하는 것이 좋다.

본 연구에서는 초등학생들에게 인공지능 개념을 학습하게 하고 실제 컴퓨터 프로그램을 해 보는 학습 모형을 개발하여 실제로 인공지능을 이해할 수 있는 프로그램을 개발한다.

컴퓨터 프로그래밍 교육에서는 학생, 교사, 프로그래밍 도구의 3의 객체간의 관계가 매우 중요하기 때문에 자기조절 학습이 매우 중요하여 본 연구에서는 자기조절 학습 모형을 이용한다.

제2장에서는 관련 연구를 자세히 설명하고, 제3장에서는 제안한 인공지능 학습을 위한 교수 학습 모델을 설명하고, 제4장에서 제안한 모델을 초등학교에 적용할 수 있는 예제를 만들었고, 제5장은 본 연구의 결론이다.

2. 관련 연구

2.1 인공지능

1956년 여름 다트머스 대학에서 존 맥카시(John McCarth) 등 여러 학자들이 생각하는 기계에 대한 토론을 하면서 새로운 이름으로 존 맥카시가 제안한 인공지능(Artificial intelligence)이라는 이름으로 사용하기로 하였다. 인공지능은 매우 많은 분야로 발전하여 왔다.

1959년 아서 사무엘(Arthur Samuel)은 명시적으로

프로그래밍 하지 않고 컴퓨터에 학습 능력을 부여한 것으로 정의하여 기계 학습을 인공지능이라고 하였다. 이 때에 주로 게임에 많이 이용되었고, 체스 게임이 대표적이며 알파 베타 가치치기(alpha-beta evaulation pruning), 민맥스(minmax) 알고리즘 등이 사용되었다. 톰 미첼(Tom M. Mitchell)은 어떤 일을 성능 P로 측정가능하고, 성능이 어떤 경험으로부터 개선된 프로그램을 기계학습으로 정의한다. 여기서 경험으로부터 어떤 일의 성능이 향상된다는 것이다.

인공지능과 프로그램은 어떤 차이가 있는지 살펴본다. 인공지능(artificial intelligence)은 사람의 뇌를 흉내되어 고안한 인공 신경망(artificial neural networks)을 이용하는 것이 보편적이다. 프로그램은 기본적으로 어떤 목적에 적합하게 사람이 지시한 명령어들을 수행하는 것에 있지만 인공지능은 어떤 목적으로 개발하는 것이 아니라 사람의 신경망을 구현하는 것이다. 야후, 페이스북, 구글, IBM, 마이크로소프트 등은 개발한 인공지능으로 바둑, 퀴즈풀이, 얼굴인식, 음성인식 등에 대해서 끊임없이 도전한다. 알파고가 모든 경우의 수를 다 입력하는 것이 아니라 스스로 기력을 향상시키는 것이다.

프로그램은 개발한 개발자의 구현한 알고리즘 이상으로 성능을 높이지 못한다. 성능을 향상시키고 싶으면 개발자가 모든 것을 개선해야 한다. 인공지능은 기계학습을 통해서 개발자가 생각한 것 이상으로 성능을 향상시킬 수 있다.

최근 구글, 페이스북, 야후 등은 인공지능 소프트웨어를 개발하여 오픈 소스로 공개하고 있다. 야후는 딥러닝 기법을 이용한 '카페온스파크(CaffeOnSpark)' 인공지능 소프트웨어를 오픈소스화했다. 카페온스파크는 이미지 처리 등 각각 데이터를 분석할 수 있는 기능을 오픈소스화 하였다. 이 오픈소스를 이용하여 음성 패턴, 이미지 패턴 등을 식별할 수 있다. 페이스북은 오픈소스인 빅서(Big Sur) 인공지능 프로그램을 공개했고, 여기서 딥러닝 모듈인 토치(Torch)를 이용하여 인공지능 프로그램을 할 수 있다. 구글은 텐서플로우(Google TensorFlow) 인공지능 프로그램을 오픈소스화하여 인공지능 프로그램을 쉽게 만들 수 있게 하였다. 이 소프트웨어는 기계학습을 위해서 만들어졌고, 데이터 플로우 그래픽 방식(Data Flow Graph) 방법을 이용하는 것이 특성으로 그래프 형식으 데이터를 표현할 수 있어서 매우 유익하다.

2.2 자기 조절 학습

자기 조절 능력(Self-Regulated Learning)은 스스로 학습을 관리하는 능력으로서 학습 계획, 목표 설정, 전략 실행 및 요약 점검 과정이다[16,17,19]. Zimmerman와 Martinez-Pons[21]는 자기조절학습과 학업성취도가 상관관계가 있고, 또한 과제 수행 결과가 좋다는 것을 검증하였기 때문에 자기조절 학습이 매우 중요하다. 자기 조절 학습 능력은 컴퓨터 교육과 관련 있는 분야에 적용한 것은 디지털 게임에 학생들의 자기 조절 학습 능력을 적용한 사례는 많았다[4,5,6]. 김갑수[9]는 컴퓨터 프로그래밍 언어 교육을 하면 창의성과 논리성을 개발할 수 있다고 하였고, Salomon G와 Perkins[18]는 컴퓨터 프로그래밍 교육이 학생들의 인지 능력의 전이가 빨리 일어난다고 하였고, 컴퓨터 프로그래밍 교육으로 수학적 기하학적 개념과 원리 향상할 수 있다고 하였고, Nong Ye과 Gavriel Salvendy[15]는 컴퓨터 프로그래밍을 수행할 때에는 목적(objective), 개념(conceptual), 기능(functional), 논리(logical), 및 물리(physical)적인 추상화의 단계로 컴퓨터 프로그래밍을 하는 것이 중요하다고 하였다. 컴퓨터 프로그래밍을 통하여 인공지능 프로그램을 만들어 보는 것은 매우 의미 있다.

자기조절 학습(SRL)은 신념, 동기, 전략, 반성의 과정 즉 학습자 스스로 자신의 학습과정을 주도적으로 지시하는 것이다[1]. 이 학습은 학생들의 다양한 인지 전략, 환경 전략 등을 필요로 하고, 학습 동기와 밀접한 연관관계가 있다.

본 연구에서 이용한 개발한 자기 조절 학습 모형은 자기 조절 학습 요소와 소프트웨어 개발 단계에 4C(copying, changing, creating, challenging) 모델[11]을 본 연구에서 프로그래밍하기 단계에 적용한다. 4C는 학생들이 어떤 학습을 할 때에 복사하기(copying), 변형하기(changing), 창작하기(creating), 도전하기(challenging)로 자기 조절 학습 능력을 기를 수 있다.

3. 인공지능 학습 모델 및 검증

본 연구에서 초등학생들에게 인공지능 학습을 위한 모델을 제안하고, 델파이 방법으로 검증한다.

3.1 인공지능 학습 모델

본 연구에서 인공지능 프로그램을 초등학생들이 학습할 수 있는 모델은 다음과 같은 단계로 나눈다.

첫 번째는 문제 이해 단계이다. 학생들이 어떤 문제를 해결할 때에 인공지능 프로그램으로 해결할 수 있는지 또는 인공지능 프로그램이 아닌 것으로 해결 할 수 있는지를 잘 이해해야 한다. 그래서 주어진 문제를 인공지능 프로그램 가능한지를 체크하는 단계가 매우 중요하다. 예를 들어 그림으로 개와 고양이를 판별하는 문제이거나 두 개의 꽃을 분류하는 것을 할 수 있다.

두 번째는 데이터 준비하기이다. 데이터가 우리주변 또는 인터넷에서 확보할 수 있다. 개와 고양이를 판별하는 프로그램을 만들고자 할 때에는 개와 고양이의 그림을 많이 준비해야 한다. 이 준비된 데이터가 훈련 데이터가 될 수 있다. 준비하는 데이터의 유형들은 데이터 유형, 데이터 길이, 데이터 범위 데이터 포맷 등등이 있을 수 있다.

세 번째는 인공지능 모델링을 결정한다. 인공지능 모델은 인공신경망을 많이 이용하지만 SVM(Support Vector Machine), 클러스터링 방법 등을 사용할 수도 있다.

네 번째는 학생들이 실제 인공지능 프로그램을 작성하는 단계이다. 이 단계에서는 김갑수[9]가 제안한 자기조절 학습 모델의 4C(copying, changing, creating, challenging)의 모형을 이용한다.

모방하기(copying) 모형은 교사가 어떤 문제를 주고 그 문제를 학생들이 그대로 따라하는 것이고, 변형하기(changing)는 학생들이 교사가 예제로 준 것을 조금 변형해 보는 것이고, 창작하기(creating)는 주어진 문제를 해결하기 위해서 모방하기, 변형하기를 거친 후에 직접 자신의 생각으로 만들어 보는 것이고, 도전하기(challenging)는 실제 우리주변에 인공지능 프로그램을 도전해 보는 것이다. 4개의 모형은 학생들의 인지적인 능력에 따라 선택할 수 있다.

마지막으로 다섯 번째는 보고서 작성하기 단계이다. 이 단계에서는 스프레드 쉬트 또는 각종 보고서들을 이용하여 결과 보고서를 작성하는 것이다. 웹의 경우에는 프로세싱 등이 유용하게 사용될 수 있다.

3.2 모델 검증

델파이 검증 방법[12]이 심층적인 의견을 결정할 때에 많이 사용하기 때문에 본 논문에서 검증도구로 사용한다. 전문가 집단으로는 초등학교 컴퓨터 교육 경험이 있고, 인공지능에 대한 개념을 학습하고 석사학위 이상을 가진 현장 교사 20명을 선정하였다.

본 연구에서 제안한 인공지능 학습을 위한 교수 학습 모형은 미국의 ACM 학회 산하의 CSTA(Computer Science Teacher Association)의 계산사고(Computational Thinking) 특성을 만족하는지에 대한 검증 도구를 기본으로 실시하였다. CSTA의 계산 사고 특성[2,3]은 문제 해결에 도움을 주기 위해서 컴퓨터나 다른 도구를 이용하여 문제들을 형식화할 수 있고, 논리적으로 데이터를 구성하고 분석할 수 있고, 데이터를 모델이나 시뮬레이션으로 표현할 수 있고, 알고리즘적 사고로 해법을 자동적으로 만들 수 있고, 가장 효과적으로 또는 효율적으로 가능한 해법을 찾아서 분석하고 구현할 수 있고, 문제를 해결하는 과정을 일반화하고 다양한 방법으로 표현할 수 있다는 것이다.

본 연구에서 제안한 교육 모델이 위의 6개의 특성에 어는 정도 만족하는지를 전문가의 델파이 조사를 수행했다. 5단계 측도를 이용하여 6개의 특성 각각에 대한 만족도를 조사하여. 매우 만족이면 5번이고 만족이면 4번이고, 보통이면 3번이고, 불만족이면 2번이고, 매우 불만족이면 1번이다.

전문가 집단에서 두 단계로 진행되었고, 1단계에서는 인공지능 교육을 위한 교수 학습 모델의 이론과 설명을 준 후에 전문가들의 평가 결과는 <Table 1>과 같았다.

<Table 1> Result of Delphili (1st)

	내용	평균	표준편차
1	문제 해결에 도움을 주기 위해서 컴퓨터나 다른 도구를 이용하여 문제를 형식화할 수 있다	4.75	0.44
2	논리적으로 데이터를 구성하고 분석할 수 있다.	4.80	0.41
3	데이터를 모델이나 시뮬레이션으로 표현할 수 있다.	4.70	0.47
4	알고리즘적 사고로 해법을 자동적으로 만들 수 있다.	4.40	0.50
5	가장 효과적으로 또는 효율적으로 가능한 해법을 찾아서 분석하고 구현할 수 있다.	4.15	0.49
6	문제를 해결하는 과정을 일반화하고 다양한 방법으로 표현할 수 있다.	4.30	0.66

2단계에서는 전문가들에게 예제 프로그램을 제시하였고, 조사 결과는 <Table 2>와 같다.

<Table 2> Result of Delphi (2nd)

	내용	평균	표준편차
1	문제 해결에 도움을 주기 위해서 컴퓨터나 다른 도구를 이용하여 문제를 형식화할 수 있다	4.65	0.49
2	논리적으로 데이터를 구성하고 분석할 수 있다.	4.85	0.37
3	데이터를 모델이나 시뮬레이션으로 표현할 수 있다.	4.55	0.51
4	알고리즘적 사고로 해법을 자동적으로 만들 수 있다.	4.25	0.44
5	가장 효과적으로 또는 효율적으로 가능한 해법을 찾아서 분석하고 구현할 수 있다.	3.90	0.45
6	문제를 해결하는 과정을 일반화하고 다양한 방법으로 표현할 수 있다.	3.95	0.39

본 연구에서 제안한 교수 학습 모델은 <Table 1>과 <Table 2>의 결과를 분석하여 보면 초등학생들에게 인공지능 교육을 위한 적합한 모델이다. 그 이유는 다음과 같다.

“문제 해결에 도움을 주기 위해서 컴퓨터나 다른 도구를 이용하여 문제를 형식화할 수 있다.”에 대한 조사 결과는 평균 4.75이고 표준편차는 0.44이다. 예제를 보여주고 평가한 2차 조사에서도 평균 4.60이고 표준편차는 0.49이다. “논리적으로 데이터를 구성하고 분석할 수 있다”에 대한 조사 결과는 평균 4.75이고 표준편차는 0.44이다. 예제를 보여주고 평가한 2차 조사에서도 평균 4.60이고 표준편차는 0.49이다. “데이터를 모델이나 시뮬레이션으로 표현할 수 있다.”에 대한 조사 결과는 평균 4.70고 표준편차는 0.47다. 예제를 보여주고 평가한 2차 조사에서도 평균 4.55고 표준편차는 0.51다. “알고리즘적 사고로 해법을 자동적으로 만들 수 있다.”에 대한 조사 결과는 평균 4.40이고 표준편차는 0.50이다. 예제를 보여주고 평가한 2차 조사에서도 평균 4.25이고 표준편차는 0.44이다. “효과적으로 또는 효율적으로 가능한 해법을 찾아서 분석하고 구현할 수 있다.”에 대한 조사 결과는 평균 4.15이고 표준편차는 0.49이다. 예제를 보여주고 평가한 2차 조사에서도 평균 3.90이고 표준편차는 0.45이다. “해결하는 과정을 일반화하고 다양한 방법으로 표현할 수 있다.”에 대한 조사 결과는 평균 4.30이고 표준편차는 0.66이다. 예제를 보여주고 평가한 2차 조사에서도 평균 3.95이고 표준편차는 0.39이다.

최종적으로 인공지능 교육을 위한 초등학생들에 교육하기 위한 모델은 계산 사고를 발전시킬 수 있는 것이다. 다만 아직 초등학생들이 인공지능의 각종 알고리즘을 모르기 때문에 “알고리즘적 사고로 해법을 자동적으로 만들 수 있다.”, “효과적으로 또는 효율적으로 가능한 해법을 찾아서 분석하고 구현할 수 있다.”는 항목과 “해결하는 과정을 일반화하고 다양한 방법으로 표현할 수 있다.”는 항목이 평가가 낮았다.

4. 적용 사례

본 연구에서 제안하는 인공지능 학습 모델은 ‘문제 이해하기’, ‘데이터 준비하기’, ‘인공지능 모델 결정하기’, ‘인공지능 프로그래밍하기’, ‘보고서 작성하기’ 등 5가지 단계로 구성된다.

4.1 문제 이해하기

인공지능 교육을 위한 첫 번째 교수 학습 단계는 ‘문제 이해하기’다. 이 단계에서는 인공지능, 특히 지도 학습(supervised learning)을 통하여 해결할 수 있는 문제가 어떤 것인지를 이해한다. 이때 쉽게 구현할 수 있는 인공지능 프로그램의 사례로 아래의 3가지 프로그램을 제시할 수 있다.

- 예제 1: 꽃받침의 길이/폭, 꽃잎의 길이/폭 등 붓꽃의 특징(feature) 값이 주어졌을 때, 어떤 종류의 붓꽃인지 분류하기
- 예제 2: 손으로 쓴 숫자 이미지가 주어졌을 때, 어떤 숫자인지 알아내기
- 예제 3: 개 또는 고양이 이미지가 주어졌을 때, 어떤 동물인지 분류하기

4.2 데이터 준비하기

두 번째 교수 학습 단계는 ‘데이터 준비하기’다. 이 단계를 통해 (1) 학습을 위해 필요한 데이터가 어떤 형태인지를 배우고, (2) 학습/테스트 데이터가 각각 ‘특징’과

'정답(label)'으로 이루어진다는 것을 인식하고, (3) 학습 데이터와 테스트 데이터의 차이가 무엇인지를 배운다.

4.1절에서 제시한 3가지 프로그램을 기반으로 각각의 데이터를 준비한다. 예제 1의 경우, <Table 3>과 같은 데이터를 준비할 수 있다[1]. 예를 들어 <Table 3>에 제시된 첫 번째 데이터에 따르면, 어떤 꽃의 꽃받침 길이, 꽃받침 폭, 꽃잎 길이, 꽃잎 폭은 각각 5.1, 3.5, 1.4, 0.2인데, 사람에 의해서 이 꽃은 붓꽃 중에서도 'setosa (iris setosa)'로 분류되었다는 뜻이다. 예제 2의 경우, (Fig. 1)와 같은 데이터를 준비한다[2]. 손으로 쓴 숫자 이미지들이 필요하고, 그 이미지들이 사람이 보기에 어떤 숫자인지를 나타낸 값이 필요하다. 마찬가지로 예제 3의 경우 (Fig. 2)와 같은 개/고양이 이미지와, 사람이 개 또는 고양이로 분류한 결과를 준비한다[3].

<Table 3> Example Training (Test) Data for Iris Flower Classification

꽃받침		꽃잎		꽃 이름
길이	폭	길이	폭	
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.3	3.3	6.0	2.5	virginica
5.8	2.7	5.1	1.9	virginica

(Fig. 1) Example Training (Test) Data for MNIST Handwritten Digit Recognition

(Fig. 2) Example Training (Test) Data for Dogs & Cats Classification

1) https://en.wikipedia.org/wiki/Iris_flower_data_set
2) http://yann.lecun.com/exdb/mnist/
3) https://www.kaggle.com/c/dogs-vs-cats

4.3 인공지능 모델 결정하기

본 단계에서는 (1) 학습에 사용할 모델을 정하고, (2) 그 모델을 어떻게 구현할 것인지 결정한다. 학습을 하기 위해서는 SVM, K-인접 이웃 분류기, 인공 신경망 등 여러 가지 방법들을 사용할 수 있다. 어떤 것을 사용하여도 교육 목적으로는 큰 문제가 없을 것으로 보이나, 최근에 알파고, 음성 인식기, 자동 번역기, 자율 주행차 등에서 폭넓게 활용되고 있는 인공 신경망을 사용하는 것이 더 학생들의 흥미를 끌 수 있을 것으로 보인다. 만약 인공 신경망을 사용하기로 하였다면 신경망을 어떻게 구성하고 학습할지를 결정해야 한다. 다행인 것은 인공 신경망에 대해 잘 모르더라도 널리 사용되는 신경망 구조를 차용하면 많은 경우에 큰 문제가 되지 않는다는 점이다. 단, 너무 복잡한 모델을 사용하지 않도록 주의해야 한다. 왜냐하면 초등학교 현장에서는 GPU가 별도로 설치된 컴퓨터가 거의 없을 것이기 때문에, 복잡한 모델은 학습하려면 매우 오랜 시간이 소요될 수 있다.

학습에 사용할 모델을 결정했다면, 그 모델을 어떻게 구현할지 정해야 한다. 인공 신경망을 사용할 경우, TensorFlow, Theano, Caffe, Keras 등 다양한 딥러닝 프레임워크 중 하나를 선택하는 것이 편리하다. 이 중에서도 특히 Keras가 초등 컴퓨터 교육 목적으로 가장 적합할 것으로 보이는데, 그 이유는 크게 2가지다. 첫째, Keras는 TensorFlow와 Theano의 래퍼(wrapper)이기 때문에 매우 사용하기 쉽다. 둘째, Keras는 (Theano backend를 사용할 경우) 64비트 Linux, 32비트 Linux, 64비트 Windows, 32비트 Windows 환경에서 별도의 가상 머신을 사용하지 않고 쉽게 사용할 수 있다. 많은 초등학교에서 64비트/32비트 Windows가 설치된 컴퓨터를 사용하고 있고, 일부 학교는 라즈베리 파이 등을 사용해 리눅스 환경에서 컴퓨터 교육을 받고 있는 것을 고려할 때, 이러한 범용성은 교육적 관점에서 매우 중요할 수 있다.

<Table 4>는 Keras로 구현된 학습 모델의 예를 나타낸다. 초등학생들은 이 코드를 이해하지 않아도 무방하며, get_data.py 파일을 만든 후 학습하고 싶은 목적에 맞도록 get_training_data()와 get_test_data() 함수만 정의하면 된다.

<Table 4> Source code for training neural networks

```
from keras.callbacks import EarlyStopping
from keras.models import Sequential
from keras.layers import Activation
from keras.optimizers import SGD
from keras.layers import Dense
from sklearn.preprocessing import LabelEncoder
from keras.utils import np_utils
from get_data import *

# (1) prepare data
train_features, train_labels = get_training_data()
test_features, test_labels = get_test_data()

# (2) refine the data
n_classes = len(set(train_labels))
le = LabelEncoder()
encoder = lambda labels: np_utils.to_categorical(
  le.fit_transform(labels), n_classes)
train_labels, test_labels =
  encoder(train_labels), encoder(test_labels)

# (3) define a neural network architecture
model = Sequential()
model.add(Dense(768,
  input_dim=len(train_features[0]), init="uniform",
  activation="relu"))
model.add(Dense(384,
  init="uniform", activation="relu"))
model.add(Dense(len(train_labels[0])))
model.add(Activation("softmax"))

# (4) train the model using SGD
print("compiling model...")
sgd = SGD(lr=0.01)

model.compile(loss="binary_crossentropy",
  optimizer=sgd, metrics=["accuracy"])
model.fit(train_features, train_labels,
  nb_epoch=iteration, batch_size=128, verbose=1)

# (5) show the accuracy on the test set
print("evaluating on test set...")
(loss, accuracy) = model.evaluate(test_features,
  test_labels, batch_size=128, verbose=1)
print("loss={:.4f}, accuracy: {:.4f}%".format(loss,
  accuracy * 100))
```

4.4 인공지능 프로그래밍하기

인공지능 모델을 결정했다면, 실제로 프로그래밍하는 과정을 수행한다. 본 논문에서는 모방하기, 변형하기, 창작하기, 도전하기 등 4단계로 구성된 프로세스를 따른다.

1단계는 '모방하기' 단계다. 4.1절의 예제 1를 구현하기 위해, <Table 5>에 나타난 코드를 그대로 따라 작성하게 된다. 이 코드에서는 get_training_data()과 get_test_data()를 정의하였으며, 이 두 함수만 작성하면 4.3절에서 제시한 모델을 활용하여 바로 학습이 가능하다. get_training_data() 함수와 get_test_data() 함수는 features 리스트와 labels 리스트를 정의하고 반환한다. get_training_data() 함수를 보면, <Table 3>에 나타낸 6개의 데이터를 features와 labels 리스트로 나누어 표현한 것을 볼 수 있다. get_test_data() 함수에는 3개의 테스트 데이터가 소스 코드에 직접 기재되어 있다. 학습을 진행하면, Intel Core i5-4590 CPU를 기준으로 하였을 때 1000번 반복 학습하는 데에(1000 epochs) 5초가 소요되지 않으며, 테스트 데이터에 대한 분류 정확도는 100%다.

<Table 5> Example of Copying Step

```
iteration = 1000

def get_training_data():
  features = [[5.1,3.5,1.4,0.2],
              [4.9,3.0,1.4,0.2],
              [7.0,3.2,4.7,1.4],
              [6.4,3.2,4.5,1.5],
              [6.3,3.3,6.0,2.5],
              [5.8,2.7,5.1,1.9]]
  labels = ['Iris-setosa',
            'Iris-setosa',
            'Iris-versicolor',
            'Iris-versicolor',
            'Iris-virginica',
            'Iris-virginica']
  return features, labels

def get_test_data():
  features = [[5.1,3.8,1.9,0.4],
              [5.6,2.7,4.2,1.3],
              [6.7,3.3,5.7,2.5]]
  labels = ['Iris-setosa',
            'Iris-versicolor',
            'Iris-virginica']
  return features, labels
```

2단계는 '변형하기' 단계다. 1단계에서 6개의 학습 데이터와 3개의 테스트 데이터를 직접 소스코드에 입력하였는데, 이 방식은 데이터가 많아지면 굉장히 번거로운

일이 될 것이다. 따라서 많은 데이터를 학습하기 위해 파일에서 데이터를 읽어오는 방법으로 소스코드를 변형하고자 한다. 학습 데이터와 테스트 데이터가 각각 ./iris_train/data.txt, ./iris_test/data.txt에 저장되어 있고, data.txt 파일의 포맷은 <Table 6>과 같이 각 라인이 하나의 데이터를 의미하며 ‘,’를 구획문자로 하여 특징 값들과 정답이 분리되어 있다고 가정하자. 이 경우 <Table 5>의 코드는 <Table 7>의 코드로 변형될 수 있다. 학습 데이터가 135개인 경우 1000번 반복 학습하는 데에 약 10초가 소요되었으며, 15개의 테스트 데이터에 대해 100%의 분류 정확도를 나타내었다.

<Table 6> Example “data.txt”

```
5.1, 3.5, 1.4, 0.2, Iris-setosa
4.9, 3.0, 1.4, 0.2, Iris-setosa
7.0, 3.2, 4.7, 1.4, Iris-versicolor
...
```

<Table 7> Example of Changing Step

```
import os

iteration = 1000

def get_training_data():
  return get_data_from_dir('iris_train')

def get_test_data():
  return get_data_from_dir('iris_test')

def get_data_from_dir(dir_name):
  features = []
  labels = []

  with open(dir_name + '/' +
    os.listdir(dir_name)[0], 'r') as file:
      for line in file:
        spl = line.split(',')
        if len(spl) > 1:
          features.append([float(i) for i in spl[:-1]])
          labels.append(spl[-1].strip())

  return features, labels
```

3단계는 ‘창작하기’ 단계다. 이 단계에서는 1단계와 2단계에서 수행한 과제를 응용하여, 4.1절의 예제 2로 제시한 ‘손으로 쓴 숫자 이미지가 주어졌을 때 어떤 숫자인지 알아내기’를 직접 구현하는 것이 목표다. 이전 단계에서는 ‘수’로 표현된 특징 값들을 이용하여 붓꽃을 분류하였다. 그런데 이번에는 이미지를 이용하여 숫자 이미지를 분류해야 한다는 점이 어렵다. 이 문제를 해결하려면 (1) 이미지는 픽셀로 표현되며, (2) 각 픽셀의 정보는 ‘수’의 리스트로 표현할 수 있다는 것을 알아야 한다. 예를 들어, MNIST 데이터셋의 숫자 이미지는 28*28 크기다. 따라서 이 이미지는 총 28*28=784개의 픽셀이 존재한다. 각 픽셀들은 RGB 값을 가지고 있으므로, 784개의 픽셀들은 784*3=2352개의 수로 표현할 수 있다. Python의 OpenCV 라이브러리를 사용하면, 28*28 크기의 이미지를 2352개의 수로 구성된 리스트로 쉽게 변환할 수 있다. 즉, 1단계와 2단계에서 features 리스트의 각 요소가 4개의 수로 구성된 리스트였다면, 이번에는 2352개의 수로 구성된 리스트로 변환되었을 뿐이다. <Table 8>는 ‘창작하기’ 단계에서 만들어낼 수 있는 소스코드의 예를 나타낸다. 학습 데이터와 테스트 데이터는 mnist_train과 mnist_test 디렉토리에 ‘0.200.jpg’와 같은 형태로 저장되어 있다고 가정하였다. 이때 ‘0’은 0을 나타내는 이미지라는 뜻이며, ‘200’은 0을 나타내는 이미지 중 200번째 이미지라는 뜻이다. mnist_train 디렉토리에는 각 숫자별 6000개(총 60000개)의 jpg 파일이 있고, mnist_test 디렉토리에는 각 숫자별 1000개(총 10000개)의 jpg 파일이 존재한다. 학습을 수행한 결과, 10번 반복 학습하는 데에 약 25초 소요되었으며, 분류 정확도는 약 92%로 나타났다. 지난 예제들보다 조금 정확도가 떨어졌는데, 그 이유는 크게 2가지다. 첫째, 붓꽃을 분류하는 것은 3개 중 하나를 선택하는 문제였으나, 이번에는 10개 중 하나를 선택해야 하는 문제였다. 둘째, 이미지 데이터를 수로 변환한 후 학습하였는데, 이 과정에서 인접한 픽셀들이 무엇인지에 대한 정보가 손실되었다.

4번째 단계는 ‘도전하기’ 단계다. 이 단계에서는 자유롭게 자신이 만들고 싶은 프로그램을 정하고, 학습하는 것이 목표다. 예를 들어 4.1절의 예제 3과 같이, ‘개 또는 고양이 이미지가 주어졌을 때, 어떤 동물인지 분류’하는 프로그램을 만들 수 있을 것이다. 숫자 분류를 할 때와 마찬가지로, 이미지를 수로 변환하면 바로 학습할 수 있다. 그러나 개와 고양이 이미지를 수집하였을 때, 예상치 못한 문제가 발생할 수 있다. 예를 들어 MNIST 데이터셋의 경우 이미지의 크기가 28*28로 동일하였지만, 웹에서 개와 고양이 이미지를 수집할 경우 이미지의 크기가 모두 제각각일 가능성이 높다. 이런 경우 모든

<Table 8> Example of Creating Step

```
import os
import cv2
import numpy as np

iteration = 10

def get_training_data():
  return get_data_from_dir('mnist_train')

def get_test_data():
  return get_data_from_dir('mnist_test')

def get_data_from_dir(dir_name):
  features = []
  labels = []

  for filename in os.listdir(dir_name):
    features.append(cv2.imread(dir_name + '/' +
      filename).flatten())
    labels.append(filename.split(".")[0])
  features = np.array(features) / 255.0

  return features, labels
```

<Table 9> Example of Challenging Step

```
import os
import cv2
import numpy as np

iteration = 50

def get_training_data():
  return get_data_from_dir('dog_cat_train')

def get_test_data():
  return get_data_from_dir('dog_cat_test')

def get_data_from_dir(dir_name):
  features = []
  labels = []

  for filename in os.listdir(dir_name):
    features.append(cv2.resize(
      cv2.imread(image_path), (32, 32)).flatten())
    labels.append(filename.split(".")[0])
  features = np.array(features) / 255.0

  return features, labels
```

이미지의 크기를 일정 크기(예를 들어 32*32)로 변환하는 작업이 필요하다. <Table 9>은 OpenCV의 resize 함수를 적용하여 이 작업을 쉽게 수행하였다. 학습 데이터와 테스트 데이터는 각각 dog_cat_train, dog_cat_test 디렉토리에 저장하였으며, 학습에 사용된 이미지는 각 종류 별 10000개(총 20000개), 테스트에 사용된 이미지는 각 종류 별 2500개(총 5000개)다. 학습 결과, 50번 반복 학습하는 데에 약 6분이 소요되었으며, 분류 정확도는 67% 수준이었다. 상당히 낮은 정확도라는 생각이 들 수도 있지만, 실제 학습/테스트 데이터를 확인해 보면, 학습하기가 쉽지 않은 이미지임을 확인할 수 있다. 만약 분류 정확도를 더 높이고 싶은 학생이 있다면, CNN (Convolutional Neural Network)을 적용해야 한다고 말해줄 수 있다. CNN을 적용하려면 <Table 4>의 내용을 수정해야 하는데, Keras를 활용할 경우 코드 작성이 어렵지는 않으나 신경망에 대한 이해를 필요로 한다.

4.5 보고서 작성하기

마지막 과정은 보고서를 작성하는 것이다. 보고서를 작성함으로써, 학습하기 위해 어떤 데이터가 사용되었는지, 특징과 정답의 차이는 무엇인지, 학습 데이터와 테스트 데이터의 차이는 무엇인지 등을 스스로 정리해 볼 수 있다.

교사가 설명하지 않은 내용을 실험해 보았다면, 그 결과를 작성하는 것도 좋다. 스스로 결과를 분석할 수 있으면 가장 좋겠지만, 그러지 못하더라도 교사의 피드백을 받을 수 있을 것이다. 예를 들어 대표적으로 다음과 같은 사항들을 기재할 수 있다.

- 수업시간에 제시한 예제보다 더 많은 데이터를 사용해 보았더니 분류 정확도가 향상되었다.
- 나쁜 품질의 데이터를 학습에 사용하지 않았더니 오히려 분류 정확도가 떨어졌다.
- 'iteration' 변수의 값을 바꾸어 더 오랜 시간 학습했더니, 분류 정확도가 점점 좋아지다가 어느 순간부터는 정확도가 오히려 떨어졌다.

본 논문에서는 <Table 4>와 같은 신경망을 직접 설계하지는 않았다. 그러나 기계 학습에 관심이 많은 학생

이라면, 신경망이 어떻게 동작하는지 스스로 유추하고 보고서에 정리하는 것도 좋은 학습 방법이다.

5. 결론 및 문제점

본 연구에서는 4차 산업 사회에서 인공지능이 매우 부각되고 있고, 2015년 개정 교육과정에서 소프트웨어 교육을 초등학교부터 해야 하는 시점에서 인공지능을 학생들에게 학습하게 하는 교수 학습 모델을 만들었다. 본 연구에서 제안한 학습 모델은 문제 이해 단계, 데이터 정리하기 단계, 인공지능 모델 정하기 단계, 프로그래밍하기 단계, 보고서 작성하기 단계로 구성되어 있고, 학생들은 수준에 적합하게 프로그래밍하기 단계에서는 복사하기, 변형하기, 창조하기, 도전하기로 나누었다. 이 단계들에 대한 전문가들의 델파이 조사는 매우 의미 있게 결과가 나왔다. 본 연구에서 제안한 모형에 따라 초등학교에서 실제 적용 가능한 예제를 만들어서 초등학생들에게 적용할 수 있다.

본 연구의 문제점은 초등학생들이 파이썬을 이해해야 한다는 점이다. 이 인공지능 프로그램을 이용하고 만들기 위해서 많은 시수가 필요하다는 점이다.

그렇지만 학생들이 한번 경험해 보는 것은 의미 있는 일이고, 앞으로 좀 더 많은 구체적인 실험을 통해서 예제를 만들어 보는 것이 의미 있다.

참고문헌

[1] Boekarts, M., Pintrich, P. R., & Zeidner, M. (Eds.), (2000). Handbook of self-regulation. San Diego, CA: Academic.

[2] Chris Stephenson and Valerie Barr(2011), Defining Computational Thinking for K - 2, The Voice of K-12 Computer Science education and its Educators 2011, 7-2

[3] CSTA(2011),K - 12 Computer Science Standards

[4] Garris, R., Ahlers, R., & Driskell, J. E. (2002). Games, motivation, and learning: A research and practice model. *Simulation & Gaming, 33*, 441-467.

[5] Gee, J. P. (2003). What video games have to teach us about learning and literacy. New York: Palgrave Macmillan.

[6] Gee, J. P. (2005). Learning by design: Good video games as learning machines. *E-Learning, 2*, 5-16.

[7] Harnad, S.(1992),The Turing Test Is Not A Trick:Turing Indistinguishability Is A Scientific Criterion. SIGART Bulletin 3(4).

[8] Harnad, S. (2000) Minds, Machines, and Turing: The Indistinguishability of Indistinguishables. Journal of Logic, Language, and Information 9(4): 425-445.

[9] Kim, Kapsu (2010), "A Study on Programming Language Instruction Strategies of Improving the creative and logical thinking for Elementary Students", Journal Of The Korean Association Of Information Education, 14(1), 89-97.

[10] Kim,Kapsu(2014). Measuring and Applying the Self-efficacy in Computer Programming Education. *Journal Of The Korean Association Of Information Education, 18*(1), 111-120.

[11] Kim, Kapsu(2015). A Self-regulated Learning Model Development in Computer Programming Education, Journal of The Korean Association of Information Education Vol. 29(1),21-30.

[12] Linstone,H.A and Turoff,M(1975). The Delphi Method: Techniques and Application, London: Addison-wesley.

[13] Miltiadou, M., & Yu, C. H. (2000). Validation of the Online Technologies Self-Efficacy Scale (OTSES)

[14] Ministry of Education (2015). The revised national curriculum 2015 for Primary and Secondary Schools, Vol.2015-75, No. 10.

[15] Nong Ye Gavriel Salvendy (1996). Expert-novice knowledge of computer programming at different levels of abstraction. *Ergonomics, 39*(3), 461-481.

[16] Pressley, M., & Ghatala, E. S. (1990). Self-regulated learning: Monitoring learning from text.

Educational Psychologist, 25, 19-34.

[17] Pressley, M., & Ghatala, E. S. (1990). Self-regulated learning: Monitoring learning from text. *Educational Psychologist, 25*, 19-34.

[18] Salomon G & Perkins D. N. (1987). Transfer of cognitive skills from programming: When and how?. *Journal of Educational computing Research, 3*, 149-169.

[19] Schraw, G. (2007). The use of computer-based environments for understanding and improving selfregulation. *Metacognition and Learning, 2*, 169-176.

[20] Turing, A. M. (1950) Computing Machinery and Intelligence, Mind 49: 433-460.

[21] Zimmerman, B. J. (1990). Self-regulated learning and academic achievement: An overview. *Educational Psychologist, 25*(1), 3-17.

저자소개

김 갑 수

1985 서울대학교 계산통계학과(학사)

1987 서울대학교 계산통계학과 전산학전공(석사)

1996 서울대학교 계산통계학과 전산학전공(박사)

1987~1992 삼성전자 사원-과장

1995~1998 서경대학교 조교수

1998~현재 서울교육대학교 컴퓨터교육과 조교수-교수

관심분야: 컴퓨터 교육, SW 공학, 정보 영재, 기능성 게임

e-mail: kskim@snue.ac.kr

박 영 기

2008.2 KAIST 전자전산학과 전산학전공(학사)

2010.2 서울대학교 컴퓨터공학과 (석사)

2015.2 서울대학교 컴퓨터공학과 (박사)

2015.3~2016.8. 삼성전자 종합기술원 전문연구원

2016.8~현재 춘천교육대학교 컴퓨터교육과 조교수

e-mail: ypark@cnue.ac.kr