Journal of The Korea Society of Computer and Information
Vol. 22 No. 12, pp. 137-143, December 2017

www.ksci.re.kr
https://doi.org/10.9708/jksci.2017.22.12.137

Design and Implementation of Routing System Using Artificial Neural Network

Jun-Yeong Kim*, Seog-Gyu Kim**

Abstract

In this paper, we propose optimal route searching algorithm using ANN(Artificial Neural Network) and implement route searching system. Our proposed scheme shows that the route using artificial neural network is almost same as the route using Dijkstra's algorithm but the time in our propose algorithm is shorter than that of existing Dijkstra's algorithm. Proposed route searching method using artificial neural network has better performance than exiting route searching method because it use several weight value in making different routes. Through simulation, we show that our proposed routing system improves the performance and reduces time to make route irrespective of the number of hidden layers.

▸Keyword: Routing algorithm, Artificial neural network ,Dijkstra algorithm

I. Introduction

2016년 말에는 구글 딥 마인드에서 인공지능 기술테스트 플랫폼인 딥마인드 랩을 외부에 무료 공개 하였다. 이는 지난해 구글의 인공지능 플랫폼 텐서플로우 공개의 연장 선상안에 있다. 어느덧 우리 주변에서 인공지능이라는 단어는 어렵지 않게 들을 수 있다[1]. 인공지능 기술 중에서도 딥러닝 분야에는 기본적으로 지도 학습과 비지도 학습으로 구분된다. 지도 학습은 정해놓은 결과를 기반으로 학습을 진행하여 결과에 최대한 가깝도록 트레이닝 과정을 거쳐서 도출하는 방식을 이야기한다. 비지도 학습의 경우 어떠한 주제를 주지 않고 해당되는 정보들의 공통점에 가까운 위치로 구성하여 정보를 분류할 때 사용하는 방식이다.

다익스트라 알고리즘은 주변의 여러 경로 중 최단경로를 탐색하고 그 탐색한 정보를 기반으로 경로를 구성 하도록 한다. 이 알고리즘의 특징은 최소의 Cost를 이용하여 최단경로를 구성하는 것에 있다. 하지만 모든 경로의 경우의 수를 매번 탐색을 하기 때문에 경로를 결정하는 데 있어서 시간이 오래 걸리는 단점이 발생한다.

인공신경망을 이용하여 경로탐색을 하는 경우 능동적으로 정보를 제공하여 해당 정보의 가중치가 변화함에 따라 최적의 경로를 탐색하는 방식을 구성하게 되게 된다. 즉 변화하는 가중치의 값에 따라 경로를 효율적으로 반응하도록 제공을 할 수 있는 것이다. 또한 한번 트레이닝 된 정보를 저장하고 추후 로드하여 다시 사용할 수 있음으로 이러한 과정을 거친 정보는 변화에도 탁월한 능력을 발휘 할 수 있다.

본 논문에서는 파이썬 프로그램과 텐서플로우를 활용하여 다익스트라 알고리즘과 인공신경망(ANN)을 활용한 경로탐색 두 가지를 비교하여 속도 문제와 정확도를 측정하고 인공신경망을 이용하여 경로탐색을 진행할 경우 트레이닝 횟수에 따른 정확도에 대한 부분을 확인하고자 한다.

본 논문은 다음과 같이 구성되어 있다. 1,2장은 인공신경망을 이용한 경로탐색을 제안한 근거 및 관련연구를 기술하였으며 3장에서는 인공신경망을 이용한 경로탐색의 구성요소에 대한 설명을 하고 4장은 제안한 시스템에 대한 구현 및 성능분석을 하였으며 5장은 결론 및 향후 계획으로 구성되어 있다.

• First Author: Jun-Yeong Kim, Corresponding Author: Seog-Gyu Kim
*Jun-Yeong Kim (jongbean@naver.com) Dept. of Information & Communication Engineering, Andong National University
**Seog-Gyu Kim (sgkion@andong.ac.kr) Dept. of Information & Communication Engineering, Andong National University
• Received: 2017. 11. 28, Revised: 2017. 12. 04, Accepted: 2017. 12. 14.
• This work was supported by a grant from 2015 Research Funds of Andong National University.

II. Preliminaries

1. Artificial neural network

1-1 인공신경망 이론

인공신경망은 인간이나 동물들이 가지고 있는 중추 신경계인 뇌를 모방한 통계학적 기반의 학습 알고리즘이다. 즉, 시냅스의 결합으로 네트워크를 형성한 뉴런이 학습을 통해 시냅스 결합의 세기(가중치)를 조정하여 변화한다. 일반적으로 인공신경망에 활용되는 다층 퍼셉트론은 입력층의 각 노드에 데이터가 제공되면 각 노드에서 전달함수를 통해 은닉층에서 변환되고 이는 출력층에서 도출된다. 이를 수식으로 표현하면 식(1)과 같다.

$$n = w_{1,1}x_1 + w_{1,2}x_2 + \ldots + w_{S,R}x_R = \sum_{j=1}^{S}\sum_{i=1}^{R} w_{ij}x_i \quad (1)$$

위 식에서 xi는 인공신경망으로 입력되는 데이터이며, wij는 각 노드에 전달 되는 데이터에 해당하는 가중치이다. 위 식을 행렬 형태로 정리하면 식(2)와 같다.

$$n = \mathrm{W}\mathrm{x} \quad (2)$$

여기서 행렬 W는 가중치 행렬로 식(3)과 같이 표현할 수 있다.

$$\mathrm{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix} \quad (3)$$

식(2)을 다음과 같은 전달함수에 적용시켜 얻은 결과를 그 처리기의 출력으로 하여 이와 연결된 다른 처리기로 보내면 식(4)과 같다.

$$\mathrm{y}^n = \mathrm{f}^n(\mathrm{W}^n \ldots \mathrm{f}^2(\mathrm{W}^2\mathrm{f}^1(\mathrm{W}^1\mathrm{x}))) \quad (4)$$

위 식은 다층 인공신경망의 경우로 n은 층의 수를 나타내고 yn은 인공신경망의 최종적인 출력을 나타낸다. 여기서 전달함수 f는 신경 세포의 반응 여부를 결정한다. 현재 다양한 신경망 모델은 계층 수, 출력형태, 데이터유형, 학습방법 그리고 전달함수와 같은 기준에 따라 분류할 수 있는데, 그 중에서 다층 인공신경망 (MFNNs: Multilayered Feedforward Neural Networks)은 인공 신경망의 구조 중에서 가장 많이 적용되는 구조이며, 시스템 식별, 제어, 패턴인식과 같은 분야에 적용되고 있다. 따라서 본 연구에서는 공정결과를 예측하기 위하여 다층 및 순환 신경망을 사용하였다

1-2 인공신경망 구조

인공신경망은 입력층(input layer), 히든층(hidden layer), 출력층(output layer)으로 구성되어 있다. 입력층에는 각각의 입력변수가 1:1로 매칭되는 뉴런(neuron)이 존재한다. 히든층에는 입력층의 뉴런과 가중치(weight)의 결합으로 생성되는 뉴런이 존재하며, 히든층에서의 층의 개수에 따라 모형의 복잡도가 결정되고 히든층의 개수가 2개 이상이 되는 경우 deep neural network 또는 deep learning이라고 칭한다.

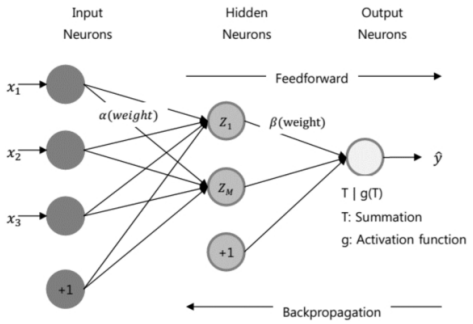

Fig. 1. Structure of Artificial neural network

출력층에는 히든층에서의 뉴런과 가중치가 결합하여 생성되는 뉴런이 존재하며, 예측하고자 하는 종속변수의 형태에 (numeric, binary or multinomial) 따라 출력층의 개수가 결정된다. 히든층과 출력층에 존재하는 뉴런은 이전 층에서의 입력값과 가중치를 합(summation)을 계산하는 기능과 뉴런의 가중합을 입력값으로 신호를 출력하는 활성화 함수(activation function)기능을 수행한다[1][2][3].

2. Path search algorithm

최단경로문제(shortest path problem)는 네트워크 이론에서 가장 기본적이고 중요한 문제 중의 하나이다. 특정의 두 교점 사이에는 여러 개의 경로가 존재한다. 두 교점 사이의 경로 중에서 가장 짧은 길이의 경로를 최단경로(shortest path)라고 부른다. 최단경로 알고리즘의 기본개념은 최적원리(principle of optimality)에 근거를 둔 등식을 만족하는 해를 구하는 것이다.

πj = 시발점 1 에서부터 교점 j 까지 최단거리
dij = (i, j) 의 거리

최단경로의 길이는 다음의 식(5)을 만족하여야 하는데 이것을 Bellman의 식(Bellman's equation)이라고 부른다.

$$\pi_1 = 0$$
$$\pi j = \min_{i \neq j} \{\pi i + dij\} \quad (5)$$

Bellman의 식에서 보듯이 특정 교점까지의 최단거리를 구하기 위해서는 그 교점뿐 아니라 모든 교점까지의 최단경로를 구해야 한다[4]. 다익스트라 알고리즘은 시작지점에서 모든 이웃하는 경로를 탐색하며, 시작 지점을 기준으로 영역을 확장하여 경로를 검색하고 목적지점까지를 탐색하는 알고리즘이다. 다익스트라 알고리즘은 최단거리 탐색을 보장하는 완전 탐색 알고리즘이다. 다익스트라 알고리즘은 수행하는데 있어서는 각 정점과 정점간의 비용을 합리적으로 결정하여 수행하는 것이 매

우 중요하다, 이를 위해 임시표지와 영구표지의 개념을 사용한다. 각 정점간의 비용을 합리적으로 결정하였을 때 더욱 정확한 결과를 얻을 수 있기 때문이다[5][6][7].

III. Artificial Neural Network Search Design

본 장에서는 인공신경망을 이용한 경로탐색 방법에 대한 내용을 구성하였다. 우선 진행되어야 할 부분은 경로를 결정해야 하는데 많은 데이터를 기반으로 하게 될 경우 정확도를 확인하기 어렵기에 본 논문에서는 Fig 2와 같이 간단한 형태의 데이터를 기반으로 제안하였다.

1. Route configuration & System environment

경로에 대한 정보는 Cost값으로 포함된 형태로 구성되어진다. Fig 2와 같이 Cost값으로 구성된 경로정보를 나타내고 이를 기반으로 한 경로정보를 이용하여 경로를 결정하게 된다. 이를 위해 우선 최단거리 알고리즘인 다익스트라 알고리즘을 이용하여 구해진 경로와 인공신경망을 이용한 경로가 일치하는지 확인을 한다. 그리고 인공신경망을 이용한 경로 결정에 따른 시간이 다익스트라 알고리즘보다 적게 나와 경로 도출에 결리는 시간이 향상됨을 확인하고자 한다.

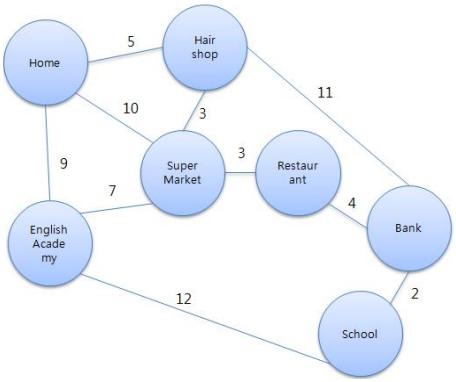

Fig. 2. Compose of Route

본 논문에서 제안하는 경로탐색을 구현하고 실험하기 위한 실험환경을 Table 1과 같이 구성하였다.

Table 1. System Environment

Item	detail
System OS	Mac OS X 10.12.6
CPU	i5-6360U 2GHz(3.1GHz)
Memory	8G
Graphic Card	intel Iris 540
Program Language	python 3.6.2
Use Library	tensorflow 1.4

Python 3.6과 Tensor Flow 1.4를 이용하여 시스템을 구성하였으며, 데이터의 내용이 크지 않으므로 MacBook Pro를 활용하여 전체적인 시스템을 완성하였다. 본 논문의 실험에서는 CNN(Convolutional Neural Network) 인공신경망을 사용하는 그래픽 작업이 방대한 데이터가 존재 하는 것이 아니므로 노트북 환경에서 시스템을 구현하고 실험하였다.

2. Design of Artificial Neural Network

인공신경망을 구성하기 위하여 우선 그래프 탐색 알고리즘을 구성하였다. 그래프 탐색 알고리즘에서 배열형태의 정보를 구성한다. 인접리스트 형태로 구성할 경우 어떤 정점에서 다른 정점에 연결되는 형태를 띄고 있어 인공신경망을 이용한 가중치를 나타내기에는 부적합한 형태를 띄고 있으므로 배열형태의 인접행렬 형태를 이용하였다. 일반적으로 인접행렬에는 일반 연결 관계와 가중치 연결 관계가 존재하는데 Fig 3에서는 가중치 연결 관계를 사용하여 배열의 구성하였다.

```
x_data = np.array(
    [[0,5,10,9,99,99,99]
     ,[5,0,3,99,99,11,99]
     ,[10,3,0,7,3,99,99]
     ,[9,99,7,0,99,99,12]
     ,[99,99,3,99,0,4,99]
     ,[99,11,99,7,4,0,2]
     ,[99,99,99,12,99,2,0]
     ])
```

Fig. 3. Array Information of Artificial Neural Network

위의 배열정보를 이용하여 정보를 구성하게 된다. 그래프 탐색 알고리즘을 이용할 경우 인접행렬의 정보를 구성하기 좋은 배열형태로 진행하였으며, 또한 인접행렬과 가중치 연결관계를 나타내기 위하여 자신의 위치일 경우는 0, 그리고 인접하지 않은 위치의 경우 99를 두어 가중치에서 제외되는 형태를 띄게 만들었으며, 이외에는 인접행렬의 가중치 값(행렬간의 거리 값)을 제공하여 위치정보를 인식하게 만들도록 정의 하였다. 위의 행렬값을 기반으로 나온 결과의 가중치를 인경신경망을 활용하여 정확도와 시간성능을 확인하고자 한다.

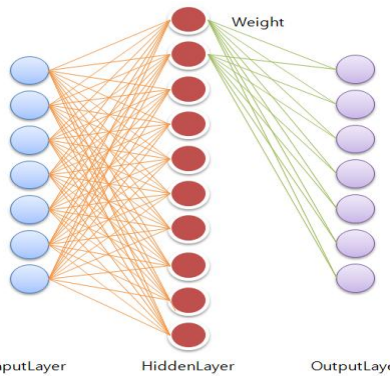

Fig. 4. Compose of Artificial Neural Network

Fig 4는 구성한 인공신경망에 대한 설명이다. 인공신경망은 기본적으로 InputLayer와 HiddenLayer 그리고 Output Layer로 구성된다. InputLayer와 OutputLayer는 동일하게 구성이 되게

된다. 입력된 정보를 기반으로 경로탐색을 진행한 후에 출력정보를 제공하기 때문에 경로의 수가 줄어들지는 않기 때문이다. 그렇다면 HiddenLayer에서 구성하는 퍼셉트론의 숫자에 따라 성능의 변화를 확인 할 수 있을 것이다. HiddenLayer에는 입력층의 뉴런과 가중치(weight)의 결합으로 생성되는 뉴런이 존재하며, HiddenLayer에서의 층의 개수에 따라 모형의 복잡도가 결정되고 히든층의 개수가 2개 이상이 되는 경우 deep neural network 또는 deep learning이라고 칭한다. 또한 HiddenLayer에서 도출된 값은 reru 활성 함수를 활용하여 값을 도출하게 된다.

인공신경망에서 퍼셉트론의 숫자를 증가시킬 때 변화를 확인하기 위해서 입력노드의 1.5배를 가설로 설정하여 10개로 퍼셉트론을 구성하였다. 4장에서는 퍼셉트론을 2배로 증가시켜 20개의 퍼셉트론에 따른 결과를 10개의 퍼셉트론으로 구성하여 탐색 능력이 향상되는지를 확인하고자 한다.

IV. Implementation and performance analysis

본 장에서는 다익스트라 알고리즘을 활용하여 경로탐색 된 결과를 확인하고 설계된 인공신경망을 이용하여 경로탐색 시스템을 실험한 결과를 나타낸다.

1. Implementation of Dijkstra algorithm

우선 그를 위해서는 다익스트라 알고리즘을 이용하여 최단경로로 구성하였을 경우 나오는 값을 확인해보았다. 우선 다익스트라 알고리즘을 구현하기 위하여 Fig 5와 같이 Pseudo Code를 기반 프로그래밍을 진행해 보았다. 다익스트라 알고리즘은 자신의 인접행렬들을 모두 검색을 하고 가장 짧은 거리를 저장함으로써 최단경로를 검색해 나가게 된다.

```
function dijkstra(G, w, s)
# initialize d and previous
# d: distance from s
# previous: previous node whose distance is minimum
for each vertex v in V(G)
d[v] := infinity
previous[v] := undefined
# Start distance = 0
d[s] := 0
S := empty set
Q := set of all vertices
while Q is not empty
u := extract_min(Q)
S += u
for each edge(u, v):
if  d[v] > d[u] + w(u,v)
d[v] := d[u] + w(u,v)
previous[v] := u;
```

Fig. 5. PseudoCode of Dijkstra algorithm

위의 PseudoCode를 기반으로 경로탐색을 해본 결과 아래의 Fig 6와 같이 경로탐색이 되었다.

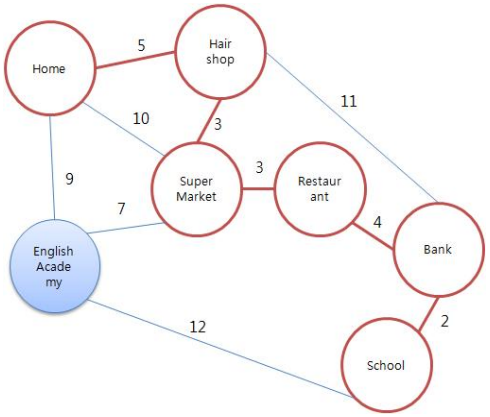

Fig. 6. Shortest path graph

Fig 6은 다익스트라 알고리즘을 이용한 결과이다. 경로는 집에서부터 학교까지의 정보를 나타내며 경로는 Home->Hairshop->SuperMarket->Restaurant->Bank->School 순으로 나타나게 되는 것을 확인할 수 있다. 하지만 다익스트라 알고리즘을 이용할 경우 한 번을 이용하여 정보는 제공 받을 수 있지만 여러 번을 사용할 경우에도 동일한 과정을 거쳐야 하므로 비효율적인 결과를 나타내게 된다.

2. Speed verification of ANN

인공신경망을 사용하였을 경우 다익스트라 알고리즘과 속도면에서 어떠한 차이를 보이는지 확인해 보았다. 다익스트라 알고리즘은 시간복잡도 면에서 O(ElogV)의 시간복잡도를 보인다. 최악의 시간복잡도가 발생할 경우 모든 경로를 검색하여 속도면에서 현저히 떨어지는 현상이 발생하게 된다. 이와 반대로 인공신경망의 경우 가중치를 중심으로 트레이닝 과정을 거치므로 트레이닝된 경로를 기반으로 검색을 하게 되는 것이다. 이러한 가정을 설정해 둔 다음 실험을 해 보았다. 실험은 각 다익스트라 알고리즘과 인공신경망을 이용한 경로 알고리즘을 각 1000번씩 실행해 보았다.

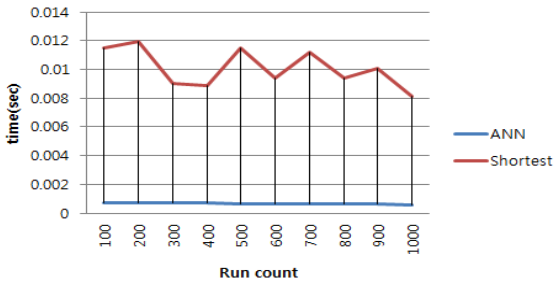

Fig. 7. Speed verification

Fig 7과 같이 속도면에서는 인공신경망이 월등히 앞서는 것을 확인할 수 있다. 그 이유로는 하드웨어의 성능의 문제가 아닌 경로 탐색에서 사용되는 Cost가 현저히 낮기 때문이다. 본 내용을 통해 시간적인 면에서는 인공신경망을 이용하는 것이 효율적임을 실증하였다.

3. A proper level of training

인공신경망의 경우 트레이닝에 따른 정확도는 데이터의 양에 따라 상이하게 나타난다. 본 경로를 기준으로 인공신경망의 트레이닝 횟수의 적절도를 확인해 보았다.

위 Fig 8과 같이 7개의 트레이닝 정보를 기반으로 할 경우 트레이닝 횟수의 적절도를 확인해 보았다. X축은 트레이닝 횟수를 나타내며 Y축은 정확도를 나타낸다. 정확도는 0에 가까울수록 정확하다고 보면 된다.

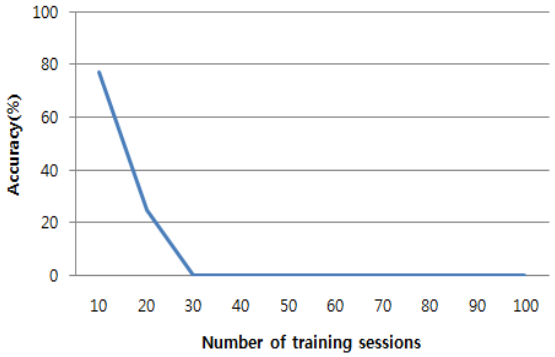

Fig. 8. Hidden Layer 1 Training Program

이와 같이 7개의 트레이닝 정보를 기반으로 하였을 경우 약 30회 정도가 적절한 것으로 확인 되었다. 30회에 들어서게 되면 정확도는 0.01퍼센트에 가깝게 이루어지게 되고 이후에는 정확도는 100프로에 이르는 형태로 진행되게 된다. 트레이닝을 많이 한다고 해서 학습능력이 향상되는 것은 아니다. 적절한 값을 찾았을 경우 해당 Weight값이 크게 변동하지 않으므로 학습도를 올리기 위해 무리한 트레이닝은 하드웨어 성능만 저해하는 결과를 가져오게 된다.

```
/Library/Frameworks/Python.framework/Versions/3.6/bi
10 77.2102
20 24.8437
30 0.00541543
40 0.0159622
50 0.0
60 0.0
70 0.0
80 0.0
90 0.0
100 0.0
예측값: [0 1 2 3 4 5 6]
실제값: [0 1 2 3 4 5 6]
정확도: 100.00

Process finished with exit code 0
```

Fig. 9. Route Exploration Results Using Artificial Neural Network

Fig 9과 같이 인공신경망을 이용할 경우 30회의 트레이닝을 통해 해당 경로를 탐색하는 것을 확인 할 수 있다. 또한 30회 진행하였을 때 유실율은 0.0159622 퍼센트에 해당하며 이 데이터는 오차범위 0.05안에 존재하게 되므로 적합한 트레이닝이 진행되었다고 볼 수 있다. 학습 데이터를 기반으로 텐서플로우의 학습 저장기법을 활용하여 학습된 데이터를 실제 운영 데이터만을 이용한 결과는 아래 Fig 10와 같다.

```
In [3]: ########
        # 결과 확인
        # 0,1,2,3,4,5,6
        ######
        prediction = tf.argmax(model, 1)
        target = tf.argmax(Y, 1)
        print('예측값:', sess.run(prediction, feed_dict={X: x_data}))
        print('실제값:', sess.run(target, feed_dict={Y: y_data}))

        is_correct = tf.equal(prediction, target)
        accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))
        print('정확도: %.2f' % sess.run(accuracy * 100, feed_dict={X: x_data, Y: y_data}))

        예측값: [0 1 2 3 4 5 6]
        실제값: [0 1 2 3 4 5 6]
        정확도: 100.00
```

Fig. 10. Path search results after learning

Fig 10에서는 실제 트레이닝 된 인공신경망의 퍼셉트론의 가중치와 스냅스 간의 가중치 정보만을 이용하여 결과 값을 도출하였다. 인공신경망을 이용하여 사용하는 가장 큰 장점이라고 한다면 한번 실행된 정보를 저장하고 다시 사용할 수 있다는 것이다. 인공신경망에서 Weight값이 일정한 트레이닝을 거친 후 고정된 후에는 다시 사용 할 때에는 추가적인 검색을 진행한 것이 아니라 Weight값에 따라 이동함으로 속도면에서 향상됨을 알 수 있다. 또한 변경되는 경로가 추가되거나 바뀌더라도 해당 경로에 대한 추가 정보를 구성한 후 운영 데이터에서 트레이닝 과정을 진행하고 이를 토대로 저장과 불러오기를 반복할 경우 동적인 인공 신경망을 구축할 수 있다는 특징을 지니게 된다.

4. Verification of Accuracy according to the Hidden Layers

인공신경망에서 HiddenLayer를 2개층으로 구성할 경우 성능에 대해서 확인을 해 보았다. 인공신경망을 구성을 Input Layer와 HiddenLayer를 구성하였다.

```
10 67.5077
20 33.6249
30 12.6843
40 1.06422
50 0.0134181
60 0.0132348
70 0.00646269
80 0.00425121
90 0.00345965
100 0.0030082
[3]: ########
     # 결과 확인
     # 0,1,2,3,4,5,6
     ######
     prediction = tf.argmax(model, 1)
     target = tf.argmax(Y, 1)
     print('예측값:', sess.run(prediction, feed_dict={X: x_data}))
     print('실제값:', sess.run(target, feed_dict={Y: y_data}))

     is_correct = tf.equal(prediction, target)
     accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))
     print('정확도: %.2f' % sess.run(accuracy * 100, feed_dict={X: x_data, Y: y_data}))
     예측값: [0 1 2 3 4 5 6]
     실제값: [0 1 2 3 4 5 6]
     정확도: 100.00
```

Fig. 11 Result of Hidden Layer 2 Training Program

7개의 경로에 대한 정보를 제공하고 1번 HiddenLayer의 퍼

셉트론을 20개 구성하고, 2번 HiddenLayer는 10개로 구성하였다. 3장에서 설계한 모델의 경우 10개의 퍼셉트론을 구성하여 HiddenLayer를 구성한 형태에서 20개의 퍼셉트론 계층과 10개의 퍼셉트론 계층을 HiddenLayer에 추가하였을 때 입력에 따른 출력이 달라지는지 확인하고, 실제 트레이닝의 효과가 있는지 확인하기 위해서 진행을 하였다. 위 Fig. 11과 같이 정확도에 관련된 성능에는 큰 변화가 보이지 않았으며 오히려 정확도가 0이 나오지 않게 되므로 오버피팅이 되어 정확도에서 떨어지는 결과를 나타내었다.

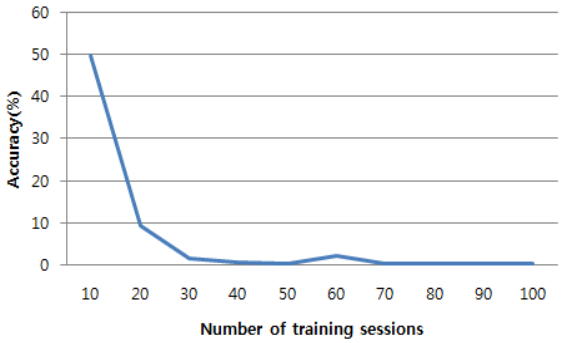

Fig. 12. Hidden Layer 2 Training Program

이와 같이 적은 양의 데이터를 기반으로 인공신경망을 구성할 때에는 Hidden Layer를 추가하여도 성능이 향상될 수 없음을 확인할 수 있다. 이는 적절한 HiddenLayer를 구성하여 학습을 진행하여야 해야 최적의 성능을 나타낼 수 있다. HiddenLayer의 개수는 Fig 12에 나타난 바와 같이 성능의 차이는 거의 나지 않았으며, Fig 12에서 나타나는 정확도 내용 중 5~10번의 정확도 결과에서 정확도가 0이 아닌 다른 수치가 나오는 것으로 오버피팅에 대한 내용을 입증하였다.

Ⅴ. Conclusions

본 논문에서는 주어진 일정 경로를 기반으로 다익스트라 알고리즘을 이용한 최단경로를 구하고 이를 인공신경망을 이용한 경로탐색 결과와 비교하여 보았다. 인공신경망을 이용할 경우 다익스트라 알고리즘을 이용하여 경로탐색을 할 경우보다 경로 탐색 횟수가 증가할수록 인공신경망의 가중치를 이용하여 경로탐색의 횟수를 줄여줌으로써 속도 측면에서 향상됨을 알 수 있다. 또한 인공신경망의 트레이닝 횟수의 경우 적절한 트레이닝 횟수를 구하여야 향상된 정확도를 기반으로 경로탐색을 할 수 있으며, 하드웨어에 무리가 가지 않는 방법을 찾을 수 있음을 알았다. 또한 앞에서 검증한 것과 같이 Hidden Layer를 무조건 추가 한다고 해서 성능이 향상되는 것이 아니란 것도 확인할 수 있었다. 또한 Hidden Layer를 무리하게 추가하거나 트레이닝 과정을 과도하게 진행할 경우 오버피팅이 발생할 수 있음을 확인하였다.

향후, 본 논문의 내용을 기반으로 텐서플로우의 학습능력을 다양한 경로탐색 알고리즘과 비교하여 성능향상에 도움이 될 방법을 추가로 고민해 보아야 하며, 텐서플로우의 지도 학습방법 이외에 비지도 학습과 강화 학습을 이용한 경로탐색 혹은 경로분류를 통한 방법을 추가로 연구할 계획이다.

REFERENCES

[[1] Chung-Hyun Nam, "Open Source AI - Artificial Intelligence Ecosystem and Open Innovation" KISDI Premium Report, KISDI Premium Report, pp.16-10, 2016.

[2] Gwi-Im Jung, Sang-Sung Park, Young-Geun Shin, Dong-Sik Jang. "Construction of Personalized Recommendation System Based on Back Propagation Neural Network". JOURNAL OF THE KOREA CONTENTS ASSOCIATION Vol. 7, No. 12, pp. 292-302, Nov. 2007.

[3] The world of electricity, "Artificial Neural Network and Development Trend", The Korean Institute of Electrical Engineers, Vol. 66, No. 8, pp.36-41, Aug. 2017.

[4] Jun-Yeong Kim, Seog-Gyu Kim. "Design and Implementation of Optimal Path Search Service Using GPS Information in Photo File", Journal of the Korea Society of Computer and Information Vol. 17, No. 12, pp.199-207, Nov. 2012.

[5] D.B. Johnson, "A note on Dijkstra's shortest path algorithm", J. ACM. Vol 20, No 3, 1973.

[6] Dijkstra, E. W., "A note on two problems in connexion with graphs", Numerische Mathematik, vol. 1, no. 1, pp. 269-271, June 1959

[7] Hoon Kim, Young-Jae Jeon, Seung-Yun Lee, Jae-Sung Kim, Jae-Chul Kim, "Outage restoration in electric distribution system using Dijkstra algorithm", The Korean Institute of Electrical Engineers C, pp.1416-1418, July. 1999.

[8] tensorflow : *https://www.tensorflow.org/*

[9] Sam Abrahams, "TensorFlow For Machine Intelli gence : A hands-on introduction to learning algorithms Kindle Edition", Bleeding Edge Press, 2016.

[10] Nick McClure, "TensorFlow Machine Learning" Cookbook, 2017.

[11] Cheol-Han Bae, "A Study of the Automatic Berthing System of a Ship Using Artificial Neural Network", Korean Institute of Navigation and Port Research, Vol. 32, No. 8,. pp. 589-596, Oct. 2008.

Authors

Jun-Yeong Kim received the B.S., M.S.. degrees Department of Information & Communication Engineering, Andong National University in 2008 and 2012. He has been working as System Management Team in Catholic Sangji College, Korea, since 2016.His research interests include Artificial intelligence, LBS and Programming Architecture.

Seog-Gyu Kim received the B.S., M.S.. and Ph.D. degrees in Electronic Engineering from Yonsei University, Korea, in 1990, 1992 and 1997, respectively He worked as senior researcher in SK Telecom from 1997 to 2004. He joined the faculty of the Department of Information & Communication Engineering, Andong National University in 2006. He is currently a Professor in the Dept. of Information & Communication Engineering, Andong National University. He is interested in next-generation network and mobile computing, AI and IoT.