ISSN: 2508-7894 © 2017 KAIA. http://www.kjai.or.kr
Doi: http://dx.doi.org/10.24225/kjai.2017.5.1.18

A Study on the Development of DGA based on Deep Learning

Deep Learning 기반의 DGA 개발에 대한 연구

[1] Jae-Gyun Park(박재균), [2] Eun-Soo Choi(최은수), [3] Byung-June Kim(김병준), [4] Pan Zhang(장범)

[1] First Author Department of Medical IT Marketing, Eulji University, Korea
[2,3] Department of Medical IT Marketing, Eulji University, Korea
[4] Corresponding Author Professor, Dept. School of Business Administration, Shandong University of political science and law, China. E-mail: zhafafa@126.com

Received: June 16, 2017. Revised: June 19, 2017. Accepted: June 20, 2017.

Abstract

Recently, there are many companies that use systems based on artificial intelligence. The accuracy of artificial intelligence depends on the amount of learning data and the appropriate algorithm. However, it is not easy to obtain learning data with a large number of entity. Less data set have large generalization errors due to overfitting. In order to minimize this generalization error, this study proposed DGA which can expect relatively high accuracy even though data with a less data set is applied to machine learning based genetic algorithm to deep learning based dropout. The idea of this paper is to determine the active state of the nodes. Using Gradient about loss function, A new fitness function is defined. Proposed Algorithm DGA is supplementing stochastic inconsistency about Dropout. Also DGA solved problem by the complexity of the fitness function and expression range of the model about Genetic Algorithm As a result of experiments using MNIST data proposed algorithm accuracy is 75.3%. Using only Dropout algorithm accuracy is 41.4%. It is shown that DGA is better than using only dropout.

Keywords : DGA, Deep Learning, Dropout, Genetic Algorithm, Overfitting, AI.

1. 서론

최근 일부 기업은 사용자의 의도를 파악하여 지식을 제공하는 서비스를 상용화 하였다. IBM의 Watson, Google의 AlphaGo, Apple의 Siri가 대표적인 사례이다. 하지만 인공지능을 주도하는 기업은 사용자에 대한 막대한 데이터를 보유하고 있다(Kim & Yu, 2016) .

인공지능의 적중률은 학습데이터가 얼마나 많은지, 적절한 알고리즘인지의 여부에 따라서 달라진다. 실제 학습에 적절한 데이터 값을 찾기 힘들 뿐만 아니라 그 양 또한 부족한 경우가 다분하다. 또한 희귀 질병, 신종 전염병 같은 특정 개체수가 적은 데이터의 경우도 마찬가지이다.

본 논문에서 제안하는 알고리즘인 DGA 는 심층학습 기법인 드롭아웃에 유전 알고리즘을 적용하여 개체수가 적은 데이터라도 높은 적중률을 얻기 위한 목적으로 설계되었다. 과거에 유전 알고리즘을 신경망(Neural Net)에 적용하는 연구가 있었으며, 초기의 히든 유닛 수 파라미터 등을 초기화 할 때 유전 알고리즘을 적용 하였다(Park, 1997). 본 논문이 제안하는 DGA 는 학습 시에 과적합으로 인한 일반화 오차를 최소화하기 위해 드롭아웃에 유전 알고리즘을 적용하는 연구를 하였다.

2. Dropout Genetic Algorithm

2.1. 유전 알고리즘

유전 알고리즘은 John Holland 에 의해 개발되었다. 유전 알고리즘은 자연 선택과 자연 유전학의 메커니즘에 기반 한 알고리즘으로 문자열 구조 중 적자생존을 정보 교환과 결합하여 검색 능력을 갖춘다. 무작위로 추출한 유전 알고리즘은 단순한 무작위 작업이 아닌, 효율적으로 수집된 정보를 이용하여 성능 향상을 위한 새로운 검색 지점을 예측한다 (Goldberg, 1989; Forrest, 1993).

유전 알고리즘의 실행과정으로는 초기화과정과 학습과정으로 구성되어있다. 초기화 과정은 L 길이의 유전자로 이루어진 N 개의 무작위 스트링들을 만들며, 유전자는 2 진 문자나 2 진수를 사용한다. 본 논문에서는 0 과 1 을 사용하였다. 학습과정으로는 무작위로 0 과 1 을 가지는 스트링 벡터로 이루어진 개체군(Population)을 생성 및 초기화한다. 그 후 정의된 적합도 함수에 의해 계산되는 적합도(Fitness)에 비례하도록 현재 개체군으로부터 두 개의 스트링을 선택 한 뒤 쌍으로 다시 조합하여 새로운 두 개의 스트링을 생성 후 자손에 돌연변이를 적용한다. 두 개의 자손을 개체에 추가하거나 토너먼트를 사용하여 두 개의 스트링을 4 명의 부모와 자손들로부터 개체에 넣는다. 기존의 부모와 새로운 세대의 크기가 같을 때 까지 반복한 뒤, 정예주의를 사용하여 부모 세대로부터 가장 적합한 스트링, 즉 정예(Elite)들을 택하고, 선택된 정예 수만큼 자손 세대의 스트링들을 선택된 정예들로 대체한다. 위의 과정은 정지 기준을 충족할 때 까지 반복한다 (Holland, 1992). 본 논문에서는 스트링과 개체 두 단어를 혼용해서 사용한다.

유전알고리즘의 학습은 적합도 함수를 사용하여 적합도가 높은 스트링을 추출하는 것을 목적으로 한다 (Mitchell, 1998). 하지만 데이터에 따라 적합도 함수를 정의해줘야 하는 번거로움이 있다. 또한 스트링은 이진 문자로 표시하기 때문에 모델의 표현범위가 한정적인 단점이 있다. 본 논문에서 제안하는 알고리즘은

신경망에 적용하는 것으로 한정되며, 2.3 섹션에서 적합도 함수를 정의 하였다. 또한 DGA 의 스트링은 노드들의 활성상태 여부만 결정하므로 표현범위에 대한 문제도 해결할 수 있다.

자손을 생성하기 위해 부모를 선택하는 방법으로는 토너먼트 선택법(Tournament Selection), 절단 선택법(Truncation Selection), 적합도 비례선택법(Fitness Proportional Selection), 룰렛 선택법(Roulette Selection) 등이 있다. 본 연구는 룰렛 선택법을 사용하였다. 룰렛 선택법이란, 자손을 생성할 때 적합도가 높은 스트링일 경우 그 값에 비례하여 개체수를 늘려 적합도가 높은 부모의 스트링이 선택될 확률을 증대시키는 방법이다(Lipowski & Lipowska, 2012). 이 때, 유전 알고리즘에서 개체군의 크기를 유지하는 것은 중요하므로 증가된 개체 수만큼 무작위로 개체들을 제거한 뒤 부모스트링을 선택한다.

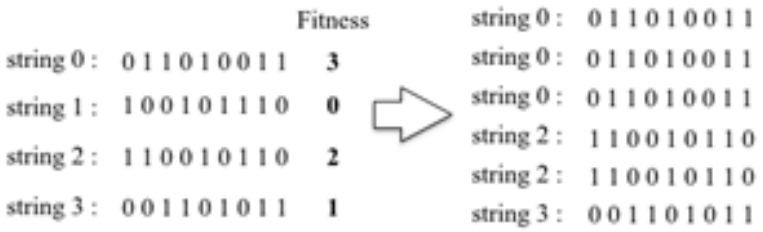

Figure 1. Roulette Selection

Figure 1 은 룰렛 선택법에 대한 예를 나타낸다. 왼쪽의 스트링들은 예비 부모로 생각 할 수 있다. 이 예비 부모들의 적합도에 비례해서 오른쪽처럼 스트링의 개수를 증가시켜준 뒤 증가된 개체군에서 무작위로 부모 스트링을 선택한다. 즉, 룰렛 선택은 스트링들이 적합도에 비례하여 부모로 선택될 확률을 증가시켜주는 방법이다. 하지만 부모를 뽑기 전에 개체 크기를 맞춰주기 위해 위 그림의 경우에서는 두 개의 스트링을 제거한 뒤 부모 스트링을 선택한다. 교배의 쌍을 선택하고 나면 두 스트링을 어떻게 합성해서 새로운 자손을 생성하기 위해 크로스오버(Crossover)를 사용한다. 크로스오버 연산자의 종류로는 단일 점 크로스오버(1-Point Crossover), 다점 크로스오버(K-Point Crossover), 유니폼 크로스오버(Uniform Crossover)가 있으며, 본 연구는 유니폼 크로스오버를 사용한다. 유니폼 크로스오버는 각 스트링의 요소를 두 부모로부터 무작위로 물려받아 자손을 생성한다(Umbarkar and Sheth, 2015).

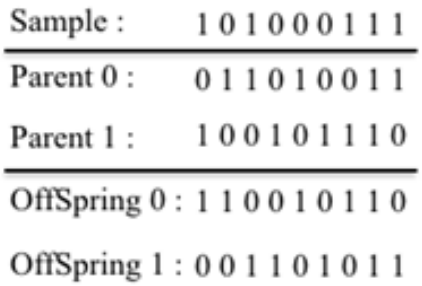

Figure 2. Uniform Crossover

Figure 2 에서 Parent 는 부모로 선택된 스트링을 말하며 이를 통해 자식 스트링인 OffSpring 을 유니폼 크로스오버로 생성하는 단계이다. 샘플을 무작위로 샘플링한 뒤 샘플이 1 이면 Parent 1 로부터 유전자를 물려받고 0 일 경우 Parent 0 으로부터 유전자를 물려받는다. 이와 같은 방법으로 OffSpring 0 이 생성되면 샘플에 보수를 취해서 같은 방법으로 OffSpring 1 을 생성한다. 여기서 개체군의 크기를 맞춰주기 위해 두 개의 부모 스트링으로부터 두 개의 자손 스트링을 생성한다.

크로스오버와 다른 유전적 연산으로는 지역 무작위 탐색을 수행하는 돌연변이(Mutation)가 있다(Elyan and Gaber, 2017). 스트링에서 어떠한 요소 값, 즉 유전자도 특정 확률 P로 변할 수 있다. 보통 특정 확률 P 는 스트링 길이 L에 대해서 $P \approx 1/L$의 값이 사용되어 각 스트링에서 하나의 돌연변이가 생긴다.

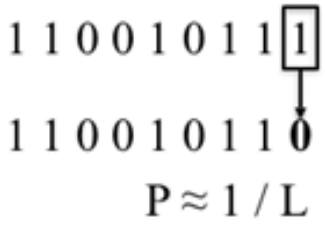

Figure 3. Mutation

Figure 3 은 유전자마다 P ≈ 1/L 의 확률로 보수를 취하는 돌연변이 과정을 나타낸다. 돌연변이는 개체의 다양성을 증가시키기 위한 목적으로 사용되며 학습 시에 지역 극점 수렴의 문제를 해결해주는 효과도 기대할 수 있다(Whitley, 1994). 또한 개체의 다양성을 확보하기 위해 유전학에서의 돌연변이에 비해 비교적 높은 확률을 설정한다.

유전 알고리즘 학습 시에 초기에 좋은 스트링을 얻었을 지라도 반복학습 시에 좋은 스트링이 손실될 수 있다. 이를 해결하기 위한 방법으로 자손이 생성된 후 부모 개체군 중 적합도가 우수한 개체들을 제거하지 않고 다음세대로 물려받아 유전 알고리즘의 성능을 향상시키는 방법으로는 정예주의(Elitism), 토너먼트(Tournament), 틈새(Niching) 등이 있다(Marsland, 2015). 본 연구는 정예주의를 사용하였다. 정예주의는 한 세대에서 적합도가 우수한 정예(Elite) 스트링들을 찾아서 다음 세대에 그대로 적용하는 것인데 무작위로 선택하거나 가장 좋지 않은 적합도를 보이는 정예 수만큼의 스트링들을 선택해서 대체한다.

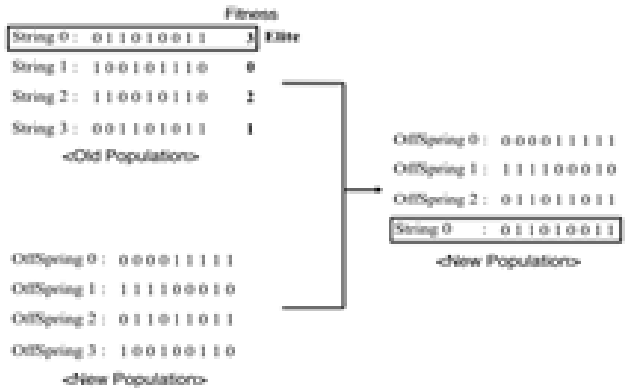

Figure 4. Elitism

Figure 4 는 이전 세대의 개체군(Old Population)과 이로부터 다음 세대의 개체군(New Population)을 생성할 때 정예주의를 적용한 예이다. 정예 수는 사용자 매개변수이며 이에 따라 이전 세대의 개체군에서 적합도가 높은 순으로 정예 스트링들을 선택하여 다음 세대의 개체군에 추가하며 추가된 스트링의 수만큼 스트링을 무작위로 제거하여 개체군의 크기를 유지한다.

2.2. 드롭아웃 (Dropout)

기계 학습에서는 과적합이 문제가 되는 일이 잦다. 과적합이란 신경망이 훈련 데이터에만 지나치게 적응되어 그 외의 데이터에는 제대로 대응하지 못하는 상태를 말한다(Srivastava et al., 2014).

기계학습은 범용 성능을 지향하여 훈련 데이터에 포함되지 않는, 아직 보지 못한 데이터가 주어져도 빠르게 식별해내는 모델을 생성해야 한다. 데이터가 많을 경우 과적합 문제를 해결할 수 있지만, 실제 데이터가 부족한 경우가 많이 있다. 따라서 과적합을 억제하는 기술이 중요하다. 과적합은 매개변수가 많고 표현력이 높은 모델 혹은 훈련 데이터가 적을 경우에 발생한다.

과적합을 억제하기 위하여 사용하는 방법으로는 가중치 감소(Weight Decay), 드롭아웃(Dropout)등이 있다. 가중치 감소는 간단하게 구현할 수 있고 어느 정도 지나친 학습을 억제할 수 있으나 신경망 모델이 복잡해지면 가중치 감소만으로는 대응하기 어렵다. 본 연구에는 드롭아웃에 유전 알고리즘을 적용하여 과적합을 억제하였다.

드롭아웃은 뉴런을 임의로 비활성상태로 만들어 학습하는 방법으로, 훈련 때 은닉층의 노드를 무작위로 골라 비활성화 시킨다. 비활성상태인 노드는 신호를 전달 받지도 전달하지도 않는다. 또한 비활성상태의 노드에 대한 가중치들은 그 단계에서는 수정되지 않아 이전 데이터로 수정된 가중치를 유지하고 있다. 즉, 이전 데이터에 대한 정보를 가지고 있는 노드들을 무작위로 남겨놓는다고 해석할 수 있다.

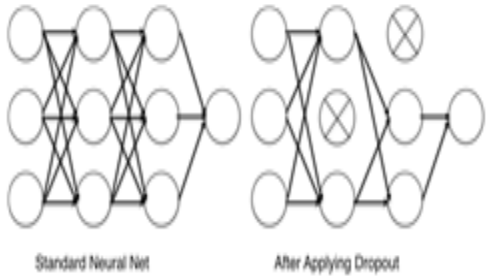

Figure 5. Applying Dropout

드롭아웃은 위의 Figure 5 와 같이 사용자 지정 확률에 따라 노드의 활성상태를 결정하는 방법이다. 기존 신경망의 경우 데이터의 수가 적을 시 과적합이 발생하기 쉽고 이를 방지하기 위해 무작위로 노드들을 비활성 상태로 만들어 학습하는 방법이다. 가중치를 수정할 때 마다 노드들의 활성상태가 무작위로 설정된다. 학습이 완료된 후 테스트단계에서 신호를 전달할 때는 노드가 비활성화 될 확률에 따라서 신호를 조절한다(The Mnist Database, 2017).

2.3. DGA (Dropout Genetic Algorithm) 제안

본 연구에서 제안하는 알고리즘인 DGA 는 드롭아웃에서 노드들을 확률에 따라 비활성상태로 만드는 대신 유전알고리즘을 사용하여 활성상태여부를 결정한다. 이를 통해 좀 더 일관적인 학습을 할 수 있다. 또한 스트링을 통해 활성상태여부를 결정하는 것에 대한 적합도 함수만 정의하면 되므로 사용자가 직접 적합도 함수를 정의할 필요가 없다.
DGA 는 데이터를 충분히 확보하지 못한 경우에 높은 적중률을 얻기 위해 제안된 알고리즘으로써 본 논문에서는 알고리즘에 대한 시간복잡도는 고려하지 않았다.

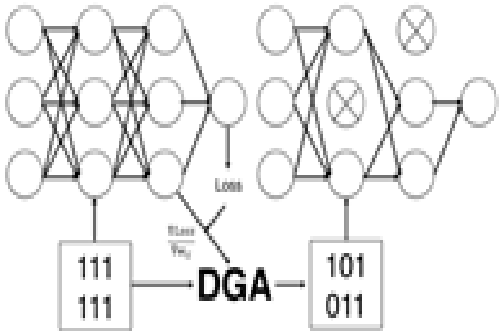

Figure 6. DGA Design

Figure 6 는 DGA 의 설계도이다. 위 그림과 같이 은닉층의 수가 두 개일 때 스트링이 두 개가 필요하며 해당되는 스트링에 따라서 노드의 활성상태가 결정된다. 먼저 모든 노드가 활성상태일 때 한번의 전향 단계와 후향 단계를 거쳐 얻어진 미분 값들과 은닉층에 대한 스트링을 사용하여 유전 알고리즘을 적용한다.

드롭아웃에 유전 알고리즘을 적용하기 위해 적절한 적합도 함수가 필요하다. 이에 대한 적합도 함수는 본 논문에서는 다음과 같이 정의하였다.

$$F_s = \sum_{i=1}^{rnN} \sum_{j=1}^{lnN} \left| g_j^s \cdot \frac{\nabla L}{\nabla w_{ji}} \right| \qquad (1)$$

L은 손실함수를 나타내며 g_j^s는 s번째 스트링의 j번째 원소 즉, 유전자를 나타낸다. Figure 7은 은닉층에서 은닉층으로의 신호 전달 과정을 나타낸다.

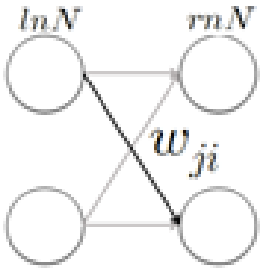

Figure 7. Signal Transduction Process between hidden layers

여기서 왼쪽 은닉층의 노드 수를 식 1 에서 rn, 오른쪽 은닉층의 노드 수를 ln으로 표기하며, Figure 7 의 경우는 2 이다. w_{ji}는 왼쪽 은닉층의 i 번째 노드에서 오른쪽 은닉층의 j 번째 노드를 연결하는 가중치를 표기한다. 즉, 적합도는 스트링에 손실함수에 대한 가중치의 미분 값을 행렬 곱하여 절대 값을 취하여 모든 원소를 더한 값으로 사용한다. 절대 값은 미분 값의 크기만을 고려하기 위하여 사용하였다.

Algorithm 1: DGA

Initialize :
Iter : Train Iteration max number
PN : Population Number
rnN_k : Node number of line k hidden layer
S ⟵ $g_1, g_2, \ldots, g_i, \ldots, g_{rnN} \in [0,1]$ (String)
P ⟵ $S_1, S_2, \ldots, S_s, \ldots, S_{PN}$ (Population)
t ⟵ 0
Train :
while t not liter do
  t ⟵ t+1
  …
  $F_s \leftarrow \sum_{i=1}^{rnN} \sum_{j=1}^{lnN} \left| g_j^s \cdot \frac{\nabla L}{\nabla w_{ji}} \right|$
  newF_s, new$F$ ⟵ GA(F_s, F)
  bestFit ⟵ argsort(newF_s) DESC
  best ⟵ new$F$[bestFit]
  m ⟵ 0
  while m not len(best) do
    if sum(best[i]) > len(best) / 2
      bestString ⟵ best[i]
      break
  end while
  …
  mask ⟵ best*ones(net.shape)
  net ⟵ net*mask
  …
end while
return

Figure 8. DGA Pseudo Code

Figure 8 은 DGA 의 의사코드이며 기존 드롭아웃 방법에서 DGA 가 적용되는 부분만 작성하였다. S 는 0 또는 1 의 값을 가지는 벡터로 초기화된다. 개체군은 무작위로 초기화되며, 한번의 학습단계를 거친 뒤에 얻어진 미분 값을 통해 두 번째 학습단계부터 DGA 가 적용된다. 특정 은닉층의 노드들이 모두 비활성화상태가 되는 것을 방지하기 위해 선택된 스트링의 1 의 수가 스트링의 길이의 반보다 큰 스트링을 고르도록 설계하였다. net 은 해당 은닉층으로 들어오는 신호를 말하며 mask 는 노드의 활성상태를 결정하는 스트링이다.

3. 실험 및 결과

3.1. 실험환경

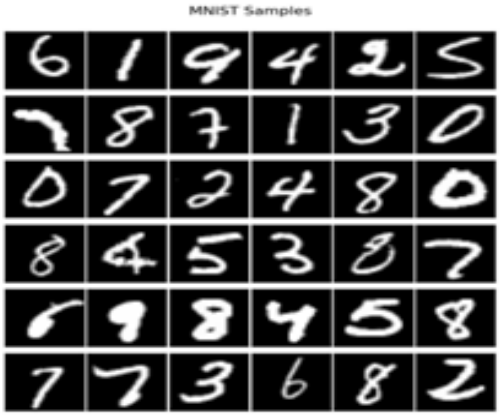

Figure 9. MNIST Dataset

데이터는 Figure 9 의 MNIST 데이터를 사용하였으며 MNIST 의 훈련데이터는 55000 개로 이루어져 (The Mnist Database, 2017). 본 실험에서는 과적합을 유도하기 위해 무작위로 300 개의 데이터만 훈련데이터로 사용하였다. 3 층 신경망과 은닉층 각각 100 개의 노드수를 사용하였다. 출력층의 노드 수는 위의 데이터를 분류하기 위해 10 개로 설정하였다. 개체군의 크기는 30 으로 설정하였고, 실험 환경은 Jupyter Notebook 을 사용하여 Python 언어로 실험하였다. 위의 환경으로 드롭아웃만 적용한 신경망의 학습결과와 DGA 의 학습결과를 비교하였다.

3.2. 실험결과

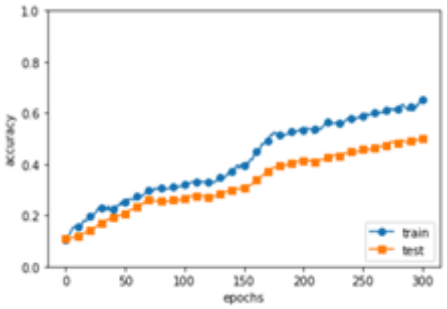

Figure 10. Standard Dropout

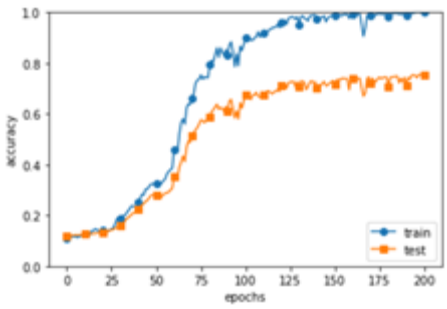

Figure 11. Applying DGA

위의 그림은 학습 반복에 따른 학습 적중률을 나타낸다. Figure 10 은 드롭아웃만 사용한 기존의 방법에 대한 적중률을 나타내며, Figure 11 은 본 논문에서 제안한 DGA 를 사용하여 학습한 모델의 적중률을 나타낸다. 기존 방법은 과적합은 거의 일어나지 않지만 낮은 적중률을 보였다. 그에 반해 DGA 는 높은 적중률을 보였다.

Table 1. Experiment results of Standard Dropout and DGA

	Loss	Train Acc.	Test Acc.
Existing method 200 round	1.74	53.3%	41.4%
Existing method 300 round	1.08	65.3%	50.2%
DGA 200 round	0.022	100%	75.3%

Table 1 은 반복수에 따른 손실 값과 적중률의 비교를 나타낸다. Loss 는 손실 값 Acc 는 적중률을 나타낸다.

4. Conclusion

학습데이터의 양과 알고리즘의 적합성에 따라 인공지능의 적중률이 결정된다. 학습데이터의 양이 충분한 경우에는 알고리즘의 시간복잡도가 우선적으로 고려된다. 하지만 데이터의 양이 충분하지 않은 경우에는 일반화 오차를 최소화하는 것이 중요하다. 본 논문은 데이터가 충분하지

않은 경우에 초점을 맞추어 연구하였다. 실제 희귀병환자에 대한 이미지 데이터 같은 데이터 수가 적지만 분류결과가 좋아야 하는 경우가 있다. 이처럼 데이터의 수가 충분하지 않은 경우들이 많이 발생되는 데 이런 경우에는 DGA 를 사용하기에 적합하다. DGA 는 심층학습 기법의 드롭아웃에 머신러닝 기법인 유전 알고리즘을 적용한 알고리즘이다. DGA 는 유전 알고리즘의 룰렛 선택, 유니폼 크로스오버, 돌연변이, 정예주의를 적용한 알고리즘이다. DGA 의 적합성을 평가하기 위해 MNIST 데이터를 활용하여 분류학습을 하였다. 실험 결과, 기존의 신경망에 드롭아웃만 적용하여 학습할 경우 41.4%의 적중률이 보이는 것에 비해, 제안된 알고리즘 DGA 는 75.3%의 적중률을 보였다. 이를 통해 DGA 가 일반화 오차를 감소시키는 것을 확인 할 수 있었다. 또한 DGA 는 확률적인 방법인 드롭아웃의 비일관성에 대한 문제를 개선할 수 있었다.

References

Kim, Y, J., & Yu, B. E. (2016). Future society change that artificial intelligence technology development will bring, KISTEP 12(lnl), 52-65. Retrieved September 30, 2010, from http://www.cnpc.com.cn/resource/english/images1/2009.pdf

Park, J. S. (1997). Designing Neural Network Using Genetic Algorithm. *The Transactions of the Korea Information Processing Society, 4*(9), 2309-2314.

Goldberg, D. E.(1989). *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley.

Forrest, S. (1993). Genetic algorithms- Principles of natural selection applied to computation. *Science*, 261(5123), 872-878.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

Mitchell, M.(1998). *An introduction to genetic algorithms*. MIT press.

Lipowski, A., & Lipowska, D. (2012). Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications, 391*(6), 2193-2196.

Umbarkar, A. J., & Sheth, P. D. (2015). Crossover Operators in Genetic Algorithms: a review. *ICTACT Journal on Soft Computing, 6*(1), 1083-1092.

Elyan, E., & Gaber, M. M. (2017). A Genetic Algorithm Approach to Optimizing Random Forests Applied to Class Engineered Data. *Information Sciences, 384*, 220-234.

Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and computing, 4*(2), 65-85.

Marsland, S. (2015). *Machine learning: an algorithmic perspective*. CRC press.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research, 15*(1), 1929-1958.

The Mnist Database (2017). Retrieved Jun 15, 2017, from http://yann.lecun.com/exdb/mnist/