Journal of Digital Convergence
Vol. 18. No. 2, pp. 261-268, 2020

ISSN 2713-6434 / eISSN 2713-6442
https://doi.org/10.14400/JDC.2020.18.2.261

학습률(Step-Size)변화에 따른 디지털 신호의 기계학습 방법 개선

지상민[1], 박지은[2*]
[1]충남대학교 수학과 박사과정 학생, [2]대구대학교 인문교양대학 교수

Improvement of existing machine learning methods of digital signal by changing the step-size

Sangmin Ji[1], Jieun Park[2*]
[1]Student, Department of Mathematics, Chungnam National University
[2]Professor, Seongsan Liberal Arts College, Daegu University

요 약 기계학습은 주어진 디지털 신호 Data로부터 비용함수를 만들고, 그 비용함수를 최소화함으로 학습이 이루어진다. 비용함수는 디지털 신호 Data의 양과 인공신경망의 구조에 따라 비용함수에 부분 최솟값(local minimum)들이 생기게 된다. 비용함수의 부분 최솟값들은 학습을 방해하는 요소가 된다. 이러한 방법을 해결하는 여러 방법 중 우리의 제안 방법은 학습률(Step-size)을 변화시키는 방법이다. 학습률을 고정된 상수로 이용하는 기존의 방법과는 다르게 비용함수를 이용한 다변수함수를 이용함으로써 불필요한 기계학습이 이루어지는 것을 방지할 수 있으며, 최솟값으로 가는 최적의 길을 찾을 수 있다. 수치적 실험을 통하여 기존의 방법보다 우리가 제안하는 방법을 이용하여 약 3%(88.8%→91.5%)의 성능이 향상하는 결과를 얻었다.

주제어 : 최적화, 디지털 신호, 기계학습, 학습률, 분류

Abstract Machine learning is achieved by making a cost function from a given digital signal data and optimizing the cost function. The cost function here has local minimums in the cost function depending on the amount of digital signal data and the structure of the neural network. These local minimums make a problem that prevents learning. Among the many ways of solving these methods, our proposed method is to change the learning step-size. Unlike existed methods using the learning rate (step-size) as a fixed constant, the use of multivariate function as the cost function prevent unnecessary machine learning and find the best way to the minimum value. Numerical experiments show that the results of the proposed method improve about 3%(88.8%→91.5%) performance using the proposed method rather than the existed methods.

Key Words : Optimization, Digital signal, Machine Learning, Step-size, Classification

*This work was supported by the National Research Foundation of Korea (NRF-2017R1E1A1A03070311)
*Corresponding Author : Jieun Park(writer2yah@daegu.ac.kr)
Received November 26, 2019 Revised December 30, 2019
Accepted February 20, 2020 Published February 28, 2020

1. 서론

기계학습은 지금까지 사람이 하던 작업을 아주 빠른 속도로 대신하고 있는 실정이다. 더 나아가 기계학습은 인간의 경험을 뛰어넘는 디지털 신호 Data 분석 능력을 갖추어가고 있다. 이러한 상황에서 기계학습의 작동 원리와 이것을 바로 알고 적용하는 것은 아주 중요한 문제이다.

기계학습의 작동 원리는 주어진 디지털 신호 Data로부터 비용함수를 만들고, 그 비용함수를 최소화 하는 변수들(parameters)을 찾는 과정이 기계학습이다. 그러므로 기계학습은 두 가지 문제로 나눌 수 있다. 첫 번째 비용함수를 정의하는 방법, 그리고 이 비용함수를 최소화 하는 방법으로 나눌 수 있다.

우선 이 비용함수를 만드는 방법에 관하여 생각해 보자. 우리가 학습하고자 하는 디지털 신호 Data로부터 도입 값(Input Data)과 출력 값(Output Data)으로 학습 Data를 분류할 수 있다. 여기서 도입 값을 인공신경망을 통하여 계산과정을 거치고 나온 결과 값을 우리는 기계학습을 통하여 나온 값으로 정의한다. 그러면 이렇게 기계학습으로 나온 값이 우리가 기대하는 값 또는 기대하고자 한 값과 일치하는 지를 확인하는 과정을 정의하게 되는데, 이 과정이 바로 비용함수의 정의이다. 즉 기계학습을 통하여 나온 값과 원본 Data로 부터의 기대 값과의 차이의 제곱을 통하여 비용함수를 정의하게 된다. 이러한 방법으로 정의된 비용함수는 학습 Data가 많아질수록 부분 최솟값(local minimum)의 개수가 증가한다[1]. 이러한 부분 최솟값들은 비용함수의 일차 미분 값을 0으로 하는 값으로 학습에 방해 요소가 된다. 기계학습은 비용함수를 최소로 만드는 것을 목표로 했는데 이러한 부분 최솟값에서 학습이 멈추어버리면 기계학습이 잘 이루어졌다고 할 수 없다. 기존의 학습 방법들은 비용함수의 일차 미분을 기반으로 하고 있기 때문에 부분 최솟값에서 더 이상 학습이 이루어지지 않는 현상이 발생한다. 이러한 방법을 해결하고자 우리는 학습률(Step-Size)을 변화시키는 방법을 도입해 본다[2-4].

기존의 방법들은 학습률의 부분을 고정된 값으로 정의하고 있으며, 비용함수의 일차 미분을 기반으로 하고 있다[5-8]. 기존의 방법에서 가장 널리 쓰이고 있으며, Google의 Tensorflow에 장착 되어 있고, 지금도 널리 쓰이고 있는 방법이 Adam 방법이다. 처음 기계학습의 시작은 GD 방법이었으며 이 방법은 비용함수가 오목(Convex)인 상황을 전제로 출발한 개념이었다[9,10]. 그러나 학습의 양이 많아지거나 Deep 구조로 가면 비용함수는 오목인 상황이 아니다[11]. 이러한 문제를 해결하고자 Momentum 방법이 도입되었고, 이 Momentum 방법은 최솟값(Global minimum)에 수렴하지 않는 경우가 많이 발생한다[12]. 사람들은 Momentum 방법을 더욱 발전시켜, Adadelta, RMSprop이 개발되었다[13]. 그리고 이 방법들의 학습 효과를 높이기 위해서 Adadelta 방법과 RMSprop 방법을 결합한 Adam 방법이 만들어졌다[14]. 이처럼 개발된 Adam 방법 또한 학습이 진행될수록 학습률의 변화를 충분히 반영하지 못하고 있다. 즉, 부분 최솟값인 부분에서 학습이 더 이상 이루어지지 않는 일이 발생한다. (Adam방법은 Adadelta와 RMSprop을 합친 방법인 만큼 수치적 실험의 결과가 매우 유사하다. 그렇기 때문에 이 논문에서는 Adam의 성능을 비교하지만 Adadelta와 RMSprop은 비교하지 않는다).

우리는 이 학습률 부분을 비용함수의 변화에 따르는 값으로 변화시키므로 기존의 방법이 가지고 있는 문제점을 해결하고자 한다. 즉 부분 최솟값에서 일차 미분값이 0이므로 발생하는 문제, 즉 학습이 멈추는 문제를 해결한다. 따라서 부분 최솟값에서 비용함수의 값이 크다고 하면 학습은 계속 이루어져야 한다. 그러기 위해서 학습률을 큰값으로 주어 부분 최솟값인 구간을 회피하도록 한다. 또 비용함수의 최솟값이 있는 구간에서는 학습률을 작게 하여 더 이상 학습이 이루어지지 않도록 한다. 즉 학습률이 비용함수와 비례하는 함수로 정의하고자 한다. 따라서 기존의 학습률을 η라 할 때, 우리는 학습률을 비용함수와 비례하는 함수로 $\eta_0 C(w)$ 로 정의하고자 한다. 여기서 $C(w)$는 비용함수를 나타내며, w는 변수이다.

2. 기계학습 방법론

기계학습의 기본은 주어진(학습하고자 하는) Data로부터 예측되는 즉 인간이 생각했을 때 기대되는 결과를 만들어 내기 위하여 기계를 프로그램화 하는 것이다. 이를 위하여 주어진 Data로 부터 결과를 얻을 수 있는 함수를 생각하게 되는데 이때 우리는 인공 신경망 구조를 도입하여 함수 N_f 를 만든다. 우선 가장 단순한 형태는 $N_f = \sigma(wx+b)$ 의 형식을 가지게 된다. σ는 활성화 함수(Activation function)라고 우리는 부른다. 이 함수는 연속 함수를 주로 사용하나, 꼭 연속함수일 필요는 없으며 경우에 따라서 다양한 함수를 도입하여 사용하게 된다. 앞에서 이야기 했듯이, 우리는 인공 신경망을 거친

함수의 값(N_f)이 우리가 기대하는 값(y를 우리가 기대하는 값이라 정했을 때)과 같이 나오기를 원함으로 두 값의 차이가 없도록 하고 싶다. 이로써 우리는 식 $y-N_f$을 만들 수 있고 이것으로부터 비용함수 $C(w)=\sum(y-N_f)^2$의 기본적인 함수를 만들 수 있다. 비용함수는 w가 학습됨에 따라 변화하게 되며, 비용함수 $C(w)$가 최소가 되는 순간에 w의 값을 고정하여 이를 통해 결정된 w로 우리는 $N_f=\sigma(wx+b)$의 기계학습을 완성하게 되는 것이다.

Fig. 1은 이장에서 이야기한 단순 구조에서 더 발전한 복잡한 깊은 구조(Deep structure)의 구조를 도식화한 것이다.

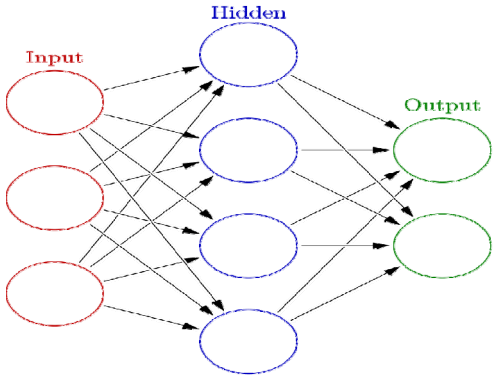

Fig. 1. Deep Neural Netwok (from : https://en.wikipedia.org/wiki/Artificial_neural_network#/media/File:Colored_neural_network.svg)

w를 결정하는 방법은 여러 가지가 있을 수 있는데 여기서는 수학적인 방법인 비용함수의 일차 미분의 값을 0으로 만드는 방법을 사용하고자 한다. 즉 식 $\frac{\partial C}{\partial w}=0$을 만족하는 w의 값을 찾아야하는 문제로 바뀌는 것이다. 이 논문에서 소개하는 방법들은 기본적으로 수학에서 쓰이는 고정점 반복법을 사용하여 문제를 해결하게 된다. 고정점 반복법은 식 $w_{i+1}=w_i-\frac{\partial C}{\partial w}$을 반복적으로 수행하여 결과적으로 수열이 수렴하면 식 $w=w-\frac{\partial C(w)}{\partial w}$을 만족하게 되어 우리가 원하는 $\frac{\partial C}{\partial w}=0$의 식을 얻게 되는 방법이다. 하지만 $\frac{\partial C}{\partial w}$의 크기에 따라서 변수 w가 원하는 값으로 수렴하지 않을 수 있어 hyper parameter η를 도입하여 $w_{i+1}=w_i-\eta\frac{\partial C}{\partial w}$ 얻는다. 각 방법에 따라 다양한 hyper parameter가 있으며 주어진 학습 데이터에 따라서 최적의 hyper parameter를 계산할 수 있다. 하지만 학습 데이터가 추가되거나 바뀌면 hyper parameter를 다시 계산해야 하는 문제가 있어 여러 번의 수치적 경험을 통하여 얻은 값들로 한다. 이 방법을 기본으로 하여 다음에 소개하는 방법들이 연구되어 왔다.

2.1 GD method

고정점 방법에 변수 w의 변화에 가중치(Step size)를 주는 방법으로, 식으로 쓰면 다음과 같다. $w_{i+1}=w_i-\eta\frac{\partial C}{\partial w}$. 여기서 η 값에 따라 w의 변화가 다르게 이루어지게 된다. 특히 η의 값의 결정이 중요하게 되는데 이 값의 정의는 경험적 값으로 결정짓게 된다[11].

2.2 Momentum method

GD 방법이 가지고 있는 문제점인 부분 최솟값인 부분에서 학습이 이루어지지 않는 것을 극복하고자, $\frac{\partial C}{\partial w}$의 값을 상수의 비율로 더하는 방법으로 식을 만든 것이 Momentum 방법이다. 이 방법은 각 학습 단계별 계산되는 비용함수의 일차 미분 값을 쌓아가는 방법으로 $m_i=\alpha m_{i-1}+\beta\frac{\partial C}{\partial w}$식으로 계산된다. 여기서 α와 β는 적당한 상수이다. 이렇게 계산된 m값을 가지고 다음과 같은 식을 얻는다. $w_{i+1}=w_i-\eta m_i$. m의 값의 효과로 $\frac{\partial C}{\partial w}$의 값이 계속 누적되어 w에 영향을 미치는 방법이다[12].

2.3 Adam method

Momentum 방법은 상당히 힘을 포함하고 있어 부분 최솟값인 부분에서 학습이 잘 이루어지나 지나치게 힘이 강한 문제를 가지고 있다. 이점에 착안하여 학습의 정도에 따른 학습 강도를 조절할 수 있는 방법이 Adam 방법이다. 이 방법의 계산은 $w_{i+1}=w_i-\eta\frac{\hat{M}_i}{\sqrt{\hat{V}_i+\varepsilon}}$ 식으로 이루어지며, 여기서 $\hat{M}_i=M_i/(1-\beta_1^i)$이고 $\hat{V}_i=V_i/(1-\beta_2^i)$이다. 또한 식 $M_i=\beta_1 M_{i-1}+(1-\beta_1)\frac{\partial C(w_i)}{\partial w}$을 통하여 M_i의 값을 정할 수 있다. 같은 방법으로 식 $V_i=\beta_2 V_{i-1}+(1-\beta_2)\left(\frac{\partial C(w_i)}{\partial w}\right)^2$

을 통하여 V_i의 값을 정할 수 있다. $\widehat{M}_i$과 $\widehat{V}_i$ 비율이 비슷하며, 이것을 서로 나누는 과정이 있어 Adam 방법은 일정 비율로 학습이 이루어지는 효과가 나타난다[14].

2.4 AdaMax method

AdaMax 방법은 Adam 방법이 가지고 있는 학습의 조정능력이 $\widehat{M}_i$과 $\widehat{V}_i$ 비율이 비슷하여 일정하게 이루어지는 점을 착안한 방법이다. 즉 조정능력의 변화를 좀 더 크게 만들고자 분모에 해당되는 $\widehat{V}_i$의 값을 $u_i = \max(\beta_2 u_{i-1}, \left|\frac{\partial C}{\partial w}(w_i)\right|)$ 로 바꾸어 계산하는 방법이다. $w_{i+1} = w_i - \eta\frac{\widehat{M}_i}{u_i}$ 다른 변수들은 2.3에서 정의한 변수들과 같다[14, 15].

3. 비용함수를 이용한 학습률 변동 방법

이처럼 기존의 방법들은 학습률의 계산에서 상수 값을 적용하였다. 우리는 이 학습률을 비용함수의 변화에 비례하는 값으로 바꾸어 방법론을 만든다. 다시 말해서, 기존의 각 방법에서의 학습률 η를 비용함수의 변화에 상응하는 값으로 바꾼다. $\eta \Rightarrow \eta(\mathrm{w}) = \mathrm{C}(\mathrm{w})\eta_0$ 이처럼 변화를 주는 방법을 도입함으로 비용함수의 변화에 대한 학습률을 변화시키고자 한다. 학습률을 변화하는 방법은 각 방법에 적용가능하다.

3.1 GD method의 학습률 변화방법(S-GD)

기존의 GD 방법 $w_{i+1} = w_i - \eta\frac{\partial C}{\partial w}$에서 $w_{i+1} = w_i - C(w_i)\eta_0\frac{\partial C}{\partial w}$로 학습률을 변화시킨다.

3.2 Momentum method의 학습률 변화방법 (S-Momentum)

기존의 Momentum 방법$w_{i+1} = w_i - \eta m_i$에서 $w_{i+1} = w_i - C(w_i)\eta_0 m_i$로 학습률을 변화시킨다.

3.3 Adam method의 학습률 변화방법 (S-Adam)

기존의 Adam 방법$w_{i+1} = w_i - \eta\frac{\widehat{M}_i}{\sqrt{\widehat{V}_i}+\varepsilon}$에서 $w_{i+1} = w_i - C(w_i)\eta_0\frac{\widehat{M}_i}{\sqrt{\widehat{V}_i}+\varepsilon}$로 학습률을 변화시킨다.

3.4 AdaMax의 학습률 변화방법(S-AdaMax)

기존의 AdaMax의 방법$w_{i+1} = w_i - \eta\frac{\widehat{M}_i}{u_i}$에서 $w_{i+1} = w_i - C(w_i)\eta_0\frac{\widehat{M}_i}{u_i}$로 학습률을 변화시킨다.

4. 수치적 결과를 통한 성능 평가

이 장에서는 기존의 방법들과 우리가 제안한 학습률의 변화에 따른 각 방법들의 수치적 실험의 결과를 보일 것이다. 다양한 방법이 있으나 우선은 non-convex의 상황에서의 수치적 결과들의 비교, 이를 위한 1차원 문제와 2차원 문제 그리고 일반적인 디지털 영상의 구분에 관한 문제(개와 고양이 구분)에 관하여 실험한다.

4.1 1차원 비용함수 문제

1차원 비용함수를 이용한 실험에서는 최솟값과 부분 최솟값이 1개씩 있는 경우에 대하여 우리의 방법이 보다 효과적임을 보일 것이다. 4.1.1에서는 각 방법들이 부분 최솟값을 지나서 최솟값에 도달할 수 있는지를 실험할 것이며, 4.1.2에서는 최솟값 다음에 부분 최솟값이 위치한 비용함수를 이용하여 부분 최솟값에 도달하였을 때 최솟값이 아니라면 도달했던 부분 최솟값에서 빠져나올 수 있는지를 실험할 것이다.

4.1.1 부분 최솟값 후 최솟값

우리는 비용함수를 왼쪽에는 부분 최솟값이 있고 오른쪽에는 최솟값이 있는

$C(w) = (w+5)(w+3)(w-1)(w-10)/800 + 3$

로 정의하고 변수의 초기 값을 -9로 설정하여 부분 최솟값을 지나서 최솟값까지 도달할 수 있는지를 실험할 것이다. Table 1은 이 실험에서 이용한 각 방법들의 hyper parameter들(여러 번의 수치적 경험을 통하여 얻은 값들)을 나타낸다.

Table 1. Initial hyper parameters in Fig. 2 and Fig. 3

Method	Learning Rate	S-Method's Learning Rate	Hyper Parameters
GD	0.2	6.0	
Momentum	0.2	0.2	$\lambda = 0.9$
Adam	0.2	6.0	$\beta_1 = 0.9,$ $\beta_2 = 0.999,$ $\epsilon = 10^{-8}$
AdaMax	0.2	6.0	$\beta_1 = 0.9,$ $\beta_2 = 0.999$

Iteration number = 100

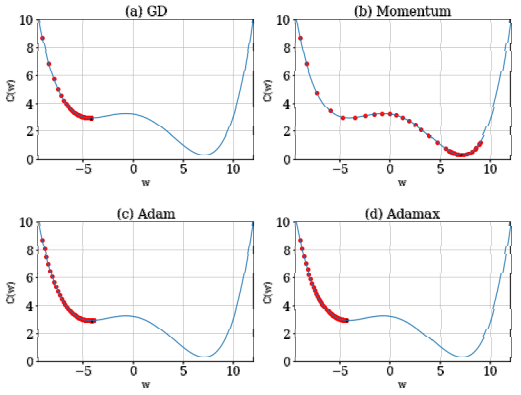

Fig. 2. Solution behaviors of (a) GD, (b) Momentum, (c) Adam and (d) AdaMax scheme.

Fig. 2는 기존의 방법들이 주어진 hyper parameter에서 학습을 하며 변수를 변화시키는 것을 그래프에 나타낸 것이다. 기존의 방법들은 부분 최솟값을 벗어나지 못하는 것을 보여준다.

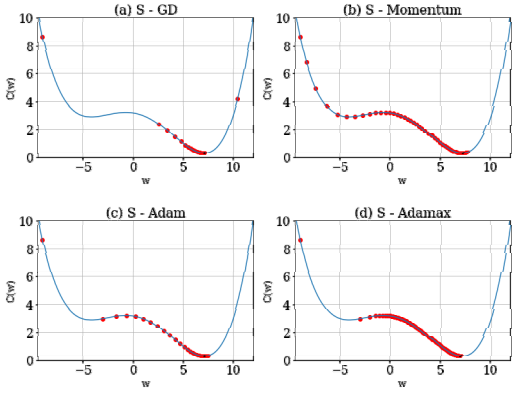

Fig. 3. Solution behaviors of (a) S-GD, (b) S-Momentum, (c) S-Adam and (d) S-AdaMax scheme.

Fig. 3은 기존의 방법에 학습률을 변화시키는 기법을 적용한 후 학습을 시킨 결과이며 모든 방법들이 부분 최솟값을 지나 최솟값으로 도달했음을 보여준다. Fig. 2와 Fig. 3을 비교하면 학습률을 상수가 아닌 함수로 바꾸어 학습을 시켰을 때 더 좋은 효과가 나타나는 것을 확인할 수 있다.

4.1.2 최솟값 후 부분 최솟값

이번에는 4.1.1의 실험과는 반대로 왼쪽에 최솟값이 있고 오른쪽에 부분 최솟값이 있는

$C(w) = 0.4(w-0.12)(w-1)(w+1.1)(w+0.5) - x^2 + 0.1x + 2$

를 비용함수로 이용하고 변수의 초기 값을 -9로 설정하여 최솟값 뒤에 부분 최솟값이 있는 상황에 대하여 실험을 하였다. Table 2는 이 실험에서 이용된 hyper parameter들을 보여준다.

Table 2. Initial hyper parameters in Fig. 4 and Fig. 5

Method	Learning Rate	S-Method's Learning Rate	Hyper Parameters
GD	0.5	0.2	
Momentum	0.3	0.34	$\lambda = 0.6$
Adam	0.5	0.7	$\beta_1 = 0.9,$ $\beta_2 = 0.999,$ $\epsilon = 10^{-8}$
AdaMax	0.5	0.7	$\beta_1 = 0.9,$ $\beta_2 = 0.999$

Iteration number = 100

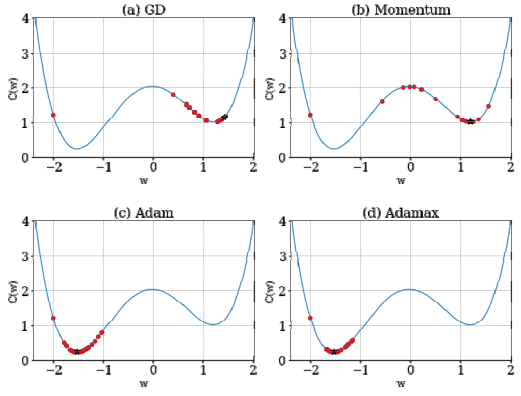

Fig. 4. Solution behaviors of (a) GD, (b) Momentum, (c) Adam and (d) AdaMax scheme.

Fig. 4는 기존의 방법들에 Table 2에 주어진 hyper parameter들로 설정을 한 후 학습시킨 결과를 나타낸

다. Fig. 4-(a)와 (b)를 보면 GD와 Momentum은 최솟값을 지나 부분 최솟값에 도달하였지만 부분 최솟값에 그대로 머물러 최솟값으로 돌아오지 못하는 것을 보여준다.

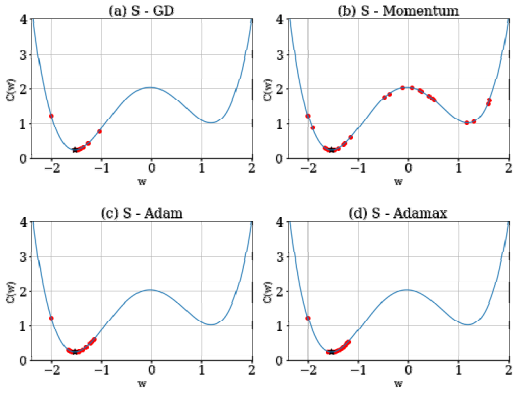

Fig. 5. Solution behaviors of (a) S-GD, (b) S-Momentum, (c) S-Adam and (d) S-AdaMax scheme.

Fig. 5는 기존의 방법들에 학습률을 변화시키는 기법을 적용한 후 학습을 시킨 결과를 나타낸다. Fig. 5-(a)와 (b)를 보면 Fig. 4-(a), (b)와는 다르게 GD는 최솟값에 도달한 후 부분 최솟값으로 가지 않으며, Momentum은 최솟값을 지나 부분 최솟값 방향으로 가지만 다시 최솟값 방향으로 돌아오는 것을 알 수 있다. 이것은 기존의 방법들로 학습을 시킬 때 부분 최솟값에 빠져 학습이 잘 되지 않는 상황이 오더라도 학습률을 변화시키는 기법을 적용한다면 이러한 문제점에서 빠져나올 수 있다는 것을 보여준다.

4.2 2차원 문제

이 실험에서는 non-convex 2변수 함수를 비용함수로 이용하여 parameter들이 잘 학습되는지 확인하기 위하여 Styblinski-Tang 함수라고 불리는

$$C(w_1, w_2) = \frac{1}{2}((w_1^4 - 16w_1^2 + 5w_1) + (w_2^4 - 16w_2^2 + 5w_2))$$ 를 비용함수로 이용한다.

이 함수는 3개의 부분 최솟값과 1개의 최솟값을 가지고 있으며 최솟값은 (-2.903534, -2.903534)이다. Table 3은 이 실험에서 사용한 방법들의 hyper parameter들을 나타내며 초기 값을 (6, 0)로 하여 100번을 반복하여 학습을 진행하였다. 이에 따른 결과는 Fig. 6에 보여진다.

Table 3. Initial hyper parameters in Fig. 6

Method	Learning Rate	S-Method's Learning Rate	Hyper Parameters
GD	$5(10^{-3})$	0.03	
Momentum	$5(10^{-3})$	0.004	$\lambda = 0.9$
Adam	$5(10^{-1})$	5.0	$\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$
AdaMax	$5(10^{-1})$	10.0	$\beta_1 = 0.9$, $\beta_2 = 0.999$

Iteration number = 100

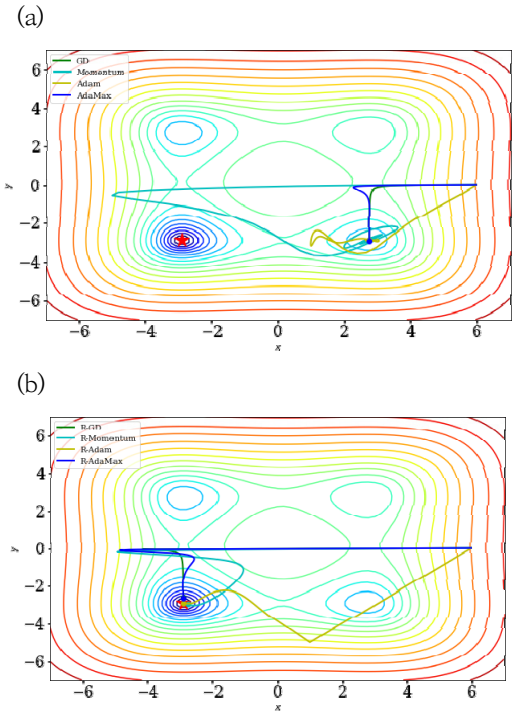

Fig. 6. Comparing S- scheme with others for four layers. starting point (6, 0)

Fig. 6의 (a)는 기존 방법들이 주어진 설정에서 변수들을 학습시킨 결과를 나타내며 (b)는 학습률을 변화시킨 방법들이 주어진 설정에서 변수들을 학습시킨 결과를 나타낸다. 기존의 방법들은 Fig. 5의 (a)에서 보여 지는 것처럼 초기 값 주변에서의 부분 최솟값을 찾아서 이동한다. 하지만 학습률을 함수로 변화시킨 기법을 적용한 방법들은 초기 값에서 떨어져 있는 최솟값을 찾아간다. 이것은 다변수 비용함수에서도 학습률을 변화시키는 방법이 효과가 있음을 보여준다.

4.3 디지털 영상신호 구분 방법(개, 고양이 사진을 이용한 구분 방법)

이 실험에서는 CIFAR-10 (Canadian Institute For Advanced Research) 이라고 불리는 공개된 학습 데이터를 이용하여 실험을 하였다. CIFAR-10은 비행기, 자동차, 새, 고양이, 사슴, 개, 개구리, 말, 배, 트럭의 10종류의 이미지를 가지고 있으며 주로 분류 문제를 해결할 때에 많이 이용되는 데이터 집합이다. 하지만 이 실험에서는 10가지 종류를 모두 이용하지 않고 개와 고양이의 이미지만 이용한다. CIFAR-10의 하나의 이미지는 크기가 가로 32, 세로 32인 RGB 데이터이다. 그렇기 때문에 하나의 이미지는 3072차원 벡터로 볼 수 있다. 같은 이진 분류 문제를 풀더라도 학습 데이터의 차원이 높아지면 학습이 잘 되기 위해서 더 복잡한 학습 모델을 필요로 한다.

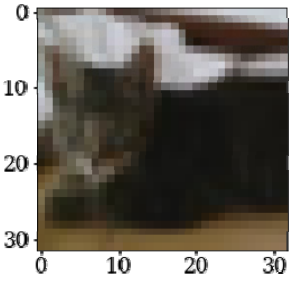

Fig. 7. One of cat category image of CIFAR-10 data set

Fig. 7은 CIFAR-10의 고양이 이미지 중 하나이다. 우리는 각 방법들의 성능을 실험하기 위하여 히든 레이어가 8개인 NN 모델을 위에서 언급한 CIFAR-10의 개와 고양이 데이터 집합으로 500번 학습시킨 후 Fig. 7의 고양이 이미지를 이용하여 학습이 잘 되었는지를 확인하였다.

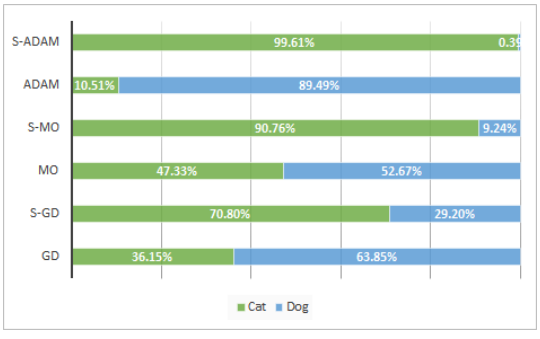

Fig. 8. Result of subsection 4.3 with input data Fig. 7

Fig. 8은 그 결과를 나타낸다. Fig. 8에서 녹색 부분은 각 방법들이 Fig. 7을 고양이로 판단한 확률이며 파란색은 개로 판단한 확률이다.

해당 이미지에 대하여 테스트를 하였을 때 학습률에 변화를 주는 기법을 적용한 방법들은 적용하지 않은 방법들보다 좋은 결과를 나타내었다.

5. 결론

효과적인 기계학습이 이루어지기 위해서는 처음의 의도대로 비용함수의 최솟값(Global minimum)을 찾아가는 방법으로 학습이 이루어져야한다. 우리는 이러한 전제하에 학습률(Step-size)을 조절하는 방법을 도입하였다. 우리의 방법은 우리의 의도대로 비용함수의 최솟값을 찾아가는데 있어 별 무리가 없으며, 학습의 원활한 진행이 수치적 방법들을 통하여 확인되었다. 특히 우리의 방법은 학습이 이루어지는 과정에서 학습의 크기(size)의 변화만을 바꾸는 방법으로 기존의 모든 방법에 적용 가능한 용이함이 있다. 또한 비용함수의 변화에 따른 크기의 변화로 비용함수가 최소의 값에 도달하면 자동적으로 학습을 멈추는 효과가 있다. 이 모든 결과는 수치적 실험과 이론을 통하여 확인된다. 또한 디지털 신호는 기계학습 과정을 통하여 다양한 값으로 변화하여 계산된다. 따라서 디지털 신호가 가지고 있는 고유의 특성(정수화)을 유지하기 위해서는 출력 값의 다변화, 벡터화 등이 필요하다. 영상이나 이미지에서 어떤 대상을 분류할 때에 이용하는 기계학습 방법은 출력값을 벡터화하여 변수들을 학습하기 때문에 디지털 신호의 특성을 유지한다고 볼 수 있으며 이것은 디지털 신호를 이용하는 많은 곳에 기계학습을 적용시킬 수 있다는 것을 뜻한다. 따라서 디지털 신호에서 좋은 효과를 얻기 위해서는 기계학습에서의 변수들의 최적화 과정이 중요하며 우리가 제안하는 방법을 이용하여 보다 효율적인 최적화를 할 수 있다.

REFERENCES

[1] D. Yi, S. Ji & S. Bu. (2019). An Enhanced Optimization Scheme Based on Gradient Descent Methods for Machine Learning. *symmetry, 11*(7), 942-959.

[2] H. Zulkifli. (2018). *Understanding Learning Rates and How It Improves Performance in Deep Learning.*

[Online]
https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10?gi=e082fbb7c7a9.

[3] S. Lau. (2017). *Learning Rate Schedules and Adaptive Learning Rate Methods for Deep Learning. Towards Data Science*. [Online]
https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1.

[4] G. Aurélien. (2017). *Gradient Descent*. Hands-On Machine Learning with Scikit-Learn and TensorFlow. O'Reilly. pp. 113-124. ISBN 978-1-4919-6229-9.

[5] J. Duchi, E. Hazan & Y. Singer. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res, 12*, 2121-2159.

[6] Y. LeCun, L. Bottou, Y. Bengio & P. Haffner. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE, 86*, 2278-2324.

[7] R. Pascanu & Y. Bengio. (2013). Revisiting natural gradient for deep networks. *arXiv:1301.3584*.

[8] J. Sohl-Dickstein, B. Poole & S. Ganguli. (2014). Fast large-scale optimization by unifying stochastic gradient and quasi-newton methods. *In Proceedings of the 31st International Conference on Machine Learning*. (pp. 604-612). Beijing, China.

[9] P. Baldi & K. Hornik. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks, 2(1),* 53-58.

[10] M. Zinkevich. (2003). Online convex programming and generalized infinitesimal gradient ascent. *In Proceedings of the Twentieth International Conference on Machine Learning*. (pp. 928-936). Washington, DC, USA.

[11] C. T. Kelley. (1995). *Iterative methods for linear and nonlinear equations(Volume 16)*. In Frontiers in Applied Mathematics; SIAM: Philadelphia, PA, USA.

[12] I. Sutskever, J. Martens, G. Dahl & G. E. Hinton. (2013). On the importance of initialization and momentum in deep learning. *In Proceedings of the 30th International Conference on Machine Learning*. (pp. 1139-1147). Atlanta, GA, USA.

[13] M. D. Zeiler. (2012). Adadelta: An adaptive learning rate method. arXiv:1212.5701.

[14] D. P. Kingma & J. L. Ba. (2015). Adam: A Method for Stochastic Optimization. *In Proceedings of the 3rd International Conference for Learning Representations*. (pp. 7-9). San Diego, CA, USA.

[15] M. J. Kochenderfer & T. A. Wheeler. (2019). *Algorithms for Optimization*. London : The MIT Press Cambridge.

지 상 민(Sangmin Ji) [정회원]

· 2017년 2월 : 충남대학교 수학과(이학사)
· 2017년 3월 ~현재 : 충남대학교 수학과 대학원 석박통합과정
· 관심분야 : 인공지능, 기계학습, 수치해석, 대수학
· E-Mail : smji@cnu.ac.kr

박 지 은(Jieun Park) [정회원]

· 1998년 2월 : 이화여자대학교 과학교육과(화학전공) (학사)
· 2013년 2월 : 이화여자대학교 과학교육학과 (박사)
· 2015년 3월 ~ 현재 : 대구대학교 인문교양대학 조교수
· 관심분야 : 문제해결력, 표상, 인공지능
· E-Mail : writer2yah@daegu.ac.kr