MỘT PHƯƠNG PHÁP TĂNG TỐC KHẢ NĂNG HỘI TỤ ĐỐI VỚI GRADIENT DESCENT

Hoa Tất Thắng[1*], Trần Văn An[1], Đoàn Văn Hòa[2],
Lê Hoàng Minh[2], Hoàng Xuân Trung[3]

Tóm tắt: *Bài toán tối ưu hóa là bài toán tìm kiếm lời giải tốt nhất trong các lời giải khả thi. Tối ưu hóa có nhiều ứng dụng trong lĩnh vực học sâu và có nhiều ứng dụng trong đời sống thực tế như các bài toán phân loại, nhận dạng ảnh, các bài toán tối đa hóa doanh thu hay giảm chi phí, thời gian sản xuất. Phương pháp gradient descent thường được sử dụng để nhanh chóng tìm được nghiệm tối ưu của bài toán. Trong bài báo này, tác giả sử dụng một phương pháp mới để nhanh chóng tìm kiếm tham số học (learning rate) hợp lý dựa trên ý tưởng của nguyên lý điều khiển luồng và chống tắc nghẽn trong mạng viễn thông nhằm tăng tốc khả năng hội tụ của bài toán so với phương pháp gradient descent thông thường.*

Từ khóa: Gradient descent; Máy học; Tham số học; Điểm khởi tạo; Hàm mất mát.

1. ĐẶT VẤN ĐỀ

Ngày nay, rất nhiều các bài toán khoa học và kỹ thuật ứng dụng rộng rãi trong đời sống xã hội có thể xem như các bài toán tối ưu. Học máy và tối ưu hóa ngày càng có nhiều ứng dụng rộng rãi [2, 7]. Phần lớn các vấn đề học máy có thể xem như bài toán tối ưu của hàm số được biết đến với tên là hàm mất mát [1]. Hơn nữa, tối ưu hóa đóng một vai trò quan trọng trong lĩnh vực học sâu, đặc biệt là trong các vấn đề liên quan đến hình ảnh (ví dụ, phân loại hình ảnh [8], nhận dạng ảnh [5]. Điều này là do hầu hết mọi quá trình chủ yếu là nhằm cực tiểu hóa hoặc cực đại hóa một số lượng nào đó: tối đa hóa doanh thu, độ chính xác, hiệu suất hoặc giảm thiểu sai sót, chi phí, thời gian.

Hiện nay, có một số phương pháp tìm nghiệm Gradient Descent (GD) trong đó điển hình như: Batch Gradient Descent, Fast GD, Stochastic GD,... Mỗi phương pháp đều có những ưu nhược điểm riêng. Điểm chung trong các phương pháp này là việc tìm nghiệm còn phụ thuộc và điểm khởi tạo và tham số học. Bài báo đưa ra phương pháp tiếp cận mới là tìm ra tham số học tối ưu khi chọn điểm khởi tạo ban đầu, từ đó có thể sử dụng các phương pháp sau đó để tìm được nghiệm GD.

Sử dụng learning rate [3] trong gradient descent tỏ ra hiệu quả trong rất nhiều trường hợp. Tuy nhiên, một vấn đề chính ở đây là việc lựa chọn learning rate. Nếu learning rate chúng ta chọn quá lớn, điểm x_t chỉ loanh quanh điểm x^* mà không bao giờ tiến tới được x^*, thậm chí trong một số trường hợp càng ngày càng xa điểm x^* hơn. Nếu learning rate chúng ta chọn quá nhỏ thì rất lâu mới có thể tiếp cận được điểm local minimum. Một vấn đề nữa của learning rate là nếu chúng ta chọn điểm khởi tạo xa với điểm local minimum thì cũng cần phải qua rất nhiều bước chúng ta mới có thể tới điểm hội tụ. Vấn đề đặt ra ở đây là cần cải tiến cách chọn learning rate để tăng cường khả năng hội tụ của thuật toán gradient descent.

Một số phương pháp đã nghiên cứu đề xuất một số giải thuật tối ưu đối với Gradient Descent. Như momentum [10] hỗ trợ tăng tốc nhằm giúp phương pháp Gradient Descent thoát khỏi điểm cực tiểu địa phương. Tuy nhiên, phương pháp này nếu ngay từ đầu chọn learning rate không phù hợp (quá lớn) thì sẽ không tìm được nghiệm. Hay Adagrad [6] là phương pháp tìm tham số học thích nghi, sử dụng learning rate bé đối với các thuộc tính thường xuyên sử dụng và tham số học lớn đối với các thuộc tính ít sử dụng. Tuy nhiên, việc phân biệt này là khó thực hiện, và trên thực tế Adagrad thường sử dụng tham số học bé (0.01). Trong "Cyclical Learning Rates for Training Neural Networks." [9], Leslie N.

Smith và công bố [11] đề xuất có thể sử dụng giá trị learning rate thay đổi theo theo chu kì (tam giác, hình sin, parabol,...). Phương pháp này cũng hiệu quả so với phương pháp sử dụng learning rate cố định ở chỗ số vòng lặp ít hơn, tốt hơn trong 1 số bài toán tìm cực trị khi phải nhảy qua các điểm yên ngựa. Đây là một giải pháp tốt tuy nhiên ở đây cần phải xác định được điểm giới hạn trên và giới hạn dưới cho learning rate và chiều dài của chu kì lặp, nhiều khi chỉ là ước lượng. Thêm nữa là cơ sở toán học việc thay đổi theo chu kì tham số học learning rate cũng chưa được chứng minh rõ ràng.

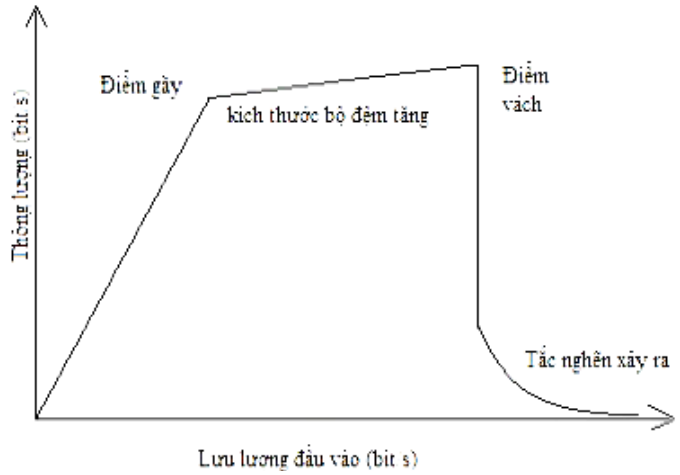
Từ những vấn đề nêu trên, nhóm tác giả đưa ra đề xuất ý tưởng cài đặt đơn giản hơn để chọn learning rate phù hợp dựa trên *phương thức điều khiển luồng và chống tắc nghẽn trong mạng viễn thông TCP (transmission control protocol)[3]*. Thuật toán sẽ tăng giá trị tham số học nếu giá trị tham số học là nhỏ và giảm giá trị tham số học nếu giá trị này lớn đến một giá trị phù hợp để có thể áp dụng và tăng tốc khi áp dụng vào Gradient Descent.

2. TĂNG KHẢ NĂNG HỘI TỤ DỰA TRÊN TÌM THAM SỐ HỌC PHÙ HỢP

2.1. Bài toán điều khiển luồng và chống tắc nghẽn trong mạng viễn thông TCP

Điều khiển luồng trong mạng viễn thông là phương pháp kiểm soát thông tin giữa hai thiết bị đầu cuối cụ thể, nó là một trong những biện pháp giúp cho lưu thông lưu lượng giữa các thiết bị thu và phát.

Tắc nghẽn là hiện tượng mà thông lượng của mạng giảm và trễ tăng lên khi lượng thông tin đi vào mạng tăng. Điều khiển luồng cung cấp cơ chế giới hạn lưu lượng thông tin đi vào mạng nhằm tránh hiện tượng tắc nghẽn, đảm bảo việc truyền thông tin của phía phát không vượt quá khả năng xử lý của phía thu, tránh tràn vùng đệm của người nhận.

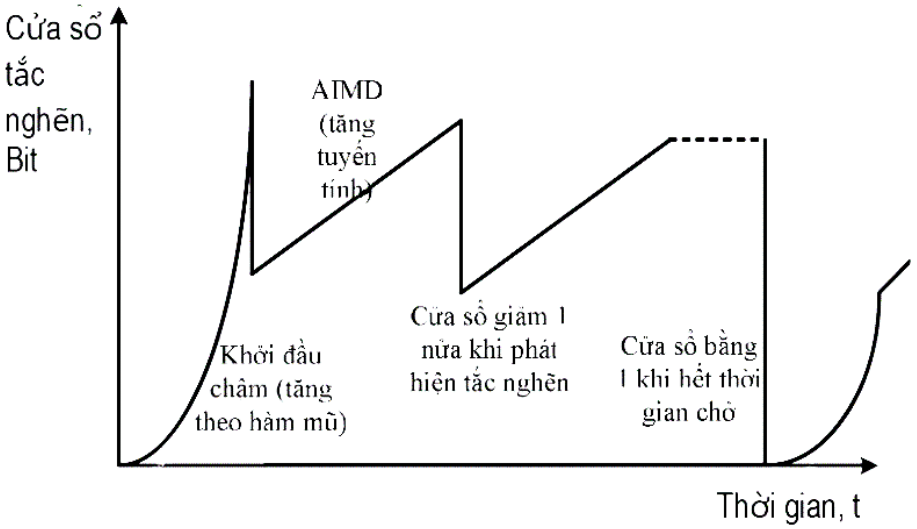


Hình 1. *Quá trình xảy ra tắc nghẽn trong mạng viễn thông.*

Quá tải làm thông lượng suy biến như được chỉ ra trên hình 1. Đồ thị biểu hiện mối quan hệ giữa thông lượng với lưu lượng đưa vào. Khi lưu lượng đưa vào nhỏ, thông lượng tăng tuyến tính ở bên trái điểm gãy. Thông lượng lớn nhất khi lưu lượng đưa vào gần với độ lớn băng thông gây ra hiện tượng thắt cổ chai và thông lượng tăng chậm theo kích thước bộ đệm. Khi lưu lượng đưa vào tiếp tục tăng là lúc xảy ra hiện tượng nghẽn và thông lượng giảm đột ngột từ điểm vách xuống một giá trị rất nhỏ.

Phương pháp điều khiển tắc nghẽn trong TCP đi vào mô hình chậm khi bắt đầu kết nối. Trong suốt quá trình khởi đầu chậm, phía gửi tăng tốc độ theo hàm mũ. Khi bắt đầu khởi đầu chậm, cửa sổ tắc nghẽn thiết lập là 1 đoạn, phía gửi gửi 1 đoạn và chờ nhận được xác nhận từ phía nhận. Khi bên nhận được xác nhận, phía gửi tăng cửa sổ chống tắc nghẽn lên 1, gửi 2 đoạn, và đợi xác nhận tương ứng. Mỗi khi xác nhận đến, phía gửi lại gửi gấp đôi lên 2 đoạn,

4 đoạn,... dẫn đến tăng theo hàm mũ của cửa sổ chống tắc nghẽn (hình 2). TCP thoát khỏi khởi đầu chậm khi đoạn bị mất. Khi đó, phía gửi giảm cửa số tắc nghẽn đi một nửa.



Hình 2. *Cửa sổ tắc nghẽn TCP.*

Lấy ý tưởng từ mô hình này, nhóm tác giả cũng đưa ra ý tưởng tìm learning rate phù hợp cho thuật toán GD với nguyên tắc khởi đầu chậm và tăng theo hàm mũ.

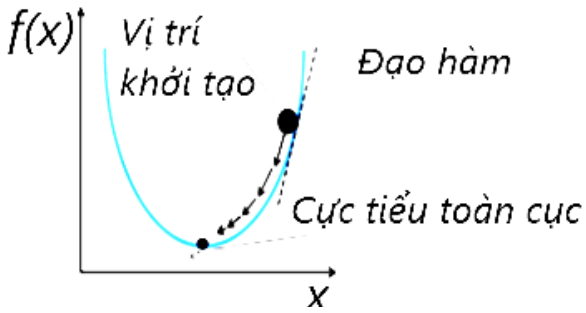
2.2. Gradient descient và thuật toán tìm tham số học

Hầu hết các vấn đề tối ưu hóa có thể được xây dựng giống như sau:

$$x^* = \arg\min f(x) \tag{1}$$

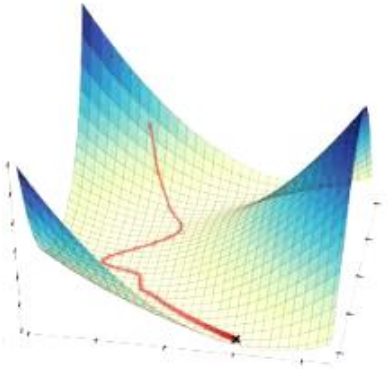
Trong đó, x^* là điểm cực trị (min), f là hàm mất mát. Nói chung, việc tìm các điểm minimum toàn cục thường gây ra khó khăn không nhỏ đối với bài toán tìm nghiệm tối ưu. Trong một số trường hợp là bất khả thi, do vậy, người ta cố gắng tìm các điểm local minimum, và ở một mức độ nào đó, điểm local minimum này có thể coi như là nghiệm của bài toán (hình 3).

Các điểm local minimum có đặc điểm chung là tại đó đạo hàm của chúng đều bằng không. Và sử dụng gradient descent là phương pháp phổ biến để giải quyết bài toán tìm nghiệm tối ưu kiểu này. Thông thường người ta chọn một điểm mà chúng ta có thể coi là gần với nghiệm của bài toán, sau đó dùng các phép lặp để tiến đến điểm tối ưu cần tìm, càng gần với điểm tối ưu thì đạo hàm của chúng càng có giá trị gần 0.



Hình 3. *Gradient descent trong không gian hai chiều.*

Trong không gian nhiều chiều, nguyên tắc làm việc của gradient descent có thể được hình dung như ở hình 4.



Hình 4. *Gradient trong không gian ba chiều.*
(Nguồn: http://dsdeepdive.blogspot.com/2016/03/optimizations-of-gradient-descent.html).

Minh họa trên hình 1 để phân tích gradient descent. Tư tưởng chung của kỹ thuật sử dụng gradient descent là giả sử x_t tìm được sau vòng lặp thứ t thì ta cần tìm thuật toán để đưa x_t càng về gần $x*$ càng tốt. Sử dụng Gradient Descent (GD) có nghĩa là chúng ta phải di chuyển ngược dấu với đạo hàm

$$x_{t+1} = x_t + \Delta \tag{2}$$

Trong đó, Δ là một đại lượng ngược dấu với đạo hàm $f'(x_t)$ giá trị x_t bên phải $x*$ thì $f'(x_t)$ lớn hơn 0 và ngược lại. Đại lượng di chuyển Δ trực quan nhất, là tỉ lệ thuận với - $f'(x_t)$.

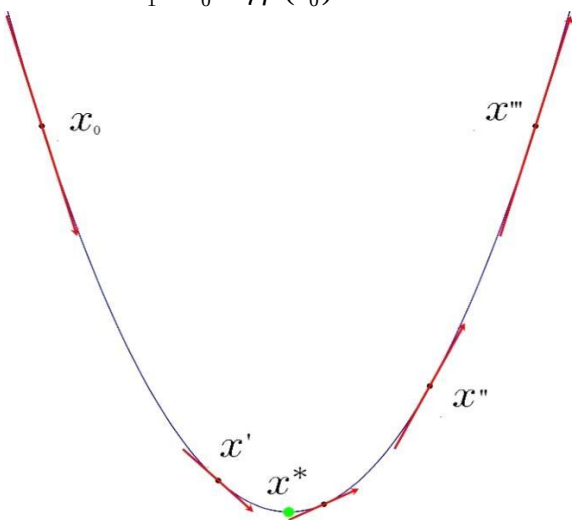
Từ đó, ta có thể cập nhật công thức:

$$x_{t+1} = x_t - \eta f'(x_t) \tag{3}$$

Trong đó, η là một số dương được gọi là learning rate (tham số học). Dấu trừ thể hiện việc phải đi ngược với đạo hàm.

Trước khi đi vào cụ thể, ta quan sát lại theo đồ thị của một hàm cơ bản có điểm cực tiểu $x*$ là điểm thuật toán cần hội tụ. Xuất phát từ một điểm x_0 với hệ số learrning rate η bất kỳ. Ta có công thức cập nhật x_1 theo thuật toán GD là:

$$x_1 = x_0 - \eta f'(x_0) \tag{4}$$



Hình 5. *Đồ thị mô tả sự tương quan giữa các vị trí của biến.*

Quan sát trên hình 5 ta thấy sẽ có 2 trường hợp tổng quát xảy ra:

Trường hợp 1: Với η nhỏ không đủ để bước nhảy đưa x_1 vượt qua vị trí của x^* thì x_1 sẽ nằm cùng bên với x', đạo hàm của 2 điểm này sẽ cùng dấu, suy ra tích của đạo hàm 2 điểm x_1, x_0 là $f'(x_1) \cdot f'(x_0) > 0$.

Trường hợp 2: Nếu x_1 nằm ở vị trí x'' hay x''' như hình 5 thì $f'(x_1) \cdot f'(x_0) < 0$. (Trường hợp $f'(x_1) \cdot f'(x_0) = 0$ chỉ xảy ra khi $x_1 = x^*$, trường hợp này rất hiếm khi xảy ra).

Đến đây, với trường hợp 2 điểm x_1, x_0 cùng bên tức là η nhỏ, để tránh thuật toán mất quá nhiều vòng lặp mới đến được đích ta sẽ thực hiện cập nhật giá trị learning rate theo công thức $\eta = 2 * \eta$. Sau đó, với η mới ta thực hiện tính toán lại vị trí x_1, nếu x_1 vẫn cùng bên với x_0 ta tiếp tục tăng gấp đôi giá trị η, cứ như vậy cho đến khi x_1 và x_0 khác bên với learning rate là η_t thì ta lấy giá trị $\eta = \eta_t / 2$ là giá trị learning rate sử dụng cho tính toán theo GD.

Với trường hợp 2 điểm khác bên giống như điều khiển luồng bị tắc nghẽn. Ta thực hiện cập nhật η theo công thức $\eta = \eta_t / 2$ cho đến khi khi x_1 và x_0 cùng bên. Thì giá trị η khi đó chính là η cần tìm.

Thuật toán tìm kiếm tham số học có thể được mô tả như sau:

Algorithm: Gradient Descent with learning *rate finding*

Input: x_{init}, η_{init}.
Output: x^*
$x_{new} = x_{init} - \eta_{init} * f'(x_{init})$
$flag = f'(x_{init}) * f'(x_{new})$
while($sign(flag) * f'(x_{init}) * f'(x_{new}) > 0$):
    $\eta_{init} = \eta_{init} * 2^{sign(flag)}$
    $x_{new} = x_{init} - \eta_{init} * f'(x_{init})$
end while

if($sign(flag) = 1$):
    $\eta_{best} \longleftarrow \eta_{init} / 2$
else
    $\eta_{best} \longleftarrow \eta_{init}$
end if

%Tiếp theo thay giá trị x_{new} cuối cùng và η_{best} vào GD thông thường%

for $i = 1$ to **max_iter**
    $x = x_{new} - \eta_{best} * f'(x_{new})$
    if (($\| f'(x_{new} \|_2 < \varepsilon$)):
        break;
    $x_{new} = x$
end for;
$x^* \longleftarrow x$

Để xác định xem điểm x_{new} có nằm cùng phía với x_{init} hay không ta sử dụng biến $flag = f'(x_{init}) * f'(x_{new})$. Nếu giá trị *flag* dương thì hai điểm này cùng phía. Nếu giá trị này âm thì hai điểm này nằm khác phía. Như vậy, hàm *sign(flag)* sẽ nhận một trong hai giá trị (+1 nếu hai điểm cùng phía và -1 nếu hai điểm khác phía) Điều kiện while thể hiện vòng lặp sẽ tiếp tục thực hiện khi tìm thấy giá trị η cuối cùng mà tại đó x_{init} và x_{new} cùng phía.

Sau khi tìm được giá trị η_{best}, ta thay giá trị tìm được này và giá trị x_{new} vào phương pháp gradient thông thường. Ở đây, ***max_iter*** là số bước lặp tối đa cho phép. Nếu chuẩn 2 của vector đạo hàm nhỏ hơn một giá trị epsilon cho trước thì thuật toán dừng và chúng ta tìm được giá trị $x*$ chính là giá trị x cuối cùng.

3. THỬ NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ ĐỀ XUẤT

Thuật toán được thử nghiệm trên máy tính xách tay với cấu hình CPU Intel Core i7, ram 8Gb, hệ điều hành windows 64 bit, sử dụng ngôn ngữ lập trình Python. Thực nghiệm thứ nhất được triển khai đối với hàm nhiều biến $f(x,y) = (x^2 + y - 7)^2 + (x - y + 1)^2$. Hàm này có 2 điểm cực trị (2;3) và (-3;-2) áp dụng phương pháp mới (**myGD)** có so sánh với phương pháp thông thường Batch Gradient Descent (BGD) khi chưa áp dụng tìm kiếm learning rate, cả hai phương pháp được phép sử dụng tối đa 1000 vòng lặp. Điểm khởi tạo và giá trị learning rate ban đầu được chọn ngẫu nhiên trong cột **Input.** Kết quả được thể hiện như ở Bảng 1 dưới đây.

Bảng 1. *Kết quả so sánh trên hàm nhiều biến.*

№	Input		Kết quả						
			BGD			myGD			
	x_{init}	η_{init}	$x*$	cost	it	$x*$	cost	it	η_{best}
1	(1;2.5)	0.01	(2;3)	0.000001	200	(2;3)	0.000001	**45**	0.04
2	(1;2.5)	0.05	(2;3)	0.000001	35	(2;3)	0.000001	35	0.05
3	(1;2.5)	0.057	(2;3)	0.003421	999	(2;3)	0.003421	999	0.057
4	(1;2.5)	0.1	None	None	none	(2;3)	0.000001	**35**	0.05
5	(-1.9;-1.9)	0.01	(-3;-2)	0.000002	420	(-3;-2)	0.000002	**211**	0.02
6	(-1.9;1.9)	0.04	(-3.08; -0.32)	7.944861	999	(-3;-2)	0.000002	**211**	0.02

Thử nghiệm được thực hiện với các giá trị tọa độ khởi tạo x_{init}, và các tham số học η_{init} khác nhau. Ta nhận thấy phương pháp mới khi áp dụng tìm kiếm tham số học phù hợp thì tốc độ hội tụ thể hiện qua số vòng lặp (it) nhanh hơn rất nhiều so với phương pháp BGD (thể hiện ở phần in đậm) và đối với những thử nghiệm này thì các tham số học đều có sự thay đổi (thể hiện ở phần gạch chân). Với phương pháp này sau khi tìm được giá trị learning rate phù hợp η_{best} ta bắt đầu tìm nghiệm với giá trị khởi tạo x_{init} mới bằng giá trị x_{new} là giá trị cuối cùng khi tìm η_{best}. Một điểm ưu việt nữa là đối với một số tham số khởi tạo như mục số 4 thì phương pháp BGD thậm chí không thể tìm ra nghiệm trong khi phương pháp mới tìm ra nghiệm tối ưu sau chỉ 35 bước lặp.

Thử nghiệm thứ 2 được thể hiện với bài toán dạng hồi quy tuyến tính. Bài toán này sử dụng mảng đầu vào x là 1000 số ngẫu nhiên trong khoảng (0,1) và y là mảng các số được cho theo công thức $y = 4 + 3x + noise$ (giá trị nhiễu là rất bé trong khoảng từ -0.2 đến 0.2). Cần tìm một đường thẳng với các giá trị trọng số w_1 và w_2 sao cho phương trình $y = w_1 + w_2 x$ mô tả tốt nhất mỗi quan hệ giữa x và y.

Bảng 2. *Kết quả so sánh trên hàm hồi quy tuyến tính.*

№	Input		Kết quả						
			BGD			myGD			
	x_{init}	η_{init}	x^*	cost	it	x^*	cost	it	η_{best}
1	(2;1)	0.1	(4;2.99)	0.018937	510	(4;2.99)	0.018937	**127**	0.4
2	(2;1)	0.4	(4;2.99)	0.018937	127	(4;2.99)	0.018937	127	0.4
3	(2;1)	1.6	Không hội tụ		1000	(4;2.99)	0.018937	**127**	0.4
4	(2;1)	2.0	none	None	none	(4;2.99)	0.018938	**101**	0.5

Thử nghiệm trên cũng được thực hiên với các giá trị tọa độ khởi tạo tạo x_{init} và các tham số học η_{init} khác nhau. Ta cũng có thể nhận thấy rằng, phương pháp mới khi có áp dụng tìm kiếm tham số học tối ưu thì số vòng lặp giảm đi rất nhanh (thể hiện ở phần in đậm) và trong một số trường hợp đối với tham số khởi tạo thì phương pháp BGD không thể tìm ra nghiệm (thử nghiệm 3, 4) trong khi phương pháp mới vẫn cho ra được nghiệm tối ưu. Ở những trường hợp giá trị vòng lặp của phương pháp đề xuất tốt hơn ở cả hai thử nghiệm ta đều thấy có sự thay đổi về tham số learning rate η_{best} so với tham số η_{init} ban đầu. Điều đó cho thấy, việc tìm kiếm tham số learning rate có ý nghĩa rất quan trọng trong việc tiệm cận nhanh hơn tới nghiệm của bài toán

4. KẾT LUẬN

Với phương pháp mới đề xuất, nhóm tác giả đã giới thiệu phương pháp tìm kiếm tham số học (learning rate) phù hợp dựa trên ý tưởng của giải thuật chống tắc nghẽn trong mạng viễn thông TCP. Với phương pháp này, mặc dù trước khi áp dụng tìm nghiệm theo Gradient Descent thì ta phải đi tìm tham số học, tuy nhiên, việc tìm tham số học xảy ra rất nhanh do tăng giảm theo lũy thừa của 2. Thông thường thì kết thúc trong vài bước lặp. Sau khi áp dụng tham số học này với phương pháp Gradient Descent thì số bước lặp để tìm thấy nghiệm được giảm rõ rệt, thậm chí giải quyết được nhiều trường hợp không tìm thấy nghiệm đối với phương pháp thông thường.

TÀI LIỆU THAM KHẢO

[1]. Bottou, L. *"Online learning and stochastic approximations. On-line learning in neural networks"*. (1998).

[2]. Boyd, S. *"Global optimization in control system analysis and design. Control and Dynamic Systems V53: High Performance Systems Techniques and Applications: Advances in Theory and Applications"*. (2012).

[3]. Chiu, Dah-Ming; Raj Jain. *"Analysis of increase and decrease algorithms for congestion avoidance in computer networks. Computer Networks and ISDN systems"*. (1989).

[4]. Darken, C., Chang, J., & Moody, J. *"Learning rate schedules for faster stochastic gradient search"*. Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop, (September), p. 1–11. (1992).

[5]. Dong, C., Loy, C.C., He, K., and Tang, X. (2016). *"Image super-resolution using deep convolutional networks"*. IEEE transactions on pattern analysis and machine intelligence, 38(2), p. 295–307. (2016).

[6]. Duchi, J., Hazan, E., & Singer, Y. (2011). *"Adaptive Subgradient Methods for Online Learning and Stochastic Optimization"*. Journal of Machine Learning Research, 12, 2121–2159. Retrieved from http://jmlr.org/papers/v12/duchi11a.html.

[7]. Granichin, O., Volkovich, V., and Toledano-Kitai, D. *"Randomized Algorithms in Automatic Control and Data Mining"*. Springer. (2015).
[8]. Hinton, G.E. and Salakhutdinov, R.R. *"Reducing the dimensionality of data with neural networks. science"*, 313(5786), p. 504–507. (2006).
[9]. Leslie N. Smith. (2017). *"Cyclical Learning Rates for Training Neural Networks"*. IEEE Winter Conference on Applications of Computer Vision (WACV).
[10]. Qian, N. (1999). *"On the momentum term in gradient descent learning algorithms"*. Neural Networks : The Official Journal of the International Neural Network Society, 12(1), 145–151.
[11]. Basel Alyafi, Fakrul Islam Tushar, Zafar Toshpulatov (2018). *"Cyclical Learning Rates for Training Neural Networks With Unbalanced Data Sets"*. Jmd in medical image analysis and applications - pattern recognition module 2018.

ABSTRACT

AN ACCELERATED METHOD OF GRADIENT DESCENT

Optimization problem is the problem of finding the best solution in best solutions. Optimization has many applications in deep learning real life such as classification problems, image recognition, problems to maximize revenue or reduce costs, production time vv. The gradient descent method usually used to find the optimal solution of a problem quickly. In this report, the author uses a new method to quickly find reasonable learning rate based on the idea of flow control and anti-congestion principle in telecommunication networks to speed up the ability convergence of the problem compared to the conventional gradient descent method.

Keywords: Gradient descent; Machine learning; Learning rate; Point initialization; Function loss.

Nhận bài ngày 12 tháng 3 năm 2020
Hoàn thiện ngày 20 tháng 4 năm 2020
Chấp nhận đăng ngày 03 tháng 8 năm 2020

Địa chỉ: [1]Khoa CNTT, Học viện KTQS;
[2]Viện CNTT, Viện KH – CN QS;
[3]Đại học Kinh Doanh và Công nghệ Hà Nội.
*Email: hoatatthang@gmail.com.