TRƯỜNG ĐẠI HỌC QUY NHƠN
KHOA TOÁN VÀ THỐNG KÊ

TIỂU LUẬN MÔN HỌC
LÝ THUYẾT TỐI ƯU

ĐỀ TÀI

THUẬT TOÁN TỐI ƯU ADAM

GVHD: **TS. TRẦN NGỌC NGUYÊN**
HỌC VIÊN: **NGUYỄN QUỐC DƯƠNG**
MÃ SỐ HỌC VIÊN: 8242548005
LỚP: KHOA HỌC DỮ LIỆU ỨNG DỤNG K24B

Bình Định - Năm 2022

Mục lục

Chương 1

Kiến thức chuẩn bị

1.1 Giới thiệu tổng quan về tối ưu trong học sâu

Các thuật toán tối ưu trong huấn luyện các mô hình đa tầng khác các thuật toán tối ưu thuần túy trên nhiều phương diện. Tối ưu trong học máy thường là tối ưu gián tiếp. Trong học máy, ta thường quan tâm đến một độ đo hiệu năng P nào đó, được đánh giá trên tập kiểm thử và có thể có chi phí tính toán lớn. Do đó, ta chỉ tối ưu P một cách gián tiếp. Chúng ta cố gắng làm giảm một hàm mất mát khác $J(\theta)$ và hi vọng điều đó sẽ cải thiện P. Điều này trái ngược với bài toán tối ưu thuần túy, ở đó cực tiểu hóa J là hàm mục tiêu duy nhất. Các thuật toán tối ưu trong huấn luyện mô hình đa tầng thường bao gồm những đặc tính chuyên biệt tương ứng với những cấu trúc cụ thể của các hàm mục tiêu [1].

Các thuật toán học sâu cần giải quyết nhiều bài toán tối ưu trong nhiều ngữ cảnh. Ta thường dùng tối ưu dạng giải tích để chứng minh lí thuyết hoặc thiết kế các thuật toán. Trong tất cả các bài toán tối ưu liên quan đến học sâu, bài toán thách thức nhất là huấn luyện mạng neuron. Chúng ta thường phải chạy các thuật toán huẩn luyện mạng neuron trên nhiều máy tính trong một vài ngày hay thậm chí là vài tháng chỉ để huấn luyện một mạng neuron. Do tối ưu trong học sâu rất quan trọng và tốn kém, người ta đã phát triển riêng một tập các kĩ thuật tối ưu chuyên biệt để giải quyết các bài toán này [1].

Nhìn chung, các phương pháp học máy đã tránh những khó khăn trong tối ưu bằng cách thiết kế hàm mục tiêu và các ràng buộc cẩn thận sao cho bài toán tối ưu có tính lồi. Khi huấn luyện mạng neuron, chúng ta phải đối mặt với các trường hợp không lồi tổng quát. Ngay cả tối ưu lồi cũng có vấn đề của riêng nó. Một số thách thức điển hình nhất liên quan đến khâu tối ưu trong các mô hình huấn luyện đa tầng có thể kể đến như *tính kém điều*

hòa (coditioning); *cực tiểu cục bộ* (local minima); *cao nguyên, điểm yên ngựa và các vùng phẳng khác* (plateaus, saddle points and other flat regions); *vách đứng và bùng nổ gradient* (cliffs and exploding gradients); *phụ thuộc dài hạn* (long-term dependencies); *gradient không chính xác* (inexact gradients); *sự liên đới yếu giữa cấu trúc cục bộ và cấu trúc toàn cục* (poor correspondence between local and global structure). Những thuật ngữ này được trình bày chi tiết trong tài liệu số [1, 2].

1.2 Một số thuật toán tối ưu

1.2.1 Gradient Descent đa biến

Với $\mathbf{x} \in \mathbb{R}^d$, hàm mục tiêu $f : \mathbb{R}^d \to \mathbb{R}$ ánh xạ các vector tới các giá trị vô hướng [3]. Gradient tương ứng cũng là đa biến, là một vector gồm d đạo hàm riêng

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_d}\right]^\top.$$

Mỗi đạo hàm riêng $\partial f(\mathbf{x})/\partial x_i$ trong gradient biểu diễn tốc độ thay đổi theo x_i của f tại $\mathbf{x}$. Sử dụng khai triển Taylor tương ứng cho các hàm đa biến, ta được

$$f(\mathbf{x} + \epsilon) = f(\mathbf{x}) + \epsilon^\top \nabla f(\mathbf{x}) + \mathcal{O}(\|\epsilon\|^2).$$

Nói cách khác, chiều giảm mạnh nhất được cho bởi gradient âm $-\nabla f(\mathbf{x})$, các hạng tử từ bậc hai trở lên trong ϵ có thể bỏ qua. Chọn một tốc độ học phù hợp $\eta > 0$, ta được thuật toán gradient descent nguyên bản như sau

$$\mathbf{x} \leftarrow \mathbf{x} - \eta \nabla f(\mathbf{x}).$$

Ưu điểm

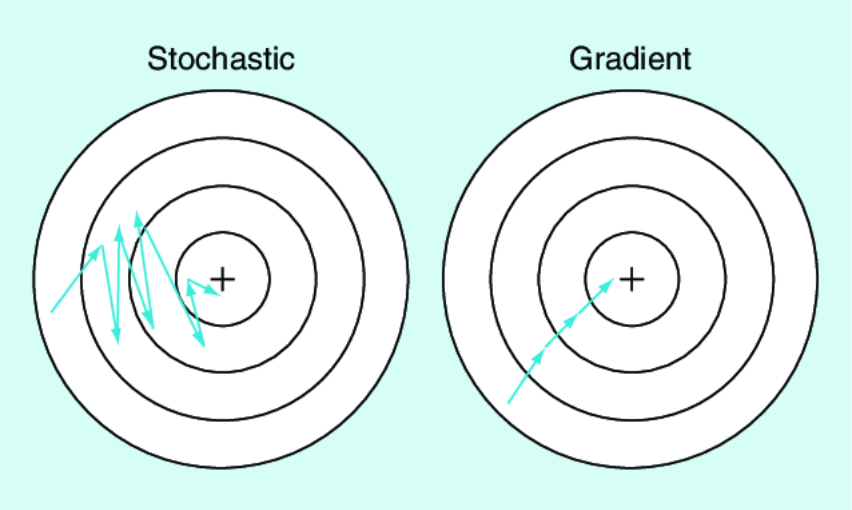
- Thuật toán gradient descent cơ bản, dễ hiểu. Thuật toán đã giải quyết được vấn đề tối ưu model neural network bằng cách cập nhật trọng số sau mỗi vòng lặp.

Nhược điểm

- Vì đơn giản nên thuật toán Gradient Descent còn nhiều hạn chế như phụ thuộc vào nghiệm khởi tạo ban đầu và learning rate.
- Ví dụ 1 hàm số có 2 global minimum thì tùy thuộc vào 2 điểm khởi tạo ban đầu sẽ cho ra 2 nghiệm cuối cùng khác nhau.

- Tốc độ học quá lớn sẽ khiến cho thuật toán không hội tụ, quanh quẩn bên đích vì bước nhảy quá lớn; hoặc tốc độ học nhỏ ảnh hưởng đến tốc độ training.

1.2.2 Stochastic Gradient Descent



Hình 1.1: So sánh Gradient Descent với Stochastic Gradient Descent

Trong học sâu, hàm mục tiêu thường là trung bình của các hàm mất mát cho từng mẫu trong tập huấn luyện. Giả sử tập huấn luyện có n mẫu, $f_i(\mathbf{x})$ là hàm mất mát của mẫu thứ i và vector tham số là $\mathbf{x}$. Ta có hàm mục tiêu

$$f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}).$$

Gradient của hàm mục tiêu tại $\mathbf{x}$ được tính như sau

$$\nabla f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{x}).$$

Nếu gradient descent được sử dụng, chi phí tính toán cho mỗi vòng lặp độc lập là $\mathcal{O}(n)$, tăng tuyến tính với n. Do đó, với tập huấn luyện lớn, chi phí của gradient descent cho mỗi vòng lặp sẽ rất cao [3].

Hạ gradient ngẫu nhiên (stochastic gradient descent - SGD) giúp giảm chi phí tính toán ở mỗi vòng lặp. Ở mỗi vòng lặp, ta lấy ngẫu nhiên một mẫu dữ liệu có chỉ số $i \in \{1, \ldots, n\}$ theo phân phối đều, và chỉ cập nhật $\mathbf{x}$ bằng gradient $\nabla f_i(\mathbf{x})$:

$$\mathbf{x} \leftarrow \mathbf{x} - \eta \nabla f_i(\mathbf{x}).$$

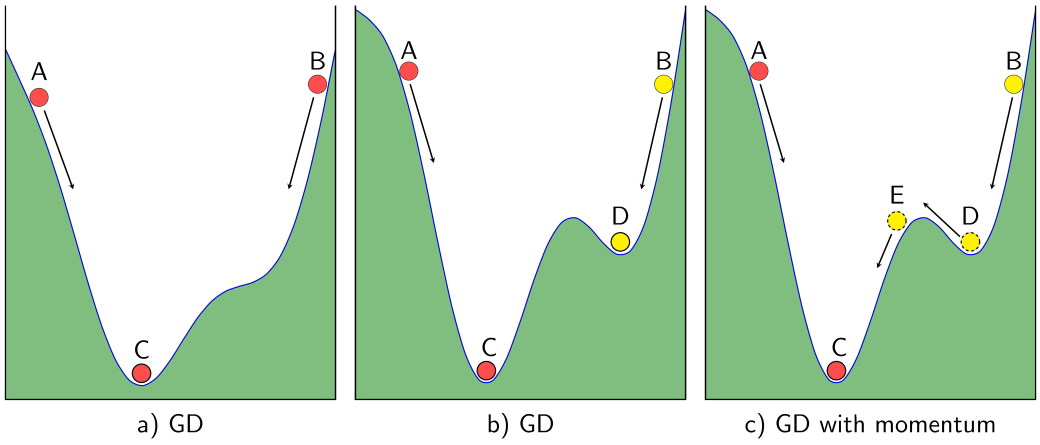
Ở đây, η là tốc độ học. Ta có thể thấy rằng chi phí tính toán cho mỗi vòng lặp giảm từ $\mathcal{O}(n)$ của hạ gradient xuống còn hằng số $\mathcal{O}(1)$. Hơn nữa, gradient ngẫu nhiên $\nabla f_i(\mathbf{x})$ là một ước lượng không thiên lệch (*unbiased*) của gradient $\nabla f(\mathbf{x})$

$$\mathbb{E}_i \nabla f_i(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\mathbf{x}) = \nabla f(\mathbf{x}).$$

Do đó, gradient ngẫu nhiên là một ước lượng gradient tốt.

Quan sát hình 1.1, ta thấy SGD có đường đi khá là zig zắc, không mượt như GD. Tuy nhiên, một điểm dữ liệu không thể đại diện cho toàn bộ dữ liệu. Chính vì lí do đó, GD có hạn chế đối với cơ sở dữ liệu lớn (vài triệu dữ liệu) thì việc tính toán đạo hàm trên toàn bộ dữ liệu qua mỗi vòng lặp trở nên cồng kềnh. Bên cạnh đó GD không phù hợp với online learning.

1.2.3 Momentum



Hình 1.2: So sánh Gradient Descent với các hiện tượng vật lý

Một vấn đề đặt ra là khi có nhiễu ta cần chọn tốc độ học như thế nào để phù hợp. Nếu ta giảm tốc độ học quá nhanh, việc hội tụ sẽ ngưng trệ. Nếu tốc độ học giảm chậm, sẽ khó hội tụ tại một kết quả đủ tốt vì nhiễu sẽ đẩy điểm hội tụ ra xa điểm tối ưu. Để khắc phục các hạn chế trên của thuật toán Gradient Descent, người ta dùng gradient descent với momentum [4].

Ví dụ minh họa dưới góc nhìn vật lý

Thuật toán GD thường được ví với tác dụng của trọng lực lên một hòn bi đặt trên một mặt có dạng như hình một thung lũng giống như hình 1.2a. Bất kể ta đặt hòn bi ở A hay B

thì cuối cùng hòn bi cũng sẽ lăn xuống và kết thúc ở vị trí C. Tuy nhiên, nếu như bề mặt có hai đáy thung lũng như Hình 1.2b thì tùy vào việc đặt bi ở A hay B, vị trí cuối cùng của bi sẽ ở C hoặc D. Điểm D là một điểm local minimum chúng ta không mong muốn.

Nếu suy nghĩ một cách vật lý hơn, vẫn trong Hình 1.2b, nếu vận tốc ban đầu của bi khi ở điểm B đủ lớn, khi bi lăn đến điểm D, theo đà, bi có thể tiếp tục di chuyển lên dốc phía bên trái của D. Và nếu giả sử vận tốc ban đầu lớn hơn nữa, bi có thể vượt dốc tới điểm E rồi lăn xuống C như trong Hình 1.2c. Đây chính là điều chúng ta mong muốn. Bạn đọc có thể đặt câu hỏi rằng liệu bi lăn từ A tới C có theo đà lăn tới E rồi tới D không. Xin trả lời rằng điều này khó xảy ra hơn vì nếu so với dốc DE thì dốc CE cao hơn nhiều. Dựa trên hiện tượng này, thuật toán momentum được ra đời nhằm khắc phục việc nghiệm của GD rơi vào một điểm local minimum không mong muốn [4, 5].

Biểu diễn momentum bằng toán học

Trong GD, chúng ta cần tính lượng thay đổi ở thời điểm t để cập nhật vị trí mới cho nghiệm (tức *hòn bi*). Nếu chúng ta coi đại lượng này như vận tốc v_t trong vật lý, vị trí mới của *hòn bi* sẽ là $\theta_{t+1} = \theta_t - v_t$. Dấu trừ thể hiện việc phải di chuyển ngược với đạo hàm. Công việc của chúng ta bây giờ là tính đại lượng v_t sao cho nó vừa mang thông tin của *độ dốc* (tức đạo hàm), vừa mang thông tin của *đà*, tức vận tốc trước đó v_{t-1} (chúng ta coi như vận tốc ban đầu $v_0 = 0$). Một cách đơn giản nhất, ta có thể cộng (có trọng số) hai đại lượng này lại, ta được

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta). \tag{1.1}$$

Trong đó γ thường được chọn là một giá trị khoảng 0.9, v_t là vận tốc tại thời điểm trước đó, $\nabla_\theta J(\theta)$ chính là độ dốc của điểm trước đó. Sau đó vị trí mới của *hòn bi* được xác định như sau

$$\theta = \theta - v_t. \tag{1.2}$$

Ưu điểm

Thuật toán tối ưu giải quyết được vấn đề: Gradient Descent không tiến được tới điểm global minimum mà chỉ dừng lại ở local minimum.

Nhược điểm

Tuy momentum giúp hòn bi vượt dốc tiến tới điểm đích, tuy nhiên khi tới gần đích, nó vẫn mất khá nhiều thời gian giao động qua lại trước khi dừng hẳn, điều này được giải thích vì viên bi có đà.

1.2.4 Adagrad

Thuật toán Adagrad điều chỉnh tốc độ học của toàn bộ tham số mô hình một cách riêng biệt bằng cách chia chúng cho nghịch đảo căn bậc hai của tổng toàn bộ các bình phương gradient trong quá khứ. Các tham số có đạo hàm riêng của hàm mất mát càng lớn thì các tốc độ học tương ứng càng giảm nhanh, trong khi những tham số có đạo hàm riêng nhỏ sẽ có tốc độ giảm chậm tương ứng. Hệ quả là thuật toán sẽ học nhanh hơn ở những hướng có độ dốc nhỏ hơn trong không gian tham số. Tuy nhiên, khi huấn luyện các mô hình mạng neuron đa tầng, việc tích lũy các bình phương gradient ngay từ khi bắt đầu quá trình huấn luyện có thể làm giảm tốc độ học quá sớm và quá mức cần thiết [1, 6].

1.2.5 RMSProp

Thuật toán RMSProp là một phiên bản sửa đổi của AdaGrad để hoạt động tốt hơn với các bài toán tối ưu không lồi bằng cách tích lũy gradient vào một trung bình động lũy thừa có trọng số. AdaGrad được thiết kế để hội tụ nhanh chóng khi được áp dụng cho một hàm lồi.

RMSprop giải quyết vấn đề tỷ lệ học giảm dần của Adagrad bằng cách chia tỷ lệ học cho trung bình của bình phương gradient.

$$E\left[g^2\right]_t = 0,9E\left[g^2\right]_{t-1} + 0,1g_t^2,$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E\left[g^2\right]_t + \epsilon}} g_t.$$

Khi được áp dụng cho hàm không lồi để huấn luyện mạng neuron thì quỹ đạo của quá trình học có thể bỏ qua nhiều cấu trúc và cuối cùng tiến đến một vùng lồi cục bộ. Adagrad giảm dần tốc độ học theo toàn bộ giá trị bình phương gradient trong quá khứ và có thể khiến tốc độ học trở nên quá nhỏ trước khi đến được một cấu trúc lồi. RMSProp sử dụng một trung bình động lũy thừa giảm để loại bỏ các dữ kiện từ quá khứ quá xa, giúp nó có thể hội tụ nhanh chóng sau khi tìm thấy vùng lồi, như thể đó là một trường hợp khi thuật toán AdaGrad được khởi tạo bên trong vùng lồi đó [1].

Qua thực nghiệm, RMSProp tỏ ra là một thuật toán tối ưu hiệu quả và thực dụng đối với các mạng neuron đa tầng. Nó đang là một trong những phương pháp tối ưu mà các chuyên gia học sâu áp dụng thường xuyên.

Ưu điểm

Ưu điểm rõ nhất của RMSprop là giải quyết được vấn đề tốc độ học giảm dần của Adagrad (vấn đề tốc độ học giảm dần theo thời gian sẽ khiến việc training chậm dần, có thể dẫn tới bị đóng băng).

Nhược điểm

Thuật toán RMSprop có thể cho kết quả nghiệm chỉ là local minimum chứ không đạt được global minimum như Momentum.

Chương 2

Thuật toán tối ưu Adam

Algorithm 1 Thuật toán Adam. Kí hiệu g_t^2 là bình phương của phép nhân element-wise $g_t \odot g_t$, các tham số khuyến nghị là $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$ và $\varepsilon = 10^{-8}$. Tất cả các toán tử trên vector là phép element-wise (tức là thực hiện tương ứng từng phần tử).

Require: α: Độ lớn mỗi bước.

Require: $\beta_1, \beta_2 \in [0, 1)$: Tốc độ suy giảm theo hàm mũ của các ước lượng moment.

Require: $f(\theta)$: Stochastic objective function with parameters θ.

Require: θ_0: Tham số khởi tạo.

$m_0 \leftarrow 0$ (Khởi tạo moment bậc 1).

$v_0 \leftarrow 0$ (Khởi tạo moment bậc 2).

$t \leftarrow 0$ (Khởi tạo thứ tự bước t.)

while θ_0 not converged **do**

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Tính gradient tại mỗi bước t).

$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Cập nhật giá trị moment bậc 1).

$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Cập nhật giá trị moment bậc 2).

$\widehat{m}_t \leftarrow m_t / \left(1 - \beta_1^t\right)$ (Hiệu chỉnh độ chệch trong moment bậc 1).

$\widehat{v}_t \leftarrow v_t / \left(1 - \beta_2^t\right)$ (Hiệu chỉnh độ chệch trong moment bậc 2).

$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / \left(\sqrt{\widehat{v}_t} + \epsilon\right)$ (Cập nhật các tham số.)

end while

return θ_t (Resulting parameters).

2.1 Phân tích và đánh giá thuật toán

Adam (Adaptive moment Estimation) là một thuật toán tối ưu tự điều chỉnh tốc độ học [7]. Cái tên "Adam" bắt nguồn từ cụm từ "adaptive moment" (moment thích nghi). Adam được coi như là sự kết hợp giữa RMSProp và momentum. Trong khi momentum có thể xem như một quả bóng đang chạy xuống dốc, Adam lại giống như một quả bóng nặng với ma sát. Adam sử dụng bình phương gradient để chia tỷ lệ learning rate như RMSProp và tận dụng "đà" giống như momentum. Adam tính toán moment cấp 1 và moment cấp 2 của gradient để ước lượng learning rate cho các tham số. Công thức cập nhật đầy đủ của Adam

$$\begin{cases} m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t, \\ v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2, \\ \hat{m}_t = \dfrac{m_t}{1-\beta_1^t}, \\ \hat{v}_t = \dfrac{v_t}{1-\beta_2^t}, \\ \theta_t = \theta_{t-1} - \eta \dfrac{\hat{m}t}{\sqrt{\hat{v}_t} + \varepsilon}. \end{cases} \tag{2.1}$$

Để tốt hơn về chi phí tính toán, chúng ta có thể viết lại 3 phương trình cuối trong hệ phương trình 2.1 như sau

$$\eta_t = \eta \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t}, \tag{2.2}$$

$$\theta_t = \theta_{t-1} - \eta_t \frac{m_t}{\sqrt{v_t} + \varepsilon}. \tag{2.3}$$

Định nghĩa 2.1.1 (Khái niệm moment)**.** Moment cấp k đối với a của biến ngẫu nhiên X là một số xác định như sau

$$v_k(a) = E\left[(X-a)^k\right].$$

Nếu $a = 0$, ta ký hiệu $v_k = v_k(0) = E\left(X^k\right)$ và gọi nó là moment gốc cấp k. Rõ ràng kỳ vọng chính là moment gốc cấp 1 ($EX = v_1$). Nếu $a = EX$, ta ký hiệu $\mu_k = v_k(EX) = E\left[(X-EX)^k\right]$ và gọi nó là moment trung tâm cấp k. Hơn nữa, phương sai là moment trung tâm cấp 2 ($VX = \mu_2$). Người ta còn dùng moment để đặc trưng cho hình dạng của mật độ phân phối.

Như đã thấy ở thuật toán, Adam sử dụng thuật ngữ khởi tạo *bias correction*. Chúng ta sẽ làm rõ điều này cho ước lượng moment cấp 2, nguồn gốc ước lượng moment cấp 1 là tương

tự. Ta thấy rằng, gradient của hàm mục tiêu có thể xem như là một biến ngẫu nhiên, vì nó thường được đánh giá trên một mini-batch dữ liệu. Moment cấp 1 là kỳ vọng, moment cấp 2 là *uncentered variance* (Có nghĩa là moment gốc cấp 2 - chúng ta không trừ đi giá trị trung bình khi tính toán phương sai). Adam tận dụng *exponentially moving average* của gradient và bình phương gradient để ước lượng các moment như sau

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\, g_t,$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\, g_t^2.$$

Trong đó *decay rate* là β_1 và β_2; m_t và v_t được gọi là moving averages, g là gradient trên mini-batch hiện tại. Để xem các giá trị này tương quan với moment như thế nào, chúng ta hãy xem xét giá trị kỳ vọng của các moving averages. Vì m và v là các ước lượng của moment cấp 1 và moment cấp 2, chúng ta muốn có tính chất sau

$$\mathbb{E}\left[m_t\right] = \mathbb{E}\left[g_t\right],$$
$$\mathbb{E}\left[v_t\right] = \mathbb{E}\left[g_t^2\right].$$

Ta khai triển v_t như sau (tương tự với m_t):

$$\begin{aligned}
&v_0 = 0,\\
&v_1 = \beta_2 v_0 + (1 - \beta_2)\, g_1^2 = (1 - \beta_2)\, g_1^2,\\
&v_2 = \beta_2 v_1 + (1 - \beta_2)\, g_2^2 = \beta_2 (1 - \beta_2)\, g_1^2 + (1 - \beta_2)\, g_2^2,\\
&\quad\vdots\\
&v_t = \beta_2 v_{t-1} + (1 - \beta_2)\, g_t^2 = (1 - \beta_2) \sum_{i=1}^{t} \beta_2^{t-i} \cdot g_i^2.
\end{aligned}$$

Do $\beta_2 \in [0, 1)$ nên ta có thể thấy rằng exponential moving average đánh trọng số cho các gradient, các gradient đầu tiên thì đóng góp ít vào giá trị tổng thể, vì chúng được nhân với β nhỏ hơn. Bây giờ chúng ta sẽ xem xét giá trị kỳ vọng của v_t để xem nó liên quan thế nào đến giá trị thực moment cấp 2 là $\mathbb{E}\left[g_t^2\right]$. Ta có

$$\begin{aligned}
\mathbb{E}\left[v_t\right] &= \mathbb{E}\left[(1 - \beta_2) \sum_{i=1}^{t} \beta_2^{t-i} \cdot g_i^2\right]\\
&= \mathbb{E}\left[g_t^2\right] \cdot (1 - \beta_2) \sum_{i=1}^{t} \beta_2^{t-i} + \zeta\\
&= \mathbb{E}\left[g_t^2\right] \cdot \left(1 - \beta_2^t\right) + \zeta.
\end{aligned}$$

Ở khai triển trên, chúng ta đã sử dụng xấp xỉ g_i thành g_t và đưa nó ra khỏi tổng. Vì sự xấp xỉ này nên ta cần cộng thêm một đại lượng ζ trong công thức để đại diện cho sai lệch do xấp xỉ. Tiếp theo, chúng ta cần điều chỉnh giá trị ước lượng của v_t để cho kỳ vọng của nó xấp xỉ moment cấp 2. Sự sai lệch đại lượng $\left(1-\beta_2^t\right)$ là do ta đã khởi tạo $v_0 = 0$. Từ đó, chúng ta có công thức hiệu chỉnh cho v như sau

$$\hat{v_t} = \frac{v_t}{1-\beta_2^t}.$$

Tương tự, thực hiện các bước trên với m_t. Cuối cùng, chúng ta có công thức cập nhật tham số cho Adam như sau

$$\theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}.$$

2.2 Đánh giá sự hội tụ

2.2.1 Sai lầm trong chứng minh hội tụ của Kingma & Ba

Trong bài báo gốc về thuật toán Adam [7], Kingma và Lei Ba đã có những sai lầm trong phân tích sự hội tụ của thuật toán Adam. Sau đây, chúng ta sẽ cùng phân tích lỗi sai của họ.

Gọi $f_1(\theta), f_2(\theta), \ldots, f_T(\theta)$ là một dãy các hàm chi phí và lồi tại vòng lặp tương ứng $1, 2, \ldots, T$. Đặt $g_t := \nabla f_t(\theta_t)$ và gọi $g_{t,i}$ là phần tử thứ i của vector gradient g_t. Đặt $g_{1:t,i} = [g_{1,i}, g_{2,i}, \ldots, g_{t,i}] \in \mathbb{R}^t$ và $\gamma := \dfrac{\beta_1^2}{\sqrt{\beta_2}}$. Gọi θ^* là tham số tối ưu của bài toán, θ_t là tham số dự đoán được tại vòng lặp thứ t. Do đó, tổng sai lệch của mô hình qua T vòng lặp là

$$R(T) = \sum_{t=1}^{T} \left[f_t(\theta_t) - f_t(\theta^*) \right].$$

Chúng ta cần chứng minh cận trên của $R(T)$ sẽ nhỏ hơn hoặc bằng $O(T)$ để từ đó có thể đánh giá được $\dfrac{R(T)}{T} \to C$ khi $T \to \infty$. Trong bài báo của Kingma và Lei Ba, các tác giả đã chứng minh và chỉ ra định lý sau.

Định lý 2.2.1. Giả sử rằng các hàm f_t có gradient bị chặn, $\|\nabla f_t(\theta)\|_2 \le G, \|\nabla f_t(\theta)\|_\infty \le G_\infty$ với mọi tham số $\theta \in \mathbb{R}^d$, và khoảng cách giữa 2 tham số θ_t bất kỳ được sinh ra bởi Adam là bị chặn, $\|\theta_n - \theta_m\|_2 \le D, \|\theta_m - \theta_n\|_\infty \le D_\infty$ với mọi m, n bất kỳ thuộc $\{1, \ldots, T\}$, và $\beta_1, \beta_2 \in [0, 1)$ thoả mãn $\dfrac{\beta_1^2}{\sqrt{\beta_2}} < 1$. Đặt $\alpha_t = \dfrac{\alpha}{\sqrt{t}}$ và $\beta_{1,t} = \beta_1 \lambda^{t-1}, \lambda \in (0, 1)$. Với mọi $T \ge 1$, ta có

$$R(T) \le \frac{D^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} \sqrt{T\hat{v}_{T,i}} + \frac{\alpha(1+\beta_1) G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} \|g_{1:T,i}\|_2 + \sum_{i=1}^{d} \frac{D_\infty^2 G_\infty \sqrt{1-\beta_2}}{2\alpha(1-\beta_1)(1-\lambda)^2}.$$

Từ định lý này, các tác giả kết luận được rằng $\frac{R(T)}{T} = O\left(\frac{1}{\sqrt{T}}\right)$. Từ đó, khi $T \to \infty$ thì $\frac{R(T)}{T} \to 0$ và hội tụ. Tuy nhiên, định lý trên là không chính xác vì các tác giả đã chứng minh sai các bổ đề trong bài báo. Các tác giả đã sử dụng các bổ đề được nêu ra sau đây.

Bổ đề 2.2.1. Nếu một hàm $f : \mathbb{R}^d \to \mathbb{R}$ là lồi thì:

$$f(y) \geq f(x) + \nabla f(x)^T (y - x),$$

với $x, y \in \mathbb{R}^d$ và $\lambda \in [0, 1]$.

Bổ đề 2.2.1 là một tính chất của hàm lồi, phổ biến và đã được chứng minh nhiều trong các tài liệu khác nên chúng ta thừa nhận kết quả này. Sau đây là chứng minh chi tiết của Kingma và Lei Ba và lỗi sai trong chứng minh các bổ đề của họ.

Bổ đề 2.2.2. Giả sử $\|g_t\|_2 \leq G, \|g_t\|_\infty \leq G_\infty$, khi đó

$$\sum_{t=1}^{T} \sqrt{\frac{g_{t,i}^2}{t}} \leq 2G_\infty \left\|g_{1:T,i}\right\|_2 .$$

Sau đây, chúng ta sẽ chỉ ra bổ đề 2.2.2 chưa chặt chẽ.

Chứng minh. Chúng ta sẽ chứng minh bằng phương pháp quy nạp. Với $T = 1$, ta có $\sqrt{g_{1,i}^2} = |g_{1,i}| \leq G_\infty$. Tuy nhiên,i Kingma & Ba đã đánh giá như sau

$$\sqrt{g_{1,t}^2} \leq 2G_\infty \left\|g_{1,i}\right\|_2 .$$

Rõ ràng, điều này chỉ đúng khi $G_\infty \geq 0.5$ và điều kiện này không được các tác giả đưa ra. Hơn nữa, bước chứng minh quy nạp cũng có một sai lầm khác trong bổ đề này. Giả sử bổ đề đúng với $T = K - 1$, ta có

$$\sum_{t=1}^{K-1} \sqrt{\frac{g_{t,i}^2}{t}} \leq 2G_\infty \left\|g_{1:K-1,i}\right\|_2 .$$

Cần chứng minh bổ đề đúng với $T = K$:

$$\begin{aligned}\sum_{t=1}^{K}\sqrt{\frac{g_{t,i}^2}{t}} &= \sum_{t=1}^{K-1}\sqrt{\frac{g_{t,i}^2}{t}} + \sqrt{\frac{g_{K,i}^2}{K}} \\ &\leq 2G_\infty \left\|g_{1:K-1,i}\right\|_2 + \sqrt{\frac{g_{K,i}^2}{K}} \\ &= 2G_\infty \sqrt{\sum_{t=1}^{K-1} g_{t,i}^2 + g_{K,i}^2 - g_{K,i}^2} + + \sqrt{\frac{g_{K,i}^2}{K}} \\ &= 2G_\infty \sqrt{\left\|g_{1:K,i}\right\|_2^2 - g_{K,i}^2} + \sqrt{\frac{g_{K,i}^2}{K}}\end{aligned}$$

Sử dụng bất đẳng thức

$$\left\|g_{1:K,i}\right\|_2^2 - g_{K,i}^2 \leq \left\|g_{1:K,i}\right\|_2^2 - g_{K,i}^2 + \frac{g_{K,i}^4}{4\left\|g_{1:K,i}\right\|_2^2} = \left(\left\|g_{1:K,i}\right\|_2 - \frac{g_{K,i}^2}{2\left\|g_{1:K,i}\right\|_2}\right)^2.$$

Từ đó

$$\begin{aligned}\sum_{t=1}^{K}\sqrt{\frac{g_{t,i}^2}{t}} &\leq 2G_\infty \left(\left\|g_{1:K,i}\right\|_2 - \frac{g_{K,i}^2}{2\left\|g_{1:K,i}\right\|_2}\right) + \sqrt{\frac{g_{K,i}^2}{K}} \\ &\leq 2G_\infty \left(\left\|g_{1:K,i}\right\|_2 - \frac{g_{K,i}^2}{2\sqrt{KG_\infty^2}}\right) + \sqrt{\frac{g_{K,i}^2}{K}} \\ &= 2G_\infty \left\|g_{1:K,i}\right\|_2 - \frac{g_{K,i}^2}{\sqrt{K}} + \sqrt{\frac{g_{K,i}^2}{K}}\end{aligned}$$

Trong chứng minh của Kingma & Ba, họ đánh giá như sau

$$2G_\infty \left\|g_{1:K,i}\right\|_2 - \frac{g_{K,i}^2}{\sqrt{K}} + \sqrt{\frac{g_{K,i}^2}{K}} \leq 2G_\infty \left\|g_{1:T,i}\right\|_2.$$

Nếu đẳng thức trên đúng, thì chứng minh theo quy nạp kết thúc. Tuy nhiên, không đảm bảo rằng đẳng thức trên sẽ đúng, đẳng thức trên tương đương với $g_{K,i}^2 \leq |g_{K,i}|$ chỉ đúng với $|g_{K,i}| \leq 1$. Do đó, bổ đề 2.2.1 được dùng để chứng minh cho bổ đề 2.2.2 là chưa chặt chẽ. □

Bổ đề 2.2.3. Giả sử $\gamma < 1$ và $\|g_t\|_2 \leq G, \|g_t\|_\infty \leq G_\infty$, thì

$$\sum_{t=1}^{T}\frac{\hat{m}_{t,i}^2}{\sqrt{t\hat{v}_{t,i}}} \leq \frac{2}{1-\gamma}\frac{1}{\sqrt{1-\beta_2}}\left\|g_{1:T,i}\right\|_2.$$

Chứng minh. Từ giả thiết, ta có $\frac{\sqrt{1-\beta_2^t}}{(1-\beta_1^t)^2} \leq \frac{1}{(1-\beta_1)^2}$. Khai triển vế trái, ta có

$$\begin{aligned}
\sum_{t=1}^{T} \frac{\hat{m}_{t,i}^2}{\sqrt{t\hat{v}_{t,i}}} &= \sum_{t=1}^{T-1} \frac{\hat{m}_{t,i}}{\sqrt{t\hat{v}_{t,i}}} + \frac{\sqrt{1-\beta_2^T}}{\left(1-\beta_1^T\right)^2} \frac{\left(\sum_{k=1}^{T}\left(1-\beta_1\right)\beta_1^{T-k} g_{k,i}\right)^2}{\sqrt{T\sum_{j=1}^{T}\left(1-\beta_2\right)\beta_2^{T-j} g_{j,i}^2}} \\
&\leq \sum_{t=1}^{T-1} \frac{\hat{m}_{t,i}}{\sqrt{t\hat{v}_{t,i}}} + \frac{\sqrt{1-\beta_2^T}}{\left(1-\beta_1^T\right)^2} \sum_{k=1}^{T} \frac{T\left(\left(1-\beta_1\right)\beta_1^{T-k} g_{k,i}\right)^2}{\sqrt{T\sum_{j=1}^{T}\left(1-\beta_2\right)\beta_2^{T-j} g_{j,i}^2}} \\
&\leq \sum_{t=1}^{T-1} \frac{\hat{m}_{t,i}}{\sqrt{t\hat{v}_{t,i}}} + \frac{\sqrt{1-\beta_2^T}}{\left(1-\beta_1^T\right)^2} \sum_{k=1}^{T} \frac{T\left(\left(1-\beta_1\right)\beta_1^{T-k} g_{k,i}\right)^2}{\sqrt{T\left(1-\beta_2\right)\beta_2^{T-k} g_{k,i}^2}} \\
&\leq \sum_{t=1}^{T-1} \frac{\hat{m}_{t,i}}{\sqrt{t\hat{v}_{t,i}}} + \frac{\sqrt{1-\beta_2^T}}{\left(1-\beta_1^T\right)^2} \frac{\left(1-\beta_1\right)^2}{\sqrt{T\left(1-\beta_2\right)^2}} \sum_{k=1}^{T} T\left(\frac{\beta_1^2}{\sqrt{\beta_2}}\right)^{T-k} \left\|g_{k,i}\right\|_2 \\
&\leq \sum_{t=1}^{T-1} \frac{\hat{m}_{t,i}}{\sqrt{t\hat{v}_{t,i}}} + \frac{T}{\sqrt{T\left(1-\beta_2\right)}} \sum_{k=1}^{T} \gamma^{T-k} \left\|g_{k,i}\right\|_2 .
\end{aligned}$$

Tới đây, Kingma & Ba đã chứng minh như sau

$$\begin{aligned}
\sum_{t=1}^{T} \frac{\hat{m}_{t,i}^2}{\sqrt{t\hat{v}_{t,i}}} &\leq \sum_{t=1}^{T} \frac{\left\|g_{t,i}\right\|_2}{\sqrt{t\left(1-\beta_2\right)}} \sum_{j=0}^{T-t} t\gamma^j \\
&\leq \sum_{t=1}^{T} \frac{\left\|g_{t,i}\right\|_2}{\sqrt{t\left(1-\beta_2\right)}} \sum_{j=0}^{T} t\gamma^j .
\end{aligned}$$

Điều này là không dễ dàng suy ra được. Rõ ràng, từ chứng mình trên, ta chỉ có thể đánh giá như sau

$$\sum_{t=1}^{T} \frac{\hat{m}_{t,i}^2}{\sqrt{t\hat{v}_{t,i}}} \leq \sum_{t=1}^{T} \frac{t}{\sqrt{t\left(1-\beta_2\right)}} \sum_{k=1}^{t} \gamma^{t-k} \left\|g_{k,i}\right\|_2 .$$

Có lẽ các bước đánh giá sai bổ đề của Kingma & Ba đã gây nhầm lẫn trong việc chứng minh hội tụ Adam. □

2.2.2 Chứng minh thuật toán Adam có thể không hội tụ

Trong báo cáo của Sashank J. Reddi, Satyen Kate và Sanjiv Kumar tại ICLR 2018 [8], các tác giả đã chứng minh thuật toán Adam không hội tụ và đề xuất ra các mở rộng, cải tiến. Trong phần này, chúng ta quy ước các kí hiệu:

- $\mathbb{S}_d^+$: tập tất cả các ma trận xác dịnh dương $d \times d$.
- $\mathbb{F}$ là tập không gian Euclid d chiều.

- Toán tử chiếu $\prod_{F,\mathbb{A}}(y)$ được định nghĩa như $\text{argmin}_{x\in\text{F}} \left\|\mathbb{A}^{1/2}(x-y)\right\|$ với mọi $y \in \mathbb{R}^d$.
- $\mathbb{F}$ có đường kính bị chặn nếu $\|x-y\|_\infty \leq D_\infty$ với mọi $x, y \in \mathbb{F}$.

Ý tưởng thuật toán gradient descent là giải quyết bài toán tối ưu

$$x^* = \text{argmin}_{x\in\text{F}}\, \mathbb{E}[f(x,z)].$$

Trong đó, $f(x,z)$ là hàm mất mát của tham số x trên mẫu z. Trong thực tế, phân phối của z là không biết nhưng chúng ta có một tập N mẫu training $\{z\}_{n=1}^N$. Từ đó, chúng ta đưa về bài toán tối ưu Empirical Risk Minimization (ERM)

$$x^* = \text{argmin}_{x\in\text{F}} \frac{1}{N}\sum_{n=1}^{N} f\left(x, z_n\right).$$

Để phân tích sự hội tụ của các phương pháp tối ưu, tại mỗi bước t, thuật toán lấy ra một điểm $x_t \in \mathbb{F}$, trong đó $\mathbb{F} \in \mathbb{R}^d$ là tập các điểm khả thi. Một hàm mất mát f_t là đã biết, và $f_t\left(x_t\right)$ là hàm mất mát của thuật toán. Tổng sai số của thuật toán tại bước kết thúc T được cho bởi

$$R_T = \sum_{i=1}^{T} f_t\left(x_t\right) - \min_{x\in\text{F}} \sum_{i=1}^{T} f_t(x).$$

Trong phần này, chúng ta giả sử rằng $\mathbb{F}$ có đường kính bị chặn và $\left\|\nabla f_t(x)\right\|_\infty$ là bị chặn với mọi $t \in [T]$ và $x \in \mathbb{F}$. Mục tiêu của chúng ta là đưa ra một thuật toán đảm bảo $R_T = O(T)$, khi đó chúng ta có thế kết luận là thuật toán hội tụ. Thuật toán đơn giản nhất cho các thiết lập này là thuật toán gradient descent online (Zinkevick 2003) có quy tắc cập nhật là

$$x_{t+1} = \prod_{\text{F}} \left(x_t - \alpha_t g_t\right).$$

Trong đó, $\prod_{\text{F}}$ biểu diễn phép chiếu $y \in \mathbb{R}^d$ trên tập $\mathbb{F}$, $\prod_{x\in\mathbb{F}}(y) = \min_{x\in\text{F}} \|x-y\|$ và $\alpha_t = \alpha/\sqrt{t}$. Bài toán tối ưu này có quan hệ gần gũi với gradient descent nói trên.

Sự không hội tụ của thuật toán Adam

Các tác giả thấy rằng vấn đề chính dẫn đến sự không hội tụ của Adam nằm ở đại lượng sau

$$\Gamma_{t+1} = \left(\frac{\sqrt{V_{t+1}}}{\alpha_{t+1}} - \frac{\sqrt{V_t}}{\alpha_t}\right).$$

Đại lượng này đo lường sự thay đổi của nghịch đảo learning rate cho các phương pháp thích nghi theo thời gian. Dễ thấy rằng, với SGD và AdaGrad thì $\Gamma_t \geq 0$ với mọi $t \in [T]$. Tuy

nhiên, điều này là chưa hẳn với các phương pháp sử dụng trung bình trượt theo cấp số nhân như Adam và RMSProp. Γ_t có thể bất ổn định với $t \in [T]$. Vi phạm này có thể dẫn đến sự không hội tụ của Adam và RMSProp.

Định lý 2.2.2. Tồn tai một bài toán tối ưu lồi online (online convex optimization) mà thuật toán Adam không hội tụ, tức là $R(T)/T \nrightarrow 0$ khi $T \to \infty$.

Chứng minh. Chúng ta xét f_t là các hàm tuyến tính và $\mathbb{F} = [-1, 1]$. Xét dãy hàm

$$f_t(x) = \begin{cases} Cx, & \text{nếu t mod } 3 = 1, \\ -x, & \text{các trường hợp khác .} \end{cases}$$

Với $C \geq 2$. Xét thuật toán Adam với các tham số sau

$$\beta_1 = 0, \beta_2 = \frac{1}{1+C^2}, \alpha_t = \frac{\alpha}{\sqrt{t}}, \alpha < \sqrt{1-\beta_2}.$$

Dễ thấy bài toán trên cùng với các tham số đã chọn thoả mãn điều kiện hội tụ trong bài báo gốc Adam. Dễ dàng thấy rằng $x = -1$ là điểm cực tiểu $R(T)$. Chúng ta khởi tạo $x_1 = 1$ và chứng minh dãy $\{x_t\}_{t=1}^{\infty}$ sinh ra từ quy tắc cập nhật Adam luôn có $x_t > 0$ với mọi $t \in \mathbb{N}, x_{3t+1} = 1$ với mọi $t \in \mathbb{N} \cup \{0\}$. Chúng ta sẽ sử dụng phương pháp quy nạp.

Với $x_1 = 1$ thoả mãn cả 2 điều kiện. Giả sử với $t \in \mathbb{N} \cup \{0\}$, chúng ta có $x_i > 0$ với mọi $i \in [3t+1]$ và $x_{3t+1} = 1$. Cần chứng minh $x_{3t+2} > 0, x_{3t+3} > 0$ và $x_{3t+4} = 1$. Thật vậy, ta có

$$\nabla f_i(x) = \begin{cases} C, & \text{với i } \mod 3 = 1, \\ -1, & \text{các trường hợp khác .} \end{cases}$$

Theo quy tắc cập nhật Adam đã xét, ta có

$$\begin{aligned} \hat{x}_{3t+2} &= x_{3t+1} - \frac{\alpha C}{\sqrt{(3t+1)\left(\beta_2 v_{3t} + (1-\beta_2)C^2\right)}} = 1 - \frac{\alpha C}{\sqrt{(3t+1)\left(\beta_2 v_{3t} + (1-\beta_2)C^2\right)}} \\ &\geq 1 - \frac{\alpha C}{\sqrt{(3t+1)(1-\beta_2)C^2}} = 1 - \frac{\alpha}{\sqrt{(3t+1)(1-\beta_2)}} > 0 \end{aligned}$$

Bất đẳng thức cuối xảy ra do $\alpha < \sqrt{1-\beta_2}$. Từ đó, ta có $1 > x_{3t+2} = \hat{x}_{3t+2} > 0$. Tương tự, ta có

$$\begin{aligned} \hat{x}_{3t+3} &= x_{3t+2} + \frac{\alpha}{\sqrt{(3t+2)\left(\beta_2 v_{3t+1} + (1-\beta_2)\right)}}, \\ \hat{x}_{3t+4} &= x_{3t+3} + \frac{\alpha}{\sqrt{(3t+3)\left(\beta_2 v_{3t+2} + (1-\beta_2)\right)}}. \end{aligned}$$

Do $x_{3t+2} > 0$ nên dễ ràng có được $x_{3t+3} > 0$. Để hoàn thành chứng minh, chúng ta cần chứng minh $x_{3t+4} = 1$. Ở đây, ta sẽ cố gắng chứng minh $\hat{x}_{3t+4} \geq 1$. Từ đó dễ dàng có được $x_{3t+4} = 1$

vì $x_{3t+4} = \prod_F (\hat{x}_{3t+4})$ và $\mathbb{F} = [-1, 1]$ với $\prod_F$ đơn giản là toán tử chiếu Euclid (trong không gian 1 chiều thì $\prod_{\mathrm{F},\sqrt{V_t}} = \prod_{\mathrm{F}}$). Ta có

$$\hat{x}_{3t+4} = \min(\hat{x}_{3t+3}, 1) + \frac{\alpha}{\sqrt{(3t+3)(\beta_2 v_{3t+2} + (1-\beta_2))}}.$$

Đẳng thức trên xảy ra do $\hat{x}_{3t+3}$ và toán tử chiếu trên miền $\mathbb{F} = [-1, 1]$. Xét 2 trường hợp:

1. Giả sử $\hat{x}_{3t+3} \geq 1$, thì dễ dàng có được $\hat{x}_{3t+4} > 1$.

2. Giả sử $\hat{x}_{3t+3} < 1$, ta có:

$$\begin{aligned}
\hat{x}_{3t+4} &= \hat{x}_{3t+3} + \frac{\alpha}{\sqrt{(3t+3)(\beta_2 v_{3t+2} + (1-\beta_2))}} \\
&= x_{3t+2} + \frac{\alpha}{\sqrt{(3t+2)(\beta_2 v_{3t+1} + (1-\beta_2)}} + \frac{\alpha}{\sqrt{(3t+3)(\beta_2 v_{3t+2} + (1-\beta_2))}} \\
&= 1 - \frac{\alpha}{\sqrt{(3t+1)(\beta_2 v_{3t} + (1-\beta_2) C^2}} + \frac{\alpha}{\sqrt{(3t+2)(\beta_2 v_{3t+1} + (1-\beta_2))}} \qquad (2.4) \\
&+ \frac{\alpha}{\sqrt{(3t+3)(\beta_2 v_{3t+2} + (1-\beta_2))}}
\end{aligned}$$

Đẳng thức (2.4) xảy ra vì $x_{3t+2} = \hat{x}_{3t+2}$. Từ đó, để chứng minh $\hat{x}_{3t+4} > 1$, chúng ta cần chứng minh

$$\begin{aligned}
T_1 := \frac{\alpha C}{\sqrt{(3t+1)(\beta_2 v_{3t} + (1-\beta_2) C^2}} \leq \frac{\alpha}{\sqrt{(3t+2)(\beta_2 v_{3t+1} + (1-\beta_2))}} \\
+ \frac{\alpha}{\sqrt{(3t+3)(\beta_2 v_{3t+2} + (1-\beta_2))}} := T_2.
\end{aligned}$$

Chúng ta đã có

$$T_1 \leq \frac{\alpha}{\sqrt{(3t+1)(1-\beta_2)}}.$$

Xét T_2, ta có:

$$\begin{aligned}
T_2 &= \frac{\alpha}{\sqrt{(3t+2)(\beta_2 v_{3t+1} + (1-\beta_2))}} + \frac{\alpha}{\sqrt{(3t+3)(\beta_2 v_{3t+2} + (1-\beta_2))}} \\
&\geq \frac{\alpha}{\sqrt{\beta_2 C^2 + (1-\beta_2))}} \left(\frac{1}{\sqrt{3t+2}} + \frac{1}{\sqrt{3t+3}} \right) \\
&\geq \frac{\alpha}{\sqrt{\beta_2 C^2 + (1-\beta_2))}} \left(\frac{1}{\sqrt{2(3t+1)}} + \frac{1}{\sqrt{2(3t+1)}} \right) \\
&= \frac{\sqrt{2}\alpha}{\sqrt{(3t+1)(\beta_2 C^2 + (1-\beta_2))}} = \frac{\alpha}{\sqrt{(3t+1)(1-\beta_2)}} \geq T_1.
\end{aligned}$$

Đẳng thức đầu tiên xảy ra do $v_t \leq C^2$ với mọi $t \in \mathbb{N}$. Đẳng thức cuối xảy ra do

$$\sqrt{\frac{\beta_2 C^2 + (1-\beta_2)}{2}} = \sqrt{1-\beta_2},$$

vì chúng ta đã chọn $\beta_2 = 1/\left(1 + C^2\right)$. Từ đó, chúng ta có $T_2 \leq T_1$ và $\hat{x}_{3t+4} \leq 1$. Vậy trong cả 2 trường hợp thì $x_{3t+4} = 1$. Nên giả thuyết quy nạp đúng. Do vậy, ta có $f_{3t+1}\left(x_{3t+1}\right) + f_{3t+2}\left(x_{3t+2}\right) + f_{3t+3}\left(x_{3t+3}\right) - f_{3t+1}(-1) - f_{3t+2}(-1) - f_{3t+3}(-1) \geq 2C - 4$. Do đó, cứ mỗi 3 bước, Adam bị mất mát 1 lượng $2C - 4$, hay $R(T) \geq (2C - 4)T/3$. Vì $C \geq 2$, nên mất mát sẽ lớn dần và $R_T/T \nrightarrow 0$ khi $T \to \infty$. Từ đó được điều phải chứng minh. □

2.3 Các mở rộng của thuật toán Adam

2.3.1 AdaMax

Quy tắc cập nhật Adam cho từng trọng số riêng sẽ tỉ lệ nghịch với chuẩn L^2 của gradient quá khứ và hiện tại. Chúng ta có thể khái quát hoá chuẩn L^2 thành chuẩn L^p cho quy tắc cập nhật của Adam. Các biến thể như vậy không ổn định cho p lớn. Tuy nhiên, trong trường hợp đặc biệt khi $p \to \infty$ thì ta lại thu được một thuật toán ổn định đáng ngạc nhiên. Trong chuẩn L^p, tại bước thứ t tỉ lệ nghịch với $v_t^{1/p}$, với

$$v_t = \beta_2^p v_{t-1} + (1 - \beta_2^p)\left|g_t\right|^p = (1 - \beta_2^p)\sum_{i=1}^{t} \beta_2^{p(t-i)} \cdot \left|g_i\right|^p .$$

Algorithm 2: *AdaMax*, a variant of Adam based on the infinity norm.
Good default settings for the tested machine learning problems are $\alpha = 0.002$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. With β_1^t we denote β_1 to the power t. Here, $(\alpha/(1 - \beta_1^t))$ is the learning rate with the bias-correction term for the first moment. All operations on vectors are element-wise.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1^{st} moment vector)
  $u_0 \leftarrow 0$ (Initialize the exponentially weighted infinity norm)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $u_t \leftarrow \max(\beta_2 \cdot u_{t-1}, |g_t|)$ (Update the exponentially weighted infinity norm)
    $\theta_t \leftarrow \theta_{t-1} - (\alpha/(1 - \beta_1^t)) \cdot m_t/u_t$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

Lưu ý rằng thuật ngữ phân rã ở đây được tham số hoá là β_2^p thay vì β_2. Nếu $p \to \infty$ và

$u_t = \lim_{p\to\infty} (v_t)^{1/p}$ thì

$$\begin{aligned} u_t = \lim_{p\to\infty} (v_t)^{1/p} &= \lim_{p\to\infty} \left((1-\beta_2^p) \sum_{i=1}^{t} \beta_2^{p(t-i)} \cdot |g_i|^p \right)^{1/p} \\ &= \lim_{p\to\infty} (1-\beta_2^p)^{1/p} \left(\sum_{i=1}^{t} \beta_2^{p(t-i)} \cdot |g_i|^p \right)^{1/p} \\ &= \lim_{p\to\infty} \left(\sum_{i=1}^{t} \beta_2^{p(t-i)} \cdot |g_i|^p \right)^{1/p} \\ &= \max \left(\beta_2^{t-1} |g_1| , \beta_2^{t-2} |g_2| , \ldots , \beta_2 |g_{t-1}| , |g_t| \right) \end{aligned}$$

2.3.2 Nadam

Adam có thể được xem như là sự kết hợp của RMSprop và momentum vì nó sử dụng các trung bình trượt theo cấp số nhân (exponential moving averages) của bình phương gradient và gradient. Mặt khác, ta đã biết Nesterov accelerated gradient (NAG) giúp cho thuật toán hội tụ nhanh hơn so với momentum thông thường. Nadam (Nesterovaccelerated Adaptive Moment Estimation) là một thuật toán kết hợp Adam và NAG. Để phát triển Nadam, ta phải điều chỉnh giá trị momentum m_t trong Adam. Đầu tiên, ta xem lại công thức cập nhật momentum

$$\begin{aligned} g_t &= \nabla\theta_t J(\theta_t), \\ m_t &= \gamma m_{t-1} + \eta g_t, \\ \theta_{t+1} &= \theta_t - m_t. \end{aligned}$$

Trong đó, $\nabla\theta_t J(\theta_t)$ là gradient của điểm trước đó, m_t là momentum của điểm trước đó, γ thường được chọn với giá trị $0.9, \eta$ là learning rate. Mở rộng phương trình trên ta được

$$\theta_{t+1} = \theta_t - (\gamma m_{t-1} + \eta g_t).$$

Có thể thấy với momentum thông thường, lượng thay đổi là tổng của hai vector: momentum vector và gradient ở thời điểm hiện tại. Với Nesterove momentum, lượng thay đổi là tổng của hai vector: momentum vector và gradient ở thời điểm được xấp xỉ là điểm tiếp theo. Ta có công thức của NAG

$$\begin{aligned} g_t &= \nabla\theta_t J(\theta_t - \gamma m_{t-1}), \\ m_t &= \gamma m_{t-1} + \eta g_t, \\ \theta_{t+1} &= \theta_t - m_t. \end{aligned}$$

Dozat đề xuất cải biến NAG như sau: Thay vì áp dụng momentum hai lần - một lần để cập nhật gradient g_t và lần thứ hai để cập nhật θ_{t+1}, chúng ta áp dụng trực tiếp momentum xấp xỉ điểm tiếp theo để cập nhật các tham số hiện tại

$$g_t = \nabla\theta_t J\left(\theta_t\right),$$
$$m_t = \gamma m_{t-1} + \eta g_t,$$
$$\theta_{t+1} = \theta_t - \left(\gamma m_t + \eta g_t\right).$$

Có thể thấy trong công thức trên, thay vì sử dụng vector momentum m_{t-1} như trong phương pháp momentum, giờ ta sử dụng vector momentum m_t để đi trước một bước.

Chương 3

Thực nghiệm

Trong chương này, em trình bày code để thực nghiệm thuật toán tối ưu Adam. Chi tiết quá trình này được thực hiện trong file notebook `https://github.com/nguyenquocduong qnu/Optimization_Theory_QNU`.

3.1 Lập trình từ đầu cho thuật toán tối ưu Adam

3.2 Hồi quy logistic trên tập dữ liệu ung thư vú từ thư viện Sklearn sử dụng trình tối ưu theo thuật toán Adam

3.3 Thực nghiệm Adam và so sánh với các thuật toán cơ sở

Các thuật toán Gradient Descent có rất nhiều vấn đề trong quá trình hội tụ, đặc biệt với những hàm số khó hội tụ tốt như hàm số có nhiều cực tiểu địa phương, nhiều điểm yên ngựa. Để so sánh Adam đã giải quyết những vấn đề trên so với những giải thuật cơ sở ta tiến hành lấy một hàm số có tính chất như vậy để thử nghiệm

$$f(x) = \log\left(1 + (|x|)^{2+sinx}\right).$$

KẾT LUẬN

Qua quá trình tìm hiểu, chúng ta có thể thấy rằng nhiều thuật toán tối ưu hóa dựa trên gradient descent đã được đưa ra và những thuật toán sau luôn được bổ sung hoàn thiện để giải quyết những tồn tại của thuật toán trước. RMSprop là một mở rộng của Adagrad với khả năng giải quyết vấn đề learning rate trở nên quá nhỏ. Sau đó, thuật toán Adam bổ sung thêm bias-correction và momentum vào RMSprop để mang lại hiệu năng tốt hơn, nó sử dụng các cập nhật gradient tỷ lệ với căn bậc hai của bình phương trung bình trượt các gradient trong quá khứ. Trong bài tiểu luận này, em đã thực hiện được một số công việc sau

- Giới thiệu tổng quan về tối ưu trong học sâu.
- Giới thiệu tổng quan một số thuật toán làm nền tảng để nghiên cứu thuật toán tối ưu Adam.
- Trình bày chi tiết bài báo gốc thuật toán tối ưu Adam và chứng minh thuật toán Adam có thể không hội tụ.
- Thực nghiệm thuật toán Adam với mô hình hồi quy Logistic và so sánh với các thuật toán tối ưu khác.

Tài liệu tham khảo

[1] Ian Goodfellow, Yoshua Bengio, Aaron Courville, *Deep Learning*, MIT Press, 2016.

[2] Charu C. Aggarwal, *Linear Algebra and Optimization for Machine Learning*, A Textbook-Springer, 2020.

[3] Aston Zhang, Zack C. Lipton, Mu Li, Alex J. Smola, *Dive into Deep Learning*, `https://d2l.ai/`.

[4] Vũ Hữu Tiệp, *Machine learning cơ bản*, Nhà xuất bản khoa học và kĩ thuật, 2019.

[5] Trần Trung Trực, *Optimizer- Hiểu sâu về các thuật toán tối ưu (GD,SGD,Adam,..)*, `https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8`.

[6] Sebastian ruder, *An overview of gradient descent optimization algorithms*, `https://ruder.io/optimizing-gradient-descent/index.html`.

[7] Diederik P. Kingma, Jimmy Lei Ba, *Adam: A Method for Stochastic Optimization*, International Conference on Learning Representations, 2015.

[8] Sashank J.Reddi, Satyen Kate, Sanjiv Kumar, *On the convergence of adam and beyond*, ICLR 2018.