

Huấn luyện mạng nơ-ron nhiều tầng ẩn bằng thuật toán Adam

Nhóm sinh viên thực hiện:

- Nguyễn Ngọc Lan Như - 1712644
- Hoàng Minh Quân – 1712688

Giáo viên hướng dẫn: Th.S. Trần Trung Kiên

Mục lục

1. Giới thiệu đề tài
2. Kiến thức nền tảng
3. Thuật toán Adam
4. Thí nghiệm
5. Tổng kết

1.

Giới thiệu đề tài

Giới thiệu đề tài

Tại sao lại sử dụng mạng nơ-ron nhiều tầng ẩn?



Giới thiệu đề tài

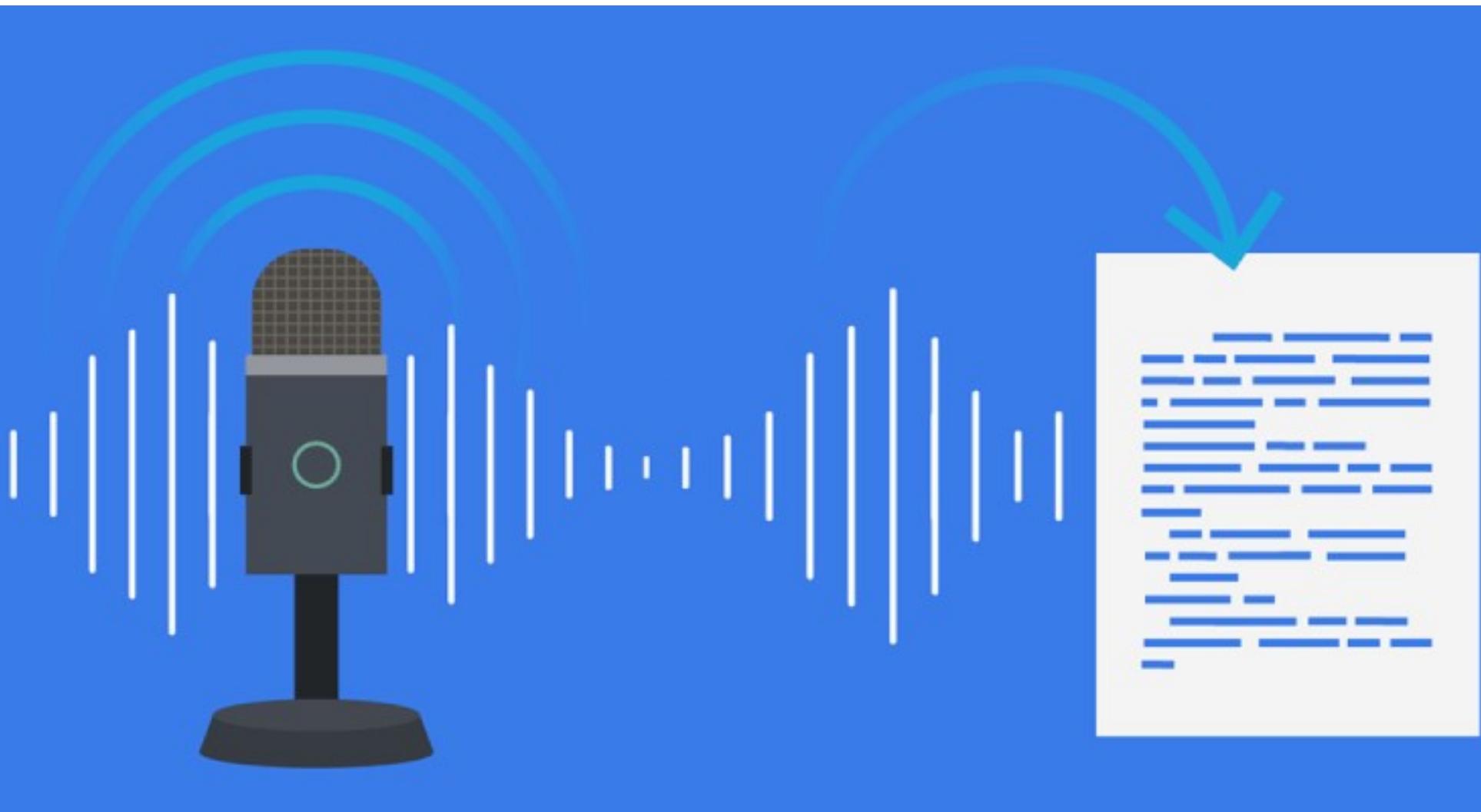
Tại sao lại sử dụng mạng nơ-ron nhiều tầng ẩn?

MACHINE TRANSLATION



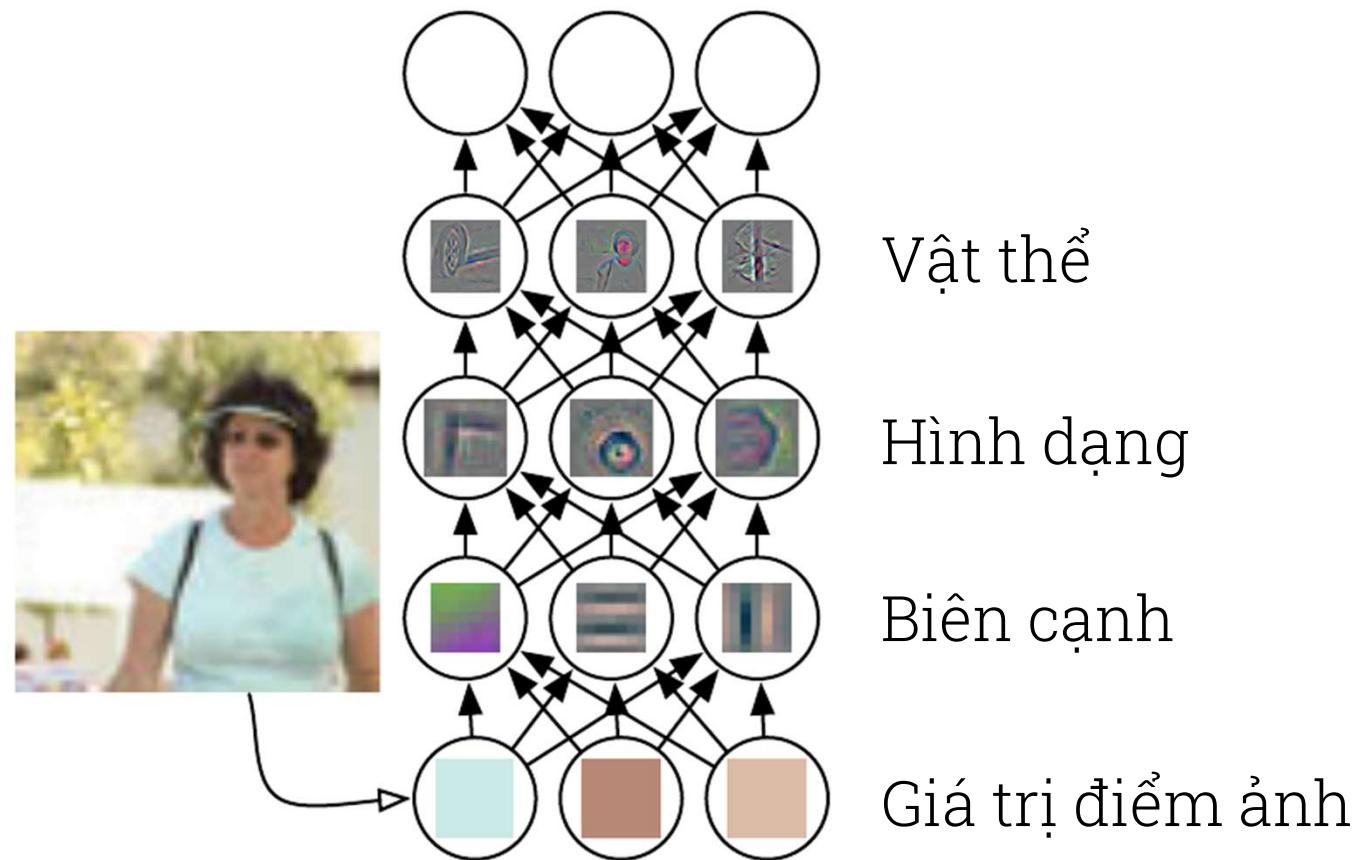
Giới thiệu đề tài

Tại sao lại sử dụng mạng nơ-ron nhiều tầng ẩn?



Giới thiệu đề tài

Tại sao lại sử dụng mạng nơ-ron nhiều tầng ẩn?



Giới thiệu đề tài

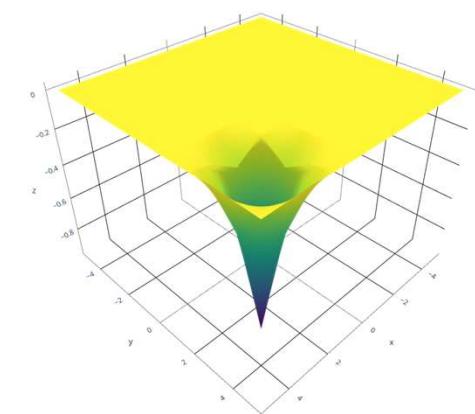
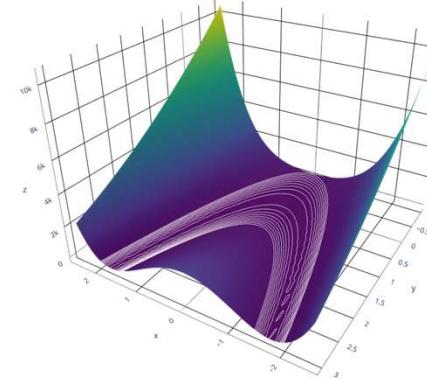
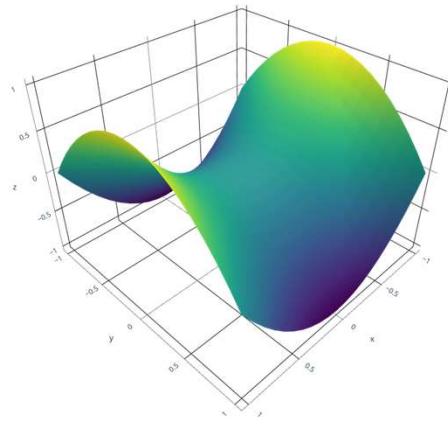
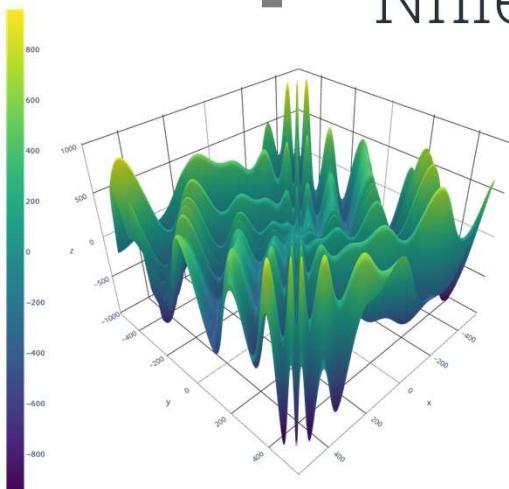
Bài toán huấn luyện mạng nơ-ron nhiều tầng ẩn

- **Input:** Hàm chi phí với tham số là các trọng số của mạng nơ-ron nhiều tầng ẩn. Hàm chi phí cho biết sự sai lệch giữa kết quả dự đoán của mạng nơ-ron so với giá trị đúng trên tập dữ liệu huấn luyện, hay *độ lỗi*.
- **Output:** Bộ trọng số của mạng nơ-ron nhiều tầng ẩn cho độ lỗi là nhỏ nhất, hoặc đủ nhỏ.

Giới thiệu đề tài

Thách thức

- Mặt phẳng lồi phức tạp:
 - Nhiều cực tiểu địa phương.
 - Nhiều điểm yên ngựa.
 - Nhiều vùng bằng phẳng.
 - Nhiều rãnh hẹp.



*Yann Dauphin et al., "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization", *Advances in Neural Information Processing Systems 27*, 2014, pp. 2933-2941.

Giới thiệu đề tài

Bài báo tìm hiểu

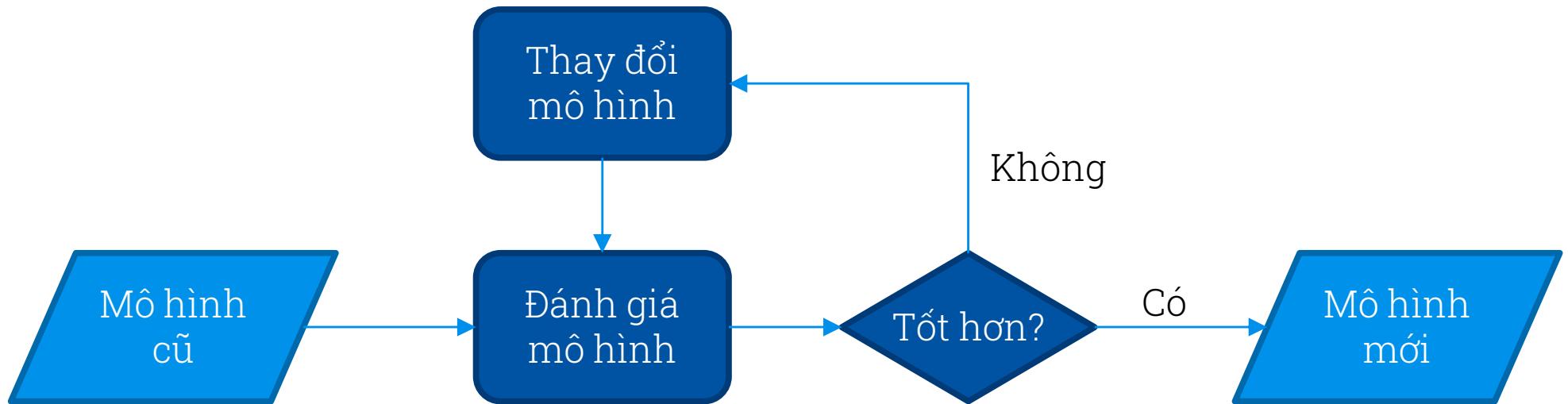
- "Adam: A method for stochastic optimization", Diederik P. Kingma, Jimmy Lei Ba (2014).
 - Sử dụng tỉ lệ học riêng biệt cho từng trọng số.
 - Hội tụ về điểm cực tiểu có độ lỗi thấp.

2.

Kiến thức nền tảng

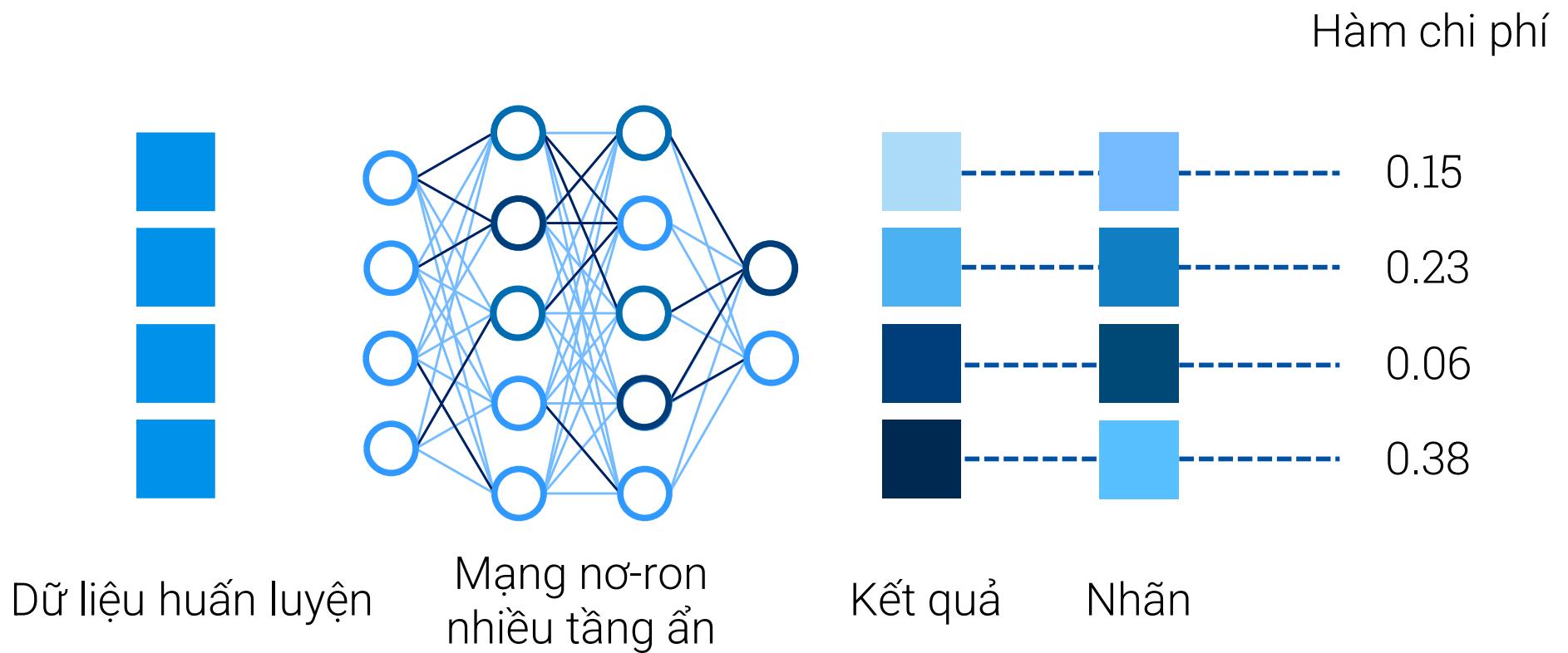
Kiến thức nền tảng

Tối ưu hóa (optimization)



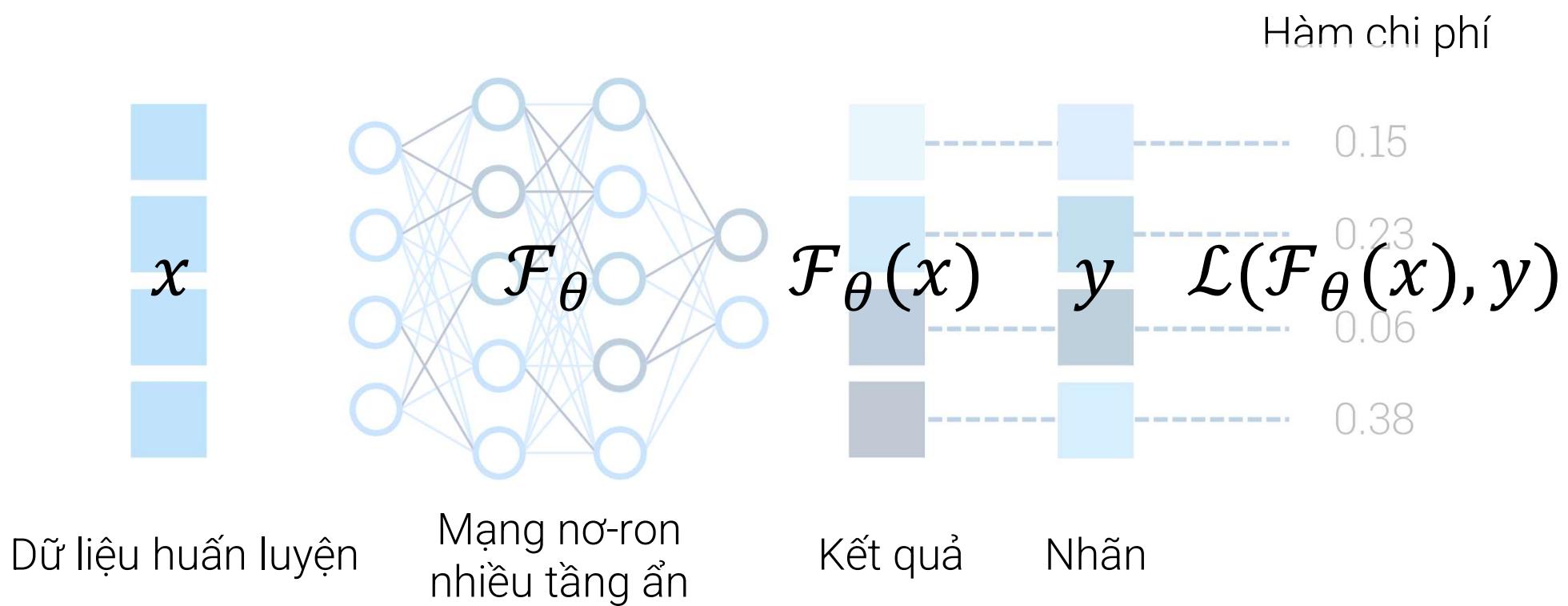
Kiến thức nền tảng

Tối ưu hóa (optimization)



Kiến thức nền tảng

Tối ưu hóa (optimization)



Kiến thức nền tảng

Gradient Descent

- Sử dụng gradient của cả tập dữ liệu để xác định hướng đi có sự thay đổi lớn nhất.
- Chi phí tính toán cao.
- Chỉ thực hiện một bước cập nhật cho mỗi lần duyệt qua tập dữ liệu.

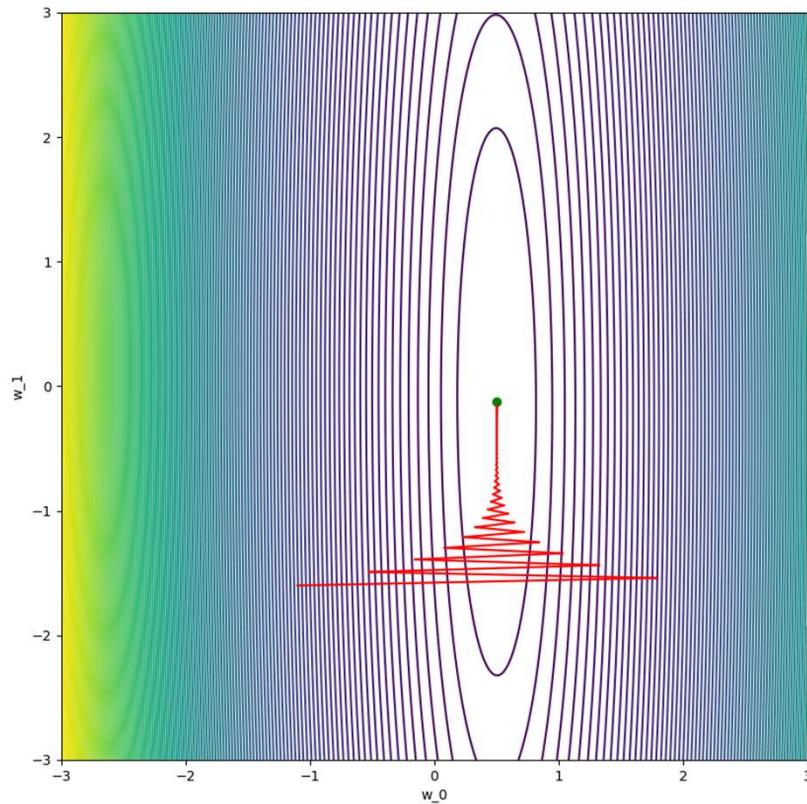
Kiến thức nền tảng

Stochastic Gradient Descent

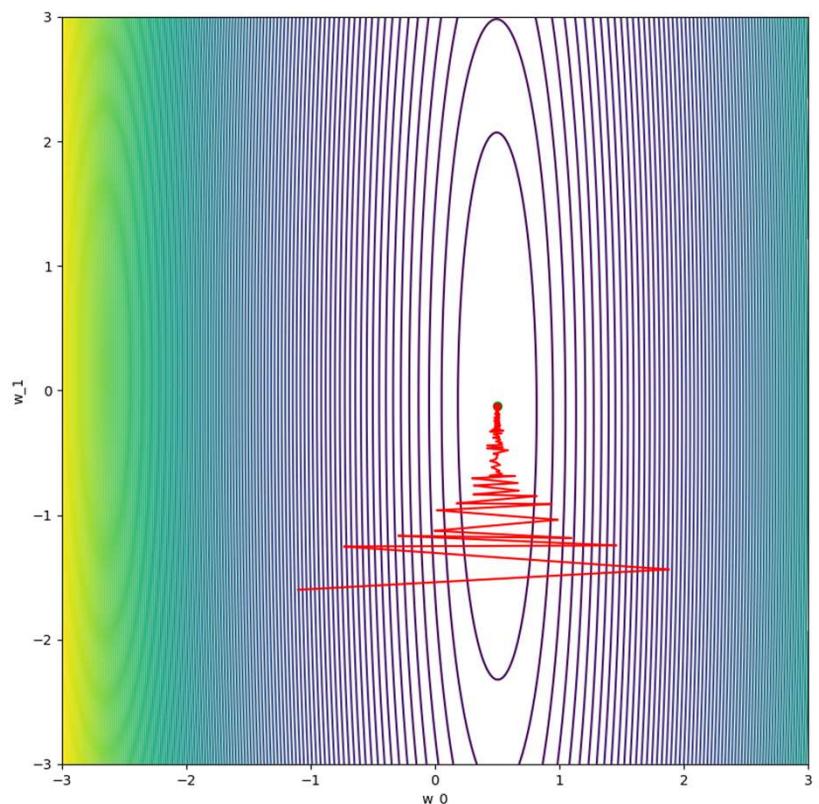
- Sử dụng gradient của một tập con của dữ liệu để xấp xỉ hướng của gradient trên cả tập dữ liệu.
- Thực hiện được nhiều bước cập nhật trong một lần duyệt qua tập dữ liệu.

Kiến thức nền tảng

Stochastic Gradient Descent



Gradient Descent



Stochastic Gradient Descent

3.

Huấn luyện mạng nơ-ron nhiều tầng ẩn

Momentum

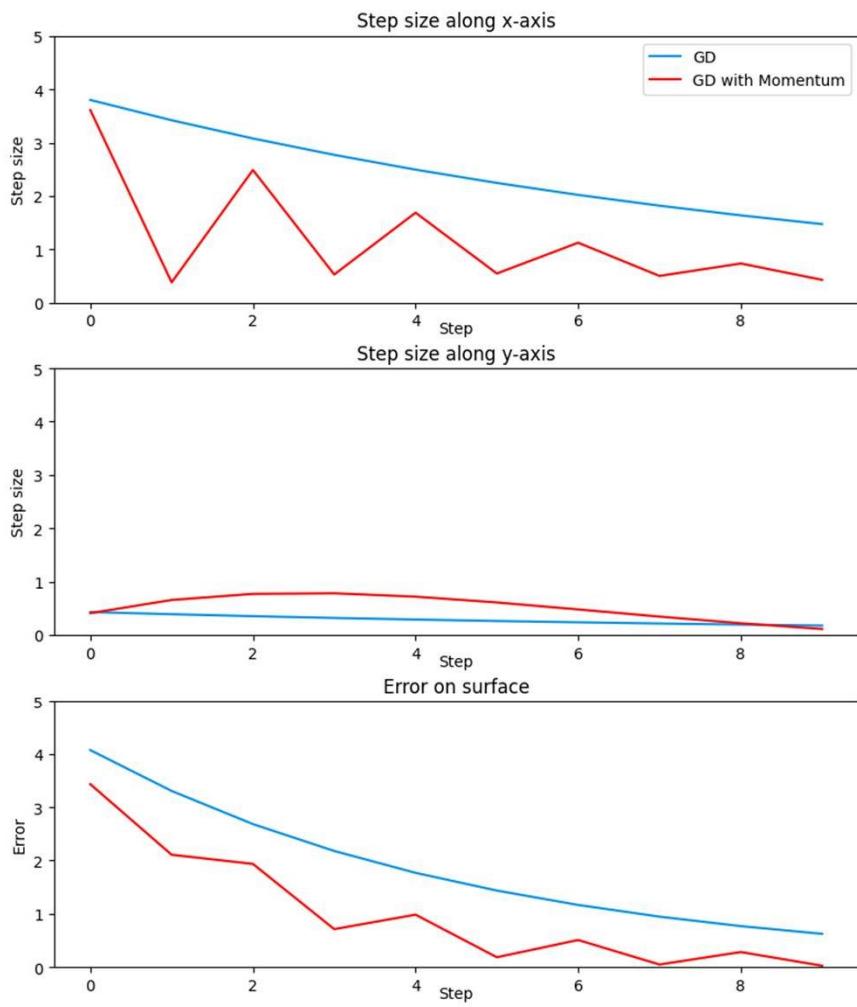
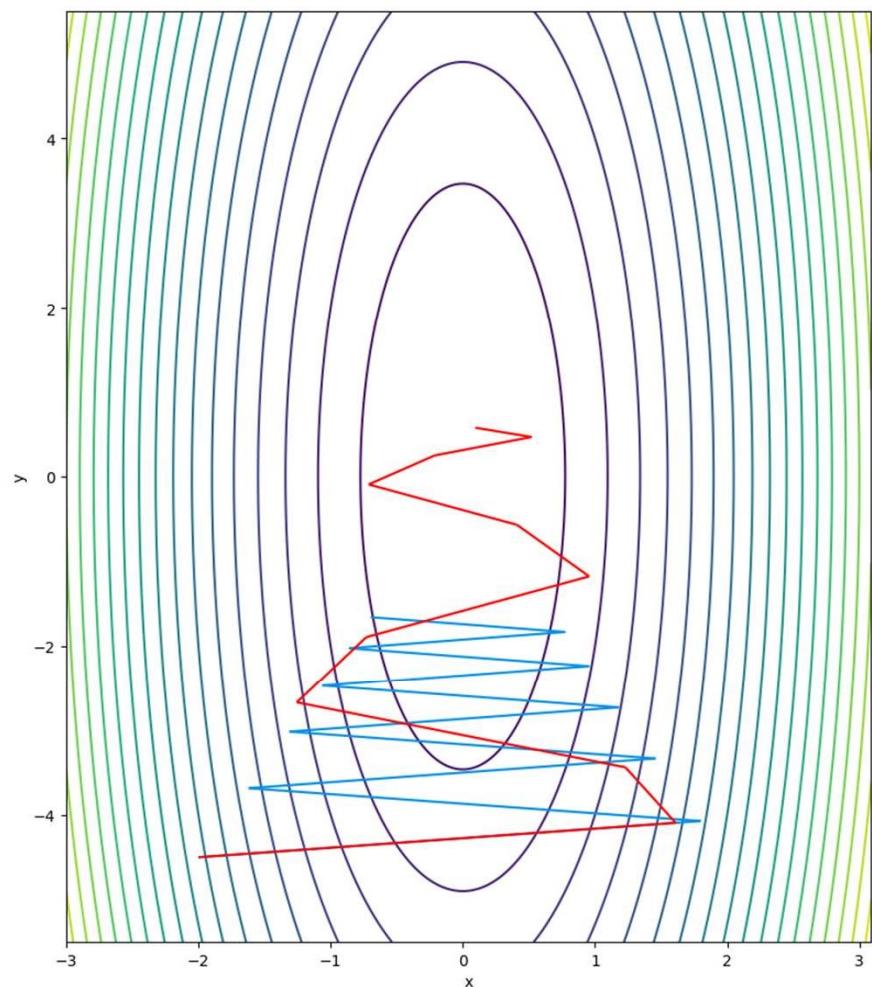
Tăng tốc Gradient Descent

- Stochastic Gradient Descent with Momentum (Momentum)*.
 - Tăng độ lớn cập nhật khi chiều gradient không đổi.
 - Giảm độ lớn cập nhật khi chiều gradient thay đổi liên tục.

*Ning Qian, "On the momentum term in gradient descent learning algorithms", *Journal of the International Neural Network Society*, 1999, vol. 12, pp. 145-151.

Momentum

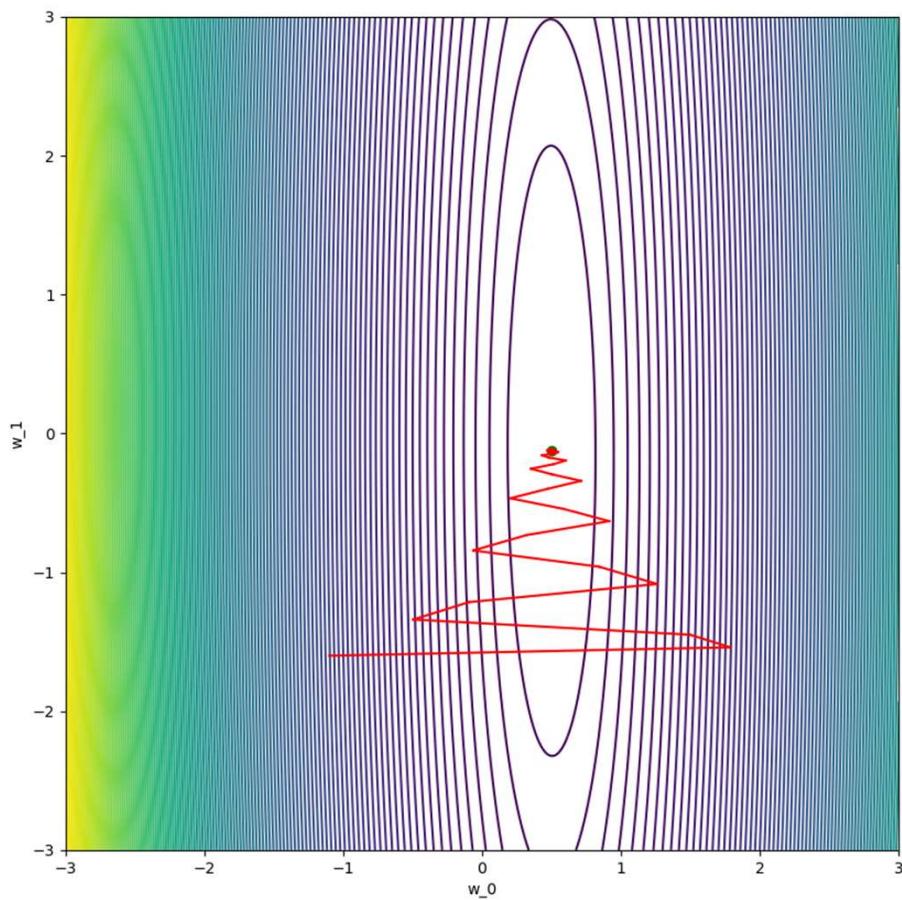
Tăng tốc Gradient Descent



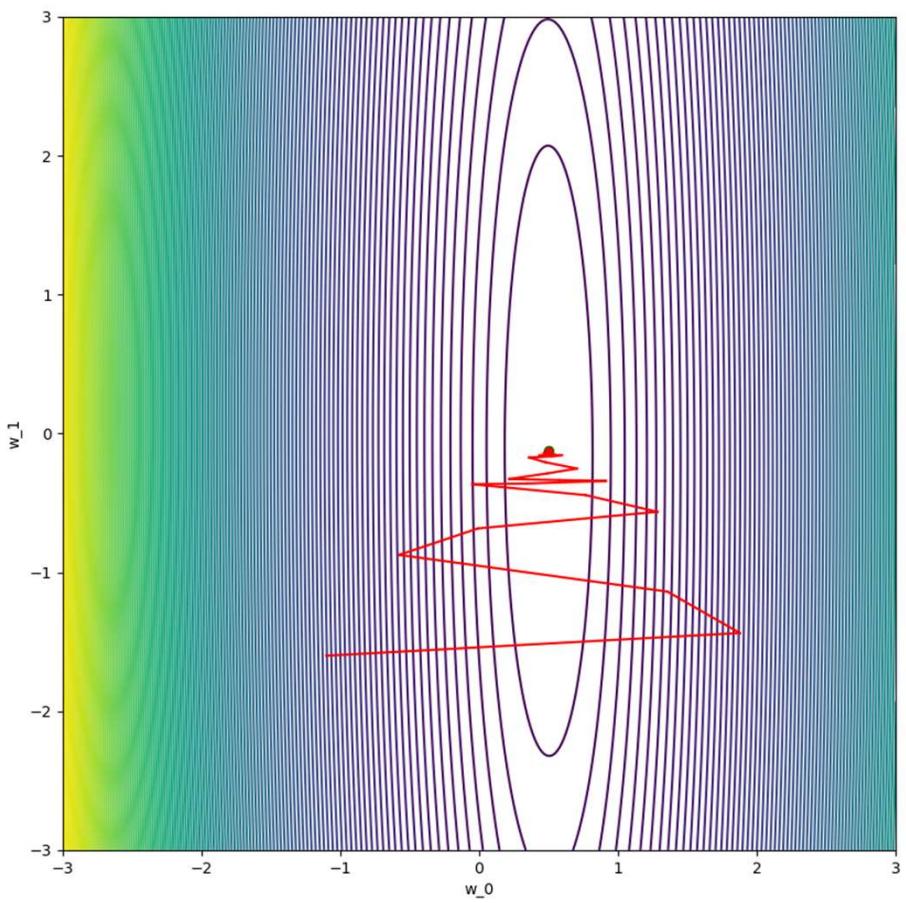
Momentum

Tăng tốc Gradient Descent

- Xấp xỉ gradient của cả tập dữ liệu tốt hơn.



GD with Momentum

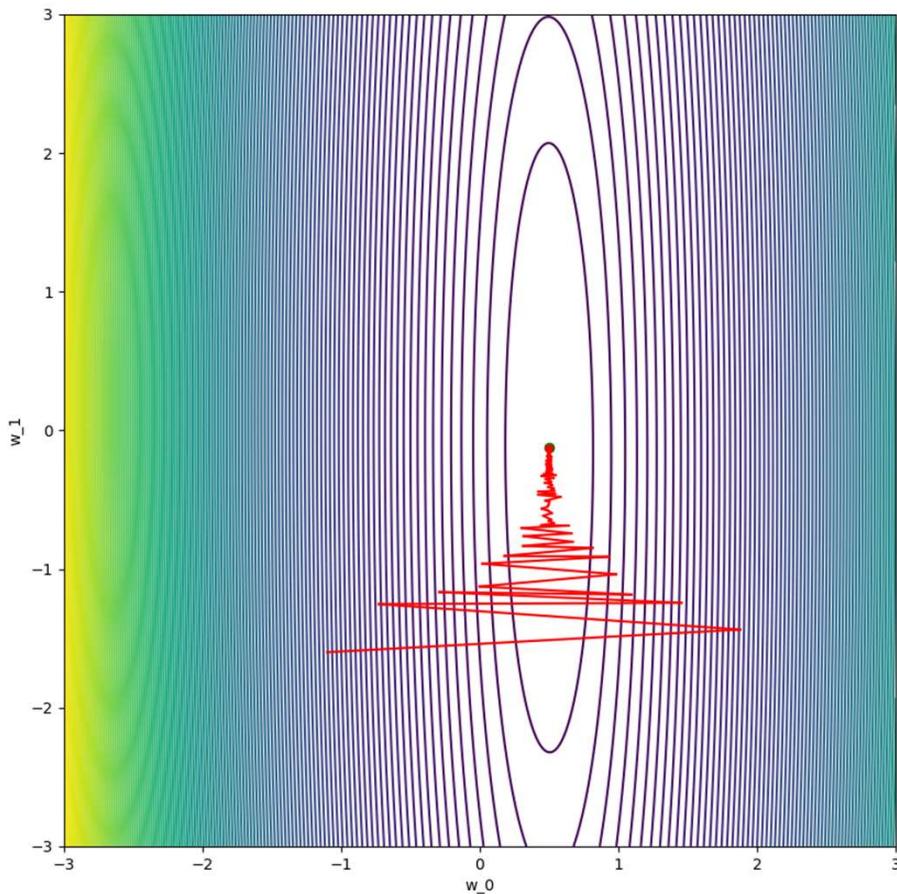


SGD with Momentum

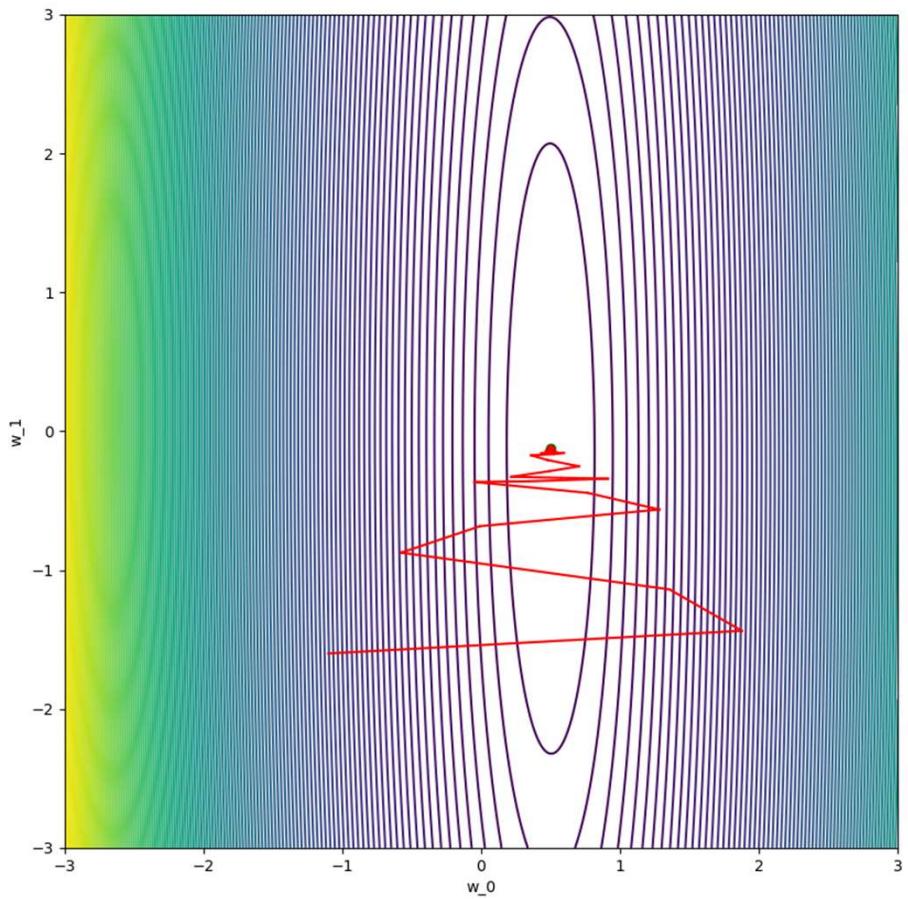
Momentum

Tăng tốc Gradient Descent

- Giảm dao động.



SGD

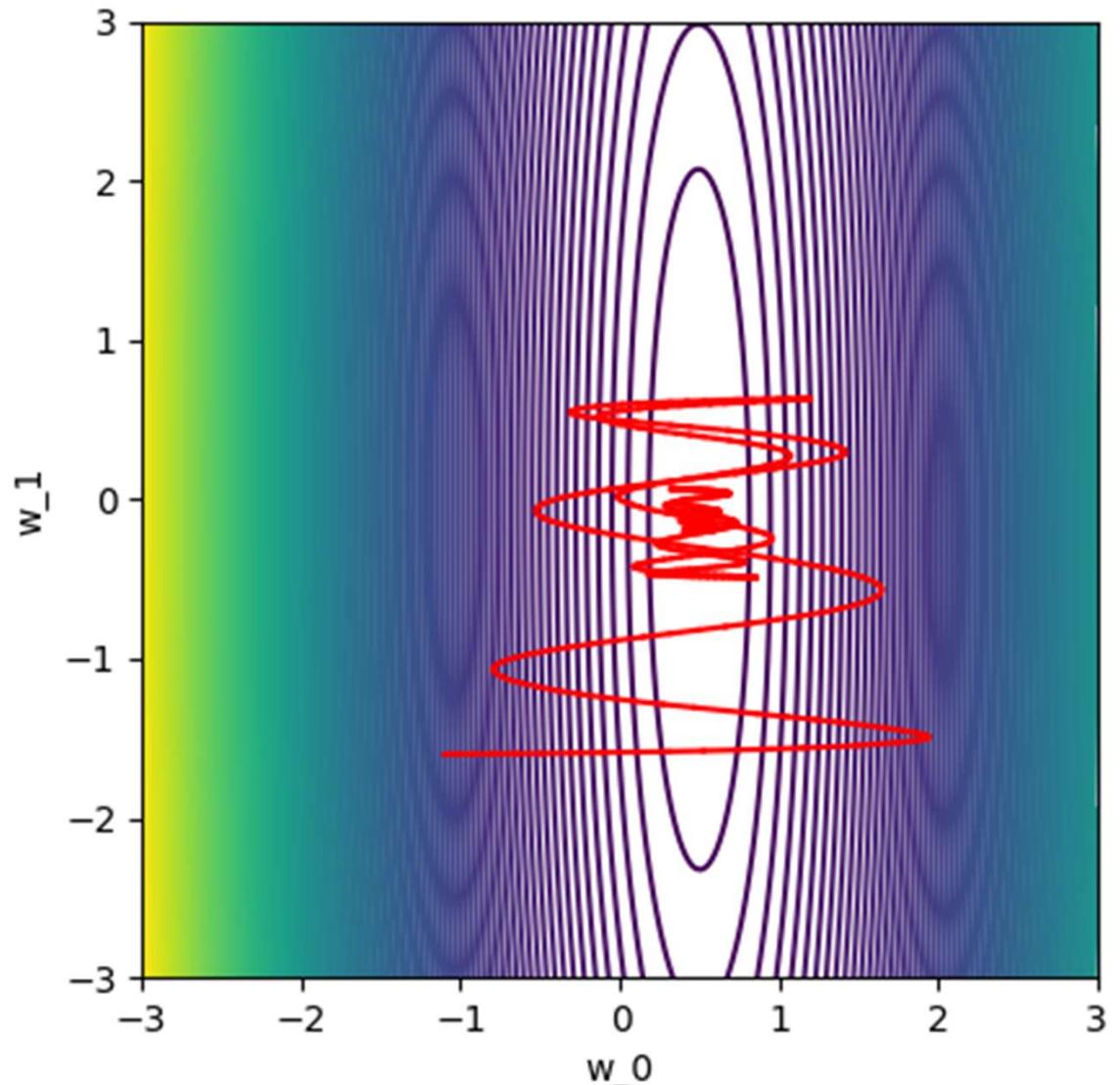


SGD with Momentum

Momentum

Tăng tốc Gradient Descent

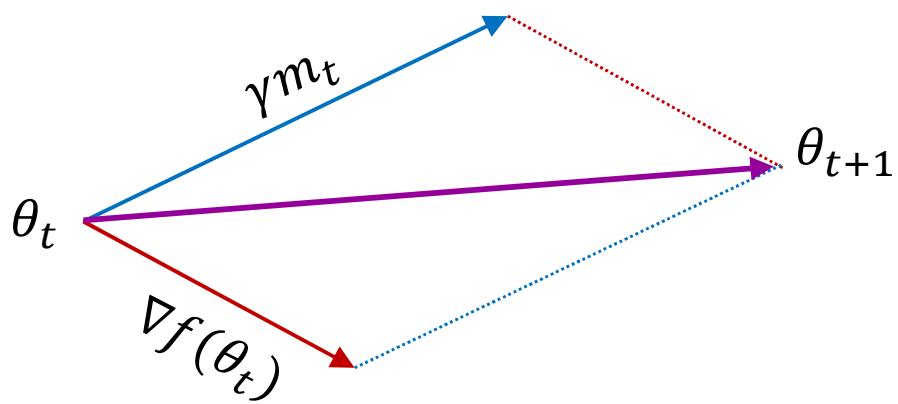
- Dao động gần cực tiểu khi hệ số quán tính quá lớn.



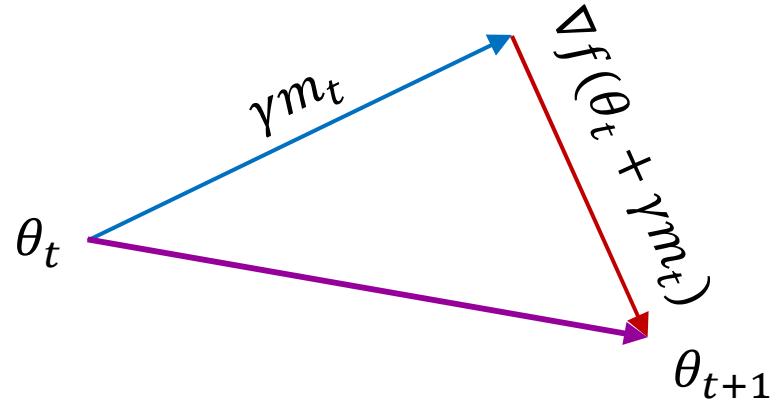
Momentum

Tăng tốc Gradient Descent

- Nesterov Accelerated Descent (NAG)*
 - Tính đạo hàm tại (**điểm hiện tại + quán tính**) để lấy hướng cập nhật tiếp theo rồi mới cộng quán tính vào lượng cập nhật.



Momentum



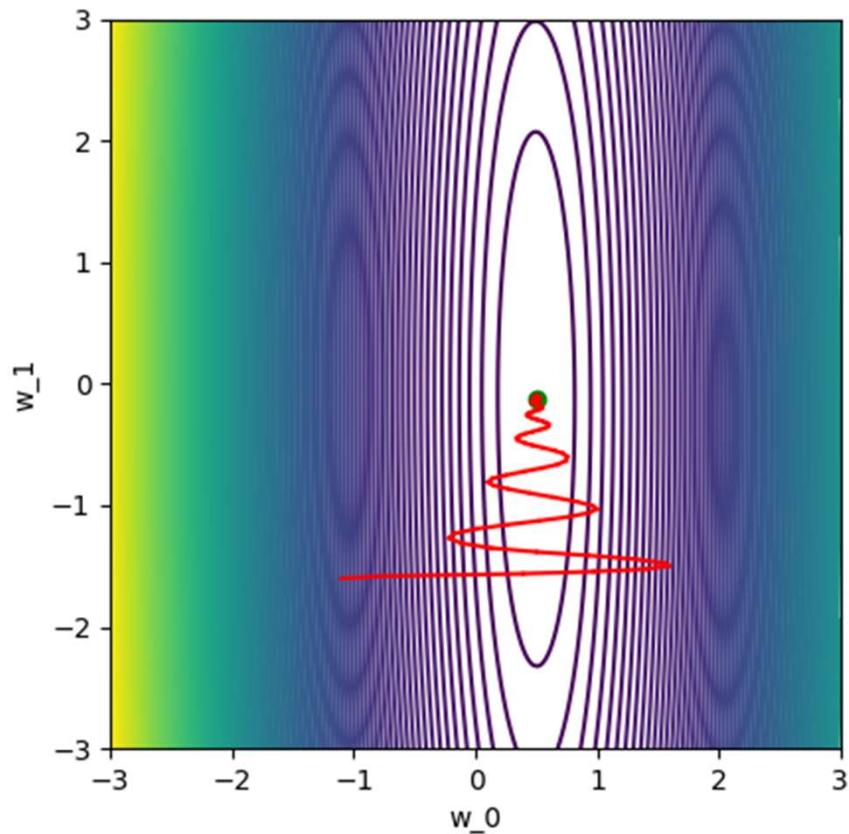
Nesterov

*Nesterov, Y., "A method of solving a convex programming problem with convergence rate $O(1/\sqrt{k})$ ", *Soviet Mathematics Doklady*, 1983, vol. 27, pp. 372-376.

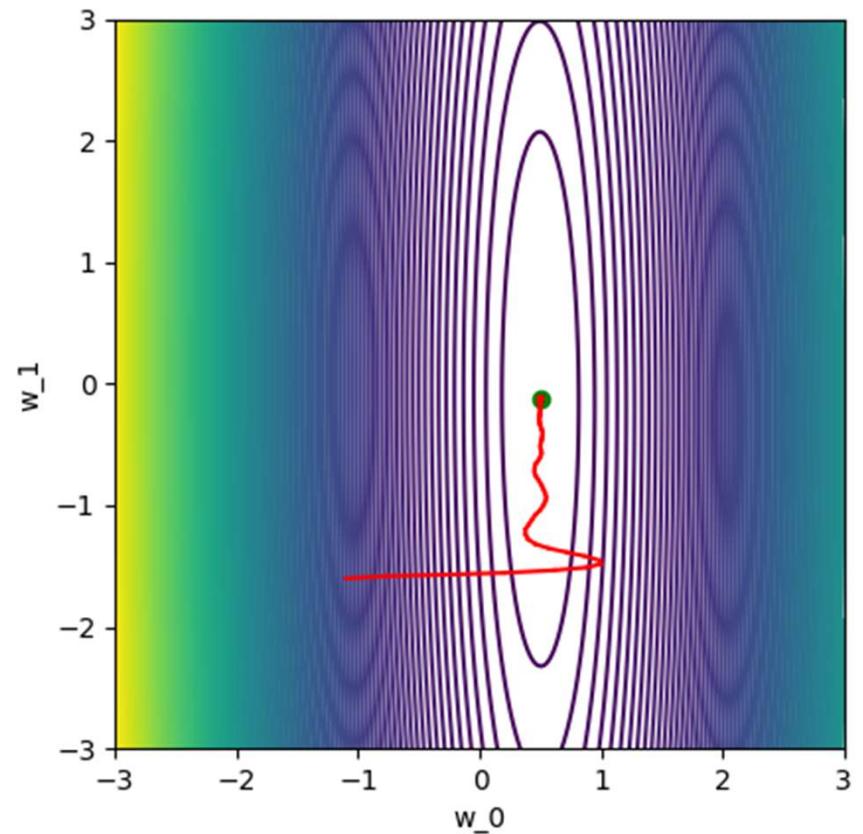
Momentum

Tăng tốc Gradient Descent

- Đường đi ổn định hơn.



SGD with Momentum

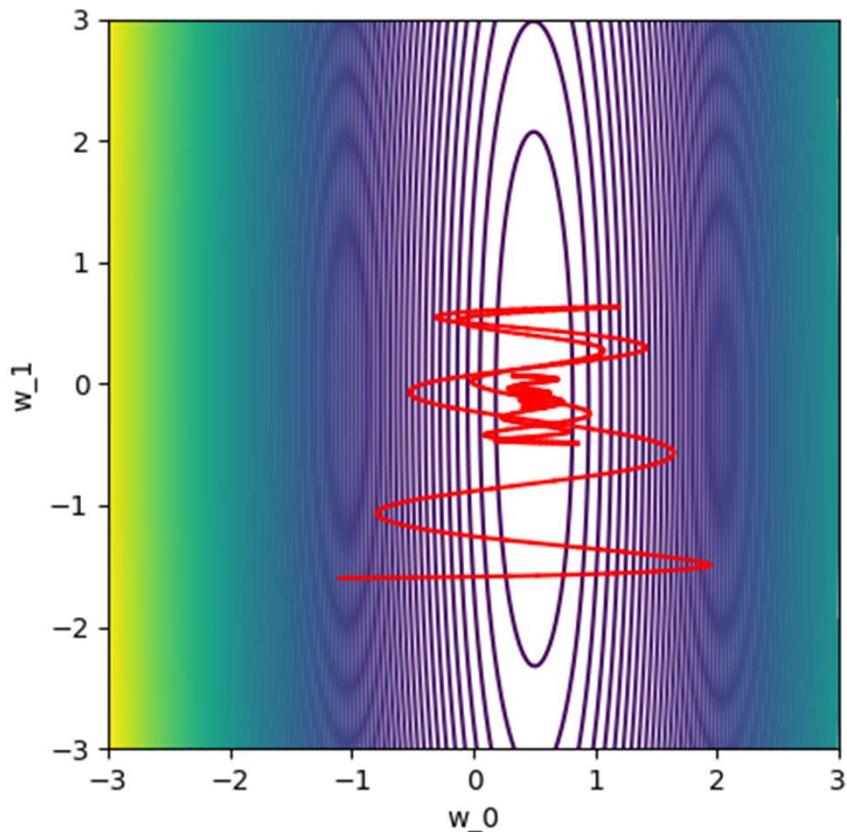


SGD with Nesterov

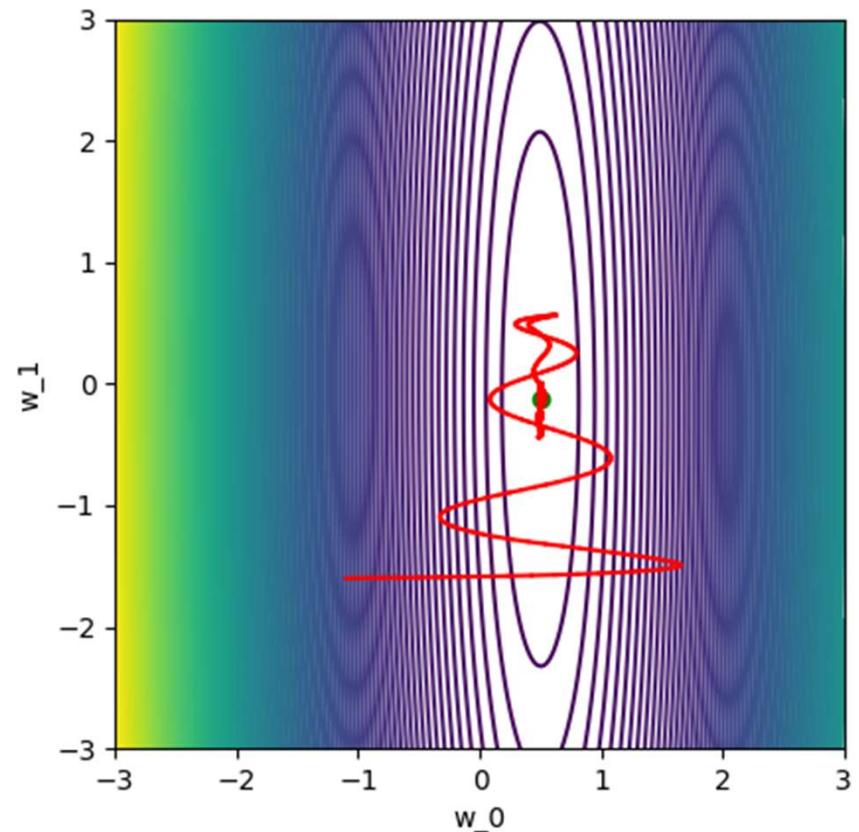
Momentum

Tăng tốc Gradient Descent

- Cho phép sử dụng hệ số momentum lớn hơn.



SGD with Momentum



SGD with Nesterov

Tỉ lệ học thích ứng

Rprop*

- Áp dụng bước nhảy (step size) riêng biệt cho từng tham số.
- Sử dụng dấu của đạo hàm.
- Chỉ hoạt động với Gradient Descent.
 - Không xấp xỉ được hướng gradient của cả tập dữ liệu do sử dụng tỉ lệ học quá lớn.

*Riedmiller, Martin, and Heinrich Braun. "A direct adaptive method for faster backpropagation learning: The RPROP algorithm." *IEEE international conference on neural networks*. IEEE, 1993.

Tỉ lệ học thích ứng

RMSprop*

- $\theta_{t+1} = \theta_t - \eta \frac{g_t}{|g_t|}$
- $G_t = \gamma G_t + (1 - \gamma) g_t^2$
- $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{g_t^2 + \epsilon}} \cdot g_t$
- $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot g_t$

*T. Tieleman and G. Hinton, "Lecture 6.5 - rmsprop," COURSERA: Neural Networks for Machine Learning, 2012.

Tỉ lệ học thích ứng

RMSprop

- Áp dụng tỉ lệ học (learning rate) riêng biệt cho từng tham số.
- Sử dụng dấu của đạo hàm.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\frac{G_t}{g_t^2} + \epsilon}} \odot \text{sign}(g_t)$$

Tỉ lệ học thích ứng

RMSprop

- Tính đạo hàm theo từng tham số.

$$g_t = \nabla_{\theta} \mathcal{L}(\theta_t)$$

- Cập nhật G ở bước hiện tại.

$$G_t = \gamma G_{t-1} + (1 - \gamma) g_t^2$$

- Cập nhật trọng số.

"leaky"

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot g_t$$

Tỉ lệ học thích ứng

RMSprop

- Tính đạo hàm theo từng tham số.

$$g_t = \nabla_{\theta} \mathcal{L}(\theta_t)$$

- Cập nhật G ở bước hiện tại.

$$G_t = \gamma G_{t-1} + (1 - \gamma) \mathbf{g}_t^2$$

- Cập nhật trọng số.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot g_t$$

Tỉ lệ học thích ứng

RMSprop

- Tính đạo hàm theo từng tham số.

$$g_t = \nabla_{\theta} \mathcal{L}(\theta_t)$$

- Cập nhật G ở bước hiện tại.

$$G_t = \gamma G_{t-1} + (1 - \gamma) \text{diag}(\mathbf{g}_t \cdot \mathbf{g}_t^T)$$

- Cập nhật trọng số.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot g_t$$

Tỉ lệ học thích ứng

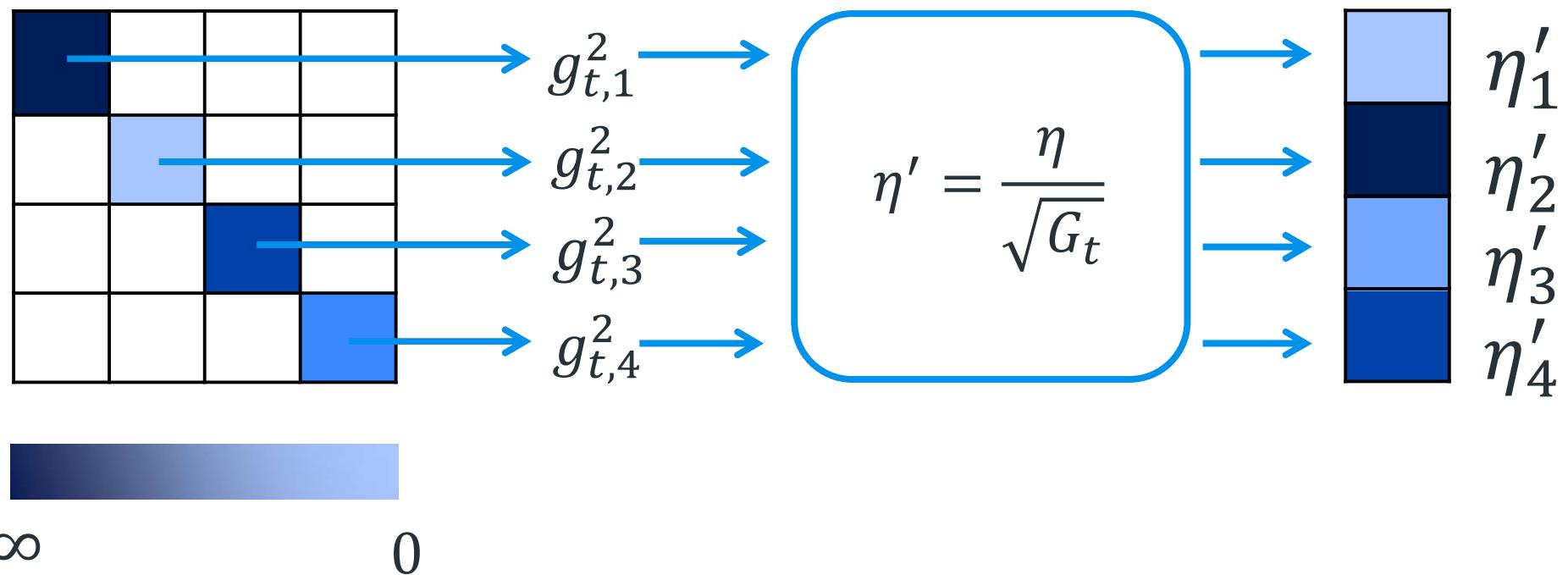
RMSprop

$$g_t = \begin{bmatrix} g_{t,1} \\ g_{t,2} \\ g_{t,3} \\ \vdots \\ g_{t,n} \end{bmatrix} \quad g_t^T = [g_{t,1} \quad \dots \quad g_{t,n}]$$

$$g_t \cdot g_t^T = \begin{bmatrix} g_{t,1} \\ g_{t,2} \\ g_{t,3} \\ \vdots \\ g_{t,n} \end{bmatrix} [g_{t,1} \quad \dots \quad g_{t,n}] = \begin{bmatrix} g_{t,1}^2 & \cdots & g_{t,1} \cdot g_{t,n} \\ \vdots & \ddots & \vdots \\ g_{t,n} \cdot g_{t,1} & \cdots & g_{t,n}^2 \end{bmatrix}$$

Tỉ lệ học thích ứng

RMSprop



Thuật toán Adam

Nguyên lý

- Kết hợp các cải tiến của các thuật toán khác.
 - Momentum.
 - Tỉ lệ học thích ứng.

Thuật toán Adam

Các bước thực hiện

Cập nhật m_t và v_t

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \quad \text{—— Momentum}$$
$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \quad \text{—— RMSprop}$$

Tính **bias-correction** của m_t và v_t

$$\widehat{m}_t = \frac{m_t}{(1 - \beta_1^t)}$$
$$\widehat{v}_t = \frac{v_t}{(1 - \beta_2^t)}$$

t tăng dần $\rightarrow \beta^t$ giảm dần
 $\rightarrow 1 - \beta^t$ tiến dần đến 1

Cập nhật **trọng số**

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t} + \epsilon}$$

4.

Thí nghiệm

Thí nghiệm

Cách thực hiện

- Khởi tạo mạng nơ-ron với các tham số ngẫu nhiên.
- Lưu các tham số này làm **điểm xuất phát chung**.
- Với mỗi thuật toán:
 - Nạp lại bộ trọng số đã lưu ở trên cho mạng nơ-ron.
 - Thực hiện tối ưu hóa mạng nơ-ron với số bước xác định.
 - Ghi nhận độ lỗi tại mỗi epoch.

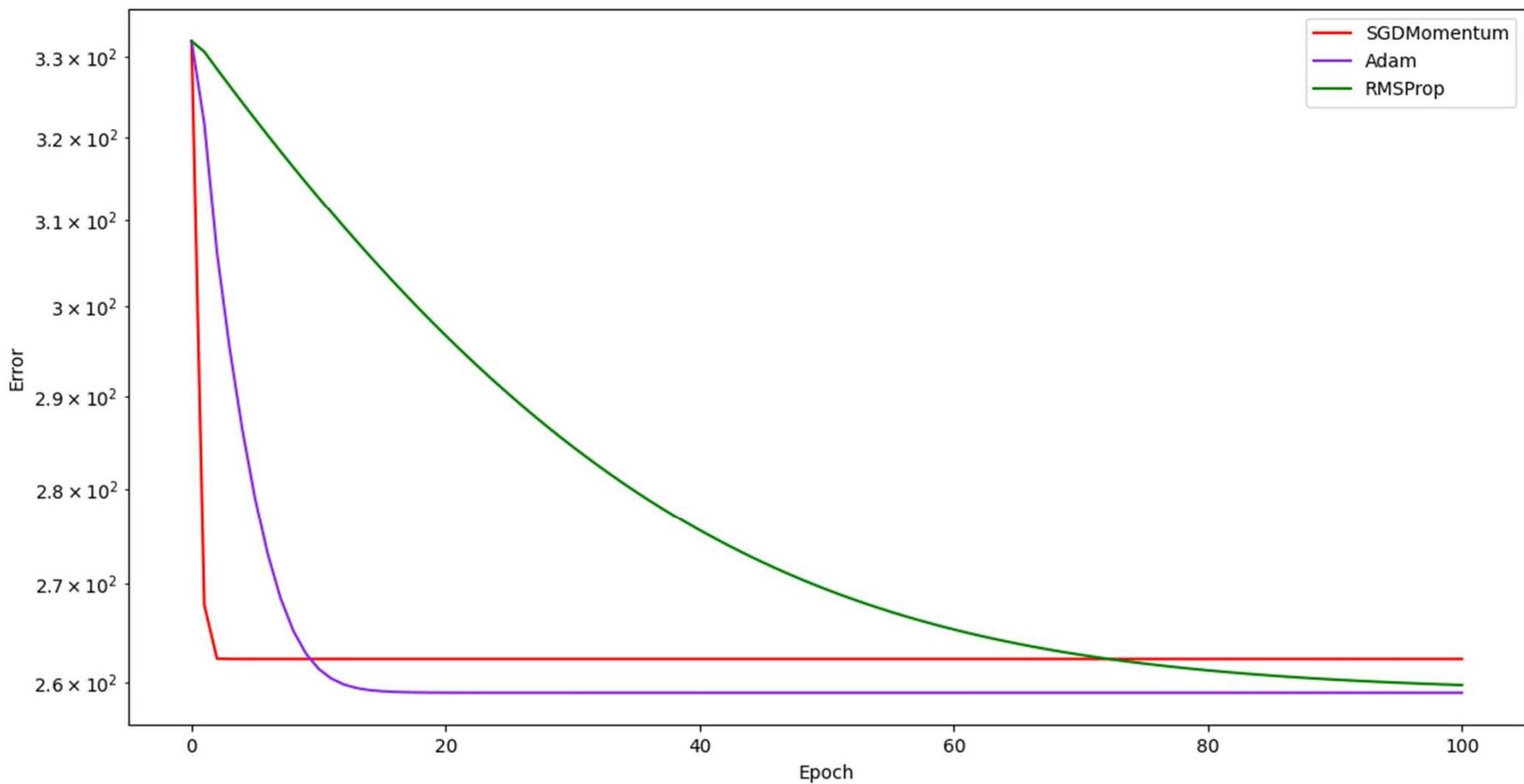
Thí nghiệm

Các thiết lập thí nghiệm

- Sử dụng ngôn ngữ lập trình Python và thư viện Pytorch.
- Sử dụng GPU NVIDIA Tesla T4 trên nền tảng Google Colaboratory để tăng tốc độ thực hiện bằng việc tính toán song song.

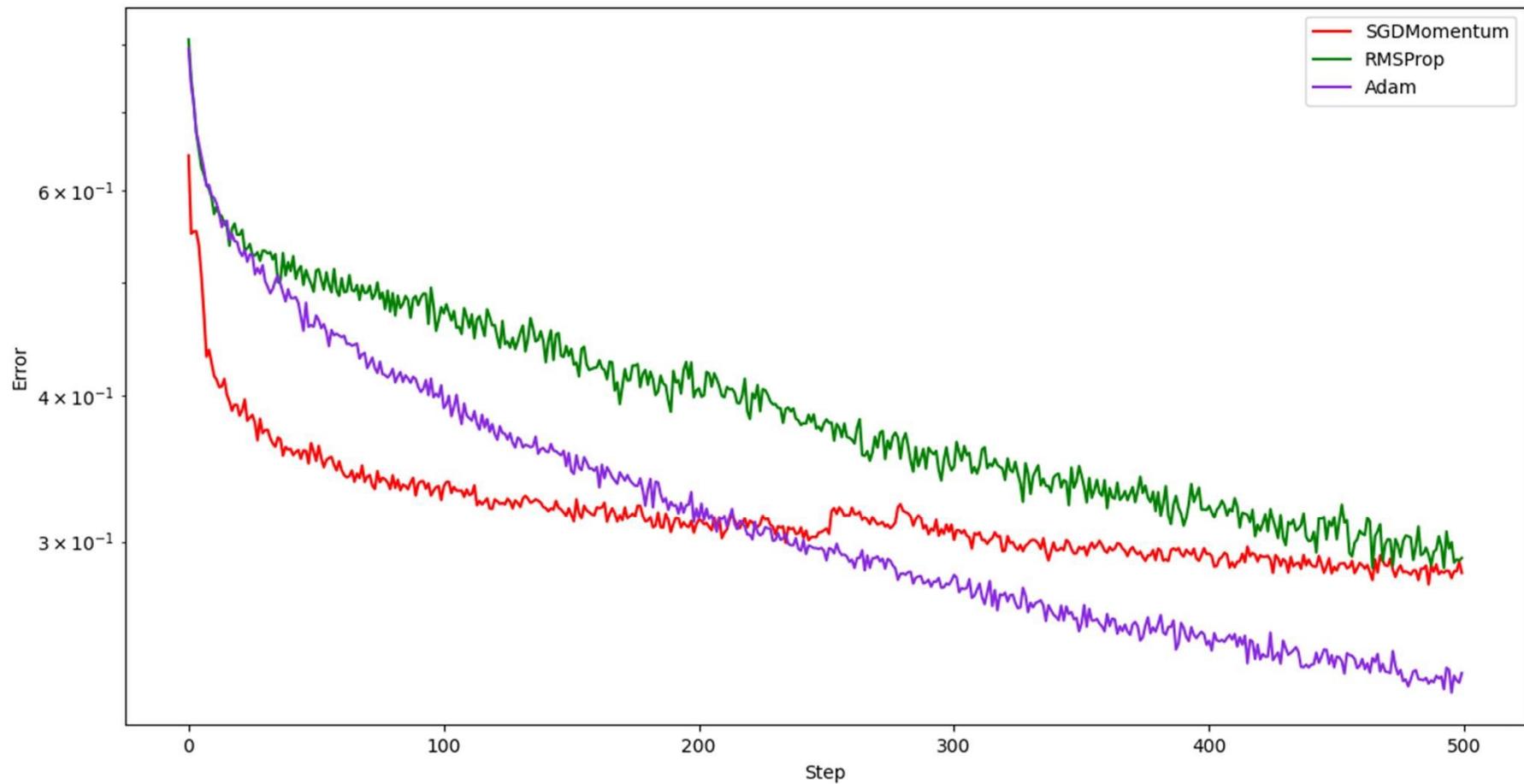
Thí nghiệm

Linear regression với độ nhiễu cao



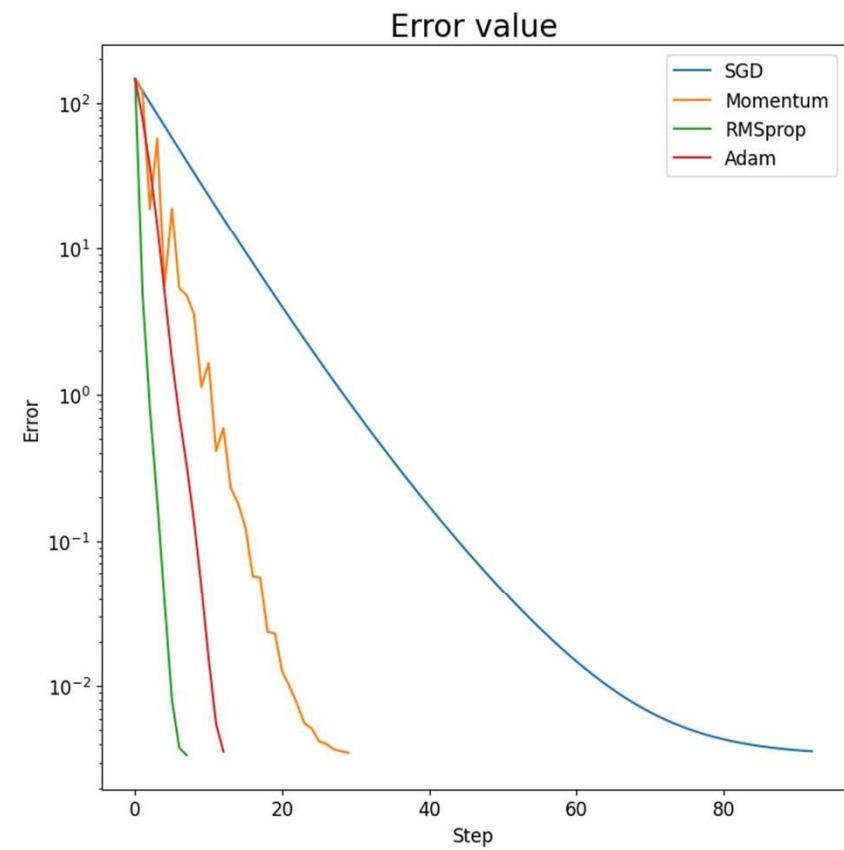
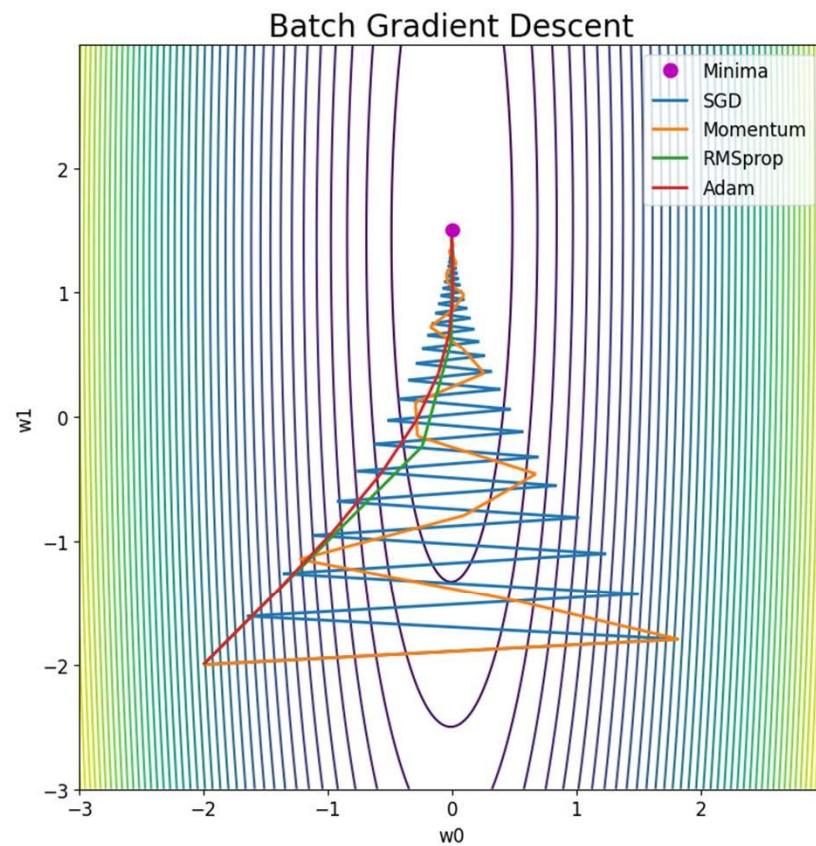
Thí nghiệm

Classification với dữ liệu thưa



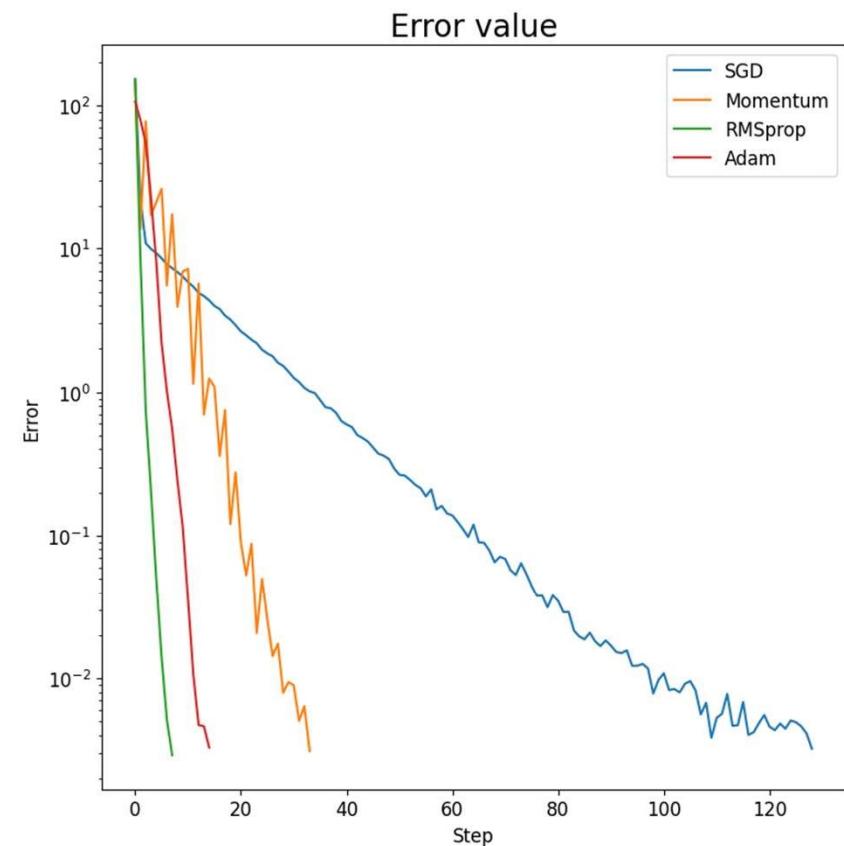
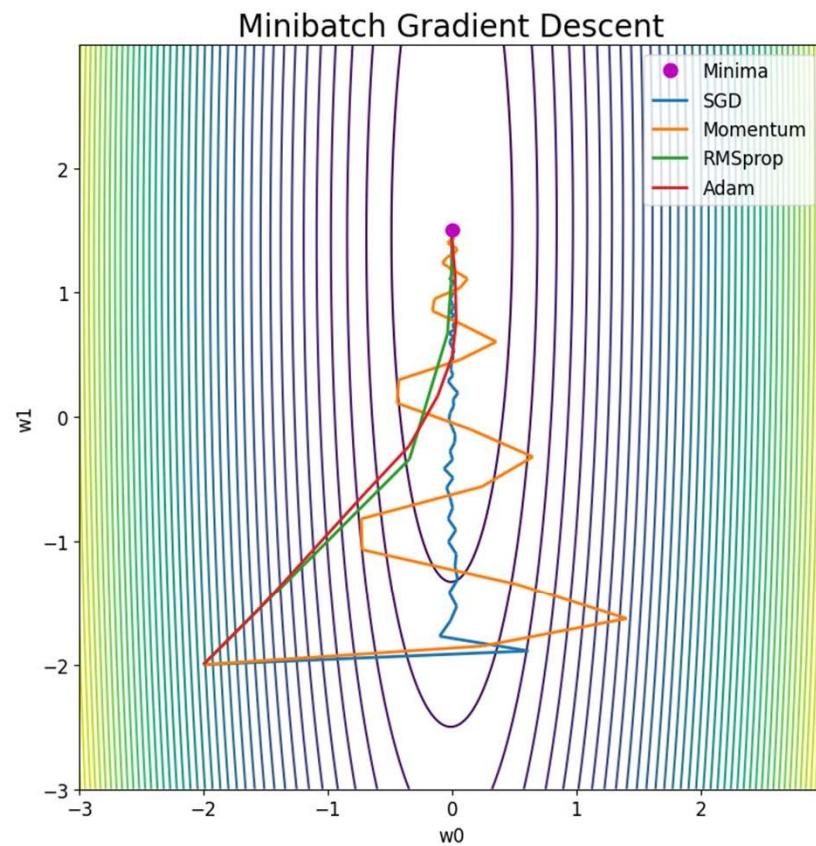
Thí nghiệm

Linear regression với tham số aligned



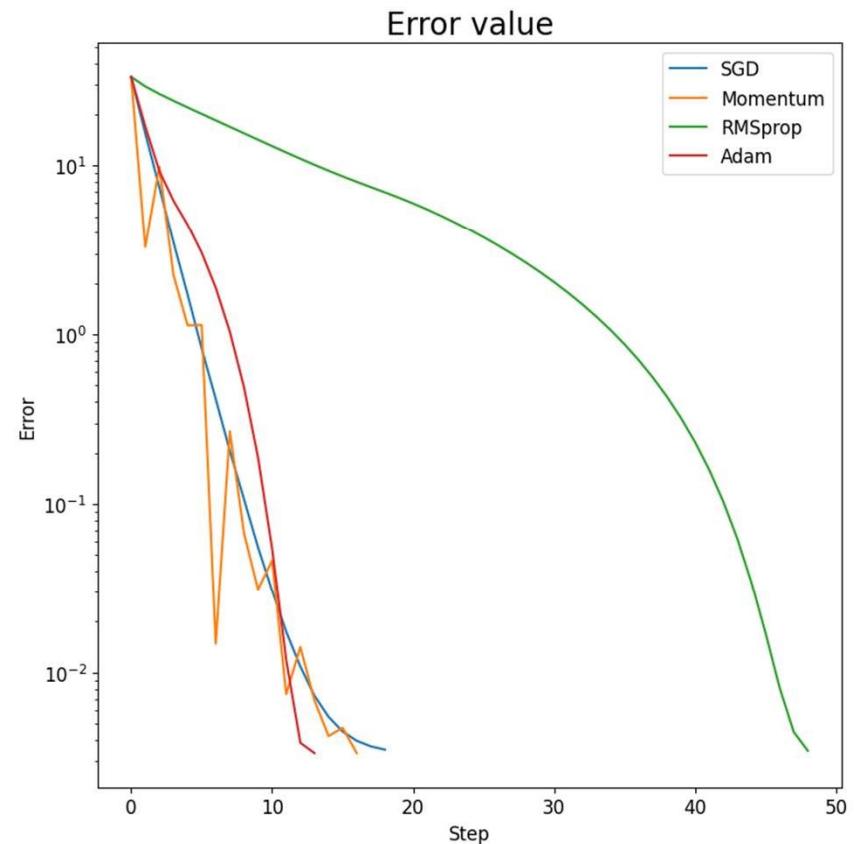
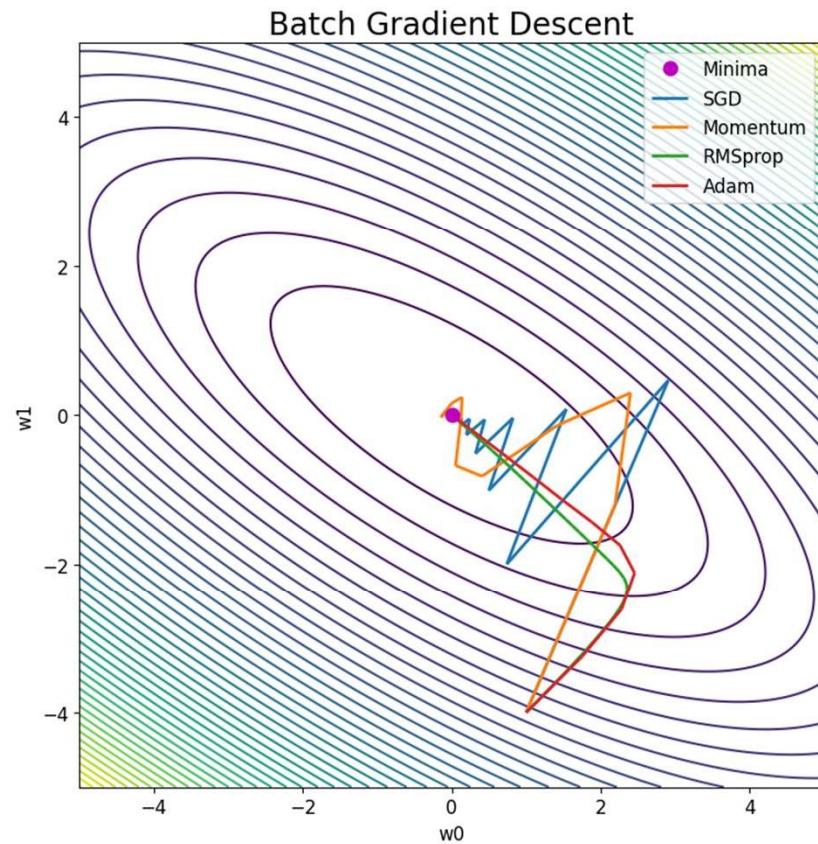
Thí nghiệm

Linear regression với tham số aligned



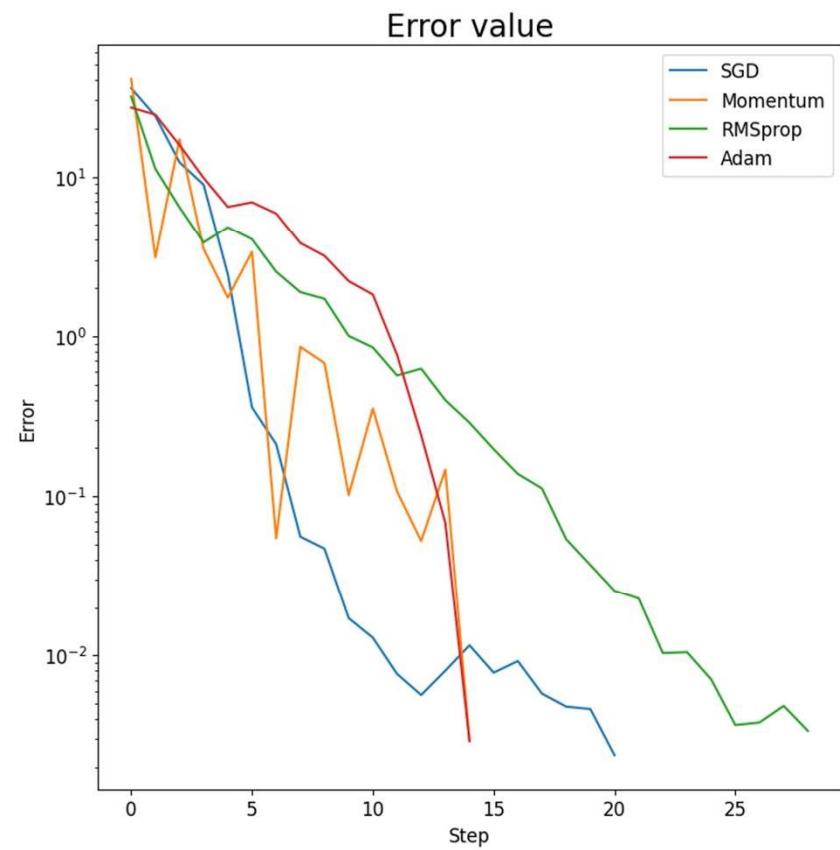
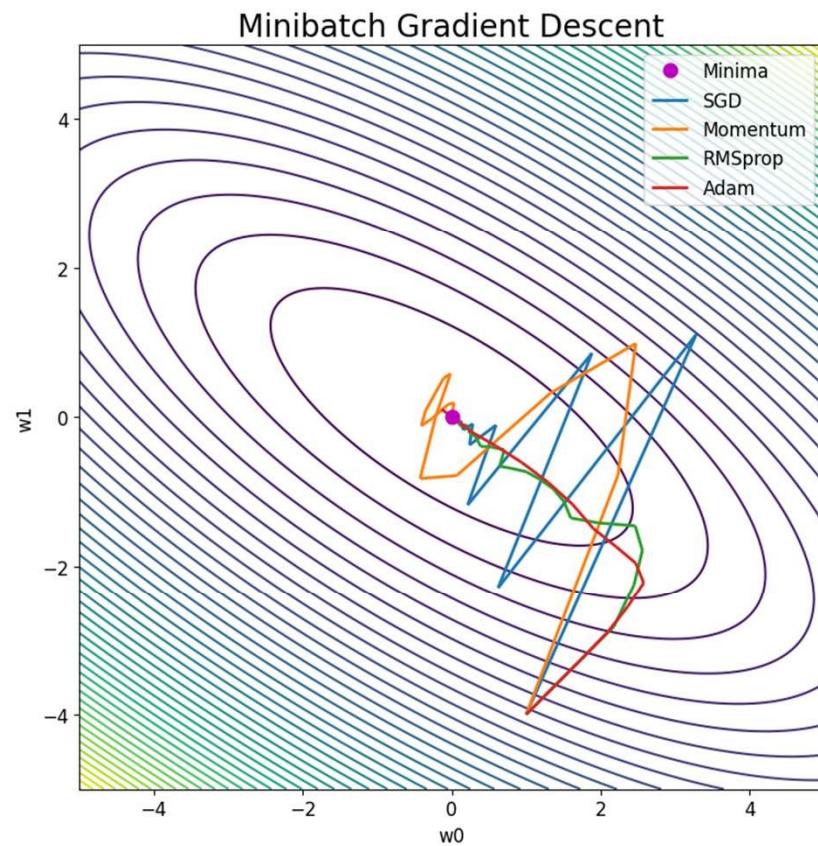
Thí nghiệm

Linear regression với tham số nonaligned



Thí nghiệm

Linear regression với tham số nonaligned



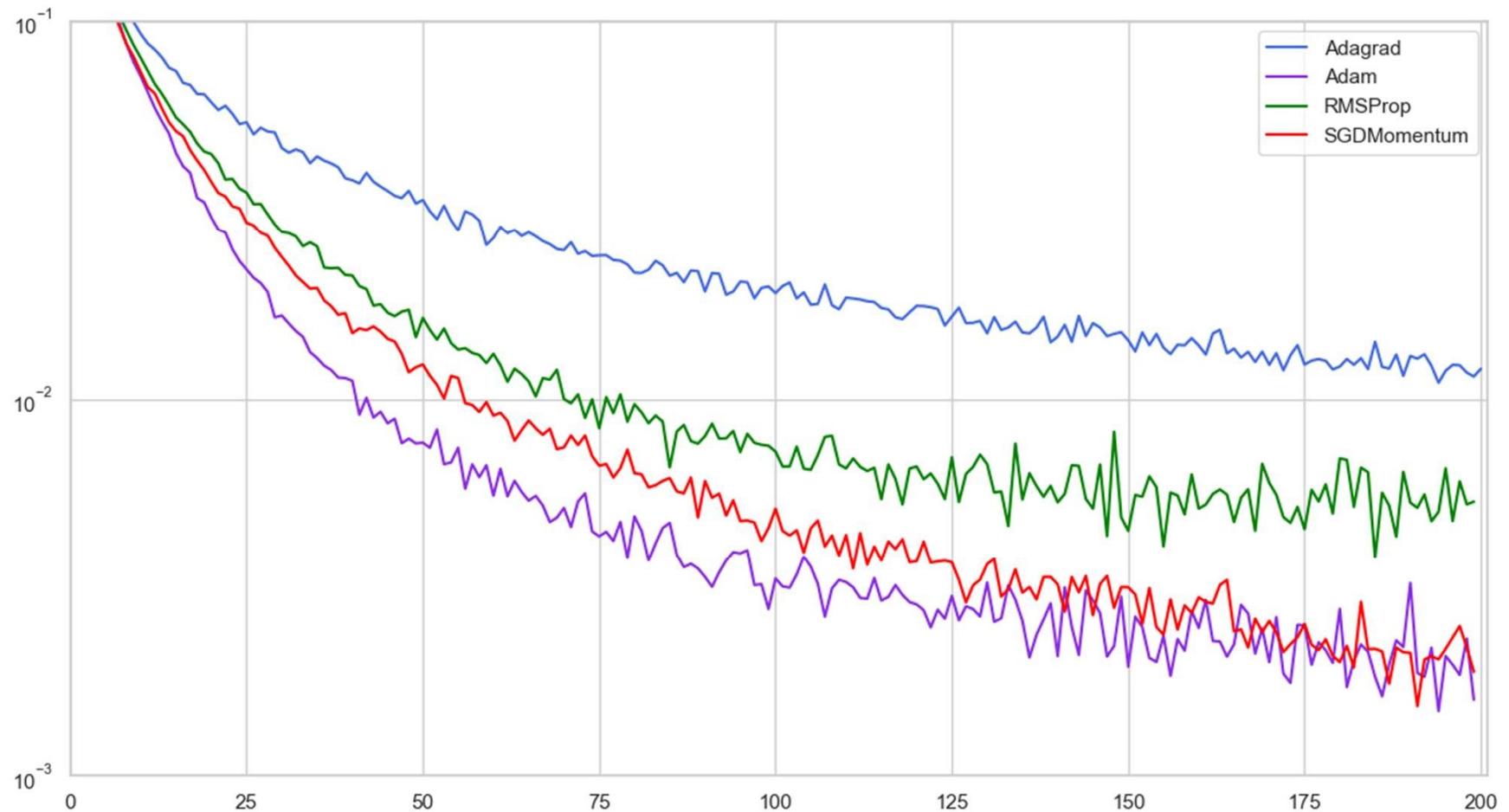
Thí nghiệm

Siêu tham số

	Learning Rate	Momentum/ Beta 1	Alpha/ Beta 2	Epsilon
SGDMomentum	0.01	0.9	-	-
AdaGrad	0.01	-	-	
RMSprop	0.0001	-	0.9	10^{-8}
Adam	0.0001	0.9	0.999	

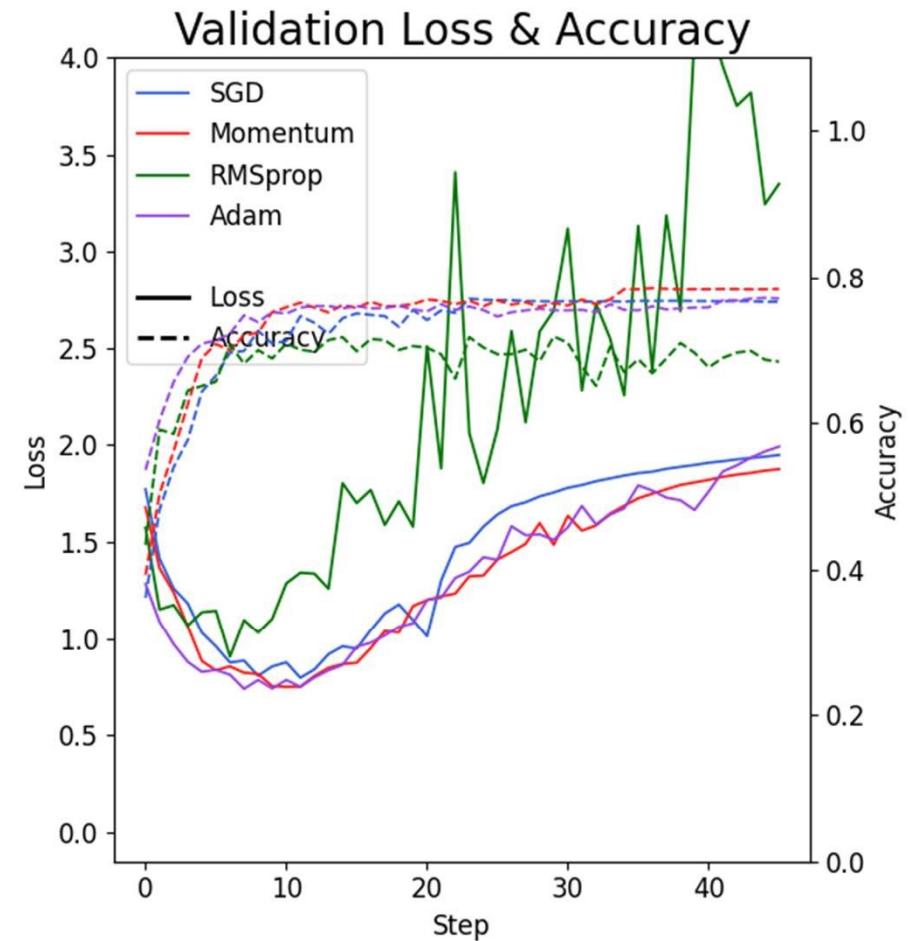
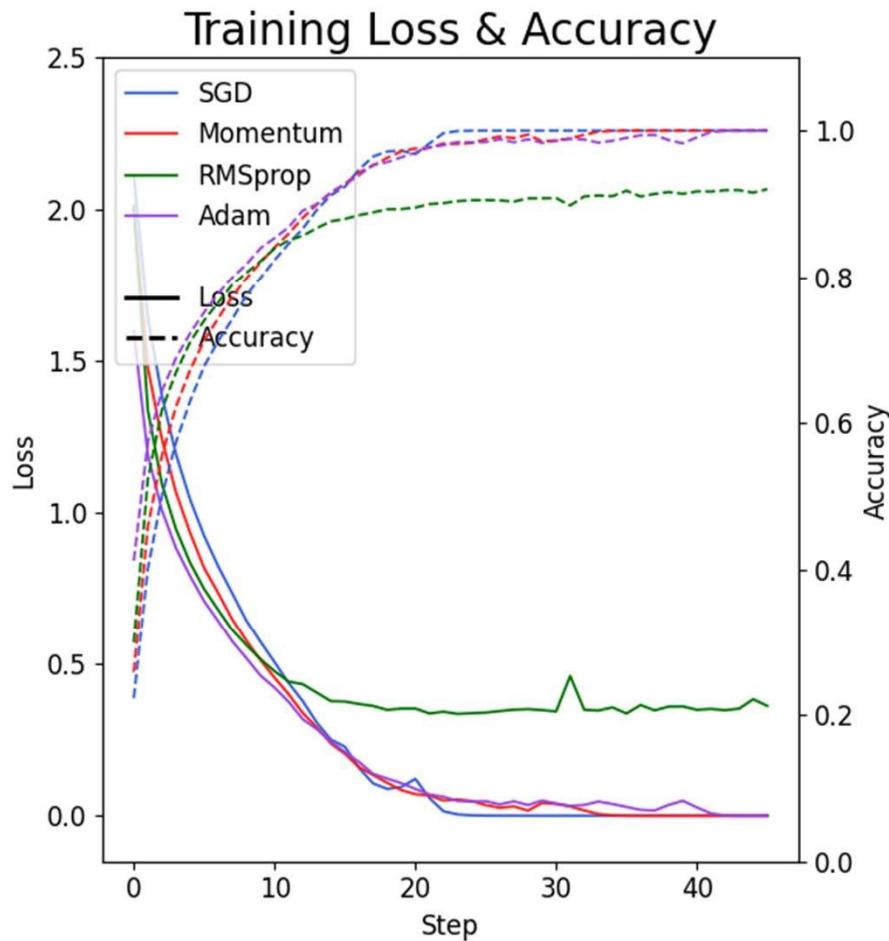
Thí nghiệm

1000fc-1000fc - MNIST



Thí nghiệm

c64-c64-c128-1000FC – CIFAR10



5.

Tổng kết

Tài liệu tham khảo

- Yann Dauphin *et al.*, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization", in *Advances in Neural Information Processing Systems 27*, 2014, pp. 2933-2941.
- Ning Qian, "On the momentum term in gradient descent learning algorithms", *Journal of the International Neural Network Society*, 1999, vol. 12, pp. 145-151.
- Nesterov, Y., "A method of solving a convex programming problem with convergence rate $O(1/\sqrt{k})$ ", *Soviet Mathematics Doklady*, 1983, vol. 27, pp. 372-376.
- J. C. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, 2011.

Tài liệu tham khảo

- T. Tieleman and G. Hinton, "Lecture 6.5 - rmsprop," *COURSERA: Neural Networks for Machine Learning*, 2012.
- D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference for Learning Representation*, vol. abs/1412.6980, 2015.