

# 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

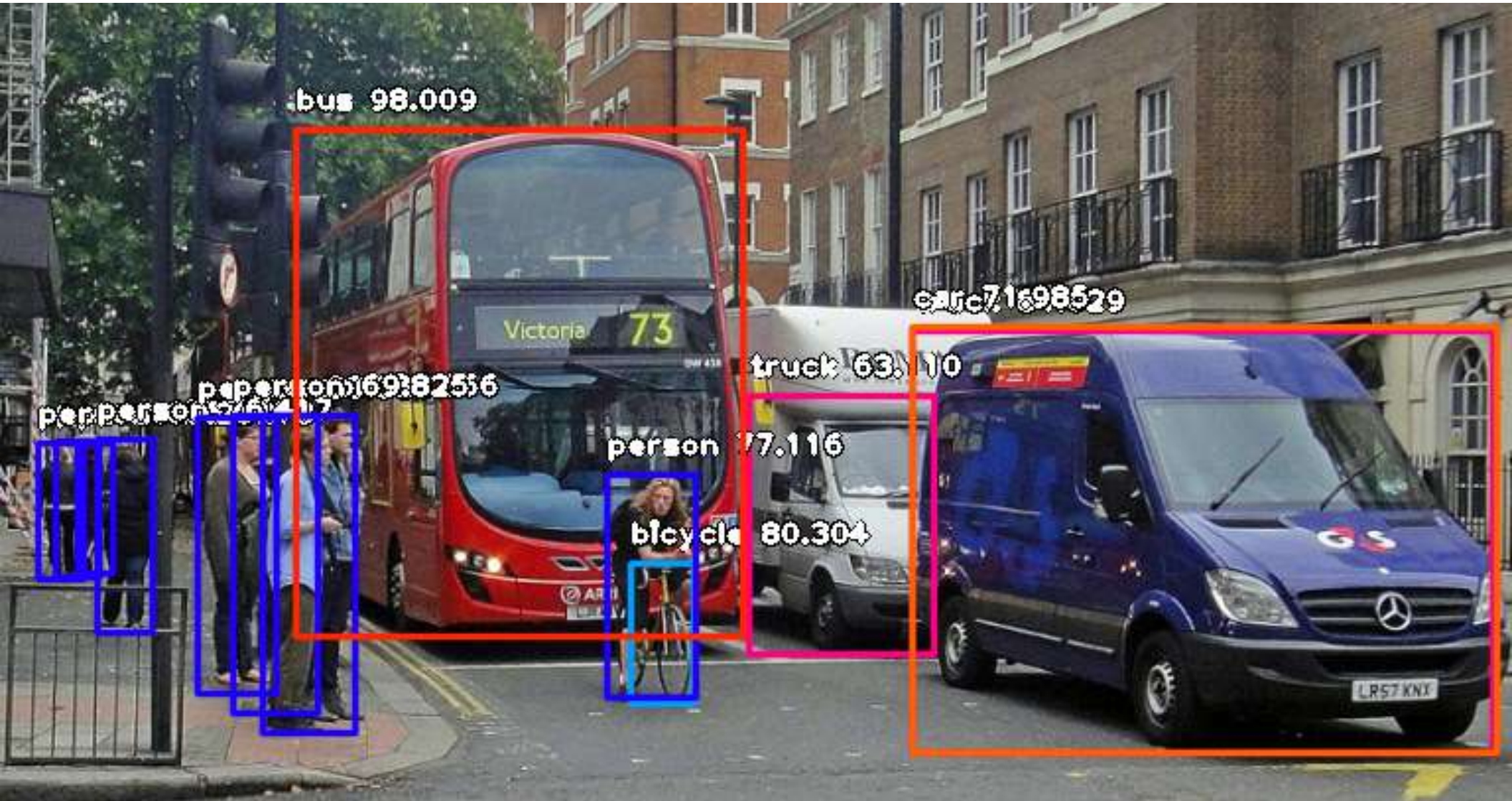
1. Giới thiệu đề tài
2. Kiến thức nền tảng
3. Thuật toán Adam
4. Thí nghiệm
5. Tổng kết

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

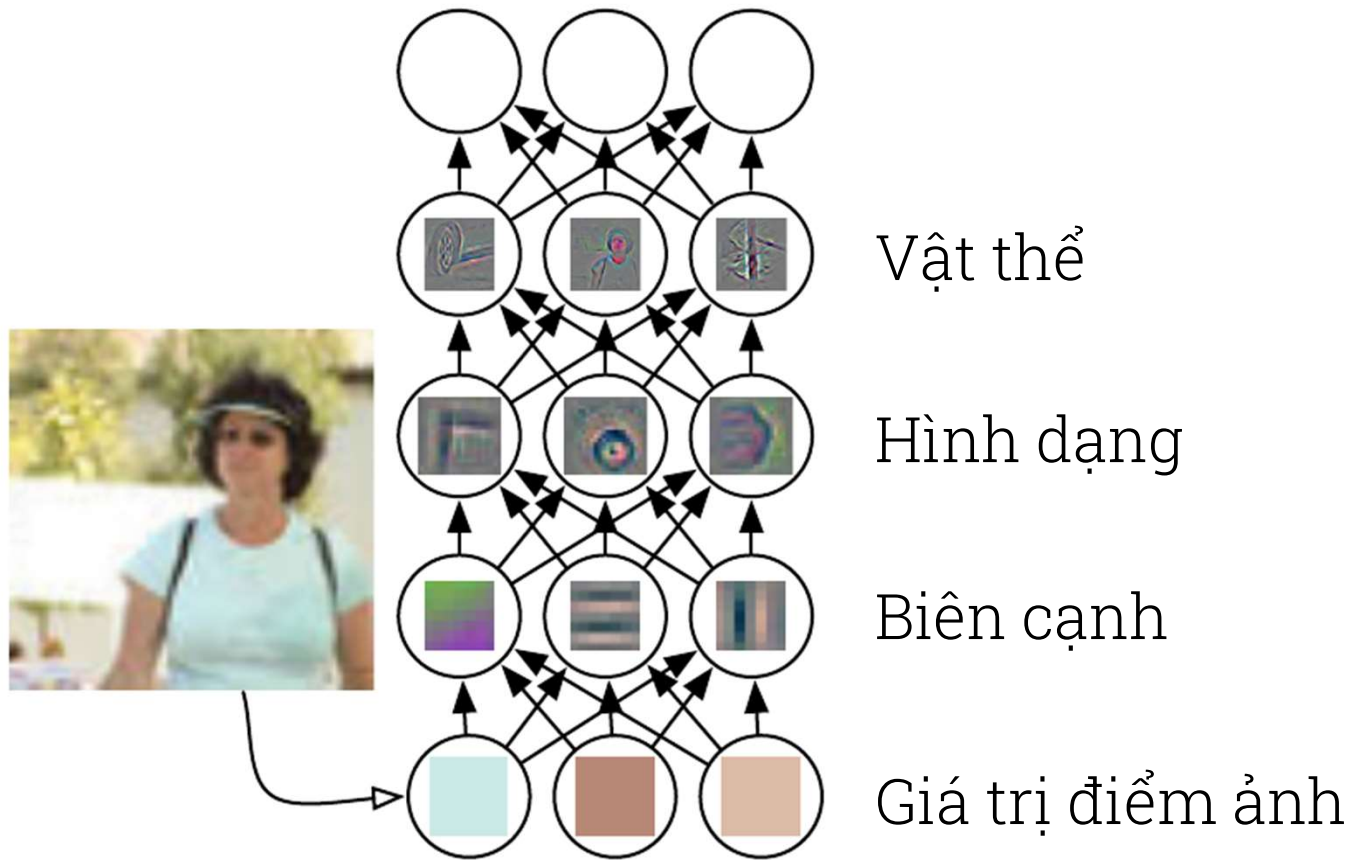
Tại sao lại sử dụng mạng nơ-ron nhiều tầng ẩn?





# 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

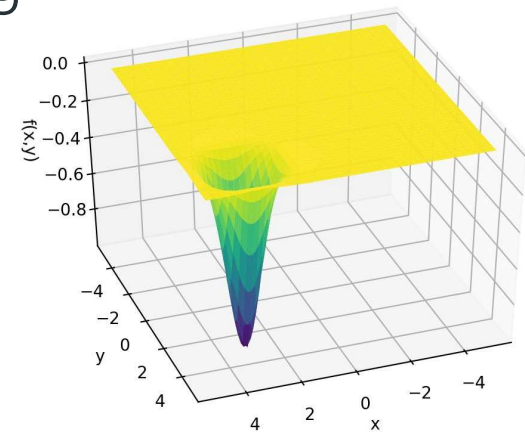
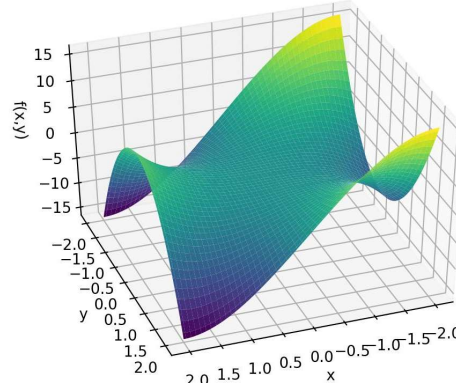
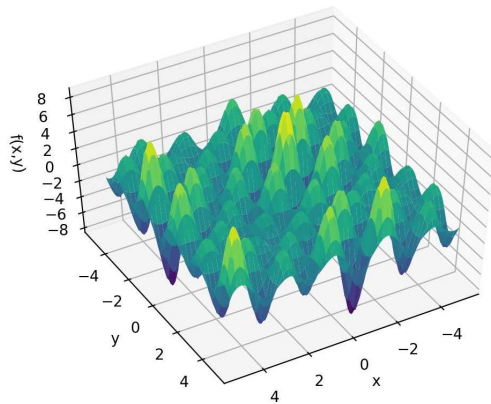
- **Input:** Hàm chi phí với tham số là các trọng số của mạng nơ-ron nhiều tầng ẩn. Hàm chi phí cho biết sự sai lệch giữa kết quả dự đoán của mạng nơ-ron so với giá trị đúng trên tập dữ liệu huấn luyện, hay *độ lỗi*.
- **Output:** Bộ trọng số của mạng nơ-ron nhiều tầng ẩn cho độ lỗi là nhỏ nhất, hoặc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



# Giới thiệu đề tài

## Thách thức

- Mặt phẳng lỗi phức tạp:
  - Nhiều cực tiểu địa phương.
  - Nhiều điểm yên ngựa.
  - Nhiều vùng bằng phẳng.
  - Mỗi hướng có đặc tính riêng.



\*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.

# Giới thiệu đề tài

## Đề tài liên quan

- Hướng tiếp cận truyền thống: **Stochastic Gradient Descent** (SGD)\*
  - Sử dụng đạo hàm bậc nhất để xác định hướng đi có sự thay đổi lớn nhất.
  - Sử dụng **tỉ lệ học** cố định.

\*Kiefer, J. and Wolfowitz, J., "Stochastic Estimation of the Maximum of a Regression Function", *Annals of Mathematical Statistics* 23, 1952, pp. 462-466.

# Giới thiệu đề tài

Đề tài liên quan: SGD

- Tạo ra sự ngẫu nhiên (stochasticity) giúp vượt qua cực tiểu địa phương có độ lỗi cao.
- Không bị mắc kẹt tại điểm yên ngựa.
- Di chuyển chậm khi gặp vùng bằng phẳng.

# Giới thiệu đề tài

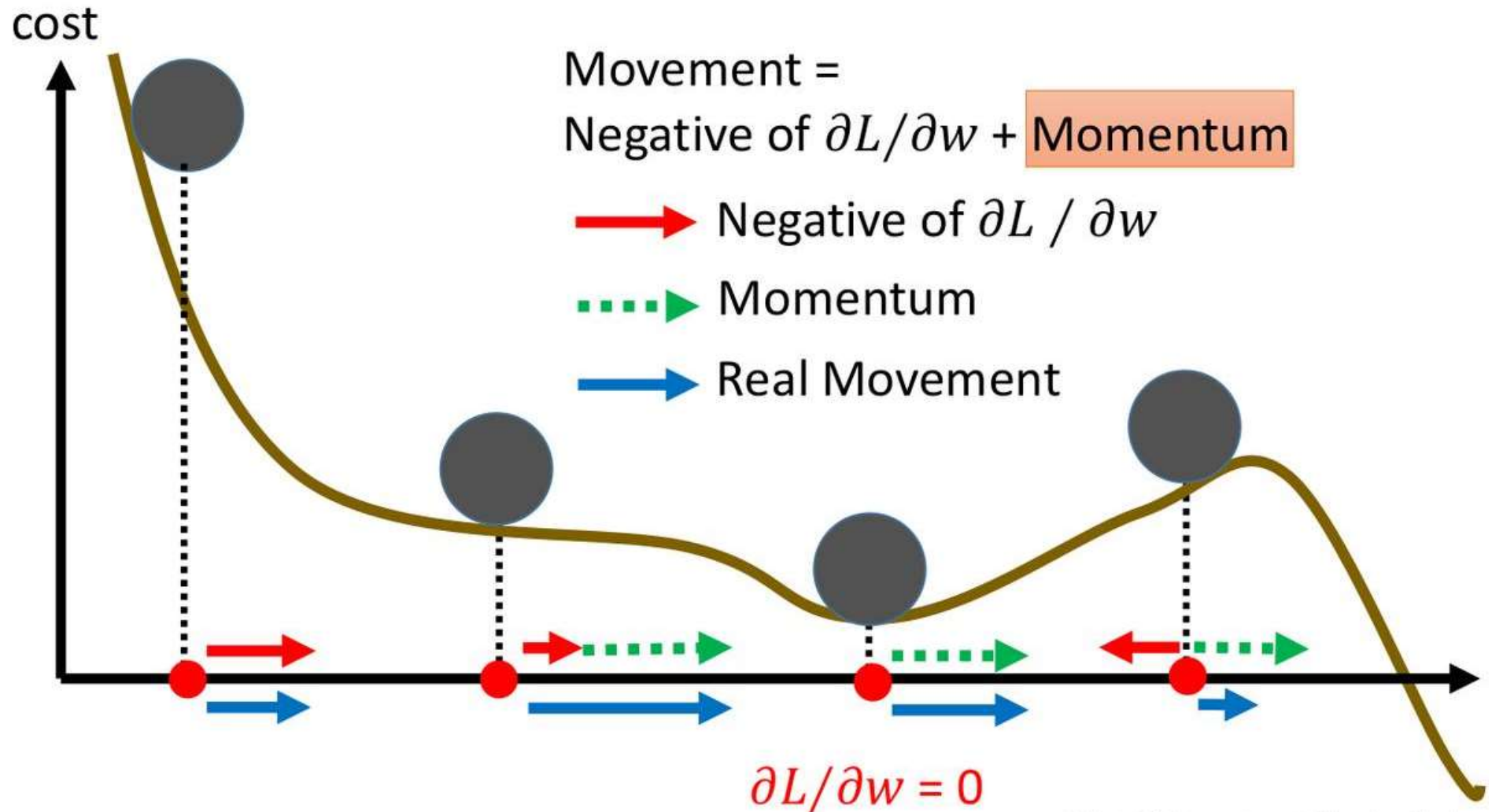
## Đề tài liên quan

- Cải tiến: Stochastic Gradient Descent **with Momentum** (Momentum)\*
  - Áp dụng nguyên lý lực quán tính.
  - Di chuyển nhanh hơn bằng cách tăng độ lớn cập nhật khi chiều gradient không đổi.

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

# Giới thiệu đề tài

Đề tài liên quan: Momentum

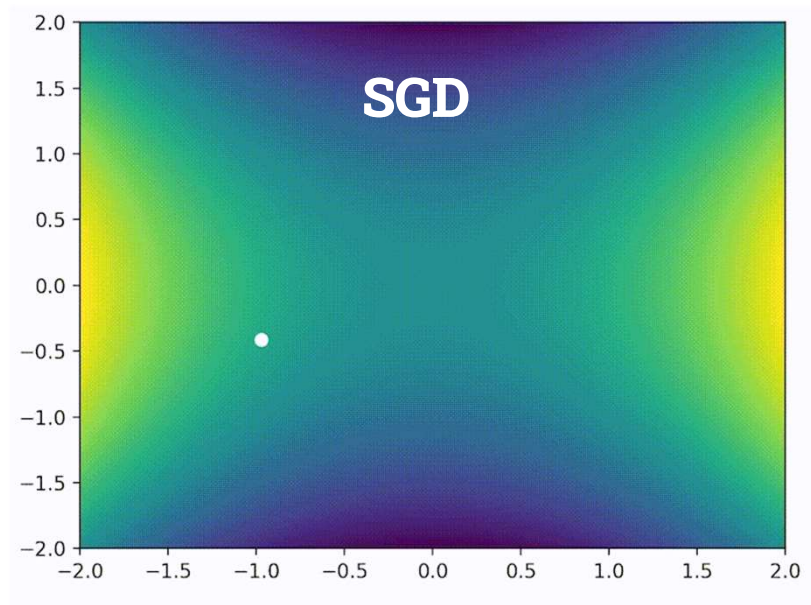
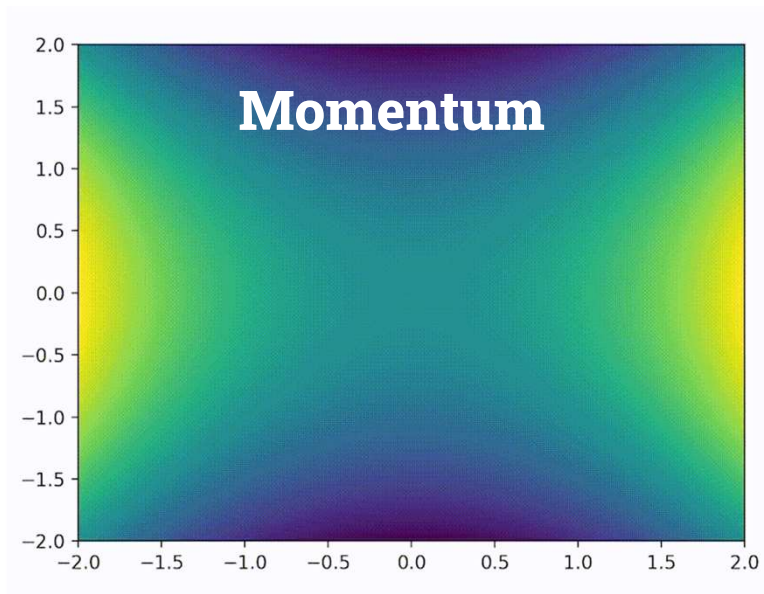


<http://blog.csdn.net/jiandanjinxin>

# Giới thiệu đề tài

## Đề tài liên quan: Momentum

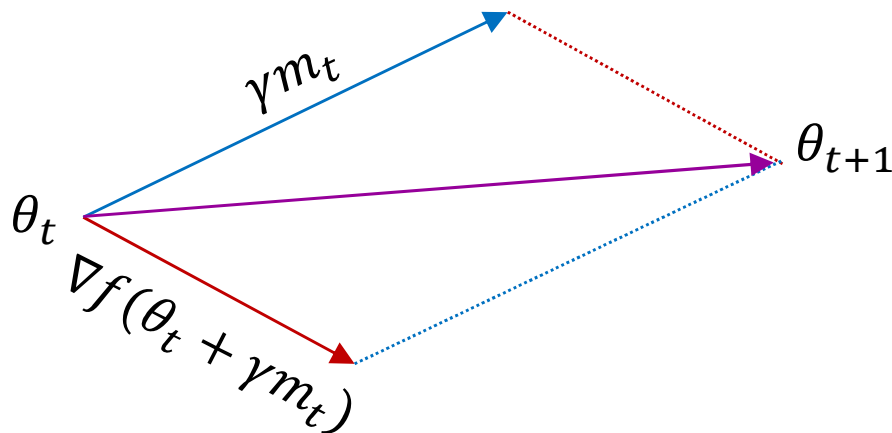
- Có thể đi vượt qua các điểm cực tiểu.
- Vượt qua điểm yên ngựa.
- Di chuyển nhanh hơn trong vùng bằng phẳng.



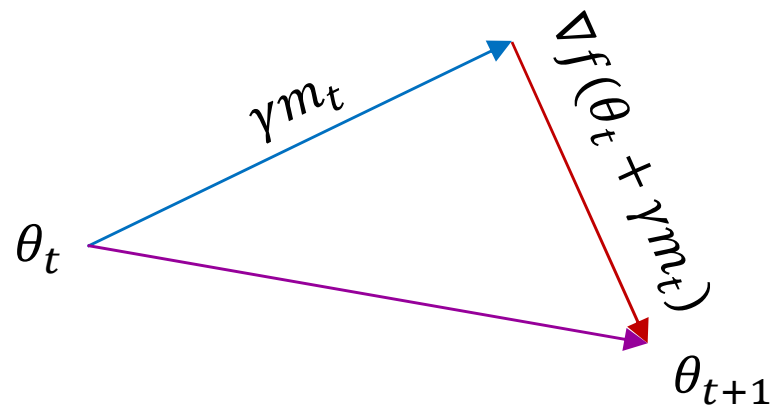
# Giới thiệu đề tài

## Đề tài liên quan

- Cải tiến: [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.



# Giới thiệu đề tài

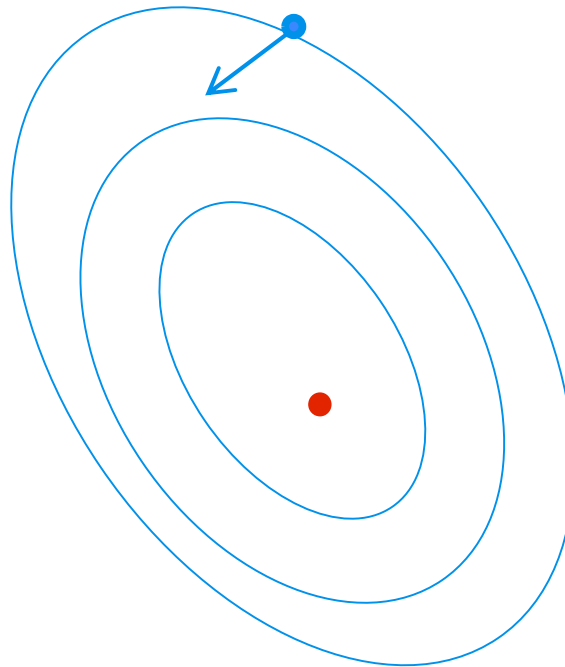
Đề tài liên quan: NAG

- Hạn chế đi vượt qua các điểm cực tiểu.
- Vượt qua các điểm yên ngựa.
- Di chuyển nhanh và ổn định trong vùng bằng phẳng.

# Giới thiệu đề tài

Đề tài liên quan: Hạn chế

- Các thuật toán trên có tốc độ hội tụ càng giảm khi số chiều của mặt phẳng lỗi càng tăng.
- Hướng của gradient trên tập con không trùng với hướng của gradient trên cả tập dữ liệu.



# Giới thiệu đề tài

## Đề tài liên quan: Hạn chế

- Các thuật toán trên có tốc độ hội tụ càng giảm khi số chiều của mặt phẳng lỗi càng tăng.
- Hướng của gradient trên tập con không trùng với hướng của gradient trên cả tập dữ liệu.
- Tỉ lệ học cố định không phù hợp cho tất cả các hướng.

→ **Cần một tỉ lệ học phù hợp cho từng hướng.**

→ **Adaptive learning rate**

# 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 nhanh về điểm cực tiểu có độ lỗi thấp.
  - Đảm bảo vượt qua các điểm yên ngựa.
  - Vượt qua vùng bằng phẳng nhanh hơn.

**2.**

## **Kiến thức nền tảng**

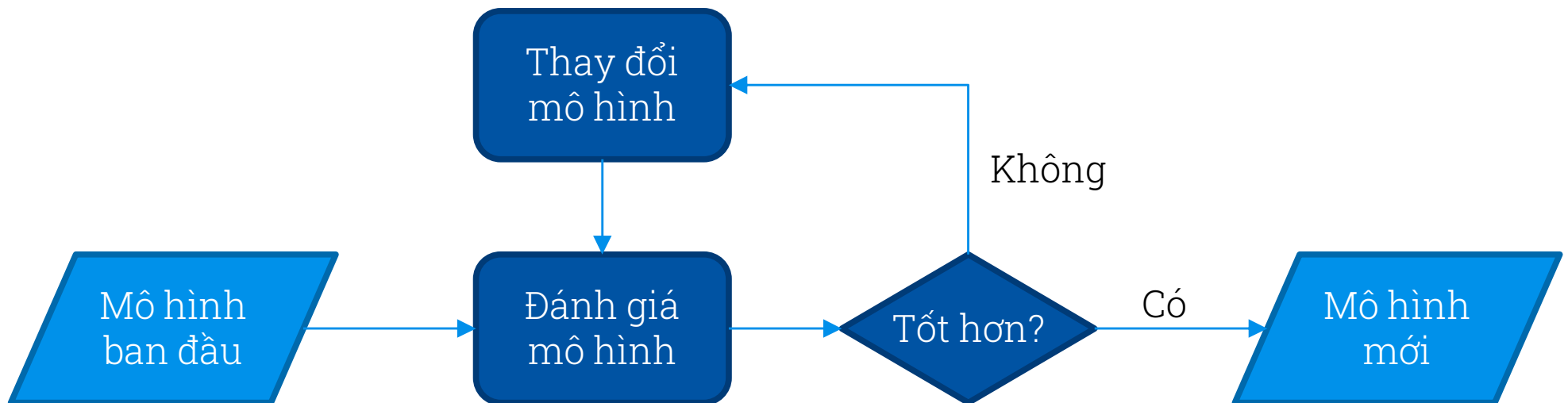
# Kiến thức nền tảng

## Tối ưu hóa (optimization)

- **Input:** một ánh xạ  $\mathcal{F}: A \rightarrow \mathbb{R}$  với  $A$  là tập bất kì.
- **Output:** một phần tử  $x_0 \in A$  sao cho
  - $\mathcal{F}(x_0) \leq \mathcal{F}(x) \forall x \in A$  (cực tiểu hóa)
  - $\mathcal{F}(x_0) \geq \mathcal{F}(x) \forall x \in A$  (cực đại hóa).

# Kiến thức nền tảng

Tối ưu hóa (optimization)





# Kiến thức nền tảng

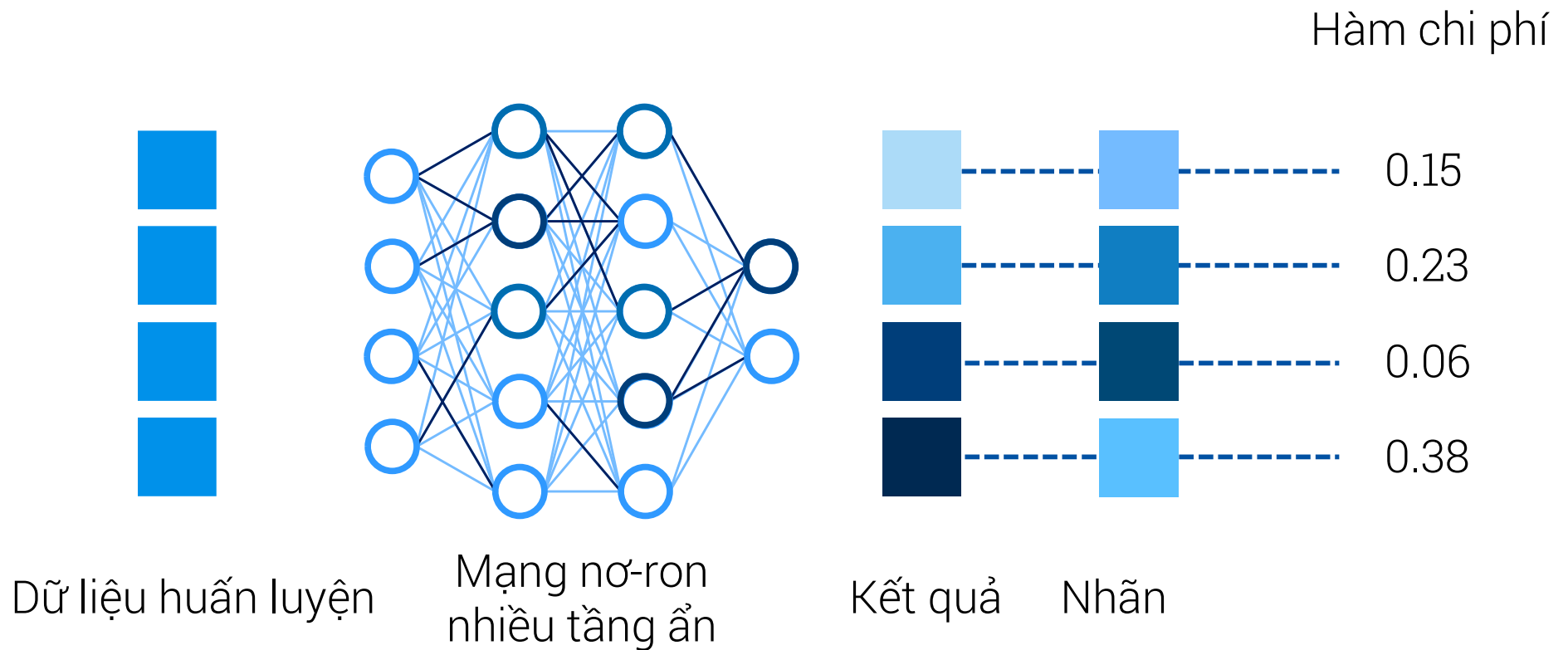
## Tối ưu hóa (optimization)

- Khi tối ưu hóa mạng nơ-ron nhiều tầng ẩn, chúng ta không tối ưu trực tiếp trên mô hình của mạng.
- Thay vào đó, chúng ta tối ưu hóa gián tiếp thông qua hàm lỗi.

$$\operatorname{argmin}_{\theta} \mathcal{L}(\mathcal{F}_{\theta}, y)$$

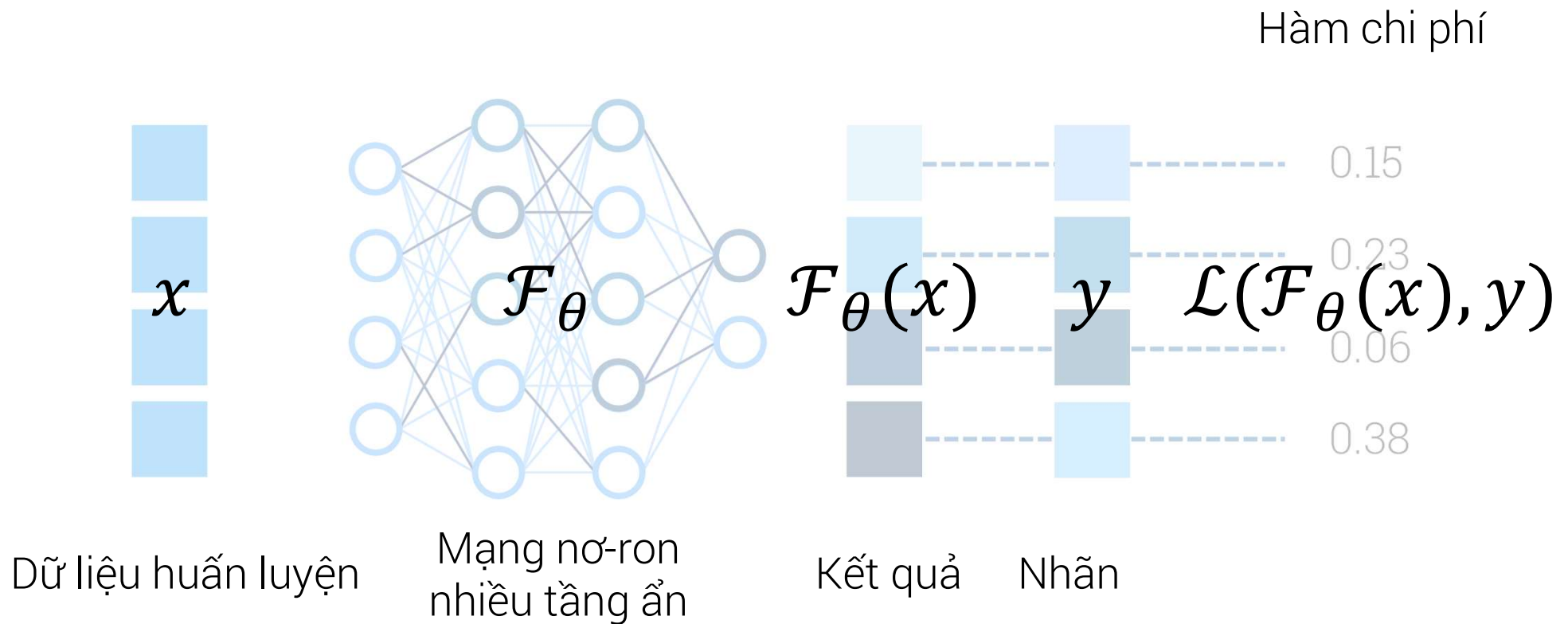
# Kiến trúc nền tảng

Tối ưu hóa (optimization)



# Kiến trúc nền tảng

Tối ưu hóa (optimization)



# Kiến thức nền tảng

## AdaGrad

- Tính đạo hàm theo từng tham số.

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

- Cập nhật  $G$  ở bước hiện tại.

$$G_t = G_{t-1} + \text{diag}(g_t \cdot g_t^T)$$

- Cập nhật trọng số.

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

# Kiến trúc nền tảng

## AdaGrad

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

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

# Kiến thức nền tảng

## AdaGrad

- Tính đạo hàm theo từng tham số.

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

- Cập nhật  $G$  ở bước hiện tại.

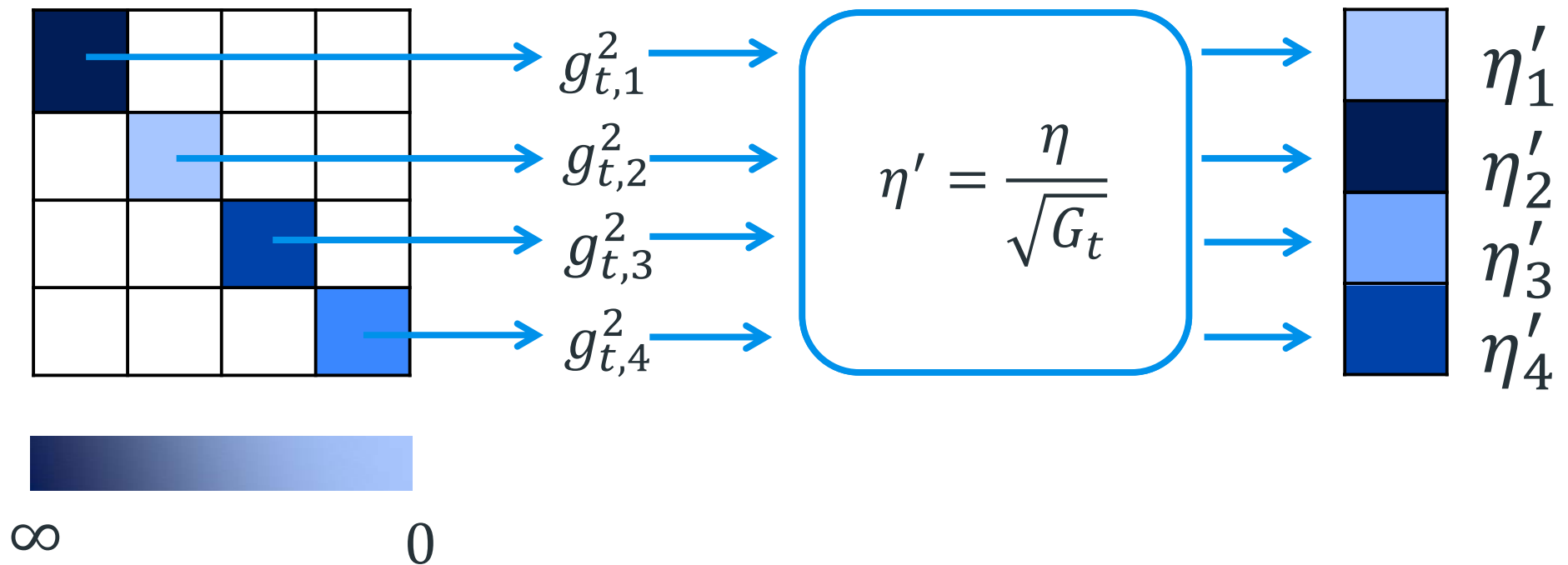
$$G_t = G_{t-1} + \mathbf{g}_t^2$$

- Cập nhật trọng số.

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

# Kiến trúc nền tảng

## AdaGrad





# Kiến thức nền tảng

## AdaGrad và RMSprop

### AdaGrad

- $g_t = \nabla_{\theta} \mathcal{L}(\theta_t)$
- $G_t = G_{t-1} + g_t^2$
- $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot g_t$

### RMSprop

- $g_t = \nabla_{\theta} \mathcal{L}(\theta_t)$
- $G_t = \gamma G_{t-1} + (1 - \gamma) g_t^2$
- $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot g_t$

**3.**

# **Thuật toán Adam**

# Thuật toán Adam

## Pseudo-code

---

**Algorithm 1:** *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation.  $g_t^2$  indicates the elementwise square  $g_t \odot g_t$ . Good default settings for the tested machine learning problems are  $\alpha = 0.001$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$  and  $\epsilon = 10^{-8}$ . All operations on vectors are element-wise. With  $\beta_1^t$  and  $\beta_2^t$  we denote  $\beta_1$  and  $\beta_2$  to the power  $t$ .

---

**Require:**  $\alpha$ : Stepsize

**Require:**  $\beta_1, \beta_2 \in [0, 1)$ : Exponential decay rates for the moment estimates

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

**Require:**  $\theta_0$ : Initial parameter vector

$m_0 \leftarrow 0$  (Initialize 1<sup>st</sup> moment vector)

$v_0 \leftarrow 0$  (Initialize 2<sup>nd</sup> moment vector)

$t \leftarrow 0$  (Initialize timestep)

**while**  $\theta_t$  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)

$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$  (Update biased second raw moment estimate)

$\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$  (Compute bias-corrected first moment estimate)

$\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$  (Compute bias-corrected second raw moment estimate)

$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$  (Update parameters)

**end while**

**return**  $\theta_t$  (Resulting parameters)

---

# Thuật toán Adam

Siêu tham số

$\alpha$ : tỉ lệ học

$\beta_1, \beta_2$ : tỉ lệ suy biến của trung bình đạo hàm và bình phương đạo hàm (mặc định lần lượt là 0.9 và 0.999)

$\epsilon$ : hệ số nhỏ

# Thuật toán Adam

Khởi tạo

$\mathbf{m}_0 = \mathbf{0}$ : khởi tạo trung bình đạo hàm

$\mathbf{v}_0 = \mathbf{0}$ : khởi tạo trung bình bình phương đạo hàm

$t = 0$ : khởi tạo bước chạy

# Thuật toán Adam

## Các bước thực hiện

Tăng bước chạy  $t$

$$t = t + 1$$

Tính đạo hàm của hàm chi phí trên từng tham số

$$g_t = \nabla_{\theta} f_t(\theta_{t-1})$$

# Thuật toán Adam

## Các bước thực hiện

Cập nhật trung bình đạo hàm  $m_t$  và trung bình bình phương đạo hàm  $v_t$

$$\begin{aligned}m_t &= \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \\v_t &= \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2\end{aligned}$$

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

$$\begin{aligned}\widehat{m}_t &= m_t / (1 - \beta_1^t) \\ \widehat{v}_t &= v_t / (1 - \beta_2^t)\end{aligned}$$

Cập nhật **trọng số**

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



**4.**

**Thí nghiệm**

# 5.

## Tổng kết