

# Hyperparameter tuning

An approach to solve problems but with better solution

Group 5

# Introduction

# What is hyperparameter

A hyperparameter is a configuration variable that is set before the training process and not updated during learning.

Examples:

- Learning rate (learning rate)
- Batch size (batch size)
- Optimizer (optimizer)
- Number of hidden layers
- Dropout rate

# Why Hyperparameter Optimization (HPO)

**Deep neural networks are powerful but hard to train and tune.**

- Hyperparameters have major impacts on accuracy and efficiency while training the model

**Incorrect hyperparameters can lead to poor accuracy or training failure.**

- Therefore it needed to be set accurately to get better and efficient results

**=> HPO aims to automate this process, increasing efficiency and reproducibility.**

**Major hyperparameters**  
**(Categories of**  
**hyperparameter)**



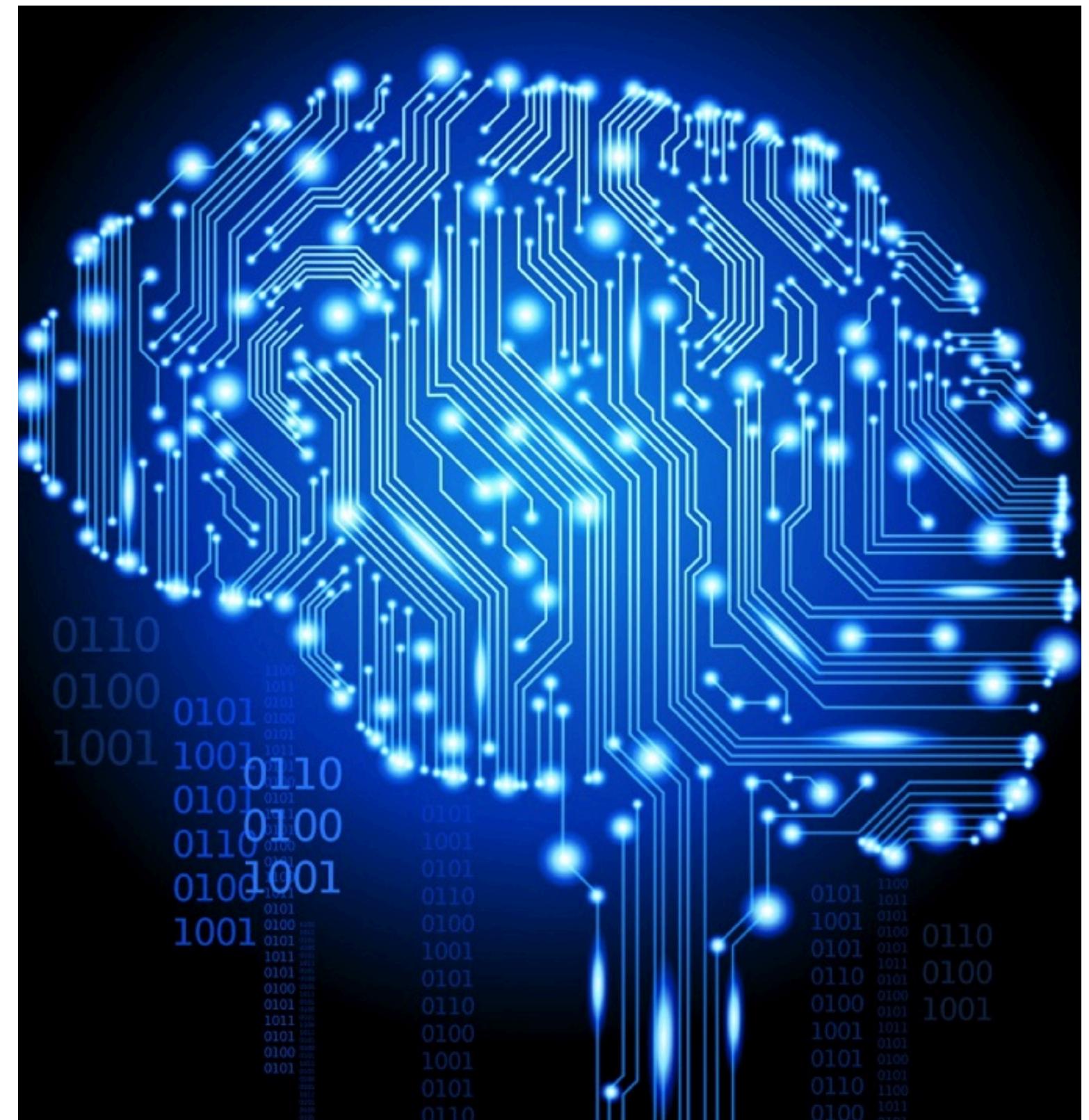


Training-related  
hyperparameters

# Affect the learning dynamics and convergence speed.

- Learning rate
- Batch size
- Optimizer & momentum

# Model structure-related hyperparameters



# Define the structure and capacity of the model.

- Number of layers
- Width of layers
- Type of activation function

# Optimizer

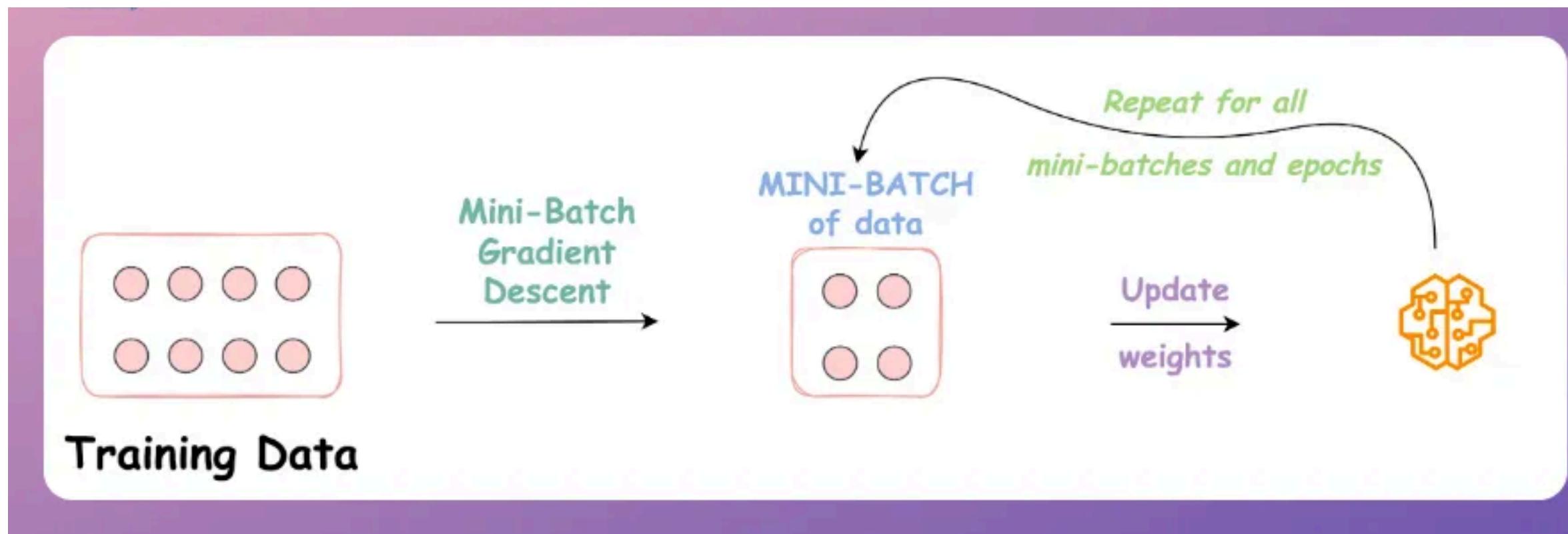
Là thuật toán được sử dụng để cập nhật weight trong quá trình huấn luyện giúp giảm loss

Ví dụ: SGD, Momentum, Adam, RMSProp, ...

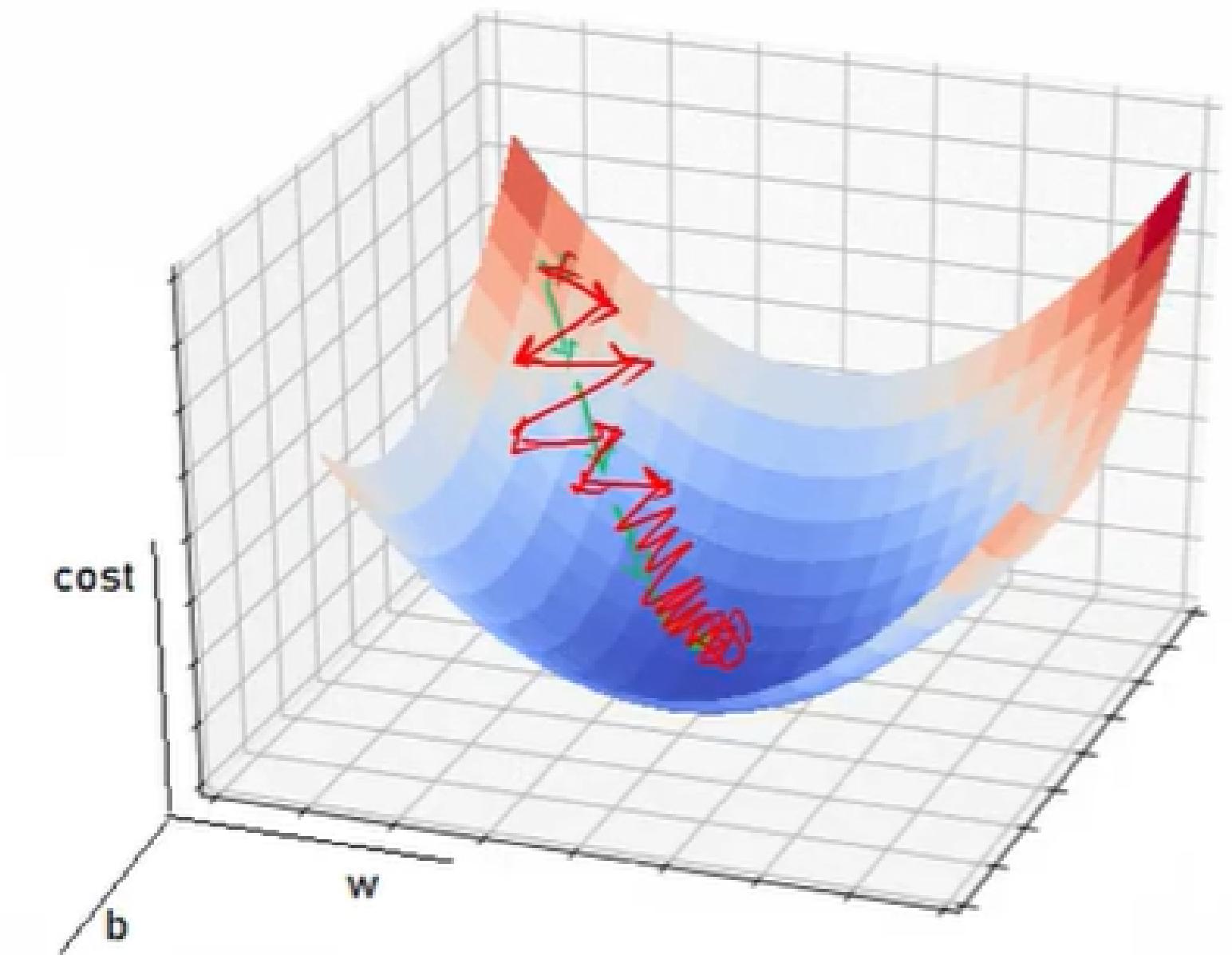
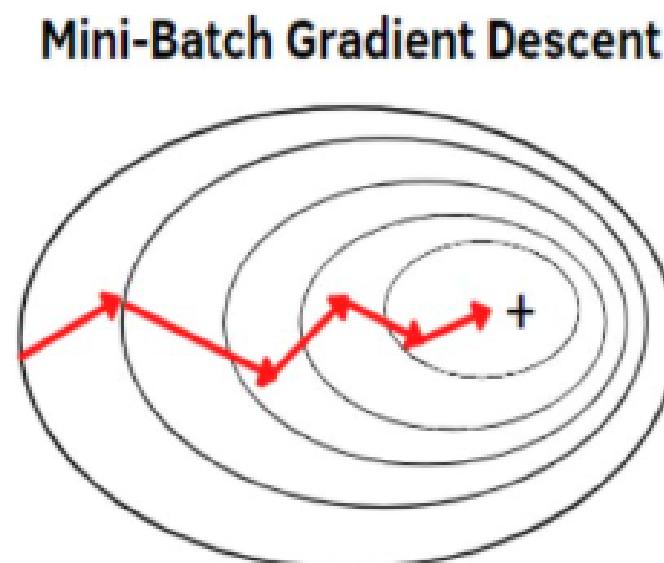
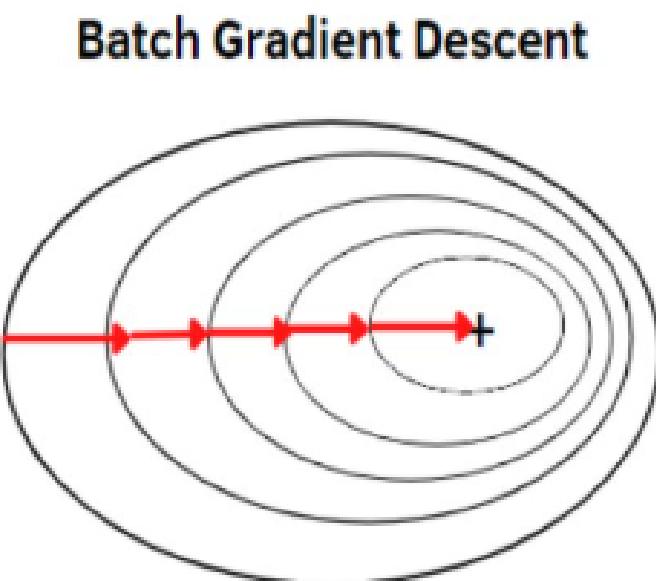
# Mini - batch gradient descent

$$g_t = \frac{1}{m} \sum_{i \in \mathcal{B}_t} \nabla \ell(f(x_i; w), y_i)$$

- $m$ : số lượng mẫu trong mini-batch
- $\mathcal{B}_t$ : mini-batch tại bước  $t$
- $\ell$ : loss trên từng mẫu



# Mini - batch gradient descent

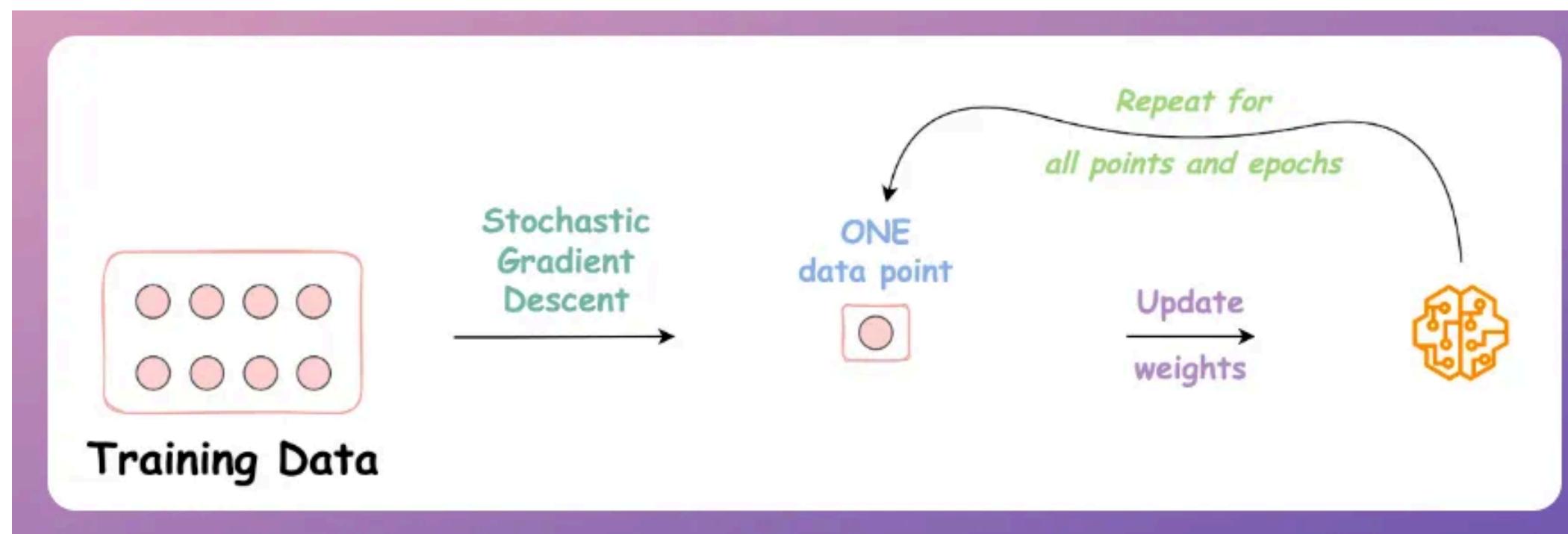


# Stochastic Gradient Descent

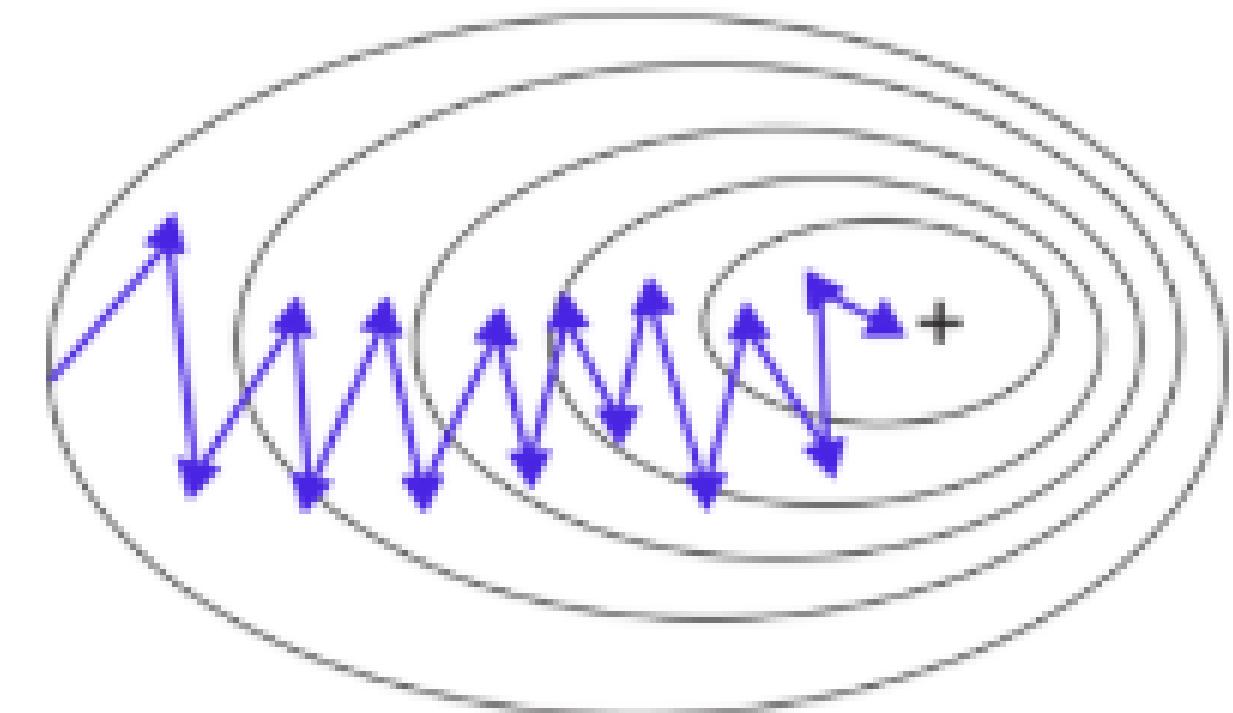
$$w := w - \eta \cdot \nabla \ell(f(x_i; w), y_i)$$

Trong đó:

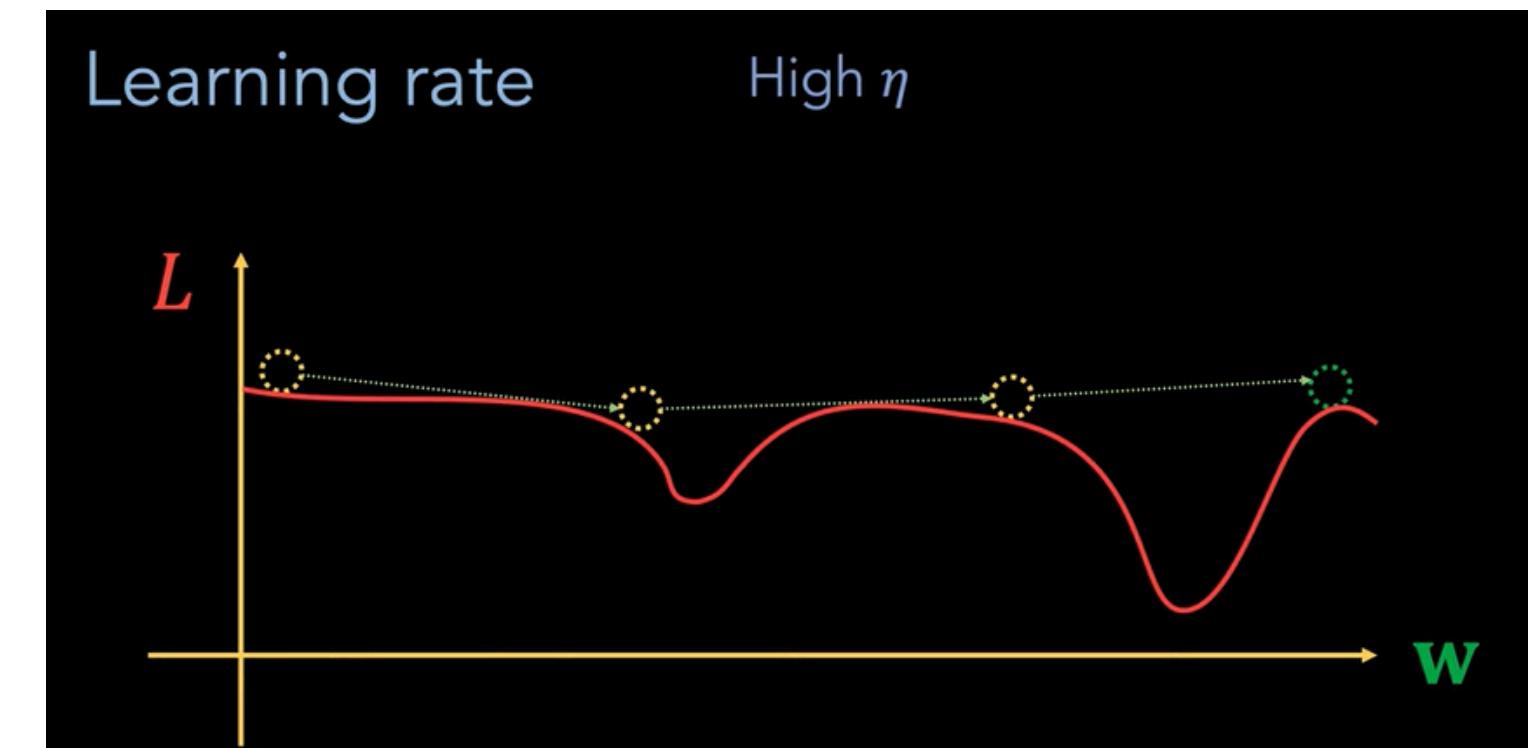
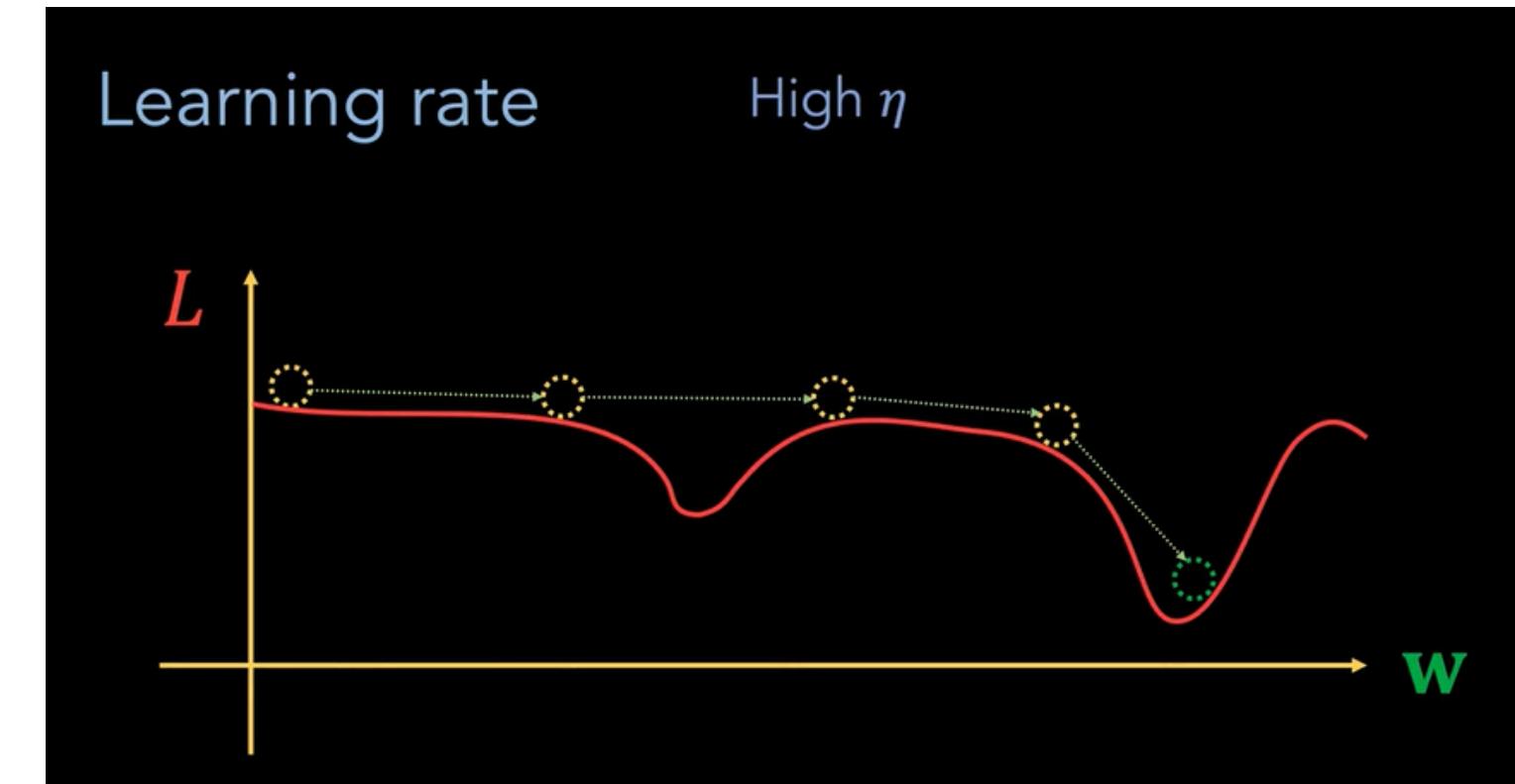
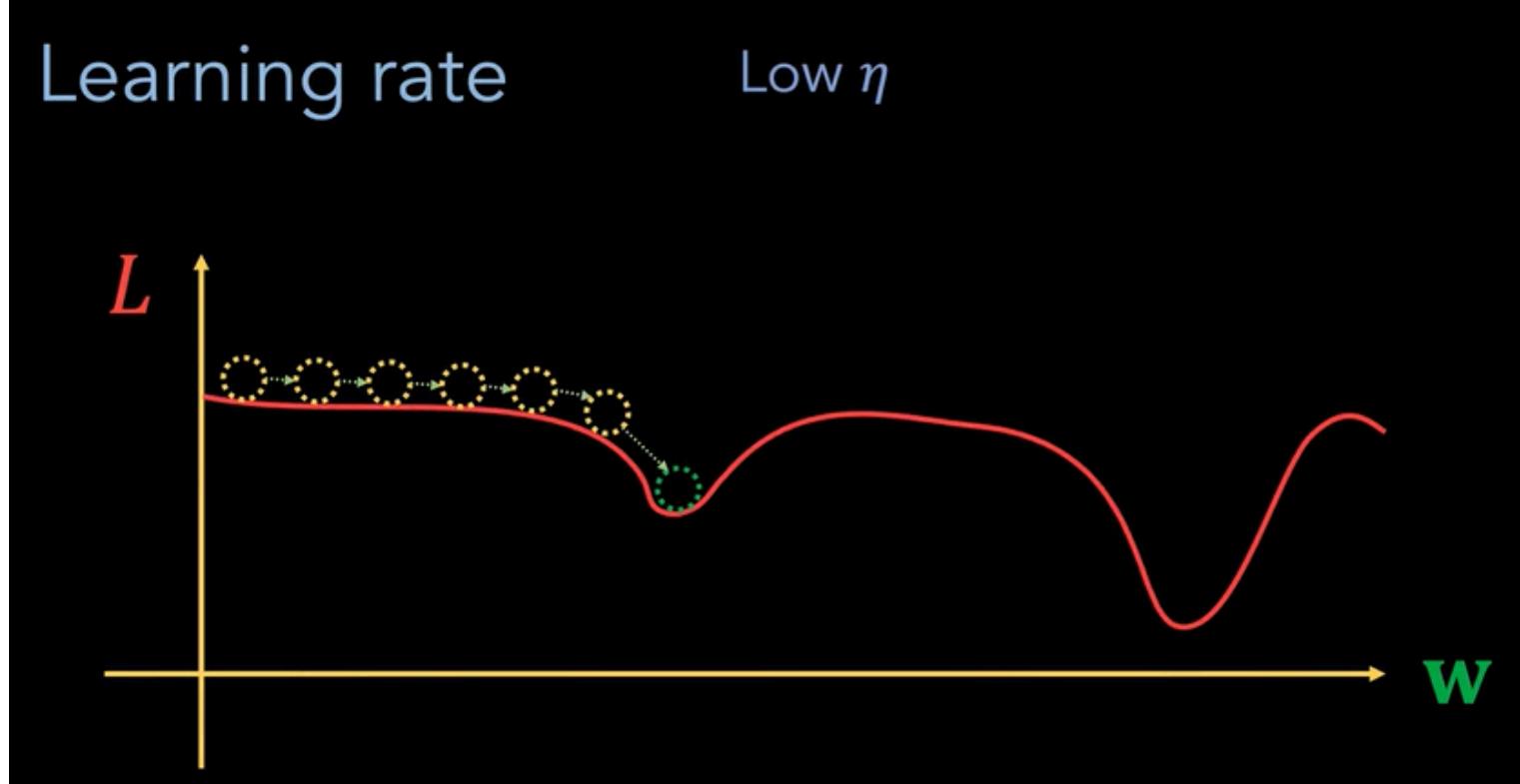
- $(x_i, y_i)$  là một mẫu duy nhất được chọn ngẫu nhiên từ tập huấn luyện.
- $\ell$ : hàm mất mát cho một mẫu.
- $\eta$ : learning rate.



**Stochastic Gradient Descent**



# Stochastic Gradient Descent



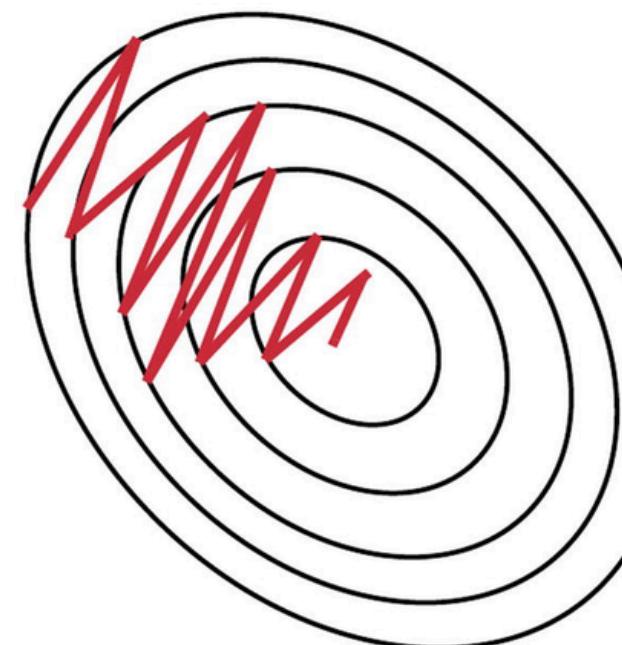
# Stochastic Gradient Descent with Momentum

$$v_t = \beta v_{t-1} + (1 - \beta) \nabla L(w_t)$$

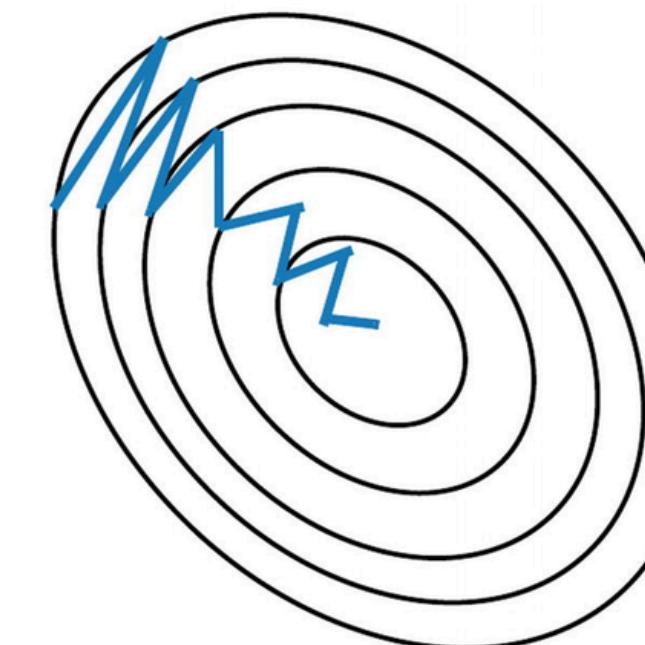
$$w_{t+1} = w_t - \eta \cdot v_t$$

Trong đó:

- $\beta \in [0, 1]$ : hệ số momentum (thường 0.9)
- $v_t$ : vận tốc cập nhật tích lũy theo thời gian

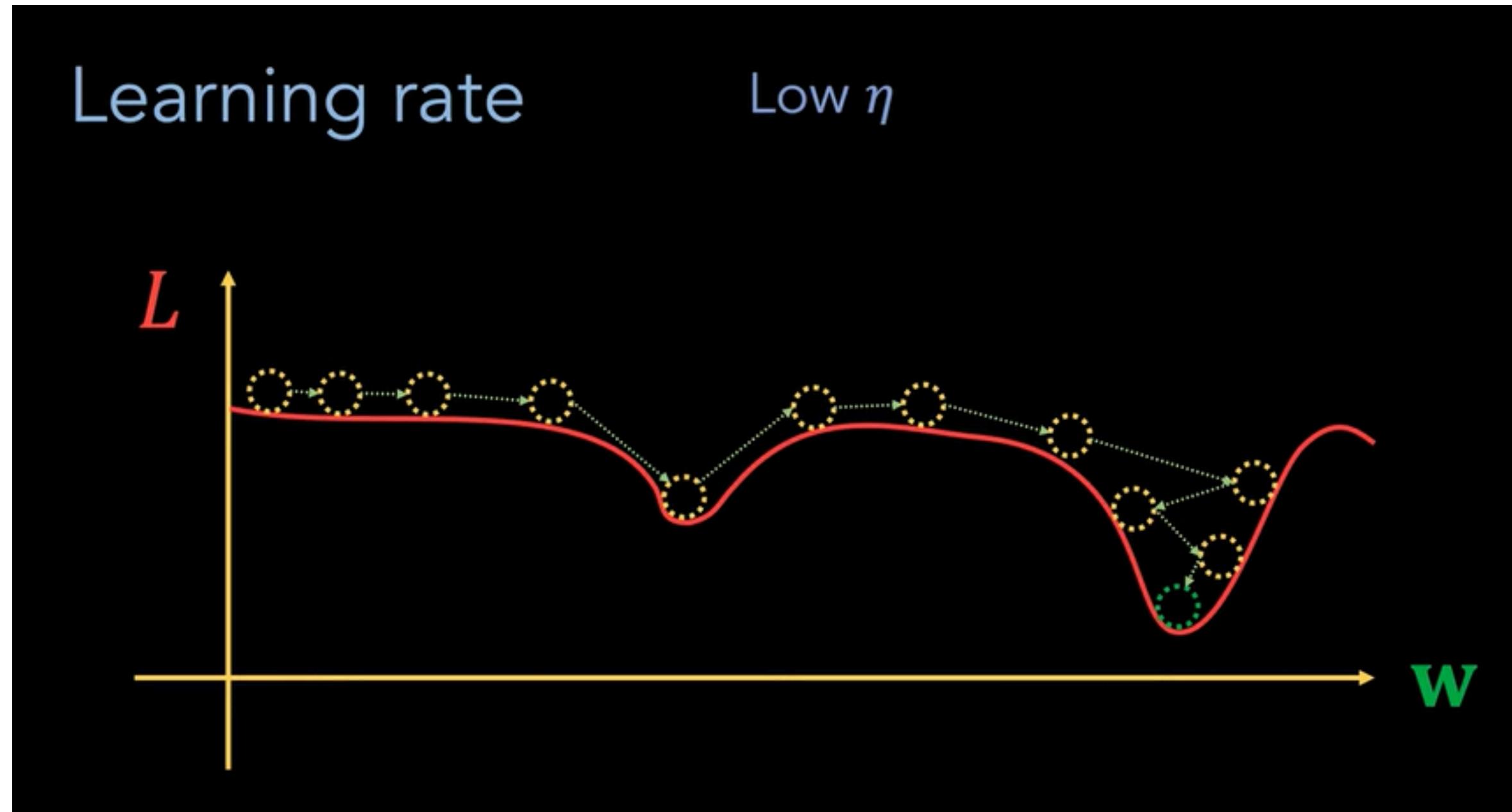


Stochastic Gradient  
Descent **without**  
Momentum



Stochastic Gradient  
Descent **with**  
Momentum

# Stochastic Gradient Descent with Momentum



# Adagrad (Adaptive Gradient)

Adagrad tự động điều chỉnh learning rate riêng cho từng feature, dựa trên tổng bình phương gradient tích lũy theo thời gian:

$$w_i := w_i - \frac{\eta}{\sqrt{G_{i,t}} + \epsilon} \cdot \nabla_{w_i} L(w)$$

- $G_{i,t} = \sum_{k=1}^t (\nabla_{w_i} L_k)^2$ : tổng gradient bình phương cho feature  $i$
- $\eta$ : learning rate ban đầu
- $\epsilon$ : số nhỏ để tránh chia cho 0

# RMSProp (Root Mean Square Propagation)

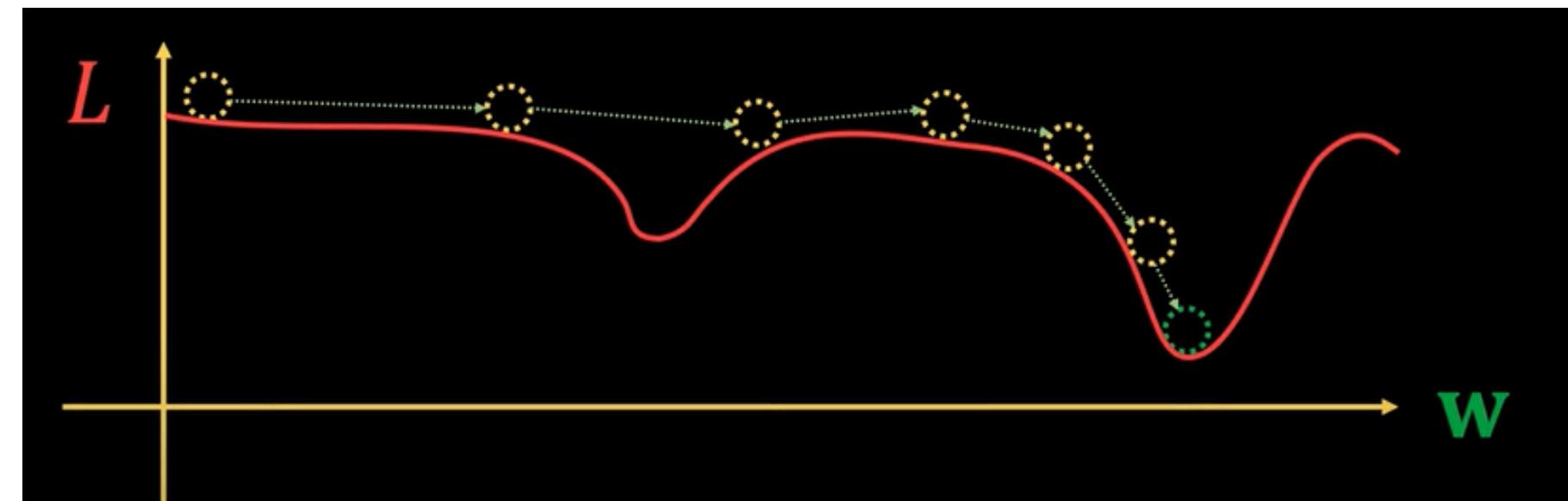
## 1. Tính trung bình động của bình phương gradient

$$E[g^2]_t = \rho E[g^2]_{t-1} + (1 - \rho) g_t^2$$

- $g_t$ : gradient tại bước  $t$
- $\rho$ : hệ số decay (thường chọn khoảng 0.9)

## 2. Cập nhật tham số

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$



- $\eta$ : learning rate gốc (tổng thể)
- $\epsilon$ : hằng số nhỏ (ví dụ  $10^{-8}$ ) để tránh chia cho 0

# Adam (Adaptive Moment Estimation)

1. **Momentum (ước lượng bậc nhất):** Tính trung bình động của gradient để làm mượt và định hướng cập nhật.

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \nabla L(w_t)$$

2. **RMSProp (ước lượng bậc hai):** Tính trung bình động của bình phương gradient để điều chỉnh tốc độ học theo độ lớn của gradient.

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot (\nabla L(w_t))^2$$

## d. Hiệu chỉnh độ lệch (bias correction):

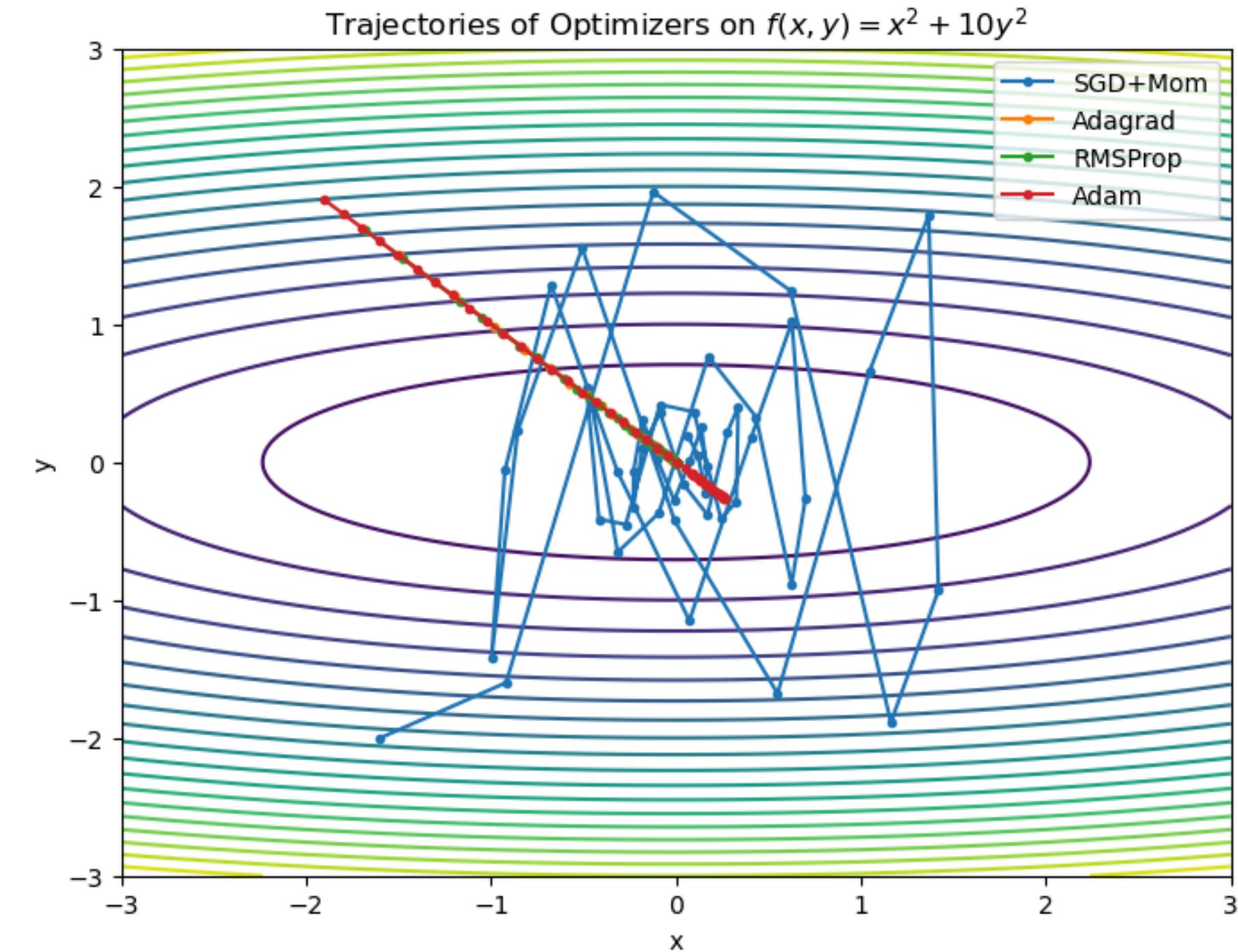
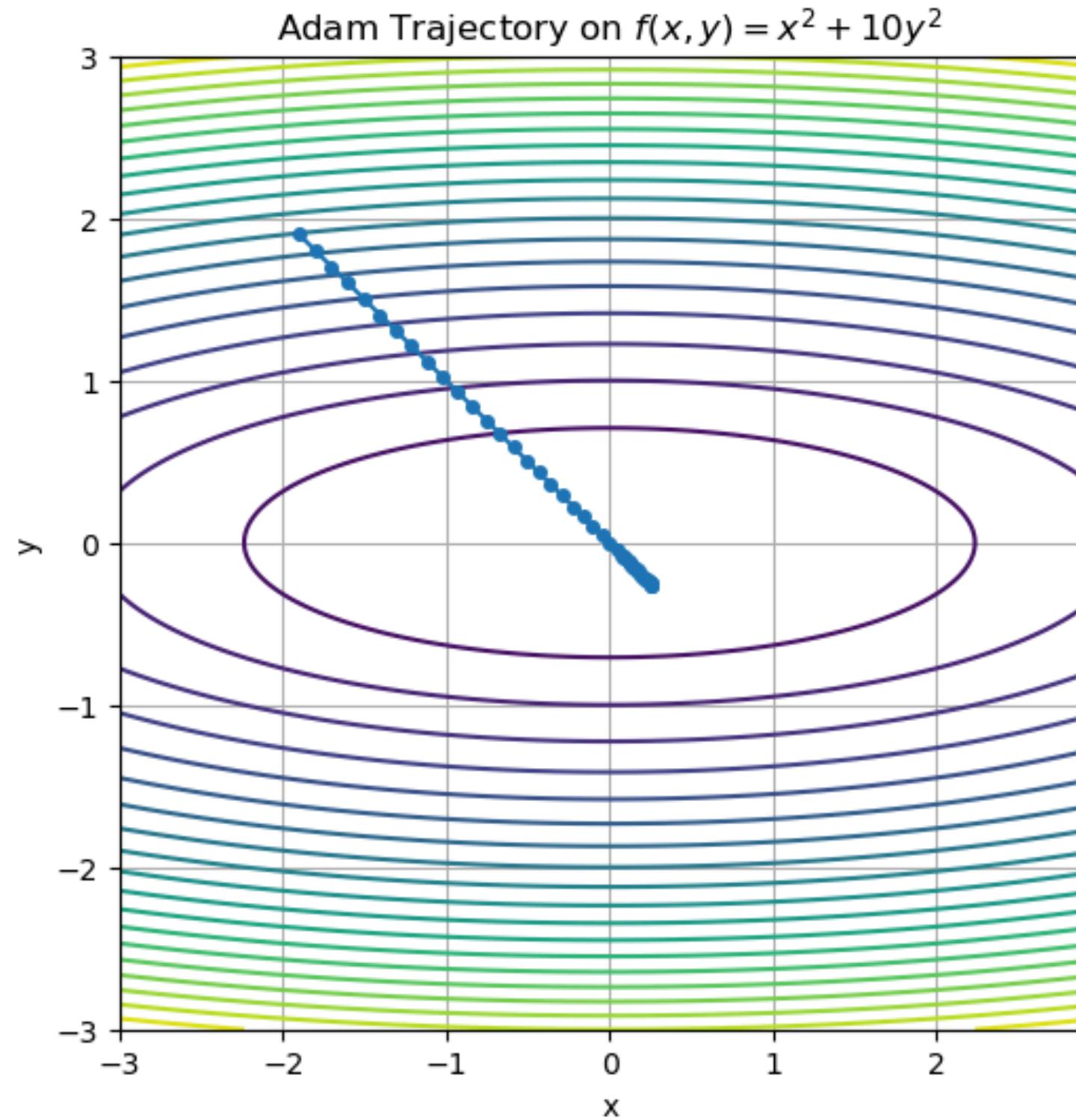
Do  $m_t, v_t$  ban đầu bị lệch về 0 nên ta hiệu chỉnh:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

## e. Cập nhật tham số:

$$w_{t+1} = w_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

# Adam (Adaptive Moment Estimation)



# Search Algorithms

Algorithms that search for hyperparameters



# Grid Search

- Grid Search is often considered a “brute force” approach to hyperparameter optimization.
- Tests all possible combinations of predefined hyperparameter values
- Builds a grid of parameter combinations and trains a model for each one
- Logs (ghi lại) performance and selects the best combination
- Expensive computing with big search space

# Grid Search

- For example:

```
num_hidden_layers = hp.Choice('num_hidden_layers', values = [1,2,3])
```

```
num_units = hp.Choice('num_units', values = [8,16,32])
```

- 9 combinations of [#hidden\_layers, #num\_units] are  
[1, 8], [1,16], ... [3,16], [3,32]

- However for some special cases, this algorithm does not work:

1. Continuous values: learning\_rate = hp.Float('learning\_rate',  
min\_value=0.0001, max\_value=0.1) => Infinite values

2. Big search space:

    num\_units: [8, 16, 32, 64, 128, 256]; activation: ['relu', 'tanh',  
'sigmoid'], optimizer: ['adam', 'sgd', 'rmsprop']  
=> Total: 54 tries

# Grid Search

## 3. Dynamic structure

When model/hyperparameters has uncertain structure/ changes

For example: #hidden\_layers change and each hidden layer can have different #units

## 4. Dependent/ Conditional hyperparameters:

```
optimizer = hp.Choice('optimizer', ['adam', 'sgd'])
```

```
if optimizer == 'adam' : beta_1, beta_2 = hp.Float('beta_1', 0.8, 0.999)
```

```
else: pass
```

```
# Optimizer: algorithm that optimizes the weights
```

```
=> Grid search can still try beta_1 = 0.9 even if optimizer = 'sgd' and we  
do not control the logic good enough
```

# Random Search

- Random Search selects values at random as opposed to the predetermined set of values
- Trains the model using different combinations for a fixed number of iterations
- Returns the best performing combination
- Use when no time, no experience, big search space
- Proved to be better than grid search but still expensive computation
- Used in early stage before guided algorithm to have better result
- Takes more time and computational resources than other guided search method

# Random Search

- Considered as an improvement of grid search for 2 reasons\*:

1. Distributing budget (tài nguyên tính toán) more flexible:

For grid search, budget is fixed for a set of hyp.para.

For random search, budget distributes depending on **distribution of search space** (hiệu suất mô hình thay đổi theo các bộ hyp.para.)

2. Better possibility of searching better results when we have more time:

Based on Monte Carlo technique, more time means more chance of best result whereas, grid search does not make sure can find better result if more time

\*: Yu & Zhu (2020), arXiv:2003.05689

# Random Search

- Steps:
  1. Define search space
  2. Choose #iterations to run the algorithm
  3. With every trial, train the model with those randomly chosen parameters, evaluate by a metric and log it (lưu lại lịch sử)
  4. Compare and give out the best result

# Bayesian Optimization

-  BO hoạt động như thế nào?
- Gồm 2 thứ chính:
- Surrogate Model (thường là Gaussian Process - GP):
- Dự đoán hàm mục tiêu (objective function) dựa trên dữ liệu đã có.
- Acquisition Function:
- Quyết định điểm tiếp theo nên thử ở đâu (explore hay exploit).
  -

# Genetic Algorithm - Giải thuật di truyền

- Genetic Algorithm (GA) simulate the process of natural selection, they simulate “survival of the fittest” among individuals of consecutive generations to solve a problem.
- Operators (Toán tử):
  - Selection Operator (Chọn lọc)
  - Crossover Operator (Lai ghép)
  - Mutation Operator (Đột biến)

# Genetic Algorithm - Giải thuật di truyền



## 1. Selection Operator – Chọn lọc



### Mô phỏng:

"Chọn cá thể tốt nhất để sinh sản" – giống như trong tự nhiên, cá thể khỏe mạnh có nhiều khả năng sinh con hơn.



### Ý nghĩa:

- Chọn ra những cá thể (nghiệm) có fitness cao hơn (nghĩa là lời giải tốt hơn) để truyền gene cho thế hệ sau.
- Tăng xác suất duy trì các gene tốt trong quần thể.



### Cách chọn phổ biến:

- Roulette Wheel Selection: xác suất chọn tỉ lệ thuận với fitness.
- Tournament Selection: chọn ngẫu nhiên một nhóm nhỏ, rồi chọn cá thể tốt nhất trong nhóm.
- Rank Selection: sắp xếp cá thể theo fitness rồi chọn dựa theo xếp hạng.



# Genetic Algorithm - Giải thuật di truyền

## 2. Crossover Operator – Lai ghép

### Mô phỏng:

"Sinh sản lai tạo" – hai cá thể giao phối để tạo ra con.

### Ý nghĩa:

- Tạo ra cá thể mới bằng cách kết hợp thông tin di truyền từ hai cá thể bố mẹ.
- Giúp khám phá kết hợp mới giữa các gene (giống recombination trong sinh học).

### Các cách thực hiện:

- Single-point crossover: cắt chuỗi tại 1 điểm rồi hoán đổi phần sau.
- Two-point crossover: cắt tại 2 điểm rồi hoán đổi phần giữa.
- Uniform crossover: từng gene con nhận ngẫu nhiên từ bố hoặc mẹ.

### Ví dụ:

Cha: 101|011

Mẹ: 110|100

Con: 101100 (cắt tại dấu | )

# Genetic Algorithm - Giải thuật di truyền

## 3. Mutation Operator – Đột biến

### Mô phỏng:

"Đột biến gene ngẫu nhiên" – thay đổi nhỏ, hiếm gặp trong tự nhiên, nhưng quan trọng để **tạo đa dạng di truyền**.

### Ý nghĩa:

- Tránh việc thuật toán bị kẹt ở **cực trị địa phương** (local optima).
- Tăng cường khám phá các vùng mới của không gian nghiệm.

### Ví dụ:

- Thay đổi một bit từ  $0 \rightarrow 1$  hoặc  $1 \rightarrow 0$ .
- Đổi giá trị 1 gene trong chuỗi đại diện cho cá thể.

### Ví dụ:

Cá thể gốc: `101101`

Sau mutation tại vị trí 2 → `100101`



# The End

Group 5