CHƯƠNG 3
TỐI ƯU HOÁ MÔ HÌNH MẠNG NEURAL (P2)

Khoa Khoa học và Kỹ thuật thông tin
Bộ môn Khoa học dữ liệu

NỘI DUNG

1. Mini-batch training.
2. Batch normalization.
3. Các thuật toán tối ưu.
4. Learning rate decay.

Batch and Mini-batch

Batch and Mini Batch

– Dữ liệu huấn luyện:

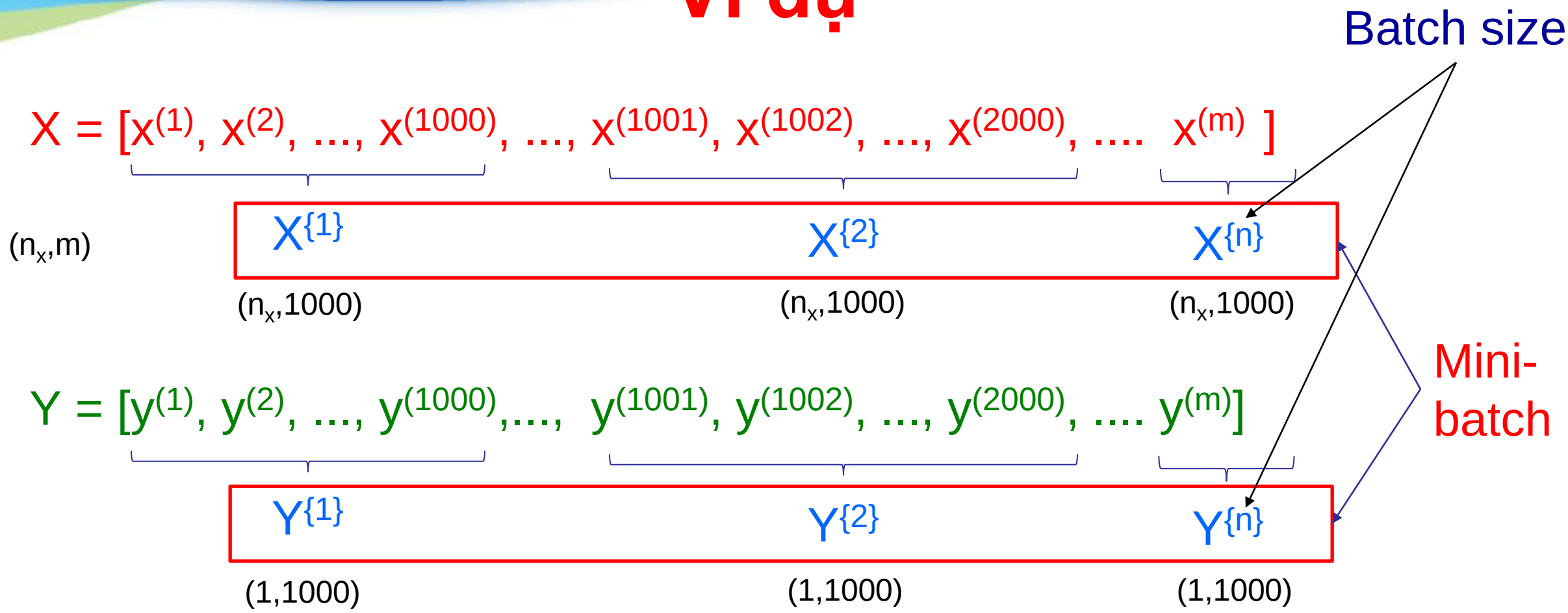
$X = [x^{(1)}, x^{(2)}, x^{(3)}, ..., x^{(m)}]$

$Y = [y^{(1)}, y^{(2)}, y^{(3)}, ..., y^{(m)}]$

Nếu như dữ liệu có kích thước quá lớn (>50M) thì việc huấn luyện một lần hết cả bộ dữ liệu không khả thi (không đủ tài nguyên, ...).

➔ Chia dữ liệu ra thành từng đoạn nhỏ, gọi là batch, lần lượt huấn luyện các batch nhỏ trên.

Ví dụ



Mỗi mini-batch chứa 1000 điểm dữ liệu (batch size: n=1000)

Epoch

For t in 1, 2, ... 5000:

1 epoch

Forward propagation:

$Z^{[1]} = W^{[1]} * X^{\{t\}} + b^{[1]}$

$A^{[1]} = g(Z^{[1]})$

...

$A^{[L]} = g(Z^{[L]})$

Compute cost: $J^{\{t\}} = \frac{1}{1000} \sum_{i=1}^{L} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2*\boldsymbol{n}} \sum_{i=1}^{L} ||W^{[l]}||_F^2$

Back propagation:

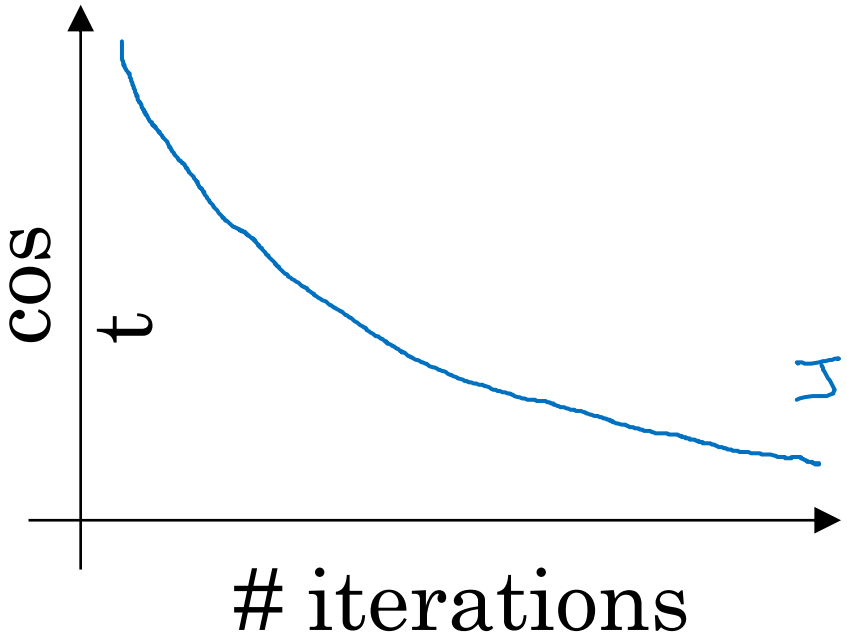
$W^{[1]} = W^{[1]} - \alpha dW^{[1]}$

$b^{[1]} = b^{[1]} - \alpha db^{[1]}$

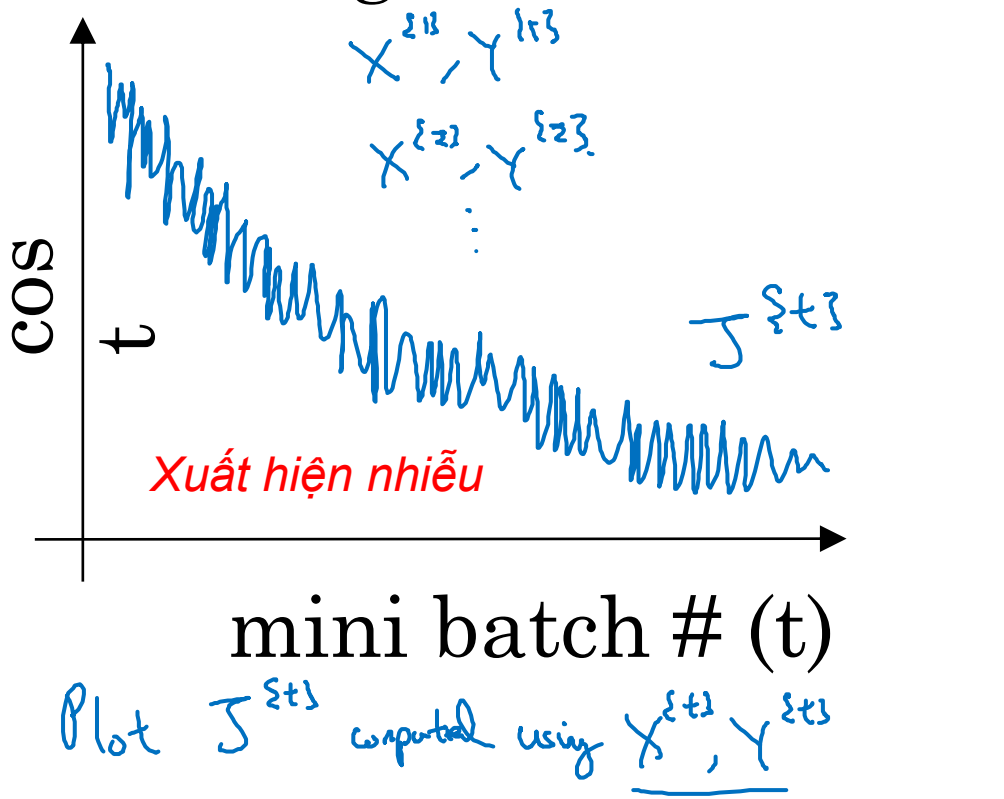
Một epoch được xem như 1 ”pass” – tức là huấn luyện xong 1 mini-batch.

Khác biệt giữa train bình thường và mini-batch train

Batch gradient descent



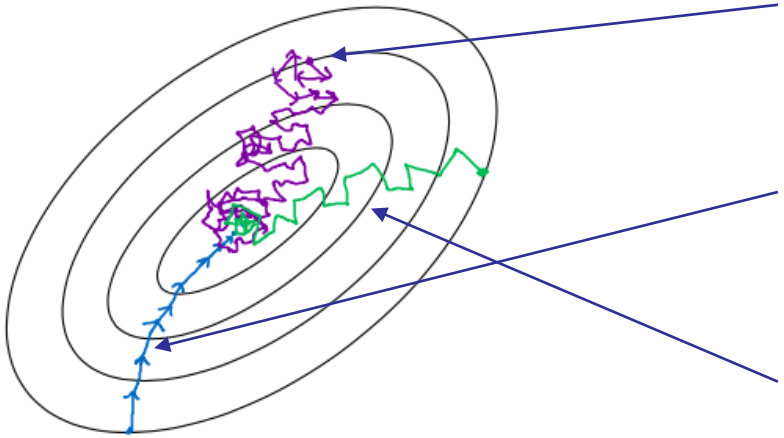
Mini-batch gradient descent



Để hạn chế nhiễu ➔ điều chỉnh batch size

Chọn batch size

- Trên thực tế, có thể chọn batch size n trong khoảng từ 1 → m.
  + Nếu chọn n = m: Batch gradient descent.
  + Nếu chọn n = 1: Stochastic gradient descent.
- Tuy nhiên, Stochastic gradient descent xuất hiện nhiễu (noise) rất nhiều, đôi khi sẽ không thể đạt đến giá trị tối tiểu toàn cục.



Stochastic gradient descent *bị nhiễu rất nhiều, và không thể đạt tới tâm (giá trị global minimum) được.*

Batch gradient descent thường hội tụ nhanh, tuy nhiên nếu dữ liệu quá lớn thì sẽ làm chậm gradient descent, và đôi khi không thể đủ tài nguyên để đáp ứng

Chọn batch size hợp lý !

Chọn batch size (tt)

- Nếu dữ liệu nhỏ (m <= 200): train hết (batch gradient descent).
- Nếu dữ liệu lớn hơn, thì nên chọn các giá trị batch size tiêu chuẩn như sau (là số mũ của 2):
  - + $64 = 2^6$.
  - + $128 = 2^7$.
  - + $256 = 2^8$.
  - + $512 = 2^9$.
  - + $1024 = 2^{10}$.

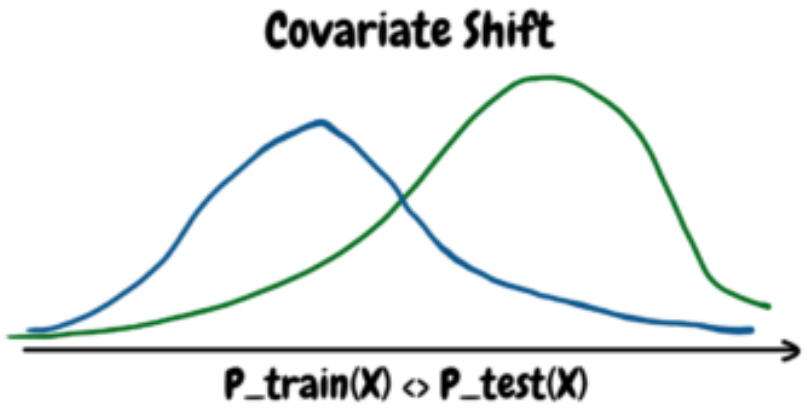
➔ *Chọn batch size phù hợp với dung lượng RAM/CPU của máy.*

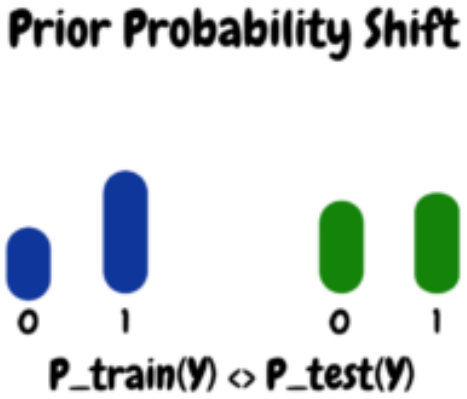
Batch normalization

Dataset shift

- Là hiện tượng dữ liệu phân bố trong tập huấn luyện (training set) và tập kiểm tra (dev/test set) khác nhau, dẫn đến sự sai lệch của mô hình trong quá trình dự đoán kết quả.
- Các dạng dataset shift:
  + Covariate Shift: hiện tượng khác nhau giữa các biến độc lập (X).
  + Prior Probability Shift: hiện tượng khác nhau giữa các biến phụ thuộc – hay còn gọi là nhãn (y).
  + Concept Shift: Sự khác nhau về mối quan hệ giữa biến phụ thuộc và biến độc lập. Hiện tượng này xảy ra khi dữ liệu huấn luyện và dữ liệu kiểm tra được thu thập khác thời điểm.

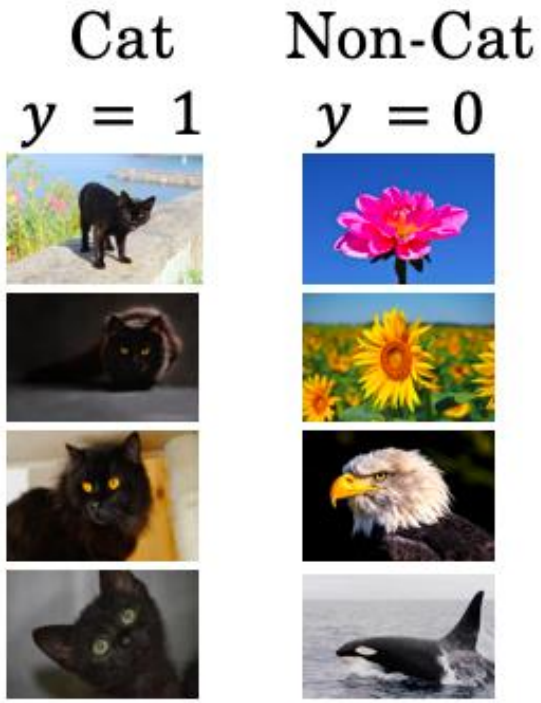
Dataset shift – minh hoạ





Covariate Shift – ví dụ

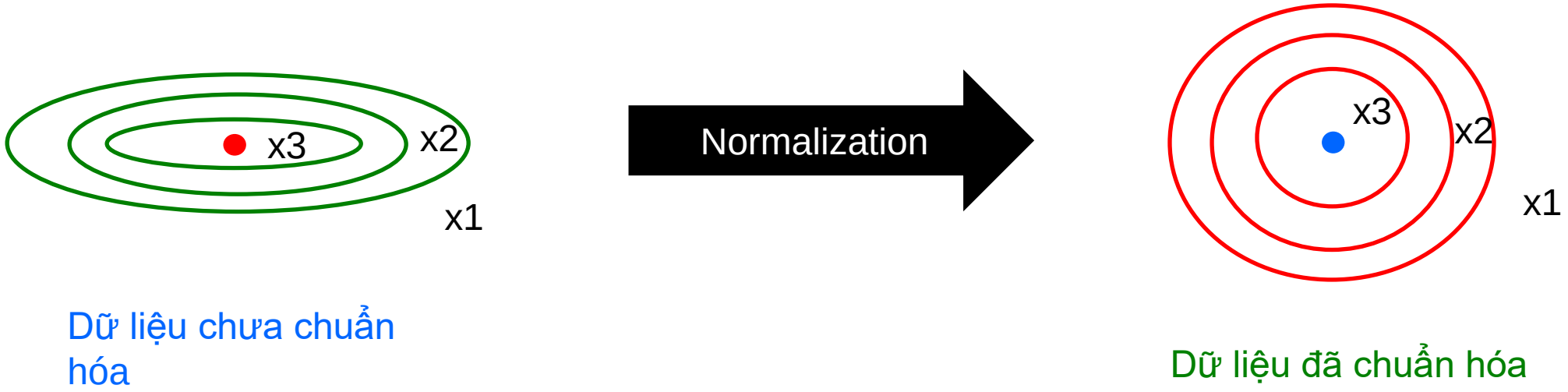
Tập huấn luyện



Tập kiểm tra

Batch normalization

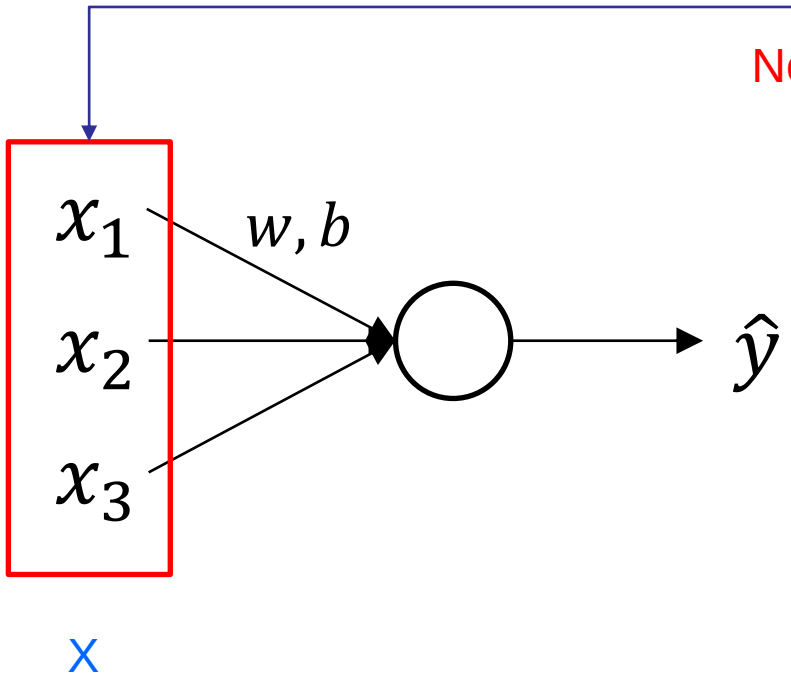
- Mục tiêu: làm cho cho quá trình học nhanh hơn:
  + Chuẩn hóa đầu vào (input) giúp cho quá trình học - điều chỉnh các tham số W và b tốt hơn.
- Minh họa:



Dữ liệu chưa chuẩn hóa

Dữ liệu đã chuẩn hóa

Normalizing input



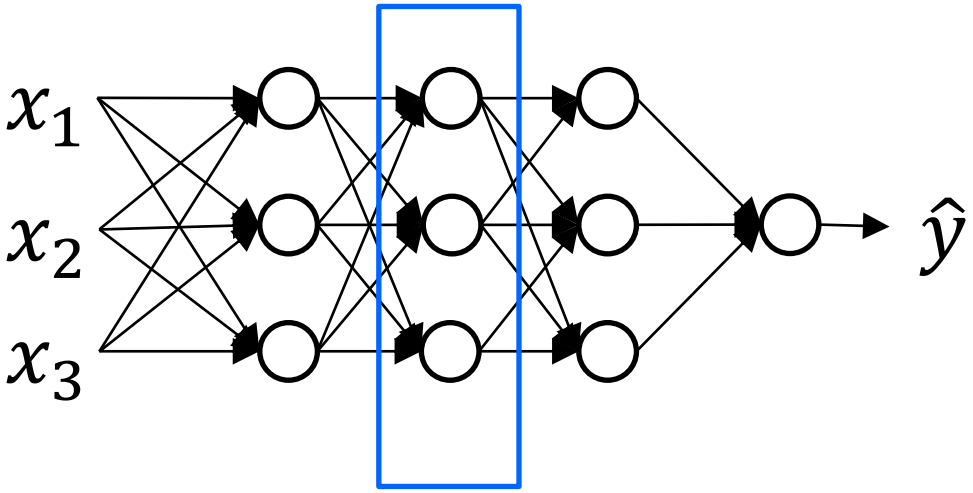
$$\mu = \frac{1}{m}\sum_i X^{(i)}$$

$$X = X - \mu$$

$$\sigma^2 = \frac{1}{m}\sum_i X^{(i)\wedge 2}$$

$$X = X / \sigma^2$$

Normalizing input



Normalizing cho 1 lớp: $Z^{[i]}$

$$\mu = \frac{1}{m}\sum_i z^{(i)}$$

$$\sigma^2 = \frac{1}{m}\sum_i \left(z^{(i)} - \mu\right)^2$$

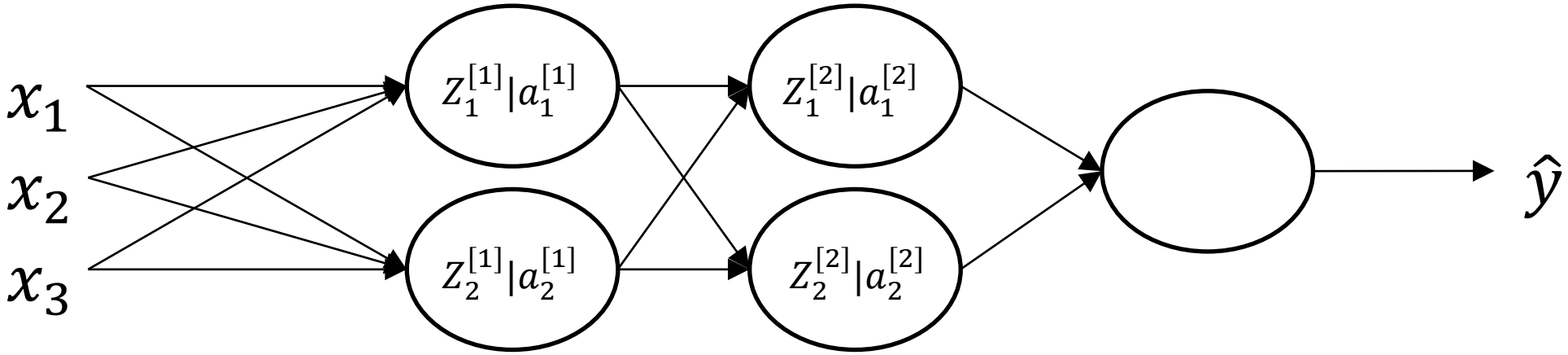
$$z^{(i)}_{norm} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

Chuẩn hóa z để tránh trường hợp giá trị của Z rơi vào phân phối chuẩn tắc.

$$\hat{z}^{(i)} = \gamma * z^{(i)}_{norm} + \beta$$

Hyper parameter: γ và β

Batch norm in neural network

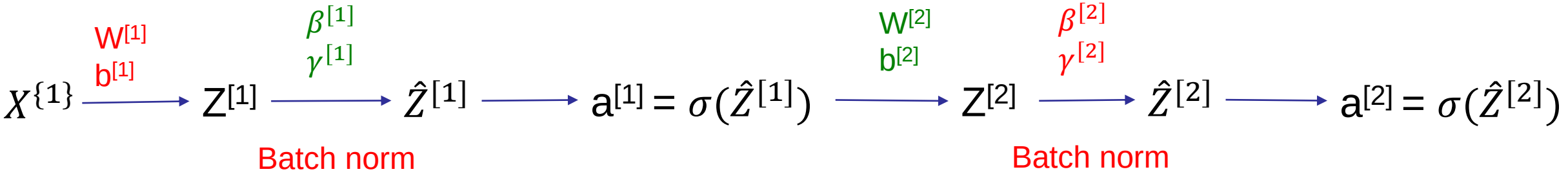


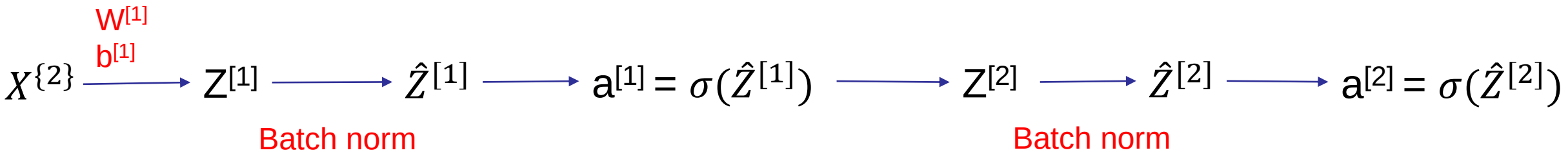
$$X \xrightarrow{W^{[1]},\ b^{[1]}} Z^{[1]} \xrightarrow[\text{Batch norm}]{\beta^{[1]},\ \gamma^{[1]}} \hat{Z}^{[1]} \rightarrow a^{[1]} = \sigma(\hat{Z}^{[1]}) \xrightarrow{W^{[2]},\ b^{[2]}} Z^{[2]} \xrightarrow[\text{Batch norm}]{\beta^{[2]},\ \gamma^{[2]}} \hat{Z}^{[2]} \rightarrow a^{[2]} = \sigma(\hat{Z}^{[2]})$$

Parameters: $W^{[1]}$, $b^{[1]}$, $\beta^{[1]}$, $\gamma^{[1]}$, $W^{[2]}$, $b^{[2]}$, $\beta^{[2]}$, $\gamma^{[2]}$, …, $W^{[L]}$, $b^{[L]}$, $\beta^{[L]}$, $\gamma^{[L]}$.

Mini-batch normalization

Sử dụng batch-norm để chuẩn hóa ứng với từng epoch:





Parameter: $W^{[l]}$, $b^{[l]}$, $\beta^{[l]}$, $\gamma^{[l]}$

Có thể bỏ qua $b^{[l]}$

Áp dụng trong Gradient descent

For t = 1 to num_minibatch:

For each l in Layer

1.Tính forward propagation cho $X^{\{t\}}$: $z^{[l]} = \hat{z}^{[l]}$

2. Tính backward propagation cho $X^{\{t\}}$: $dW^{[l]}$, ($db^{[l]}$), $\beta^{[l]}$, $\gamma^{[l]}$

3. Cập nhật trọng số:

$$W^{[l]} = W^{[l]} - \alpha * dW^{[l]}$$

$$\beta^{[l]} = \beta^{[l]} - \alpha * d\beta^{[l]}$$

$$\gamma^{[l]} = \gamma^{[l]} - \alpha * d\gamma^{[l]}$$

Batch norm – giải quyết vấn đề Covariate shift

- Với mỗi lớp trong mạng neural, các tham số W, b của mỗi lớp sẽ phụ thuộc vào giá trị tham số của lớp trước đó.
- Sự thay đổi của các tham số ở các lớp trước sẽ kéo theo sự thay đổi các giá trị trong mỗi neural (unit) ở lớp hiện tại.
  + Giá trị trong mỗi neural (unit) chính là giá trị được ánh xạ bởi các biến độc lập của dữ liệu huấn luyện ban đầu.

➔ Batch norm sẽ giảm thiểu sự thay đổi của các giá trị ở các unit trong lớp hiện tại ➔ các lớp sau sẽ không bị thay đổi (shift) nhiều.

➔ Giải quyết Covariate shift.

Các thuật toán tăng tốc

Mục tiêu

- Tăng tốc quá trình học của mô hình.
  + Tăng tốc độ hội tụ về điểm cực tiểu trong gradient descent.
- Các kỹ thuật chính:
  + Bias correction: giảm độ nhiễu của mô hình.
  + Momentum: tăng tốc độ cho GD.
- Giới thiệu 2 thuật toán chính:
  + RMSProp.
  + Adam.

Bias correction

Trung bình trượt

- Trung bình trượt (Moving average) là một kỹ thuật trong phân tích dữ liệu nhằm làm giảm các điểm dữ liệu nhiễu ra khỏi tập dữ liệu quan sát được.
- Exponential Average Weight (EAW) là một dạng trung bình trượt (MA). Bằng cách đặt vào một trọng số (greater weight) và cộng với dữ liệu gần trước đó nhất, EAW sẽ giảm được độ nhiễu của dữ liệu.

Ví dụ

$\theta_1 = 40°F$

$\theta_2 = 49°F$

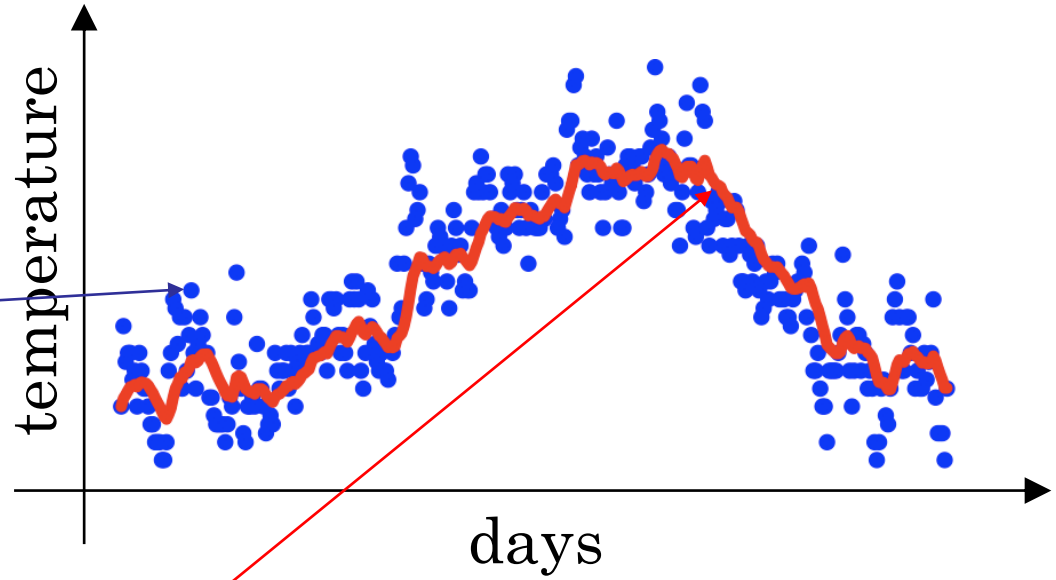
$\theta_3 = 45°F$

$\vdots$

$\theta_{180} = 60°F$

$\theta_{181} = 56°F$

$\vdots$



$V_0 = 0$

$V_1 = 0.9*V_0 + (1-0.9)*\theta_1$

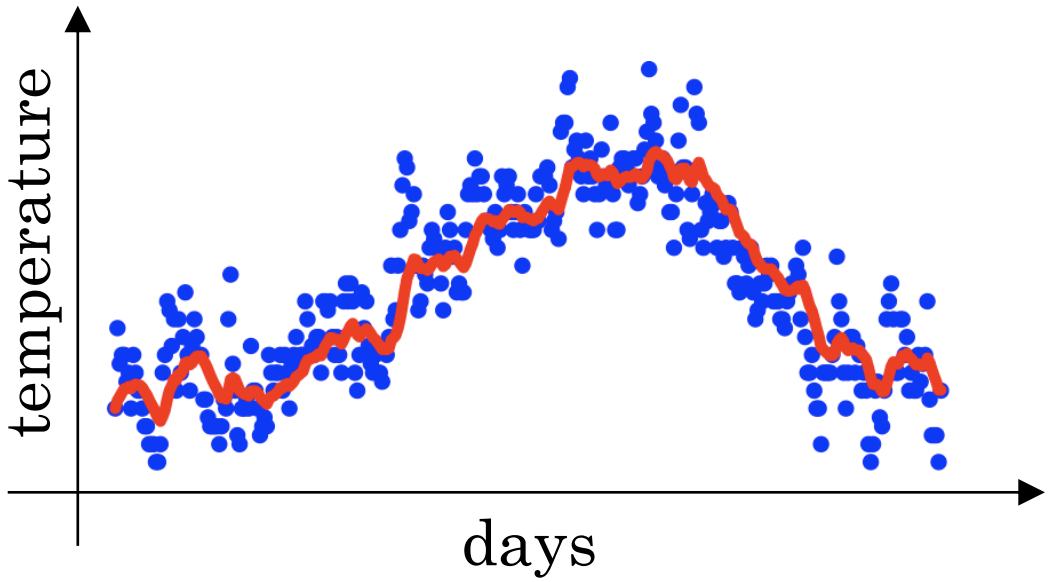
$V_2 = 0.9*V_1 + (1-0.9)*\theta_2$

$V_3 = 0.9*V_2 + (1-0.9)*\theta_3$

…

$Vt = 0.9*V_{t-1} + (1-0.9)*\theta_t$

Ví dụ



$\beta = 0.9$: Giá trị nhiệt độ trong 10 ngày
$\beta = 0.98$: Giá trị nhiệt độ trong 50 ngày
$\beta = 0.5$: Giá trị nhiệt độ trong 2 ngày.

Một cách tổng quát:

$$V_t = \beta V_{t-1} + (1-\beta)\theta_t$$

V_t là đường trung bình trượt, giá trị V_t sẽ xấp xỉ với giá trị thực một lượng là $\frac{1}{1-\beta}$

Bias correction

$\theta_1 = 40°F$

$\theta_2 = 49°F$

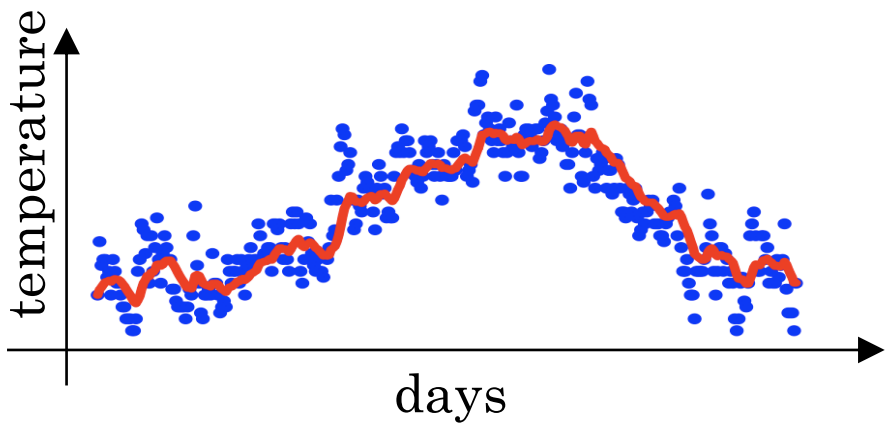
$\theta_3 = 45°F$

$\vdots$

$\theta_{180} = 60°F$

$\theta_{181} = 56°F$

$\vdots$



$V_t = \beta V_{t-1} + (1-\beta)\theta_t$

Giả sử: $\beta = 0.98$

$V_0 = 0$

$V_1 = 0.98*V_0 + 0.02*\theta_1$

$V_2 = 0.98*V_1 + 0.02*\theta_2$

$= 0.98*0.02*\theta_1 + 0.02*\theta_2$

$= 0.0196*\theta_1 + 0.02*\theta_2$

Tuy nhiên, khi tính V1 và V2:

V1 = 40*0.02 = 8

V2 = 0.0196*40+ 0.02*49 = 1.764

➔ Chênh lệch quá lớn so với θ_1 và θ_2, đồng thời, $V_1 < V_2$

Bias correction

– Bias correction:

$$V_t' = \frac{V_t}{1 - \beta^t}$$

Với t là dữ liệu ở thời điểm t hiện tại.

t = 1: $V_1' = \frac{V_1}{1-0.98^1} = \frac{0.02 * 40}{0.02} = 40$

t = 2: $V_2' = \frac{V_2}{1-0.98^2} = \frac{0.0196 * \theta_1 + 0.02 * \theta_2}{0.0396} = \frac{0.0196 * 40 + 0.02 * 49}{0.0396} = 44.5$

Bias correction

– Bias correction:

$$V_t' = \frac{V_t}{1 - \beta^t}$$

t càng lớn ➔ β^t càng nhỏ và tiến về 0.

➔Với t càng lớn, thì $V_t' \approx V_t$

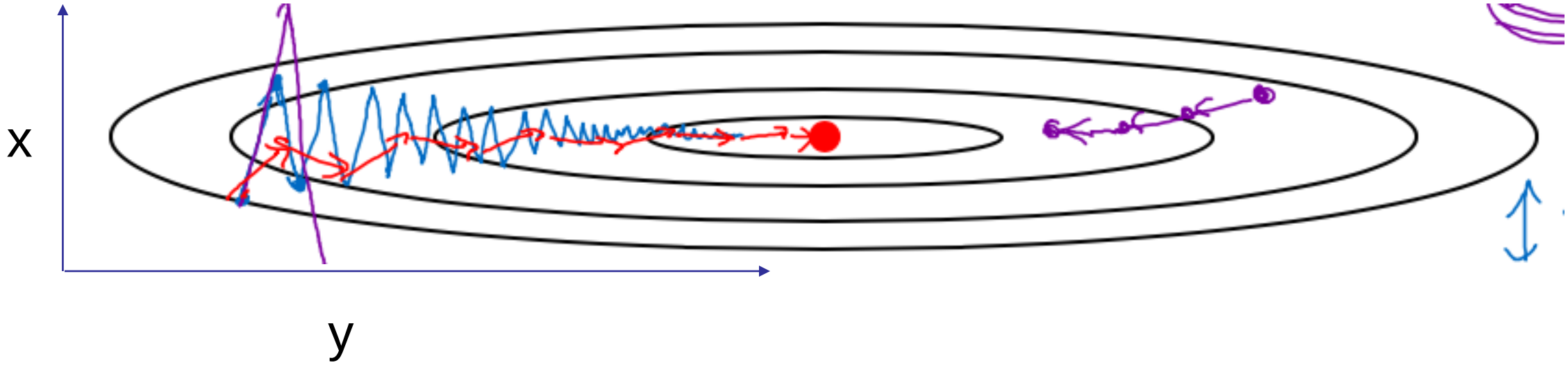
Bias correction chỉ thích hợp ở giai đoạn đầu (initial phase) của pha huấn luyện, nhằm giúp kiểm tra xem xu hướng của gradient descent có chính xác như mong muốn hay không.

Momentum

Momentum

- Momentum được tạm hiểu là động lượng, là một khái niệm vay mượn từ vật lý học. Mục tiêu của Momentum là đánh giá tốc độ thay đổi của một đối tượng nào đó trong một khoảng thời gian xác định nhằm hiểu được xu hướng (trend) của nó.

Momentum



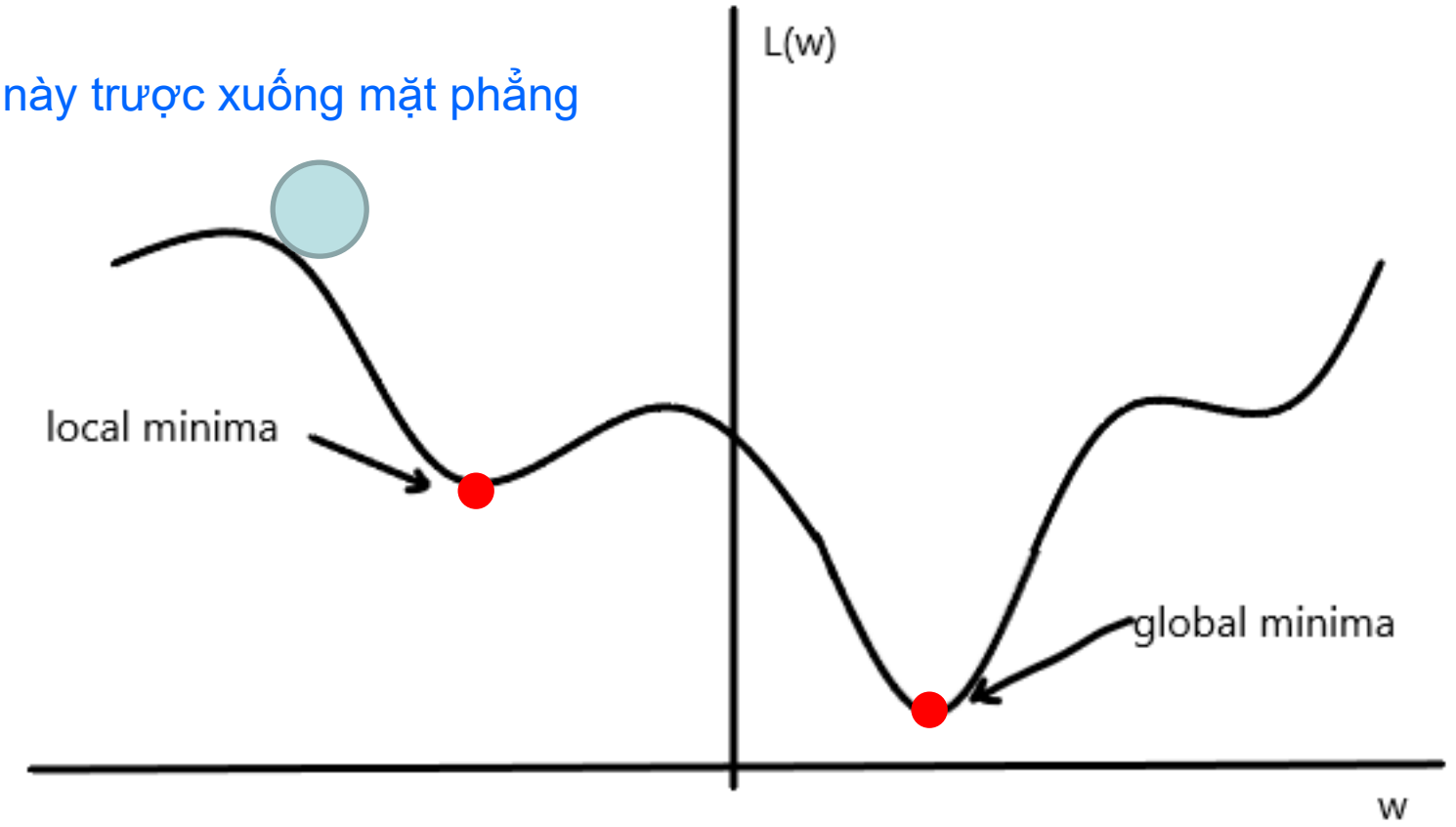
Trục x – tốc độ biến thiên sau mỗi lần lặp t.

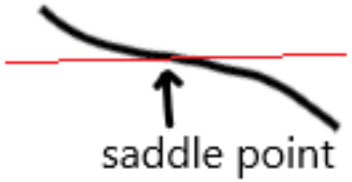
Trục y – tốc độ hội tụ về điểm tối tiểu.

***Mục tiêu**: x chậm lại, y tăng lên.*

Giải thích Momentum

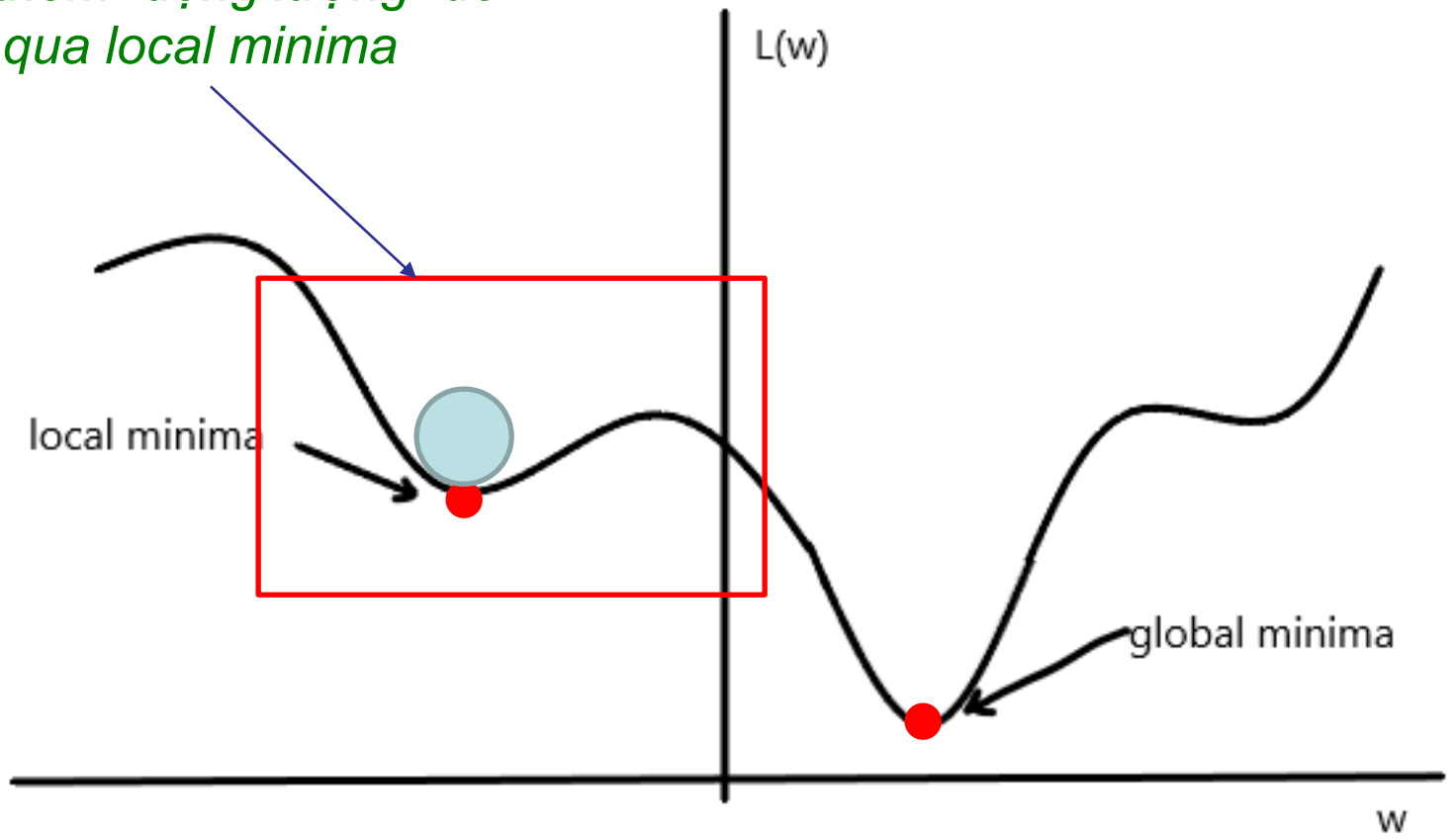
Khối này trược xuống mặt phẳng

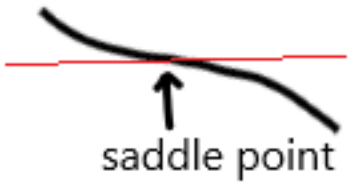




Giải thích Momentum

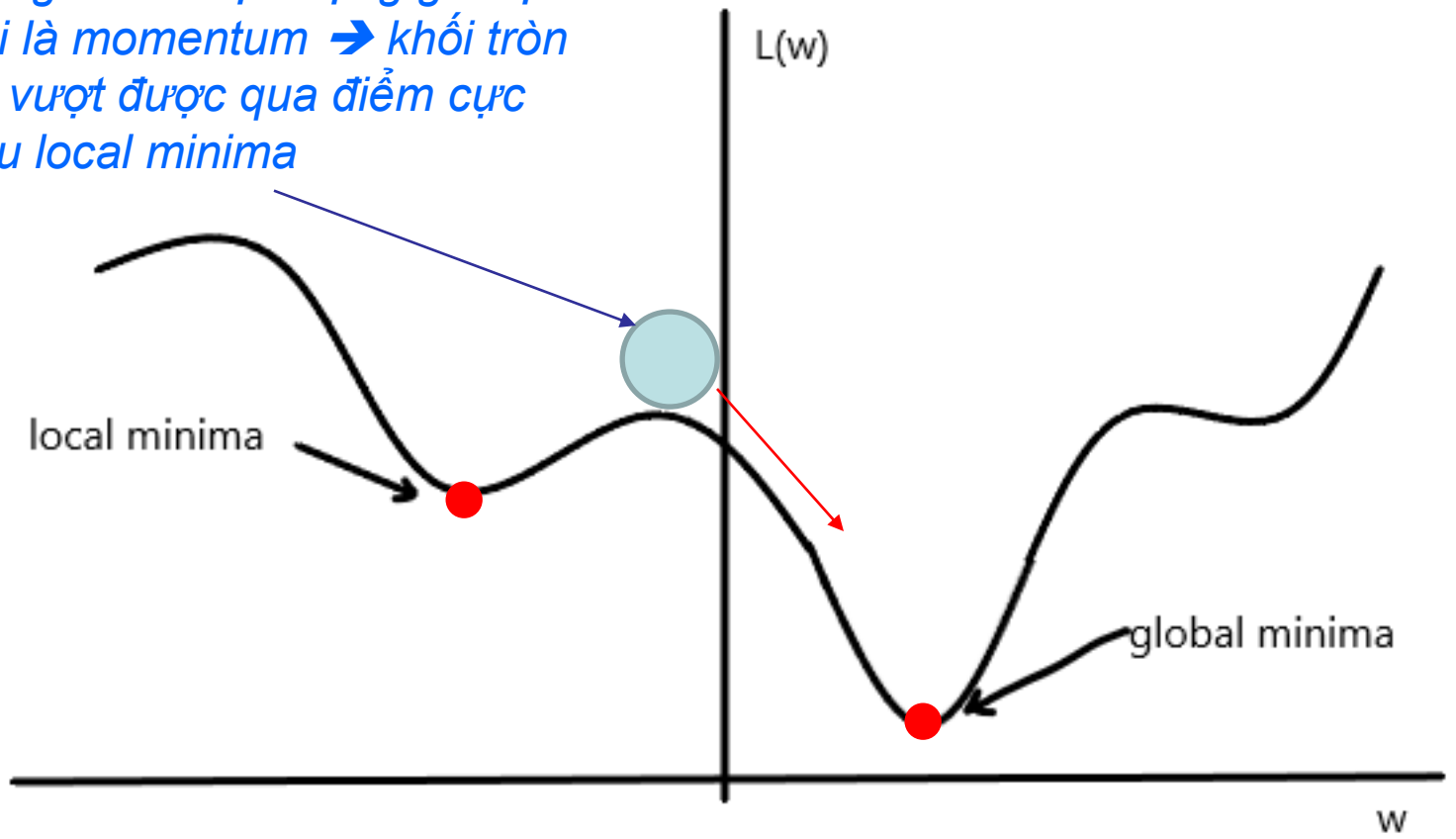
Cần thêm "động lượng" để vượt qua local minima

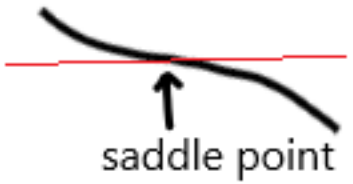




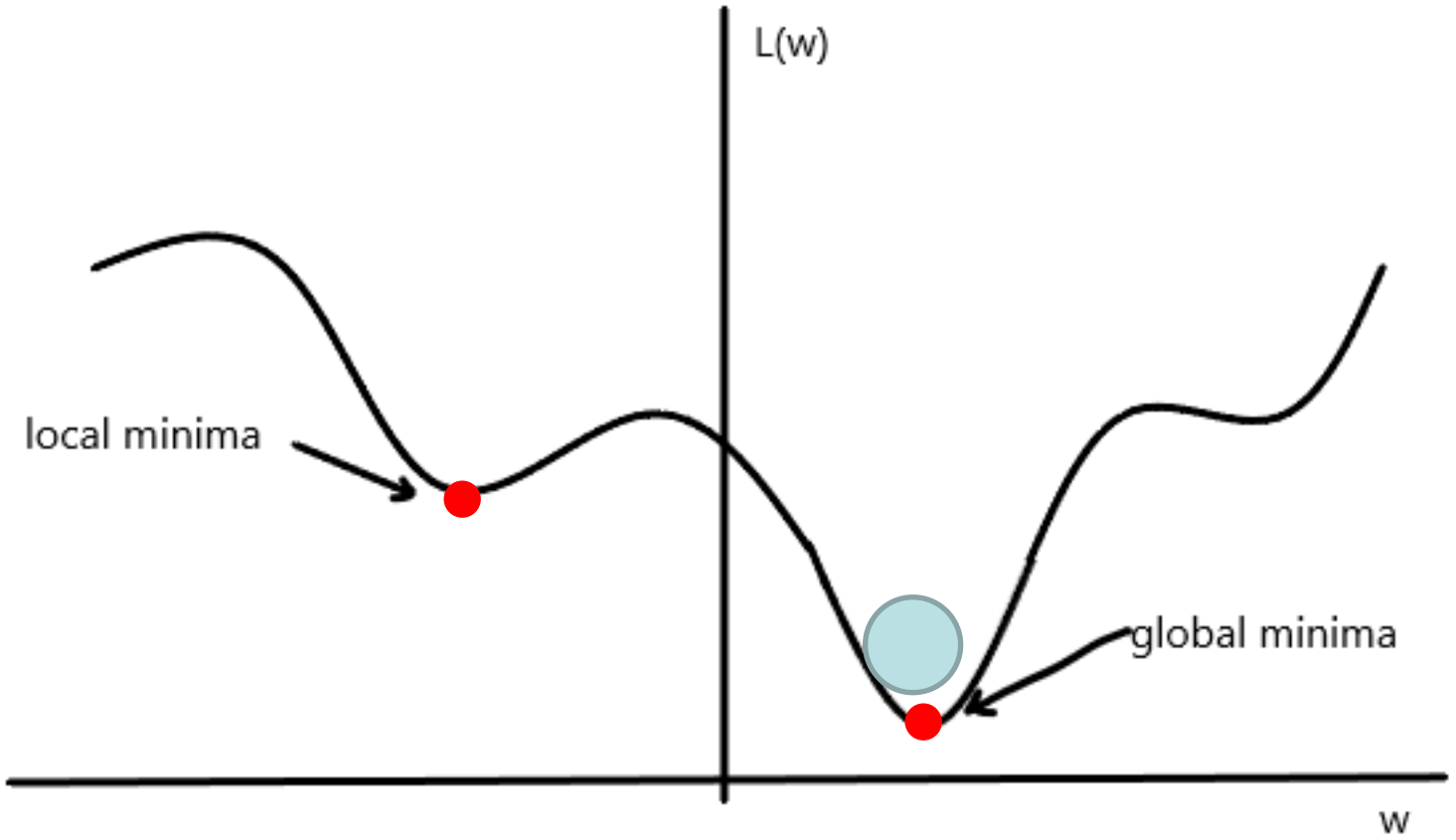
Giải thích Momentum

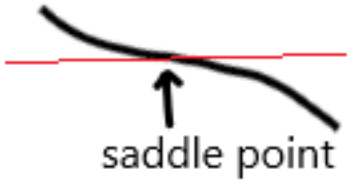
Cộng thêm một lượng giá trị gọi là momentum ➔ khối tròn sẽ vượt được qua điểm cực tiểu local minima



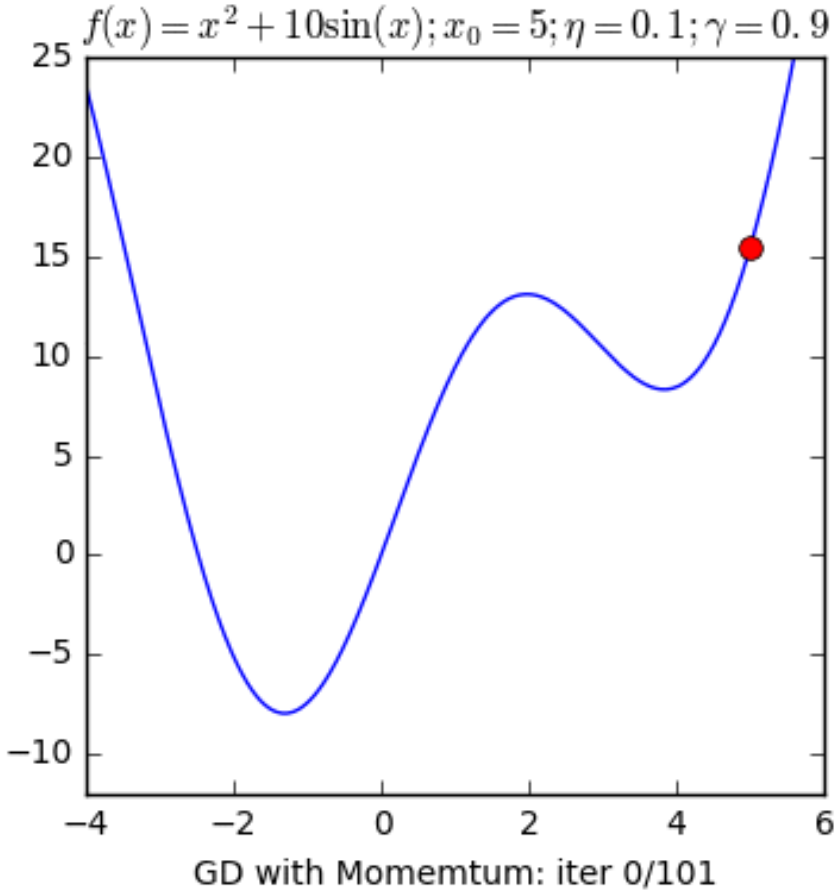


Giải thích Momentum





Monentum


$f(x) = x^2 + 10\sin(x); x_0 = 5; \eta = 0.1; \gamma = 0.9$
25
20
15
10
5
0
−5
−10
−4
−2
0
2
4
6
GD with Momemtum: iter 0/101

Momentum

On iteration t:

Compute dW, db on the current mini-batch

$$v_{dW} = \beta v_{dW} + (1-\beta)dW$$

$$v_{db} = \beta v_{db} + (1-\beta)db$$

$$W = W - \alpha v'_{dW},$$

$$b = b - \alpha v'_{db}$$

$$v'_{dw} = \frac{v_{dw}}{1-\beta^2}$$

$$v'_{db} = \frac{v_{db}}{1-\beta^2}$$

Có thể dùng bias corection, nhưng không cần thiết, vì sau khoảng 10 iteration thì kết quả v'_{dw} và v'_{db} sẽ không khác gì so với v_{dW} và v_{db}

Hyperparameters: α, β $\beta = 0.9$

Dùng momentum sẽ được lợi gì

- Giảm hiện tượng nhiễu (noise) khi sử dụng gradient descent để tìm cực tiểu, nhất là với dữ liệu được chia làm các mini-batch nhỏ.
- Có thể điều chỉnh tham số α, β để tăng tốc độ thực hiện của Gradient descent.

Stochastic Gradient Descent

Stochastic Gradient descent (SGD)

- Stochastic Gradient descent (SGD) sẽ cập nhật lại tập tham số θ bằng cách lấy ngẫu nhiên 1 điểm dữ liệu (instance) từ tập huấn luyện.
- Mục tiêu:
  + Tăng tốc quá trình huấn luyện: do chỉ tính đạo hàm cho 1 điểm dữ liệu thay vì tính cho toàn bộ tập dữ liệu huấn luyện.
  + Có khả năng huấn luyện đối với tập dữ liệu huấn luyện lớn và rất lớn: việc load 1 điểm dữ liệu vào bộ nhớ sẽ khả thi hơn là load một tập dữ liệu rất lớn (vài chục đến vài trăm gigabyte).

Stochastic Gradient descent (SGD)

– Hàm cập nhật tham số cho SGD được mô tả như sau:

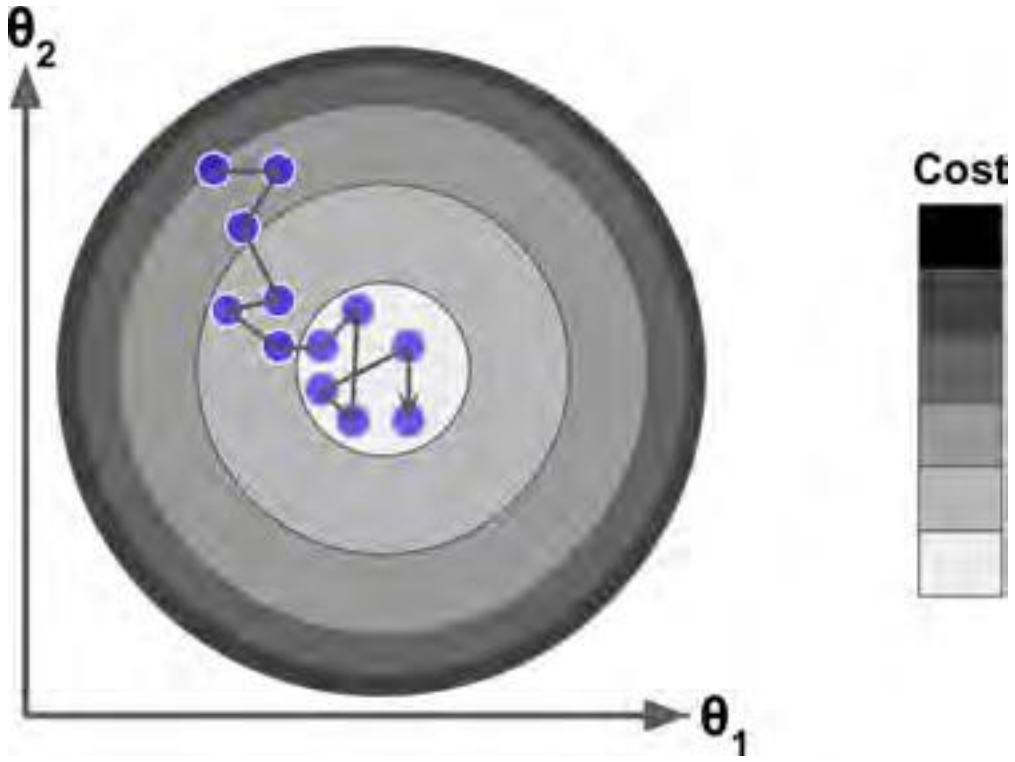
$$\boldsymbol{\theta}^{next\_step} \leftarrow \boldsymbol{\theta} - \boldsymbol{\alpha}\boldsymbol{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\mathcal{L}}(\boldsymbol{\theta}; x_i; y_i)$$

$\boldsymbol{\mathcal{L}}(\boldsymbol{\theta}; x_i; y_i)$: hàm mất mát với **1 điểm dữ liệu** $(x_i; y_i)$

– Một lần duyệt qua dữ liệu huấn luyện để cập nhật lại tham số $\boldsymbol{\theta}$ được gọi là một epoch.

– Với SGD, sẽ có tổng cộng N epoch ứng với N điểm dữ liệu trong tập huấn luyện.

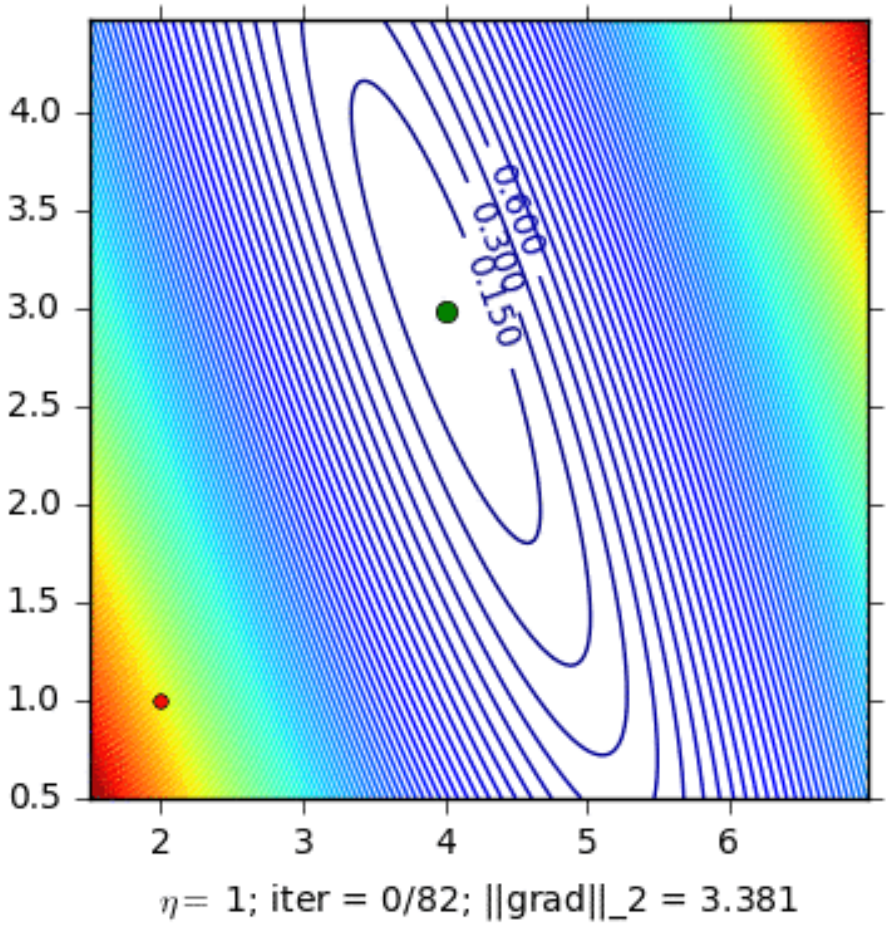
Stochastic gradient descent



- Do ảnh hưởng của quá trình lấy ngẫu nhiên, giá trị của hàm loss sẽ dao động lên xuống (nhiễu).
- Vì giá trị loss dao động lên xuống, giá trị cuối cùng tìm được sẽ chỉ gần với giá trị minimum chứ không *thực sự đạt đến giá trị minimum*.

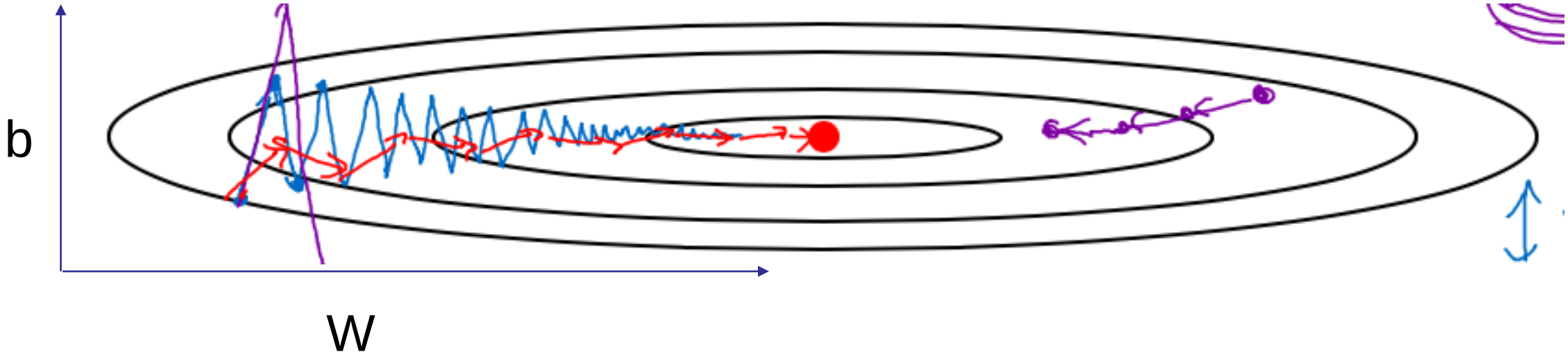
➔ Khắc phục: Tìm cách giảm bias và tăng tốc độ học để tiến về cực tiểu nhanh hơn và ổn định hơn.

Visualize Stochastic Gradient Descent



RMSProp

RMSProp



Mục tiêu: tăng tốc quá trình tiến về cực tiểu của SGD.

- Giảm tốc độ học ở trục đứng – b ➔ giảm tỉ lệ "dao động".
- **Tăng tốc độ học ở trục ngang – W** ➔ tăng tỉ lệ tiến đến điểm minimum.

Thuật toán RMSProp

On iteration t:

Compute dW, db on the current mini-batch

$dW^2 = dW \otimes dW$

$db^2 = db \otimes db$

$$S_{dW} = \beta S_{dW-1} + (1-\beta) dW^2$$

S_{dw} rất nhỏ

$$S_{db} = \beta S_{db-1} + (1-\beta) db^2$$

S_{db} rất lớn

$$W = W - \alpha \frac{dW}{\sqrt{S_{dW}}}$$

$$b = b - \alpha \frac{dW}{\sqrt{S_{db}}}$$

S_{dw} nhỏ ➔ $\frac{dW}{\sqrt{S_{dW}}}$ lớn ➔ W học nhanh hơn

S_{db} lớn ➔ $\frac{dW}{\sqrt{S_{db}}}$ nhỏ ➔ b học chậm hơn

Nhận xét

- Tốc độ học (learning rate) vẫn giữ nguyên.
- RMSProp tăng tốc được quá trình tiến đến cực tiểu so với SGD trước đó bằng cách giảm tốc độ bias và tăng tốc độ học cho W.
  + RMSProp khiến cho quá trình gradient descent ổn định hơn.
- Tuy nhiên, khi đến gần cực tiểu mà chưa chắc là cực tiểu toàn cục (điểm yên ngựa – saddle point) thì RMSProp không đủ "mạnh" để vượt qua nó.
  + Cần kết hợp thêm Momentum để tăng sức mạnh vượt qua local minimum.

Adam Optimization

Adam optimization

- Là một thuật toán kết hợp các điểm mạnh của kỹ thuật momentum và thuật toán RMSProp nhằm tối ưu hóa quá trình học của gradient descent.
- RMS prop và Adam là một trong các thuật toán tối ưu có tính tổng quát, có thể áp dụng cho nhiều kiến trúc mạng neural khác nhau.

Adam optimization

Momentum

combined

RMSProp

On iteration t:

Compute dW, db on the current mini-batch

$$v_{dW} = \beta_1 v_{dW} + (1 - \beta_1) dW$$

$$v_{db} = \beta_1 v_{db} + (1 - \beta_1) db$$

$$W = W - \alpha v_{dW},$$

$$b = b - \alpha v_{db}$$

On iteration t:

Compute dW, db on the current mini-batch

$$S_{dW} = \beta_2 S_{dW} + (1 - \beta_2) dW^2$$

$$S_{db} = \beta_2 S_{db} + (1 - \beta_2) db^2$$

$$W = W - \alpha \frac{dW}{\sqrt{S_{dW}}}$$

$$b = b - \alpha \frac{dW}{\sqrt{S_{db}}}$$

Adam optimization

$V_{dW} = 0, S_{dW} = 0, V_{db} = 0, S_{db} = 0.$

On iteration t:

Compute dW, db on the current mini-batch

$$V_{dW} = \beta_1 * VdW + (1 - \beta_1) * dW \qquad V_{db} = \beta_1 * V_{db} + (1 - \beta_1) * db$$

$$S_{dW} = \beta_2 * SdW + (1 - \beta_2) * dW^2 \qquad S_{db} = \beta_2 * S_{db} + (1 - \beta_2) * db^2$$

Momentum

RMSProp

$$V_{dw}' = V_{dw} / (1 - \beta_1^t) \qquad V_{db}' = V_{db} / (1 - \beta_1^t)$$

$$S_{dw}' = S_{dw} / (1 - \beta_2^t) \qquad S_{db}' = S_{db} / (1 - \beta_2^t)$$

Exponential Average weight

$$W = W - \alpha \frac{V_{dw}'}{\sqrt{S_{dw}' + \varepsilon}} \qquad b = b - \alpha \frac{V_{db}'}{\sqrt{S_{db}' + \varepsilon}}$$

Các siêu tham số của Adam

1. α: tốc độ học (learning rate) ➔ tùy chỉnh (tuning) khi chạy.
2. β_1: hệ số học của momentum ➔ giá trị mặc định là 0.9.
   + *First moment: tính dW*
3. β_2: hệ số học của RMSProp ➔ giá trị mặc định là 0.999.
   + *Second moment: dW^2*
4. ε: hệ số canh chỉnh ➔ giá trị mặc định là 10^{-8}.

➔ Adam: *Adaptive moment estimation. (Andrew Ng.)*

Learing rate decay

Learning rate decay

- Đây là một kỹ thuật tối ưu dành cho mạng neural bằng thực nghiệm nhằm tìm ra giá trị tốc độ học α (learning rate) tốt nhất.
- Learning rate gồm 2 bước:
  + Bước 1: Khởi tạo giá trị α. Thường là một con số lớn.
  + Bước 2: Lần lượt giảm giá trị α qua các bước lặp cho đến khi hàm mất mát hội tụ.

https://arxiv.org/pdf/1908.01878.pdf

Ví dụ

– Giả sử có n epoch. Learning rate sẽ giảm qua từng epoch với công thức sau:

$$\alpha = \frac{\alpha_0}{1 + decay_{rate} * epoch_{num}}$$

Ví dụ: α_0=0.2, decay rate = 1

Epoch 1: α = 0.2 / (1+1*1) = 0.1.

Epoch 2: α = 0.2 / (1+1*2) = 0.067.

Epoch 3: α = 0.2 / (1+1*3) = 0.05.

Epoch 4: α = 0.2 / (1+1*4) = 0.04.

Giá trị leanring rate giảm dần.

Một số công thức giảm learning rate khác

- $\alpha = 0.95^{epoch\_num} * \alpha_0$ – $exponetially\_decay$
- $\alpha = \frac{k}{\sqrt{epoch\_num}} * \alpha_0$ (có thể thay $epoch\_num$ bằng t – số bước iteration)
- Tự chỉnh bằng tay (manually)

Tổng kết

- Batch vs Mini-batch training.
  + Batch: train hết dữ liệu. Mini-batch: chía dữ liệu huấn luyện ra thành từng khoảng nhỏ. Mỗi một lần train trên khoảng nhỏ gọi là epoch.
  + Các cách chia batch.
- Exponential Weight Average và Bias Correction: giảm hiện tượng nhiễu khi chạy gradient descent.
- Các thuật toán tăng tốc Gradient descent: Momentum, RMSProp, Adam.
- Tìm ra tham số learning rate tối ưu: sử dụng kỹ thuật learning decay.

TÀI LIỆU THAM KHẢO

1. Khoá học *Neural Network and Deep learning*, deeplearning.ai.
2. Ian Goodfellow, Yoshua Bengio, Aaron Courvile, *Deep learning*, MIT Press, 2016.
3. Andrew Ng., *Machine Learning Yearning*. Link: https://www.deeplearning.ai/machine-learning-yearning/
4. Vũ Hữu Tiệp, *Machine Learning cơ bản*, NXB Khoa học và Kỹ thuật, 2018.