Ứng dụng Trí tuệ nhân tạo trong Nuôi trồng thủy sản

NGUYỄN HẢI TRIỀU[1]

[1] Bộ môn Kỹ thuật phần mềm,

Khoa Công nghệ thông tin, Trường ĐH Nha Trang

NhaTrang, September 2024

Derivatives vs Gradients

Để huấn luyện mô hình Logistic regression hoặc các mô hình Deep Learning chúng ta sử dụng thuật toán "Stochastic Gradient Descent". **Vậy *Gradient* liên quan như thế nào đến đạo hàm riêng *Partial derivative* mà chúng ta đã tìm hiểu trong chương 3.0?**

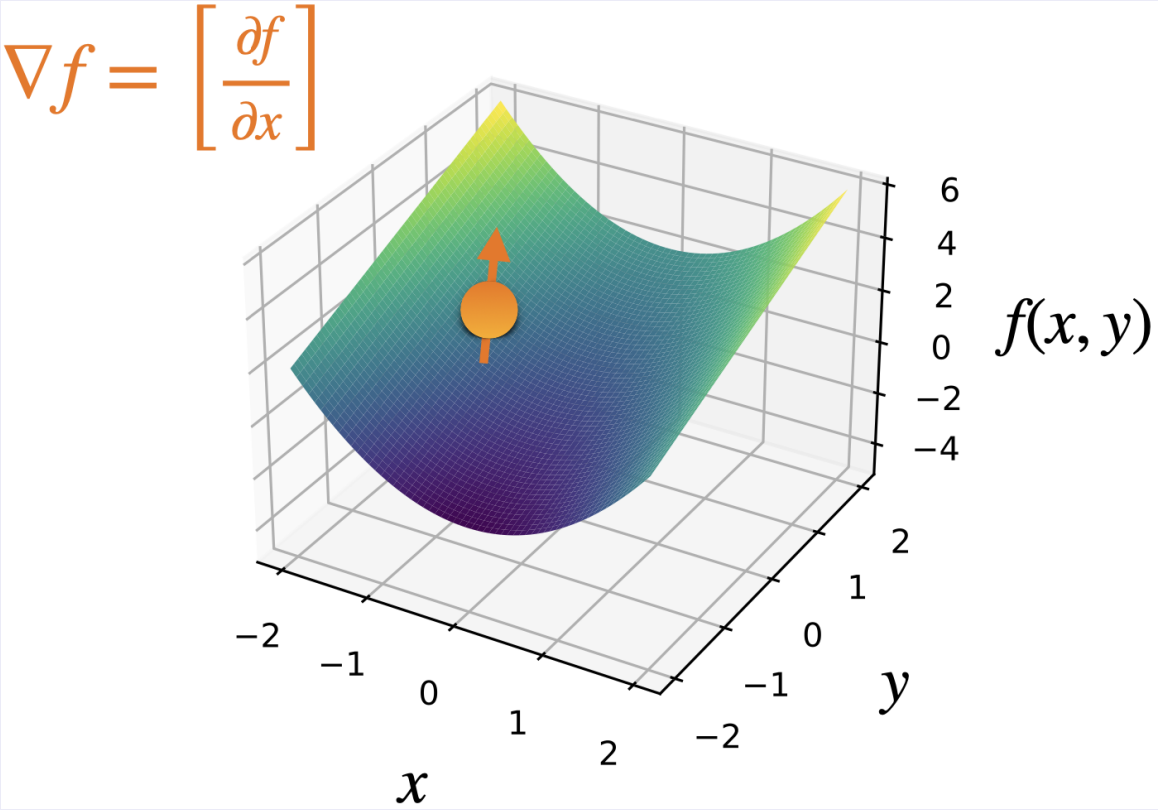
Gradient

Ví dụ một hàm 2 biến $f(x, y) = x^2 + y$, chúng ta có:

Partial derivative	Gradients
The partial derivative is essentially a derivative of one of the variables if a function consists of more than one variable	The gradient of a function is essentially just a way of writing down these partial derivatives in a vector form
$\frac{\partial f}{\partial x} = 2x, \ \frac{\partial f}{\partial y} = 1$	$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x \\ 1 \end{bmatrix}$

Gradient

The gradient of $f(x, y) = x^2 + y$ would be essentially the slope in two dimensions.

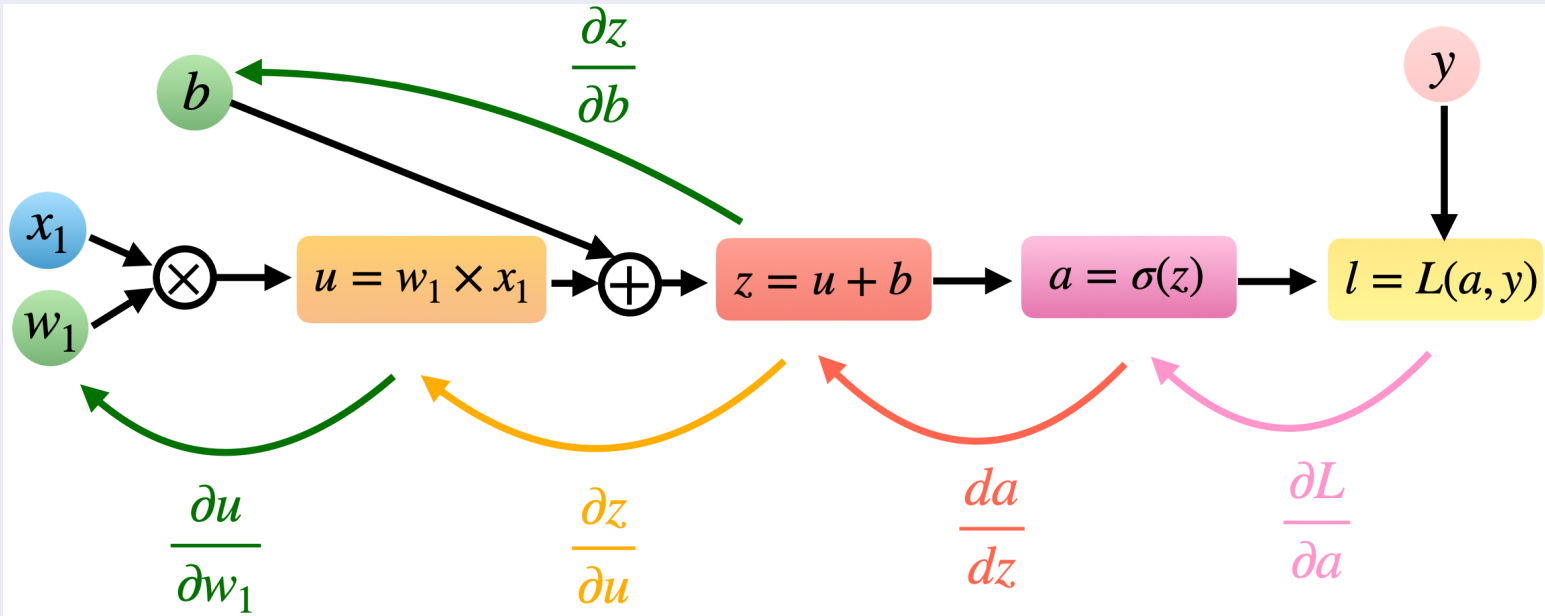


Hình 1: Biểu diễn của gradient f theo hướng x

Gradient

Trong trường hợp tính Gradient của hàm Loss trong Logistic regression ở chương 3.0, ta có thể viết lại như sau:

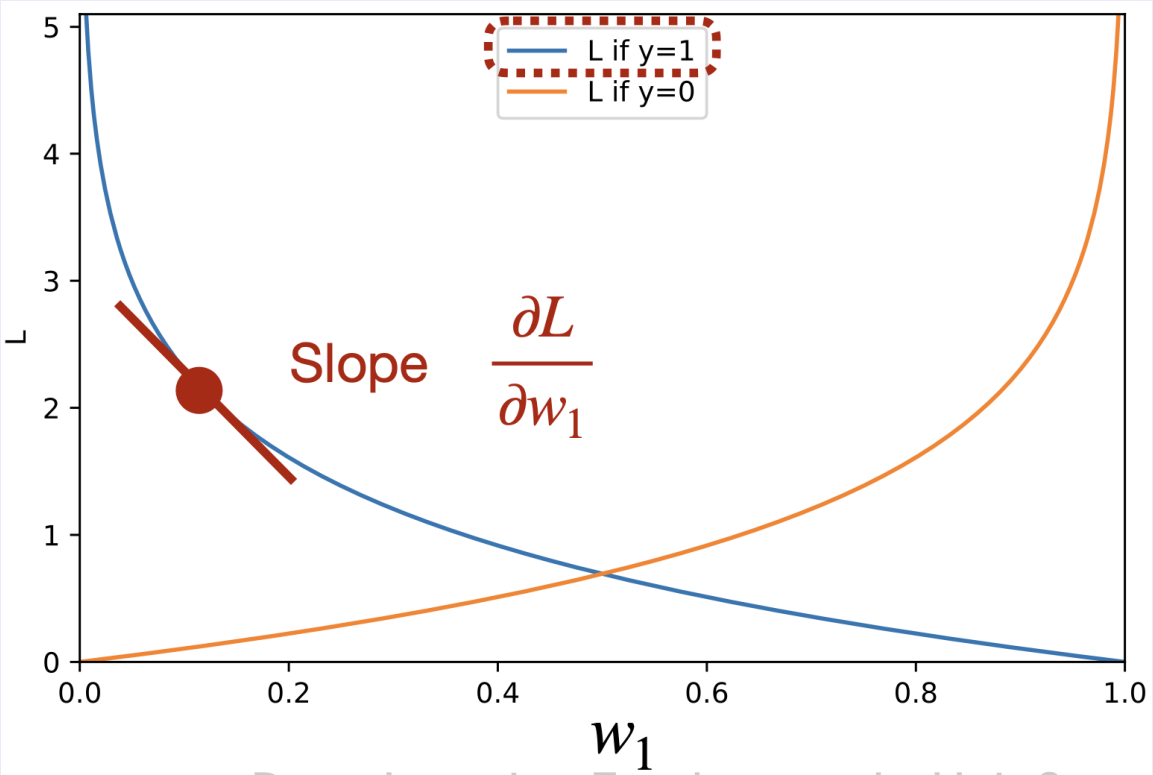
$$\nabla L = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial b} \end{bmatrix} = \begin{bmatrix} \frac{\partial u}{\partial w_1} \times \frac{\partial z}{\partial u} \times \frac{\partial a}{\partial z} \times \frac{\partial L}{\partial a} \\ \frac{\partial z}{\partial b} \times \frac{\partial a}{\partial z} \times \frac{\partial L}{\partial a} \end{bmatrix} \quad (1)$$



Trong đó, w_1, b là 2 tham số mô hình cần được cập nhật trong quá trình huấn luyện.

Minimizing the Loss Using Gradient Descent

Việc tính gradient của hàm loss L theo một trong các tham số mô hình được xem như độ dốc của hàm Loss này tại một điểm nhất định.



Chúng ta có thể sử dụng tính chất này để cực tiểu hàm L.

Thuật toán Gradient Descent

Dựa trên tính chất gradient, ta xây dựng thuật toán Gradient Descent cho hàm Loss:

Gradient descent is an iterative process

Thuật toán là quá trình lặp dựa trên số **training Epochs**

```
For each training epoch:
    overall_loss L = 0

    For each training example:
        output = σ(z)
        overall_loss L = overall_loss L + l(output, label)

    overall_loss L = overall_loss L / n
    compute ∇L_w and ∇L_b
    w = w + α∇L_w
    b = b + α∇L_b
```

Gradient descent is an iterative process

Một epoch tương ứng với thực hiện một lần lặp lại trên tập training. Thuật toán cần lặp lại trên nhiều epoch để cực tiểu loss.

```
For each training epoch:
    overall_loss L = 0

    For each training example:
        output = σ(z)
        overall_loss L = overall_loss L + l(output, label)

    overall_loss L = overall_loss L / n
    compute ∇L_w and ∇L_b
    w = w + α∇L_w
    b = b + α∇L_b
```

In each training epoch, we start with a placeholder value L equals zero.

Gradient descent is an iterative process

We iterate through the training examples.

```
For each training epoch:
    overall_loss L = 0

    For each training example:
        output = σ(z)
        overall_loss L = overall_loss L + l(output, label)

    overall_loss L = overall_loss L / n
    compute ∇L_w and ∇L_b
    w = w + α∇L_w
    b = b + α∇L_b
```

we compute the loss of *this current training example*, using the loss function between $\sigma(z)$ and the true class label. And this loss value for this training example, we add this to the overall loss.

Gradient descent is an iterative process

we normalize the overall loss by dividing by n, where n here is the size of our training.

```
For each training epoch:
    overall_loss L = 0

    For each training example:
        output = σ(z)
        overall_loss L = overall_loss L + l(output, label)

    overall_loss L = overall_loss L / n
    compute ∇L_w and ∇L_b
    w = w + α∇L_w
    b = b + α∇L_b
```

This make the loss onto a reasonable scale.

Gradient descent is an iterative process

We can then use this loss to compute the gradient with respect to the model parameters w and b.

```
For each training epoch:
    overall_loss L = 0

    For each training example:
        output = σ(z)
        overall_loss L = overall_loss L + l(output, label)

    overall_loss L = overall_loss L / n
    compute ∇L_w and ∇L_b
    w = w + α∇L_w
    b = b + α∇L_b
```

Gradient descent is an iterative process

We have these gradients, we use these gradients to update the model parameters by multiplying the gradient with α, where α is the so-called learning rate.

```
For each training epoch:
    overall_loss L = 0

    For each training example:
        output = σ(z)
        overall_loss L = overall_loss L + l(output, label)

    overall_loss L = overall_loss L / n
    compute ∇L_w and ∇L_b
    w = w + α∇L_w
    b = b + α∇L_b
```

Gradient descent is an iterative process

α is essentially a so-called hyperparameter. It's essentially just a small value that **scales the loss so that our updates are not too large, because large updates might disturb** the training.

```
For each training epoch:
    overall_loss L = 0

    For each training example:
        output = σ(z)
        overall_loss L = overall_loss L + l(output, label)

    overall_loss L = overall_loss L / n
    compute ∇L_w and ∇L_b
    w = w + α∇L_w
    b = b + α∇L_b
```

The Gradient Descent Algorithm

```
For each training epoch:
    overall_loss L = 0

    For each training example:
        output = σ(z)
        overall_loss L = overall_loss L + l(output, label)

    overall_loss L = overall_loss L / n
    compute ∇L_w and ∇L_b
    w = w + α∇L_w
    b = b + α∇L_b
```

Computes the loss over the whole epoch

The Gradient Descent Algorithm

```
For each training epoch:
    overall_loss L = 0

    For each training example:
        output = σ(z)
        overall_loss L = overall_loss L + l(output, label)

    overall_loss L = overall_loss L / n
    compute ∇L_w and ∇L_b
    w = w + α∇L_w
    b = b + α∇L_b
```

Update parameters once per epoch

Gradient Descent Vs Stochastic Gradient Descent

Bên cạnh thuật toán Gradient Descent đã được trình bày ở trước, chúng ta sẽ tìm hiểu thuật toán **Stochastic gradient descent** – một biến thể của Gradient Descent.

Stochastic Gradient Descent

Stochastic gradient descent is the flavor of gradient descent with essentially more frequent updates.

The Stochastic Gradient Descent Algorithm

Computes the loss and *updates for each training example*

1 For each training epoch:
2 For each training example:
3 output $= \sigma(z)$
4 $L = l(output, label)$
5 compute ∇L_w **and** ∇L_b
6 $\mathbf{w} = \mathbf{w} + \alpha \nabla L_w$
7 $b = b + \alpha \nabla L_b$

In stochastic gradient descent, we compute the gradient of the loss for a single training example. And then using this loss, we update the model parameters.

The Stochastic Gradient Descent Algorithm

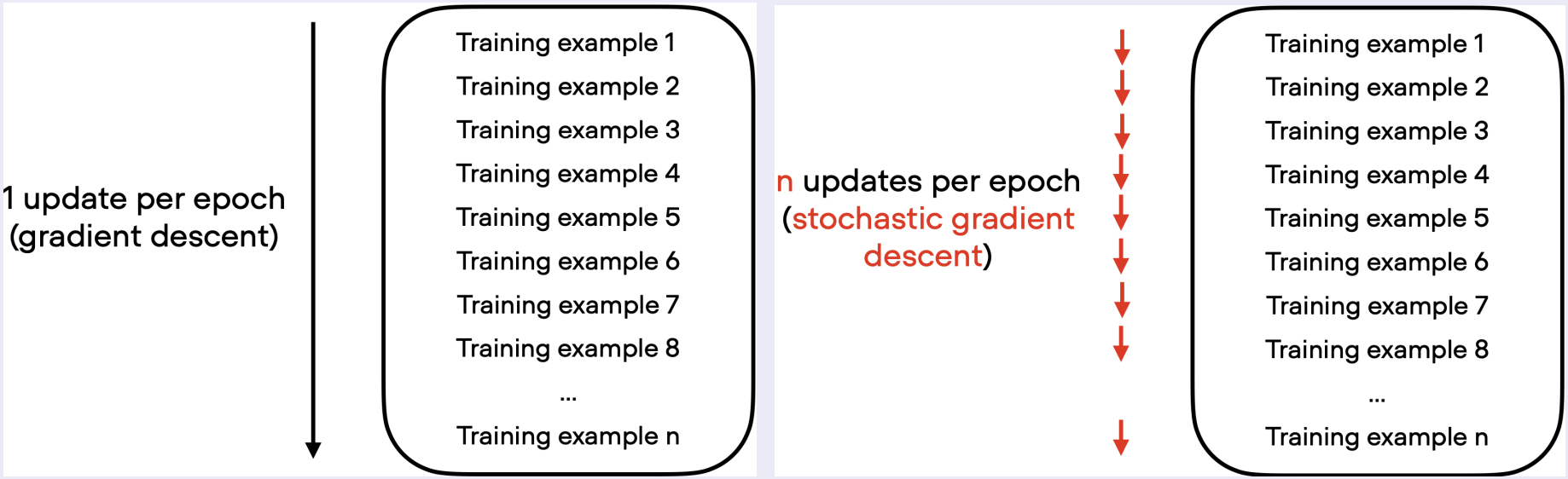
Tuy nhiên, việc cập nhật model ở các tranining example không hợp lý về mặt chi phí tính toán và dùng loss của 1 training example để xấp xỉ cho cả overall_loss trong thuật toán gradient descent.

For each training epoch:
  For each training example:
    output $= \sigma(z)$
    $L = l(output, label)$
    compute ∇L_w **and** ∇L_b
    $\mathbf{w} = \mathbf{w} + \alpha \nabla L_w$
    $b = b + \alpha \nabla L_b$

Để cải thiện nhược điểm của Stochastic gradient descent, chúng ta sử dụng Minibatch Gradient Descent.

The Minibatch Gradient Descent Algorithm

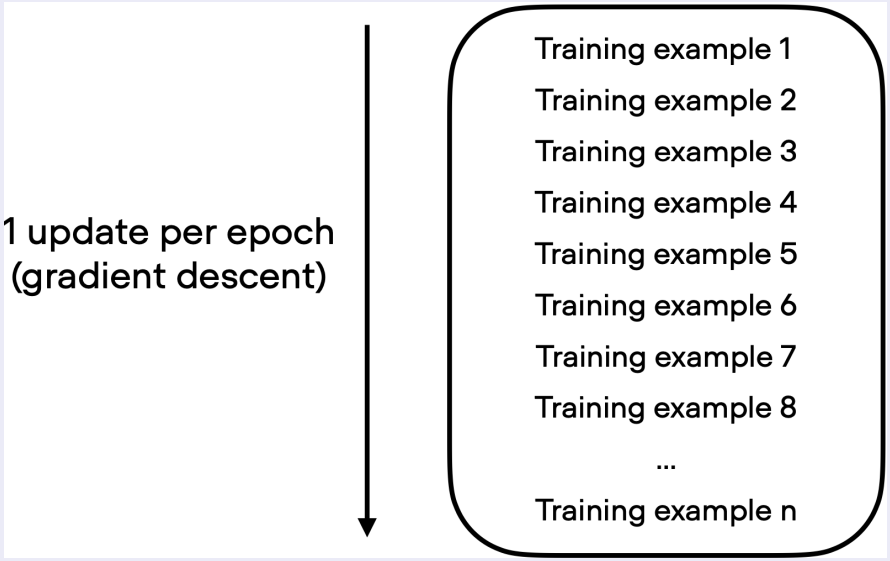
Minibatch Gradient Descent về bản chất là một xấp xỉ của Stochastic gradient descent nhưng thay vì cập nhật trọng số mô hình sau mỗi training example thì nó cập nhật sau mỗi mimibatch.



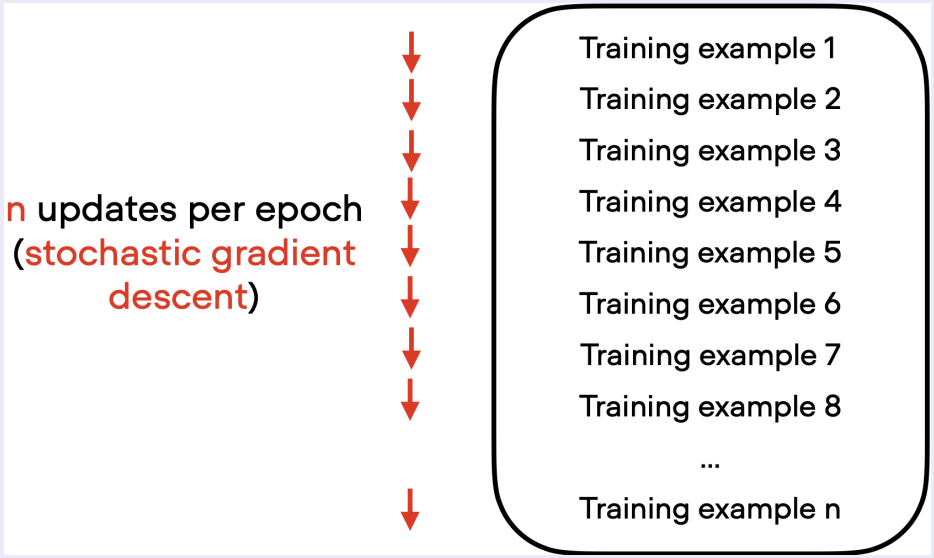
(a) (b)

Hình 2: Có thể xem Minibatch Gradient Descent như là thuật toán lai giữa Gradient descent và Stochastic gradient descent.

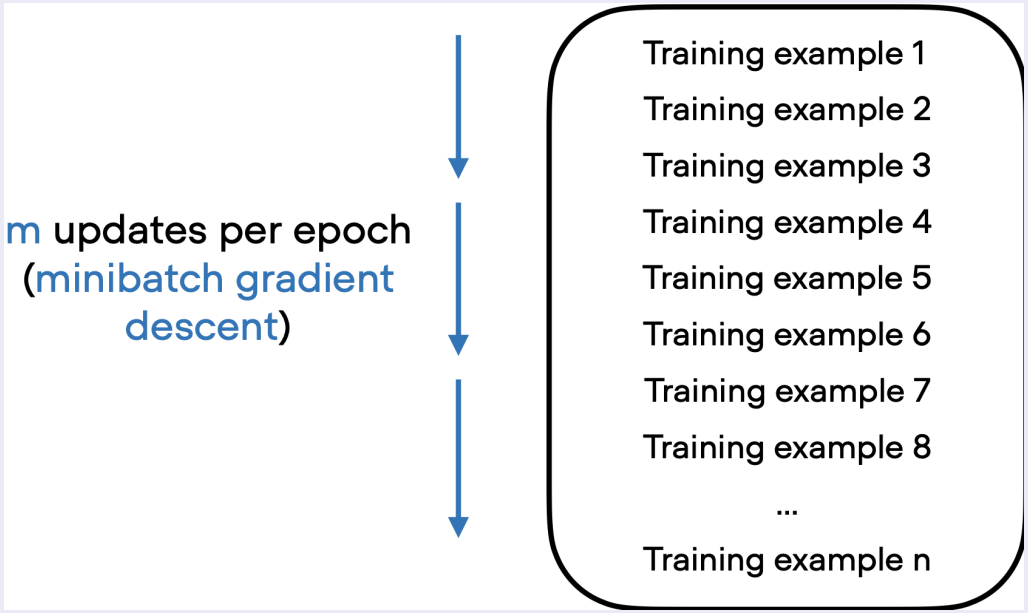
The Minibatch Gradient Descent Algorithm



(a)

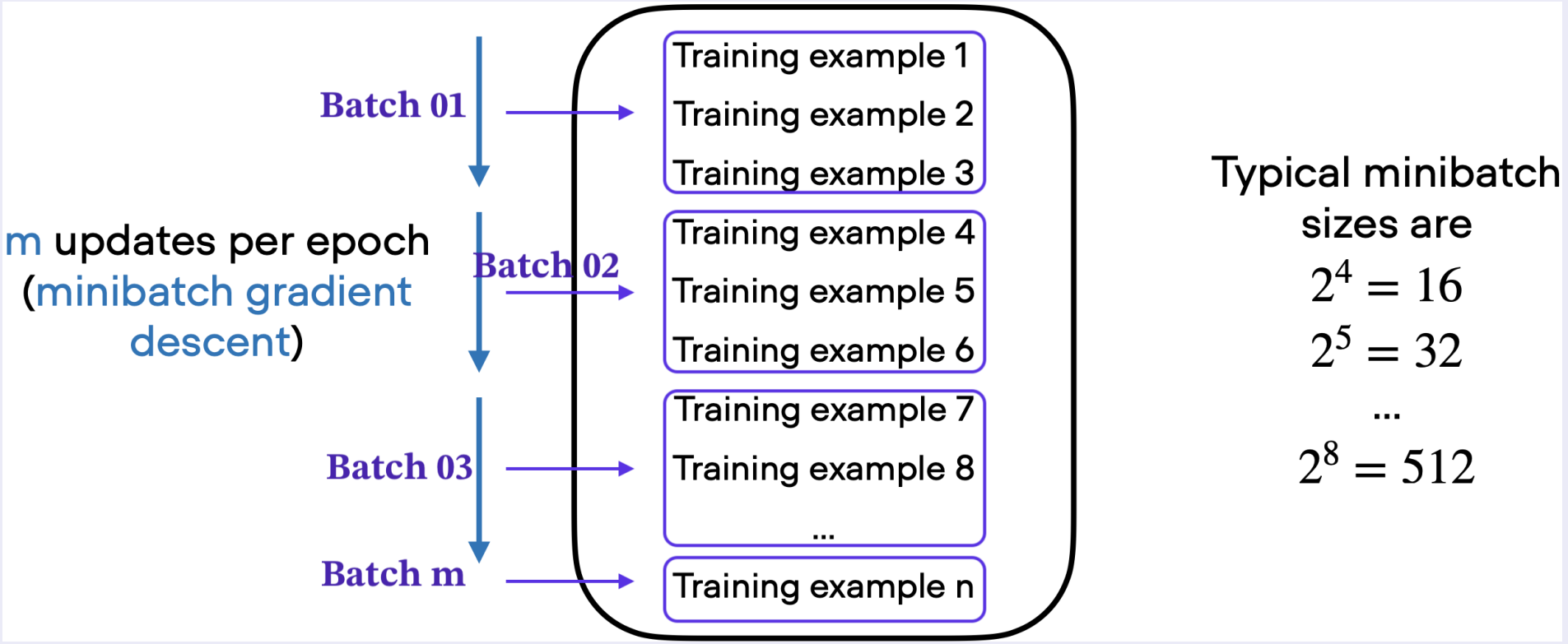


(b)

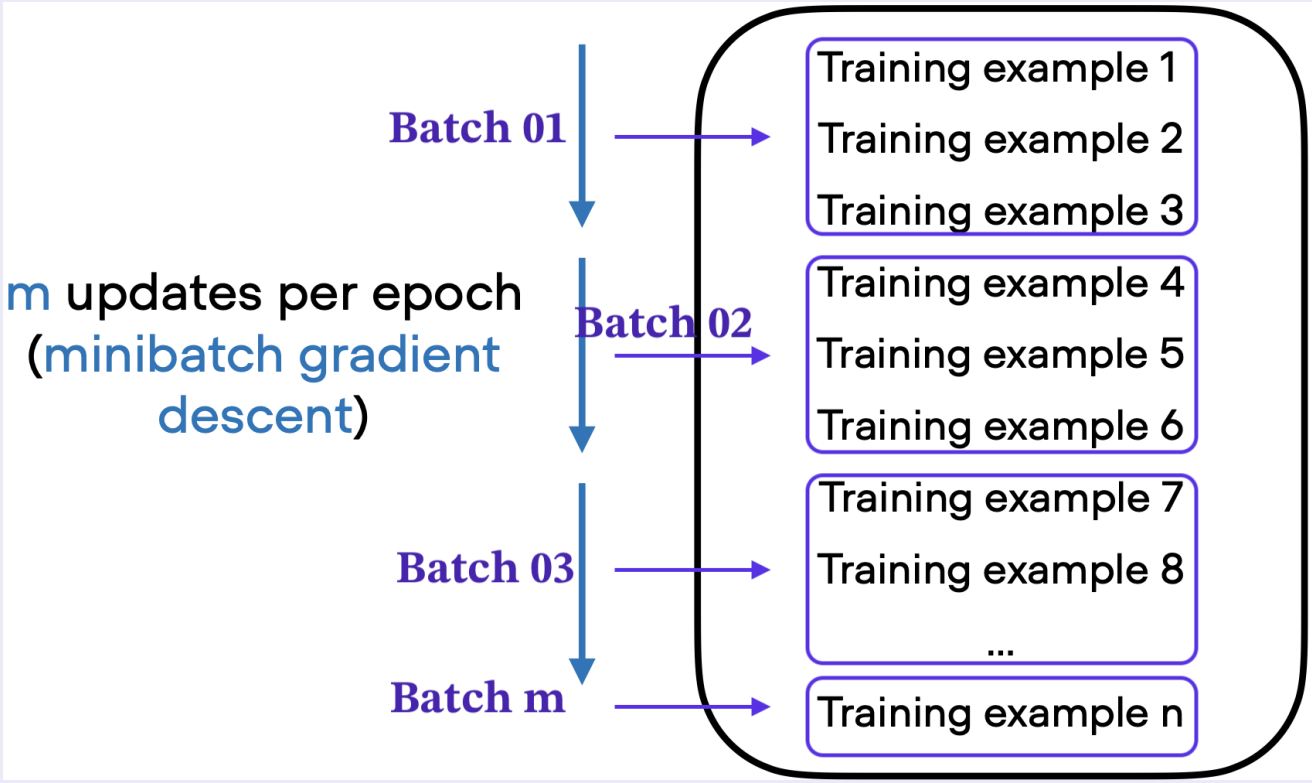


The Minibatch Gradient Descent Algorithm

Minibatch gradient descent, we form these small groups or batches of training examples, and we will make one update after each batch.



The Minibatch Gradient Descent Algorithm



Typical minibatch sizes are

$2^4 = 16$

$2^5 = 32$

...

$2^8 = 512$

Lưu ý: ở đây m là số lượng batches trên tập dữ liệu huấn luyện.
Giống như learning rate α, số lượng training example trong mỗi batch là minibatch sizes cũng là siêu tham số cần tìm đủ tốt để tối ưu huấn luyện. Thông thường ta chọn minibatch size là 2 mũ để phù hợp với kiến trúc của GPU.

Minibatch Gradient Descent Algorithm

Giống như thuật toán Gradient và Stochastic gradient descent, Minibatch Gradient Descent cũng lặp qua các training epochs

```
For each training epoch:
    For each minibatch:
        overall_loss L = 0
        For each training example in minibatch:
            output = σ(z)
            overall_loss L = overall_loss L + l(output, label)
        overall_loss L = overall_loss L/n
        compute ∇L_w and ∇L_b
        w = w + α∇L_w
        b = b + α∇L_b
```

Minibatch Gradient Descent Algorithm

Tuy nhiên, ta sẽ lặp trên số lượng các minibatches

```
For each training epoch:
    For each minibatch:
        overall_loss L = 0
        For each training example in minibatch:
            output = σ(z)
            overall_loss L = overall_loss L + l(output, label)
        overall_loss L = overall_loss L/n
        compute ∇L_w and ∇L_b
        w = w + α∇L_w
        b = b + α∇L_b
```

Minibatch Gradient Descent Algorithm

Còn lại về bản chất tính toán như Gradient descent, chỉ khác là thay vì tính toán, cập nhật trọng số cho toàn bộ dữ liệu huấn luyện thì ta thực hiện trên dữ liệu huấn luyện của từng minibatch.

```
For each training epoch:
    For each minibatch:    Gradient descent
        overall_loss L = 0
        For each training example in minibatch:
            output = σ(z)
            overall_loss L = overall_loss L + l(output, label)
        overall_loss L = overall_loss L/n
        compute ∇L_w and ∇L_b
        w = w + α∇L_w
        b = b + α∇L_b
```

Minibatch Gradient Descent Algorithm

Lưu ý: vòng lặp trong minibatch có thể sử dụng đại số tuyến tính để tính toán.

1 For each training epoch:
2 For each minibatch:
3 overall_loss $L = 0$
4 For each training example **in** minibatch:
5 output $= \sigma(z)$
6 overall_loss L = overall_loss $L + l(output, label)$
7 overall_loss L = overall_loss L/n
8 compute ∇L_w **and** ∇L_b
9 $\mathbf{w} = \mathbf{w} + \alpha \nabla L_w$
10 $b = b + \alpha \nabla L_b$

For-loop can be replaced with matrix multiplication

Minibatch Gradient Descent Algorithm

Ưu điểm của Minibatch Gradient Descent:

1. Less noisy loss compared to stochastic gradient descent with 1 training example.
2. Learn faster than gradient descent because more than 1 update per epoch.
3. Better GPU utilization than stochastic gradient descent by using linear algebra concepts (matrix multiplication).

Tài liệu tham khảo

Sebastian Raschka, Yuxi (Hayden) Liu, Vahid Mirjalili
Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python (2022). Published by Packt Publishing Ltd, ISBN 978-1-80181-931-2.

Sebastian Raschka
MACHINE LEARNING Q AND AI: 30 Essential Questions and Answers on Machine Learning and AI (2024). ISBN-13: 978-1-7185-0377-9 (ebook).

LightningAI
LightningAI: PyTorch Lightning (2024) .