HỌC MÁY – BÁO CÁO CUỐI KÌ

PHÂN LOẠI ẢNH TRANG PHỤC

BẰNG MỘT SỐ PHƯƠNG PHÁP HỌC MÁY

20127449 – Trần Quốc Bảo

20127452 – Hồ Đăng Cao

20127476 – Đỗ Đức Duy

Giảng viên hướng dẫn

Thầy. Nguyễn Ngọc Đức

5/16/2023

Mục lục

I. GIỚI THIỆU VẤN ĐỀ

1. Đưa ra bài toán

Chúng ta có một vài bức ảnh về trang phục như giày, dép, áo, túi, ... Làm sao để máy tính có thể phân loại các bức ảnh đó về đúng loại trang phục tương ứng?

Để làm được điều đó, chúng ta sẽ áp dụng các thuật toán học máy để huấn luyện trên tập dữ liệu Fashion-MNIST.

2. Giới thiệu về tập dữ liệu

2.1. Ngữ cảnh

Fashion-MNIST là một bộ dữ liệu về hình ảnh bao gồm một bộ dữ liệu huấn luyện gồm 60.000 mẫu và một bộ kiểm tra gồm 10.000 mẫu. Mỗi mẫu là một hình ảnh được liên kết với nhãn từ 10 lớp.

2.2. Nội dung

Mỗi hình ảnh có kích thước 28×28 pixels, các giá trị pixel ở mỗi dòng pixel được xép thành 1 hàng tổng cộng là 784 pixels. Mỗi pixel là một số nguyên từ 0 (màu trắng) đến 255 (màu đen), cho biết độ xám của pixel đó.

Bộ dữ liệu có 785 cột. Cột đầu tiên bao gồm các nhãn lớp tương ứng với quần áo. Các cột còn lại của mỗi dòng (mẫu) chứa các giá trị pixel của hình ảnh tương ứng.

2.3. Nhãn (lớp)

Mỗi mẫu ảnh (mỗi dòng dữ liệu) được gán cho một trong các nhãn sau:

0. T-shirt/top	2. Pullover	4. Coat	6. Shirt	8. Bag
1. Trouser	3. Dress	5. Sandal	7. Sneaker	9. Ankle boot

2.4.Tóm tắt dữ liệu

Mỗi hàng là một hình ảnh riêng biệt.

Cột 1 là nhãn lớp.

Các cột còn lại là số pixel (tổng cộng 784).

Mỗi giá trị là độ xám của pixel (1 đến 255).

3. Ý tưởng giải quyết

3.1. Đánh giá sơ bộ về bài toán

Đây là bài toán phân loại ảnh và các ảnh thuộc nhiều lớp khác nhau, gọi tắt thì đây là bài toán phân loại nhiều lớp.

3.2. Chọn thuật toán

Vì là bài toán phân loại nhiều lớp nên cần có một thuật toán thích hợp để phân lớp cho bài toán này. Trong bài báo cáo này, chúng ta sẽ sử dụng hàm **Softmax,** kĩ thuật **Mini-batch gradient descent** và **CNN** để giải quyết bài toán trên. Chi tiết về các thuật toán và lý do chọn cho sự lựa chọn này sẽ được đề cập đến ở các mục sau.

II. MINI-BATCH GRADIENT DESCENT (Mini-Batch GD)

1. Nhắc lại về Gradient Descent (GD)

Trong học máy nói riêng và toán tối ưu nói chung, việc tìm global minimum của các hàm mất mát là rất khó. Do sự phức tạp khi đạo hàm, dữ liệu có số chiều lớn, hoặc dữ liệu có kích thước lớn nên việc giải phương trình đạo hàm bằng 0 để tìm các điểm local minimum (giá trị nhỏ nhất theo 1 mức độ nào đó) là rất phức tạp, thậm chí bất khả thi.

Hướng tiếp cận phổ biến là xuất phát từ một điểm gần với nghiệm của bài toán, dùng vòng lặp để tiến dần đến điểm làm đạo hàm bằng 0.

Gradient Descent (GD) và các biến thể của nó là một trong những phương pháp được dùng nhiều nhất.

Thuật toán Gradient Descent:

2. Dự đoán một điểm khởi tạo $\theta = \theta_0$.
3. Cập nhật θ đến khi đạt được kết quả chấp nhận được:

$$\theta = \theta - \eta \nabla_\theta J(\theta)$$

với $\nabla_\theta J(\theta)$ là đạo hàm của hàm mất mát tại θ

2. Mini-Batch GD

Mini-Batch GD là một biến thể của GD. Thuật toán có nhiều lợi ích, gồm:

- Hiệu quả tính toán: Mini-batch GD có thể tính toán gradient nhanh hơn GD thông thường, vì nó chỉ tính toán gradient trên một tập con của dữ liệu huấn luyện.
- Tăng tốc độ hội tụ: Mini-batch GD thường hội tụ nhanh hơn GD thông thường, vì nó cho phép các bước cập nhật trọng số được thực hiện nhanh hơn.

- Tránh bị rơi vào cực tiểu cục bộ: vì Mini-batch GD cho phép mô hình đưa ra các bước đi lớn hơn khi gradient có giá trị lớn.

Mini-batch GD bắt đầu mỗi epoch bằng việc xáo trộn ngẫu nhiên dữ liệu rồi chia toàn bộ dữ liệu thành các mini-batch, mỗi mini-batch có n điểm dữ liệu (trừ mini-batch cuối có thể có ít hơn nếu số lượng mẫu dữ liệu không chia hết cho n). Mỗi lần cập nhật, thuật toán này lấy ra một mini batch để tính toán đạo hàm của hàm mất rồi cập nhật:

$$\theta = \theta - \eta \nabla_\theta J(\theta;\ x_{i:i+n};\ y_{i:i+n})$$

Với $x_{i:i+n}$ là dữ liệu trong mỗi mini batch. Dữ liệu này sau mỗi epoch là khác nhau vì chúng cần được xáo trộn. Mini-batch GD được sử dụng trong hầu hết các thuật toán học máy, đặc biệt là trong Deep Learning.

3. Điều kiện dừng

Giới hạn số vòng lặp (các epochs): đây là phương pháp phổ biến nhất và cũng để đảm bảo rằng chương trình chạy không quá lâu. Tuy nhiên, một nhược điểm của cách làm này là có thể thuật toán dừng lại trước khi đủ gần với nghiệm.

Trong Mini-batch GD, cách thường dùng là so sánh nghiệm sau một vài lần cập nhật.

III. SOFTMAX REGRESSION

1. Động lực nghiên cứu

Trong thực tế, các bài toán phân lớp thường có rất nhiều lớp (multi-class), các bộ phân lớp nhị phân (binary classifiers) mặc dù có thể áp dụng cho các bài toán multi-class, chúng vẫn có những hạn chế nhất định. Với bộ phân lớp nhị phân, kỹ thuật được sử dụng nhiều nhất one-vs-rest có một hạn chế về tổng các xác suất (tổng các xác suất có thể khác 1). Hồi quy Softmax là

phương pháp tổng quát của hồi quy Logistic, có thể khắc phục được hạn chế nêu trên, cũng được sử dụng rộng rãi như một phương pháp phân lớp.

Ví dụ: để phân biệt 1 bức ảnh đầu vào là chó, mèo hay chuột, Softmax sẽ đảm bảo được tổng xác suất để bức ảnh là chó, mèo hay chuột là 1.

Hồi quy Softmax phù hợp với bài toán đặt ra ở mục I.

2. Phát biểu bài toán

Các tham số:

- Một tập n dòng dữ liệu $X \in \mathbb{R}^{n\times(d+1)}$ (d là số chiều + phần tử 1 là hệ số tự do *bias)*.
- $Y \in \mathbb{N}^n$: nhãn thực tế của n dòng dữ liệu.
- C: số lớp (nhãn) để phân lớp.
- $W \in \mathbb{R}^{(d+1)\times C}$: ma trận trọng số.
- $Z \in \mathbb{R}^{n\times C} = XW$
- $A = [a_{ij}] \in \mathbb{R}^{n\times C}$, với $i = 1,2,\dots,n; j = 1,2,\dots,C$
- $a_{ij} = f(z_{ij});\ z_{ij} \in Z$

Giả sử: $P(y_i = j | x_i; w_j) = a_{ij} (y_i \in Y; x_i \in X;\ 0 < a_{ij} < 1$: là xác suất x_i rơi vào lớp j.

Ở đây, ta cần một mô hình xác suất $f(z_{ij})$ sao cho $\sum_{j=1}^{C} a_{ij} = 1$ và z_{ij} càng lớn thì a_{ij} càng cao, vậy ta cần hàm đồng biến.

Hàm *Softmax* thỏa $f(z_{ij})$ nêu trên*:* $f(z_{ij}) = \frac{\exp(z_{ij})}{\sum_{j=1}^{C} \exp(z_{ij})}, \forall j = 1,2,\dots C$

3. Hàm mất mát và tối ưu

a. One hot coding

Với ma trận đầu ra dự đoán xác suất A như trên, cần chuyển đầu ra thực sự Y thành 1 ma trận one-hot mà mỗi dòng là 1 vector có C phần tử - tương ứng C lớp và đúng 1 phần tử bằng 1 (xác suất điểm dữ liệu x_i rơi vào lớp này là 1), còn lại bằng 0.

$$Y \in \mathbb{N}^n \to Y_{one_{hot}} = [y_{ij}] \in \mathbb{N}^{n \times C}, s.t: \sum_{j=1}^{C} y_{ij} = 1$$

Hàm mất mát sẽ được xây dựng để tính độ chênh lệch giữa *đầu ra dự đoán* A và *đầu ra thực sự* $Y_{one_{hot}}$.

b. Hàm mất mát

Sử dụng Cross entropy cho tất cả cặp dữ liệu $x_i, y_i, i = 1,2, \dots n$, ta được hàm mất mát:

$$J(W; X; Y) = -\sum_{i=1}^{n}\sum_{j=1}^{C} y_{ij} \log(a_{ij}) = -\sum_{i=1}^{n}\sum_{j=1}^{C} y_{ij} \log\left(\frac{\exp(x_i w_j)}{\sum_{k=1}^{C} \exp(x_i w_k)}\right)$$

Với ma trận trọng số **W** là biến cần tối ưu.

c. Tối ưu hàm mất mát

Với mỗi cặp dữ liệu (x_i, y_i), ta có:

$$J_i(W) = J(W; x_i; y_i)$$

$$= -\sum_{j=1}^{C} y_{ij} \log\left(\frac{\exp(x_i w_j)}{\sum_{k=1}^{C} \exp(x_i w_k)}\right)$$

$$= -\sum_{j=1}^{C} y_{ij} \left(x_i w_j - \log\left(\sum_{k=1}^{C} \exp(x_i w_k)\right)\right)$$

$$= -\sum_{j=1}^{C} y_{ij} x_i w_j + \sum_{j=1}^{C} y_{ij} \log\left(\sum_{k=1}^{C} \exp(x_i w_k)\right)$$

$$= -\sum_{j=1}^{C} y_{ij} x_i w_j + \log\left(\sum_{k=1}^{C} \exp(x_i w_k)\right)$$

$$\text{Do } (\sum_{j=1}^{C} y_{ij} = 1)$$

Ta có ma trận gradient:

$$\frac{\partial J_i(W)}{\partial W} = \left[\frac{\partial J_i(W)}{\partial W_j}\right]; j = 1,2,..C$$

Mà

$$\frac{\partial J_i(W)}{\partial W_j} = -y_{ij}x_i + \frac{\exp(x_i w_j)}{\sum_{k=1}^{C} \exp(x_i w_k)} x_i$$

$$= -y_{ij}x_i + a_{ij}x_i = x_i(a_{ij} - y_{ij})$$

Nên

$$\frac{\partial J_i(W)}{\partial W} = x_i^T[a_{ij} - y_{ij}]; j = 1,2,..C$$

Suy ra

$$\frac{\partial J(W)}{\partial W} = \sum_{i=1}^{n} x_i^T(a_i - y_i) = X^T(A - Y\_one\_hot)$$

Áp dụng cho Mini-batch GD với $mini\ batch\ size = mbs$, công thức cập nhật cho W là:

$$W = W - \frac{\eta}{mbs}\ x_{i:i+mbs}^T(a_{i:i+mbs} - y_{i:i+mbs})$$

4. Hạn chế

Dễ bị overfitting: Softmax regression có thể dễ bị overfitting vì có quá nhiều tham số cần được tối ưu, đặc biệt là khi số lượng biến độc lập lớn.

Yêu cầu dữ liệu lớn: Softmax regression yêu cầu một lượng dữ liệu lớn để huấn luyện hiệu quả và tránh overfitting. Điều này làm cho phương pháp này không phù hợp khi có ít dữ liệu hoặc dữ liệu thưa.

Không thể xử lý dữ liệu phi tuyến: Softmax regression là một phương pháp tuyến tính và không thể xử lý được dữ liệu phi tuyến.

Không giải quyết được vấn đề khả năng khái quát (generalization): Phương pháp này không giải quyết được vấn đề khả năng khái quát và có thể không hoạt động tốt khi áp dụng cho các bài toán dự đoán trên dữ liệu mới mà chưa được huấn luyện.

IV. CNN

1. Động lực nghiên cứu

Mạng CNN là một trong những động lực thúc đẩy mạnh mẽ của lĩnh vực khoa học máy tính nói chung và thị giác máy tính nói riêng. Hiện nay cộng đồng nghiên cứu đã và đang tiếp tục nghiên cứu và phát triển những mạng CNN mới với hiệu năng tốt hơn.

CNN là kiến trúc lý tưởng khi giải quyết vấn đề dữ liệu hình ảnh, một trong những mô hình Deep Learning tiên tiến. Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay. CNN được sử dụng nhiều trong các bài toán nhận dạng các object trong ảnh.

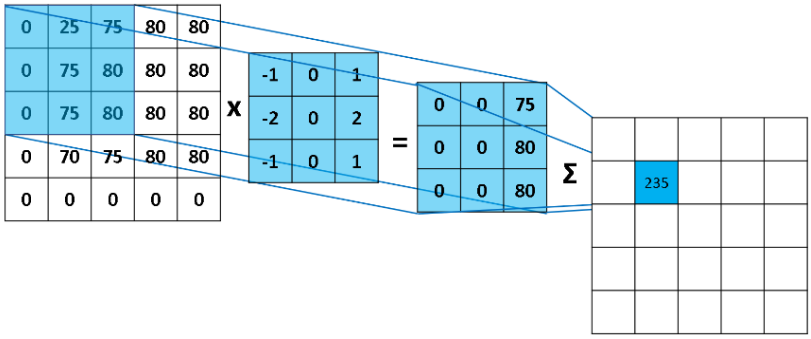
2. Phát biểu bài toán

Bài toán mà nhóm trình bày là dùng CNN để phân loại hình ảnh.

Mô hình CNN để huấn luyện và kiểm tra, mỗi hình ảnh đầu vào sẽ chuyển nó qua 1 loạt các lớp tích chập với các bộ lọc (Kernel), tổng hợp lại các lớp được kết nối đầy đủ (Full Connected) và áp dụng hàm Softmax để phân loại đối tượng có giá trị xác suất giữa 0 và 1.

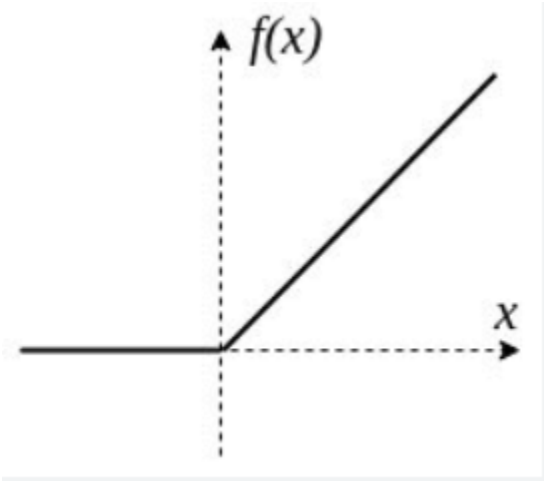
Các lớp trong CNN.

Lớp tích chập – Convolution Layer: là lớp đầu tiên trích xuất các tính năng từ hình ảnh đầu vào. Tích chập duy trì mối quan hệ giữa các pixel bằng cách tìm hiểu các tính năng hình ảnh bằng cách sử dụng các ô vuông nhỏ của dữ liệu đầu vào (gọi là kernel). Kích thước của kernel thường là các số lẻ như 3x3, 5x5.



Bước nhảy Stride: số pixel cần dịch chuyển mỗi khi trượt kernel qua bức ảnh đầu vào. Ví dụ với stride bằng 2 sẽ dịch chuyển 2 pixels mỗi lần áp dụng phép convolution.

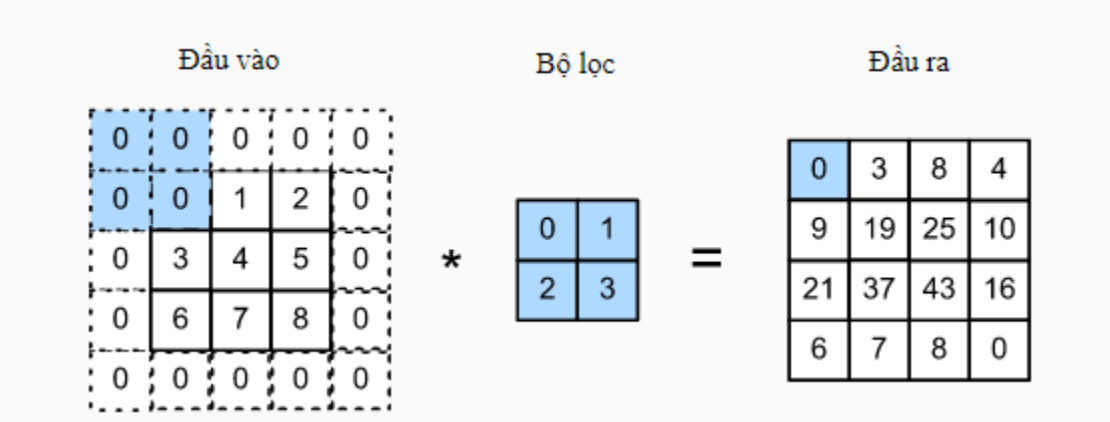
Hàm phi tuyến ReLU (Rectified Linear Unit) có đầu ra là: $f(x) = \max(0, x)$



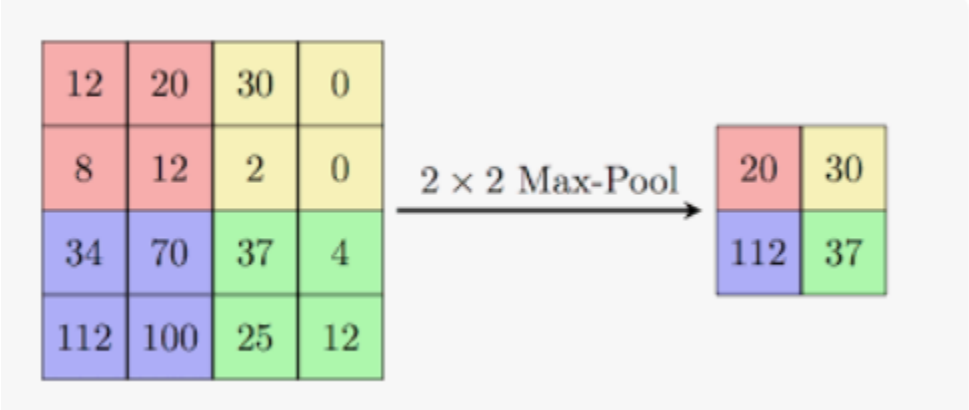
Trong activation function chúng còn có nhiều hàm khác: ReLU, Tanh, Sigmoid, Maxout, Leaky, … ReLU layer được ứng dụng phổ biến trong việc huấn luyện neuron do sở hữu nhiều ưu điểm tiên tiến.

Đường viền padding: Một vấn đề rắc rối khi áp dụng các tầng tích chập là việc chúng ta có thể mất một số điểm ảnh trên biên của ảnh. Vì chúng ta thường sử dụng các kernel nhỏ, với một phép tích chập ta có thể chỉ mất một ít điểm ảnh, tuy nhiên sự mất mát này có thể tích lũy dần khi ta thực hiện qua nhiều tầng tích chập liên tiếp. Một

giải pháp đơn giản cho vấn đề này là chèn thêm các điểm ảnh xung quanh đường biên trên bức ảnh đầu vào (được gọi là padding).

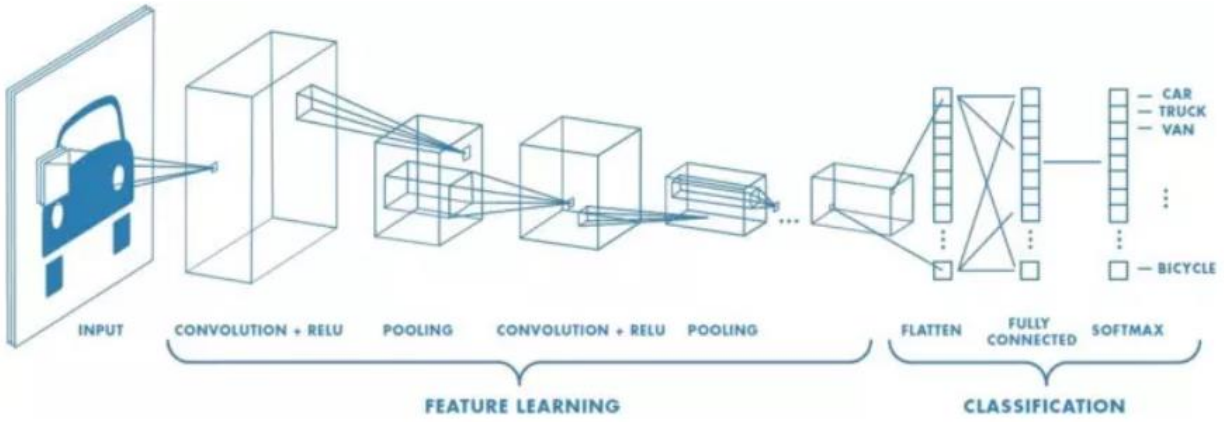


Lớp Pooling: sẽ giảm bớt số lượng tham số khi hình ảnh quá lớn. Không gian pooling còn được gọi là lấy mẫu con hoặc lấy mẫu xuống làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng. Các pooling có nhiều loại khác nhau: Max pooling, Averange pooling.



Fully connected layer: Sau khi ảnh được truyền qua nhiều convolutional layer và pooling layer thì model đã học được tương đối các đặc điểm của ảnh, thì tensor của output của layer cuối cùng sẽ được là phẳng thành vector và đưa vào một lớp được kết nối như một mạng neuron. Với FC layer được kết hợp với các tính năng lại với nhau để tạo ra một mô hình. Cuối cùng sử dụng Softmax để phân loại đầu ra.

Cấu trúc CNN đơn giản:



CNN gồm tập hợp các lớp cơ bản: convolution layer + ReLU, pooling layer, fully connected layer. Các lớp này liên kết với nhau theo một thứ tự nhất định. Thông thường, một ảnh sẽ được lan truyền qua tầng convolution layer + ReLU layer đầu tiên, sau đó các giá trị tính toán được sẽ lan truyền qua pooling layer, bộ ba convolution layer + ReLU + pooling layer có thể lặp lại nhiều lần trong network. Và sau đó được lan truyền qua tầng fully connected layer và softmax để tính sác xuất ảnh đó chứa vật thế gì.

3. Kiến trúc mạng VGG16

VGG16 là mạng CNN nổi tiếng được nộp cho ILSVRC-2014. Model sau khi được train bởi mạng VGG16 đạt độ chính xác 92,7% - top 5 test trong tập dữ liệu ImageNet gồm 14 triệu hình ảnh thuộc 1000 lớp khác nhau.

3.1. Cấu trúc

Đầu vào của VGG được đặt thành ảnh với 3 RGB channel có kích thước cố định là 224x224.

VGG16 có 3 lớp fully connected layers, 13 convolutional layer, và 5 lớp Max pooling. Tổng có 21 lớp nhưng chỉ có 16 lớp trọng số.

Convolutional layers: có kernel size là 3x3, stride =1 và padding là same. Conv-1 layer có 64 số filters, conv-2 layer có 128 filters, conv-3 có 256 filters, conv-4 và conv-5 có 512 filters

VGG-16 cũng kế thừa lại hàm activation ReLU ở AlexNet.

Max pooling có filter là 2x2 và stride là 2

Fully connnected (FC): 2 đầu tiên là lớp dense 4096 nút và lớp thứ 3 có 1000 tương ứng với 1000 lớp. Lớp cuối cùng là lớp softmax

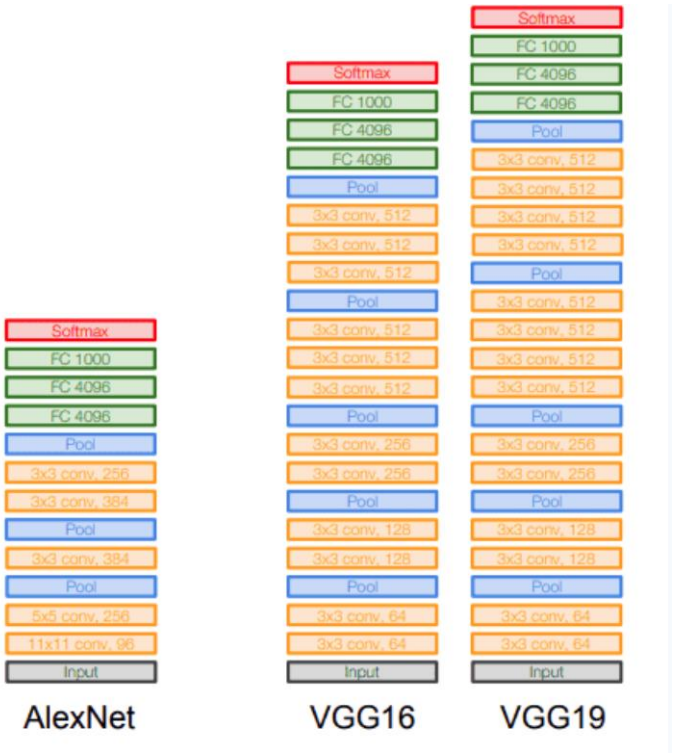
3.2. Một số hạn chế

Quá trình đào tạo khá chậm.

Chiếm khá nhiều dung lượng ổ đĩa và băng thông khiến nó không hiệu quả.

Có tới 138 triệu paramater => đòi hỏi tài nguyên của máy tính nhiều.

3.3. So sánh VGG16 với Alexnet và VGG19



Hiện nay ra đời VGG19 có nhiều hơn VGG16 3 lớp conv suy ra tham số cũng lớn hơn.

Kiến trúc VGG-16 sâu hơn, có 13 layers conv so với 5 của alexnet. Nó cải thiện bằng cách thay bộ lọc cỡ lớn (11x11 và 5x5 trong AlexNet) thành bộ lọc 3x3. Nhiều kernel size có kích thước chồng lên nhau sẽ tốt hơn so với kernel có kích thước lớn hơn vì nhiều lớp phi tuyến làm tăng độ sâu của mạng và cho phép nó tìm hiểu tính năng phức tạp hơn cũng cho chi phí thấp hơn.

V. THỰC NGHIỆM

Trong phần này chúng ta sẽ thực thi các thuật toán trên với ngôn ngữ Python cho bộ dữ liệu Fashion-MNIST. Chúng ta chỉ đề cập đến các bước tiêu biểu và bổ sung so với lý thuyết (code đầy đủ nằm trong 2 files đính kèm – file softmax_Mini-batch GD và file CNN). Cuối cùng là đưa ra kết quả, bàn luận và đánh giá.

1. **Tiền xử lý**

Chia tập train và tập test với tỉ lệ 6:1.

Chuẩn hóa dữ liệu về đoạn [0;1] bằng min-max scaling.

Chuyển train y về dạng one hot.

Riêng 2 thuật toán GD

Bằng cách quan sát các hình ảnh đầu vào, ta thử bổ sung vào X (chứa các giá trị pixel) thêm 2 đặc trưng là "intensity" và "symmetry" với hy vọng cho kết quả phân loại tốt hơn:

- "intensity" cho biết giá trị pixel trung bình của mỗi ảnh; đặc trưng này có thể giúp phân tách các hình vì có các áo quần, giày dép ít vải, nhỏ, tối hơn ("intensity" thấp) và có các áo quần, giày dép nhiều vải, to, sáng hơn ("intensity" cao).

- "symmetry" cho biết mức độ đối xứng của ảnh; đặc trưng này cũng có thể giúp phân tách các trang phục vì giày dép mức độ đối xứng thấp hơn áo quần và độ đối xứng giữa các áo quần tay dài và ngắn cũng khác nhau.

Cụ thể, "symmetry" của ảnh được tính như sau: $symmetry = -(s1 + s2) \,/\, 2$

s1: có được khi lấy ảnh trừ ảnh lật theo chiều ngang, lấy trị tuyệt đối, rồi lấy trung bình.

s2: có được khi lấy ảnh trừ ảnh lật theo chiều dọc, lấy trị tuyệt đối, rồi lấy trung bình.

2. Huấn luyện dữ liệu

- *Đối với 2 thuật toán GD*

Thực tế, ta dùng hàm *Softmax* ổn định hơn để tránh tràn số (overflow) khi các z_{ij} quá lớn.

$$\frac{\exp\left(z_{ij} - \max z_i\right)}{\sum_{j=1}^{C} \exp\left(z_{ij} - \max z_i\right)}$$

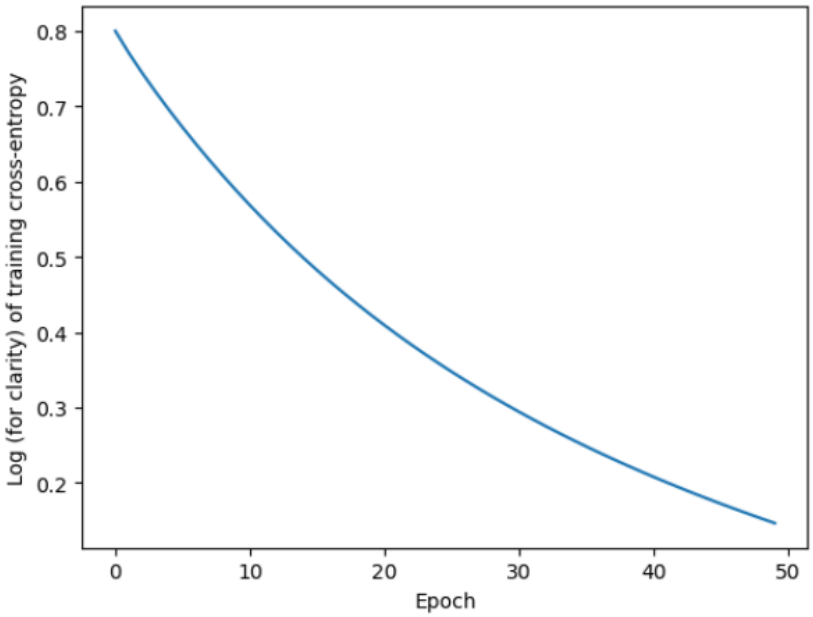
Áp dụng lý thuyết được đề cập ở các mục trên và dùng cả 2 phương pháp dừng thuật toán được đề cập ở mục 3-II để huấn luyện dữ liệu trong hàm *train_smreg*, với các tham số learning_rate = 0.03; max_epoch = 50; wanted_mbe = 13; mini-batch size là 32 đối với Mini-batch GD và bằng số dòng dữ liệu đối với GD.

- *Đối với CNN*

Với CNN thì set up như sau: gồm 2 lớp Conv + maxpooling, 1 lớp Fully- connected và cuối cùng là softmax để dự đoán cho 10 class.

3. Kết quả và đánh giá

- *Với GD:*



(Biểu đồ thể hiện cross-entropy qua mỗi epoch – biểu đồ giảm đều)

```
CPU times: total: 1min 23s
Wall time: 31.3 s
```

(Time processing)

Cross-entropy

```
train_ces[-1]
```

✓ 0.0s

```
1.1579369051139767
```

(Cross Entropy)

actual	5		6		2	4	6	7	0	4	9	6	3	6	8		1	3	2	
predict	7	9	4	0	4	2	2	9	3	3	7	3	0	8	3	7	3	4	8	0
count	490	451	366	263	257	200	195	134	107	101	68	66	52	38	34	30	29	29	25	25

(Top 20 lớp dễ bị đoán sai nhất)

Điểm số accuracy

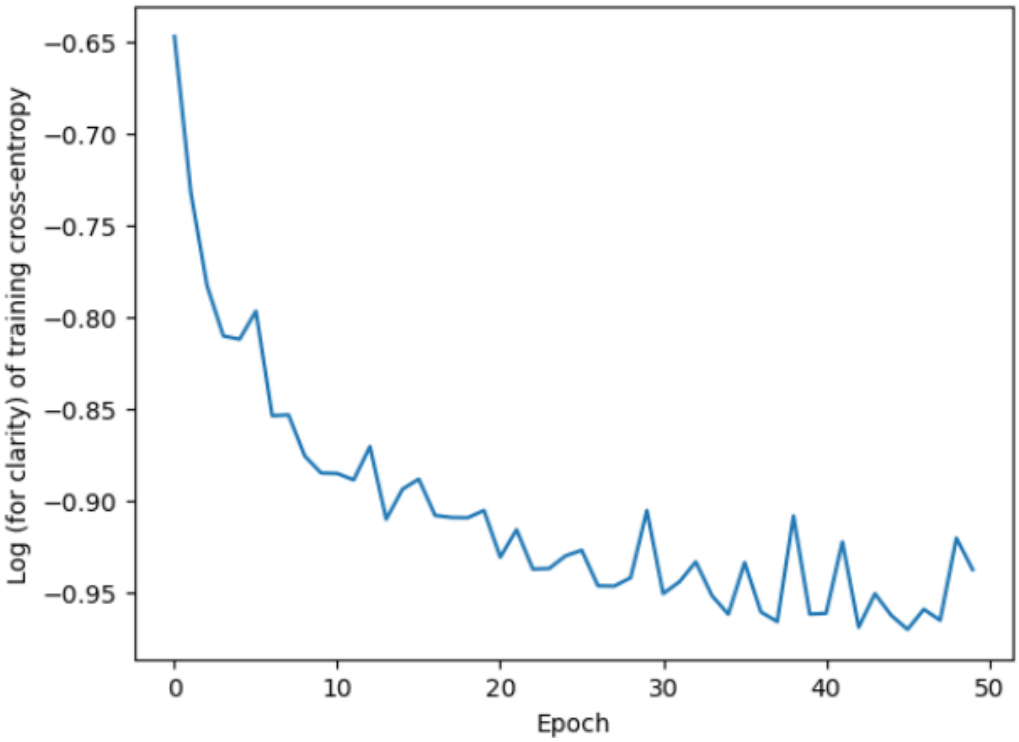
```
accuracy(y_pred, test_Y)
```

✓ 0.0s

```
0.6733
```

(Điểm số accuracy khá thấp)

- *Với mini-batch GD:*



(Biểu đồ tương quan của cross-entropy qua mỗi epoch

– biểu đồ dao động, nhưng có xu hướng giảm)

```
CPU times: total: 46.4 s
Wall time: 23.3 s
```

(Time processing)

Cross-entropy

```
train_ces[-1]
```

✓ 0.0s

```
0.3776502965306848
```

(Cross Entropy thấp hơn GD)

actual	2	0	2	6			4	7	3	5	3	0	9	7	4	5	8	3	6	3
predict	4	6	6	4	0	2	6	9	4	7	6	3	7	5	2	9	6	0	3	1
count	204	154	128	123	109	72	69	62	50	46	37	36	33	33	30	26	25	25	24	23

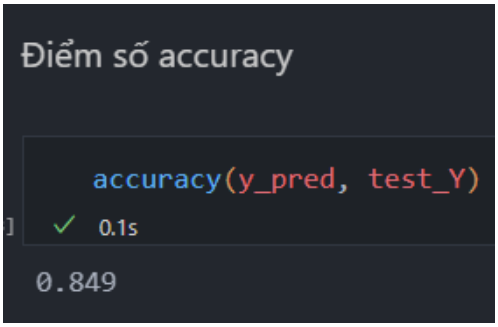
(Top 20 lớp dễ bị đoán sai nhất)

Nhận xét: 2. Pullover và 4. Coat dễ nhầm lẫn với nhau.

6. Shirt dễ nhầm lẫn với 4. Coat, 0. T-shirt/top và 2. Pullover

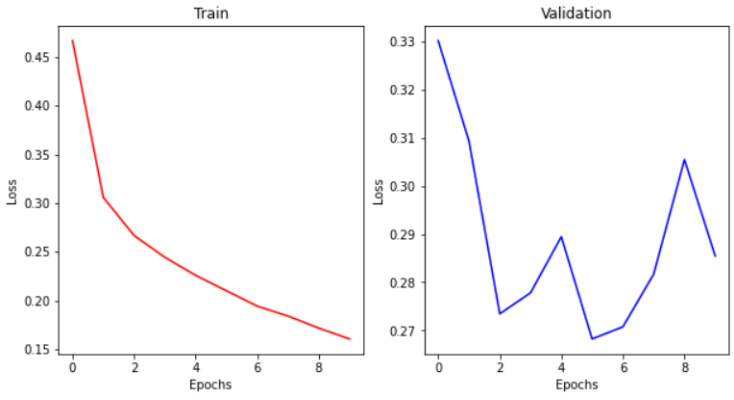
5. Sandal và 9. Ankle boot dễ nhầm lẫn với 7. Sneaker.

3. Dress dễ nhầm lẫn với 0. T-shirt/top, 2. Pullover và 4. Coat.

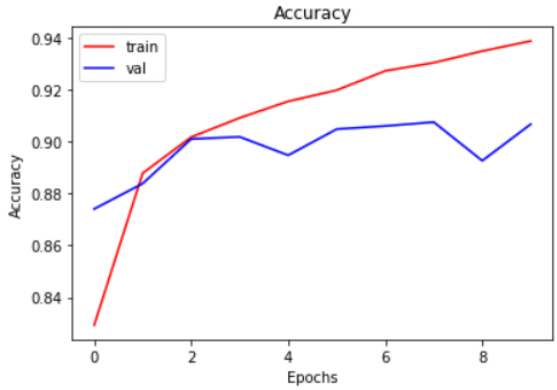


(Điểm số accuracy cao hơn nhiều so với GD)

- *Với CNN:*



(Loss trên tập train và validation)



(Accuracy trên tập train và validation)

```
test_evaluation=model.evaluate(test_X, test_Y)

313/313 [==============================] - 3s 9ms/step - loss: 0.2854 - accuracy: 0.9068
```

(Accuracy cuối cùng)

Dự đoán ngẫu nhiên 10 ảnh trong tập test phần chữ màu xanh là dự đoán dự đoán đúng, còn màu đỏ là dự đoán sai.

```
[[853   0  14  17   6   2 104   0   4   0]
 [  2 980   0  12   3   0   1   0   2   0]
 [ 20   1 858   7  86   0  28   0   0   0]
 [ 16   3   7 918  34   0  22   0   0   0]
 [  1   0  63  21 880   0  35   0   0   0]
 [  0   0   0   0   0 972   0  21   0   7]
 [103   0  80  26 101   0 684   0   6   0]
 [  0   0   0   0   0   3   0 980   0  17]
 [  6   0   2   3   1   1   9   0 977   1]
 [  0   0   0   0   0   4   1  29   0 966]]
```

Confusion_matrix: như hình ta thấy có 853 món ‘T-shirt/Top’ được dự đoán đúng cho là ‘T-shirt/Top’ và tương tự với các trường hợp còn lại.

4. Tổng hợp kết quả

Mô hình	Accuracy	Cross-entropy	Thời gian thực thi
Softmax với GD	0.673	1.158	23.3 giây
Softmax với mini-batch GD	0.849	0.378	31.3 giây
CNN	0.907	0.285	580 giây

Tham Khảo

Tiep Vu Huu, (Feb 17, 2017). Softmax Regession. *Machine Learning cơ bản*.

Tiep Vu Huu, (Jan 16, 2017). Gradient Descent (phần 2/2). *Machine Learning cơ bản*.

Hoang Tan Doan, (Nov 16, 2021). Thuật toán Convolutional Neural Network (CNN - Phần 1) – Python. AI Việt Nam.