**Phương Pháp Optimizer trong Huấn Luyện Mô Hình Học Máy và Ứng Dụng của Continual Learning và Test Production**

# Giới thiệu

Báo cáo này sẽ trình bày về hai khía cạnh quan trọng trong lĩnh vực học máy: Phương pháp Optimizer trong huấn luyện mô hình và ứng dụng của Continual Learning và Test Production. Mục tiêu là cung cấp cái nhìn tổng quan và sâu sắc về các công cụ và kỹ thuật hiện đại, giúp nâng cao hiệu quả và độ chính xác trong việc xây dựng và triển khai các giải pháp học máy.

# Phương Pháp Optimizer trong Huấn Luyện Mô Hình Học Máy

## **2.1 Định nghĩa về Optimizer**

Trong học máy, Optimizer là một thuật toán hoặc phương pháp được sử dụng để cập nhập và điều chỉnh các tham số của mô hình (như trong trọng số và bias) nhằm giảm thiểu hàm mát (loss function). Optimizer đóng vai trò quan trọng trong việc tìm kiếm điểm tối ưu của hàm mất mát, qua đó cải thiện khả năng dự đoán của mô hình.

## **Các Phương Pháp Optimizer Phổ Biến**

### **2.2.1 SGD (Stochastic Gradient Descent)**

* 1. **Khái Niệm:**

Gradient Descent: Là một phương pháp tối ưu hóa để tìm giá trị nhỏ nhất của một hàm số bằng cách di chuyển theo hướng đối diện với gradient(đạo hàm) của hàm số tại mỗi bước.

Stachastic: Từ “stochastic” (ngẫu nhiên) trong SGD chỉ ra rằng, thay vì tính toán gradient dựa trên toàn bộ dữ liệu (như trong Gradient Descent thông thường), SGD cập nhập trọng số dựa trên một tập con ngẫu nhiên của dữ liệu:

Trong đó:

* 𝜃: là tham số của mô hình.
* 𝜂: là tốc độ học.
* : là gradient của hàm mất mát J theo tham số 𝜃, tính tại điểm dữ liệu và nhãn tương ứng .
  1. **Cách Hoạt Động:**

SGD cập nhập tham số mô hình bằng cách tính toán gradient của hàm mất mát trên một tập con ngẫu nhiên của dữ liệu, sau đó điều chỉnh tham số theo hướng ngược lại với gradient.

* 1. **Điểm Mạnh:**
     + Hiệu quả với dữ liệu lớn do không cần xử lý toàn bộ dữ liệu cùng một lúc.
     + Giảm thiểu rủi ro về vấn đề cực tiểu địa phương.
  2. **Điểm Yếu:**
     + Có thể không ổn định và tốc độ hội tụ chậm.
     + Độ chính xác có thể không cao như khi sử dụng toàn bộ tập dữ liệu.
  3. **Ứng Dụng:**

SGD thường được sử dụng trong các mô hình học máy lớn và học sâu, nơi mà việc xử lý toàn bộ toàn dữ liệu là không khả thi về mặt tính toán.

### **2.2.2 Adam (Adaptive Moment Estimation)**

1. **Khái Niệm:**

Adam là một phương pháp tối ưu hóa hiệu quả, kết hợp các ưu điểm của hai phương pháp là Momentum và RMSprop. Adam không chỉ tính toán trung bình của gradient (như Momentum) mà còn điều chỉnh tốc độ học dựa trên độ lớn của gradient gần đây (như RMSprop), giúp tăng cường hiệu quả và ổn định trong quá trình huấn luyện.

Trong đó:

* : là tham số mô hình tại bước thời gian t.
* 𝜂: tộc độ học.
* : là các ước lượng bias-corrected của moment đầu tiên (trung bình của gradient) và moment thứ hai (trung bình của bình phương gradient) của gradient.
* : là một hằng số để tránh chia cho không (thường rất nhỏ, ví dụ ).

1. **Cách Hoạt Động:**

**Tính Moment đầu tiên (Momentum)**: Adam tính trung bình trượt (exponential moving average) của gradient, giúp mô hình “nhớ” hướng gradient trước đó và sử dụng thông tin này để đẩy nhanh quá trình học.

**Tính Moment thứ hai (RMSprop)**: Adam tính trung bình trượt của bình phương gradient, giúp điều chỉnh tốc độ học dựa trên độ lớn của gradient gần đây.

1. **Điểm Mạnh:**
   * + Adam tự điều chỉnh tốc độ học cho mỗi tham số, giúp quá trình học trở nên hiệu quả và ổn định hơn.
     + Phù hợp với đa số các bài toán học sâu, đặc biệt là với dữ liệu lớn và phức tạp.
2. **Điểm Yếu:**
   * + Cần phải chọn lựa cẩn than các siêu tham số như tốc độ học và các hằng số trong công thức.
     + Trong một số trường hợp, Adam có thể dẫn đến overfitting nếu không được điều chỉnh cẩn thận.
3. **Ứng Dụng:**

Adam được sử dụng rộng rãi trong các mô hình học sâu, từ mạng nơ-ron tích chập (CNN) cho đến mạng nơ-ron hồi quy (RNN), và đặc biệt hiệu quả trong các tác vụ như xử lý ngôn ngữ tự nhiên và nhận dạng hình ảnh.

### **2.2.3 RMSprop**

1. **Khái Niệm:**

RMSprop là một phương pháp tối ưu hóa được thiết kế đặc biệt để giải quyết một số vấn đề của SGD, đặc biệt là trong việc điều chỉnh tốc độ học. Nó thực hiện điều này bằng cách giữ một trung bình trượt của bình phương gradient và sử dụng nó để chuẩn đoán hóa gradient, giúp tăng cường ổn định và hiệu quả trong quá trình học.

Trong đó:

* : là tham số mô hình tại bước thời gian t.
* 𝜂: tộc độ học.
* : là trung bình trượt của bình phương gradient.
* : là gradient của hàm mất mát tại bước thời gian t.
* : là một hằng số nhỏ để tránh chia cho không.

1. **Cách Hoạt Động:**

**Tính Trung Bình Trượt của Bình Phương Gradient**: RMSprop duy trì một trung bình trượt của bình phương gradient, giúp “làm mịn” quá trình cập nhập tham số và tránh những biến động lớn.

**Chuẩn hóa Gradient**: Bằng cách chia gradient cho căn bậc hai của trung bình trượt này, RMSprop tự động điều chỉnh tốc độ học cho mỗi tham số, giúp quá trình học trở nên ổn định hơn.

1. **Điểm Mạnh:**
   * + Đặc biệu hữu ích trong các tình huống có gradient lớn và không đồng đều.
     + Giúp mô hình học nhanh hơn và ổn định hơn so với SGD truyền thống.
2. **Điểm Yếu:**
   * + Cần phải lựa chọn cẩn thận tốc độ học và các hằng số khác ron.
     + Trong một số trường hợp, RMSprop có thể mang lại hiệu suất tối ưu.
3. **Ứng Dụng:**

RMSprop thường được sử dụng trong các mô hình học sâu, đặc biệt là khi cần giải quyết vấn đề về gradient không đồng đều, như trong một số loại mạng nơ-ron hồi quy (RNN).

## **Thực Nghiệm**

### **2.2.3 Nhập Các Thư Viện Cần Thiết**

import tensorflow as tf

from tensorflow.keras.datasets import mnist

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D

### **2.2.4 Tải Bộ Dữ Liệu**

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

x\_train = x\_train.reshape(x\_train.shape[0], 28, 28, 1)

x\_test = x\_test.reshape(x\_test.shape[0], 28, 28, 1)

input\_shape = (28, 28, 1)

num\_classes = 10

y\_train = tf.keras.utils.to\_categorical(y\_train, num\_classes)

y\_test = tf.keras.utils.to\_categorical(y\_test, num\_classes)

x\_train = x\_train.astype('float32')

x\_test = x\_test.astype('float32')

x\_train /= 255

x\_test /= 255

### **2.2.5 Xây Dựng Mô hình**

batch\_size = 64

epochs = 10

def build\_model(optimizer):

    model = Sequential()

    model.add(Conv2D(32, kernel\_size=(3, 3), activation='relu', input\_shape=input\_shape))

    model.add(MaxPooling2D(pool\_size=(2, 2)))

    model.add(Dropout(0.25))

    model.add(Flatten())

    model.add(Dense(256, activation='relu'))

    model.add(Dropout(0.5))

    model.add(Dense(num\_classes, activation='softmax'))

    model.compile(loss=tf.keras.losses.categorical\_crossentropy, optimizer=optimizer, metrics=['accuracy'])

    return model

### **2.2.6 Huấn Luyện Mô Hình**

optimizers = ['Adam', 'RMSprop', 'SGD']

for i in optimizers:

    model = build\_model(i)

    hist = model.fit(x\_train, y\_train, batch\_size=batch\_size, epochs=epochs, verbose=1, validation\_data=(x\_test, y\_test))

Sau đây là kết quả phân tích sau khi chạy mô hình với kích thước batch là 64 trong 10 epoch.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Optimizer | Epoch 1  Val accurary | Val loss | Epoch 5  Val accurary | Val loss | Epoch 10  Val accurary | Val loss | Total Time |
| Adam | .9786 | .0682 | .9874 | .0366 | .9888 | .0379 | 7:1 min |
| RMSprop | .9764 | .0764 | .9866 | .0397 | .9886 | .0373 | 6:47 min |
| SGD | .9171 | .2976 | .9569 | .1459 | .9674 | .1045 | 6:35 min |

Bảng trên cho thấy độ chính xác và mất mát kiểm định ở cách epoch khác nhau. Nó cũng chứa tổng thời gian mà mô hình mất để chạy 10 epoch cho mỗi optimizer. Từ bảng trên ta có thể phân tích như sau:

* Optimizer Adam cho thấy độ chính xác tốt nhất trong một lượng thời gian hợp lý.
* RMSprop cho thấy độ chính xác tương tự như Adam nhưng với thời gian thấp hơn.
* SGD mất ít thời gian nhất để huấn luyện và cũng đạt kết quả tốt. Tuy nhiên, để đạt được độ chính xác của optimizer Adam, SGD sẽ cần nhiều lần lặp hơn, và do đó thời gian tính toán sẽ tăng lên.

## **Kết Luận**

SGD là một thuật toán cơ bản và hiện nay hầu như không được sử dụng trong các ứng dụng do tốc độ tính toán chậm. Một vấn đề nữa với thuận toán này là tốc độ học cố định cho mỗi epoch. Hơn nữa, nó không xử lý tốt các điểm yên ngựa. RMSprop cho thấy kết quả tương tự như thuật toán gradient descent với động lượng, chỉ khác ở cách tính toán gradients. Adam kế thừa những tính năng tốt của RMSprop và các thuật toán khác. Kết quả của optimizer Adam thường tốt hơn mọi thuật toán tối ưu hóa khác, có thời gian tính toán nhanh hơn và yêu cầu ít tham số điều chỉnh hơn. Vì tất cả những điều đó, Adam được khuyến khích là optimizer mặc định cho hầu hết các ứng dụng. Chọn optimizer Adam cho ứng dụng của bạn có thể mang lại khả năng cao nhất để đạt được kết quả tốt nhất.

# Tìm Hiểu về Continual Learning và Test Production

## **3.1 Continual Learning trong Học Máy**

1. **Khái Niệm:**

Continual Learning, còn được gọi là Lifelong Learning, là quá trình mà mô hình được huấn luyện để không ngừng học hỏi và thích nghi với dữ liệu mới mà không quên những gì đã học trước đó. Điều này giúp mô hình đối phó với vấn đề “catastrophic forgetting” – quên thông tin cũ khi tiếp xúc với thông tin mới.

1. **Cách Hoạt Động:**

Sử dụng các kỹ thuật như Elastic Weigh Consolidation (EWC), nơi các trọng số quan trọng của mô hình được “đóng băng” để tránh quên thông tin cũ.

Cấu trúc mô hình có khả năng thay đổi, như thêm các neuron mới khi học từ dữ liệu mới.

1. **Ứng Dụng:**
   * **Chuẩn Đoán Bệnh:** Mô hình AI được huấn luyện để nhận diện các loại bệnh ung thư từ hình ảnh X-quang. Khi có dữ liệu mới về các trường hợp bệnh lạ, mô hình cần cập nhập kiến thức của mình mà không quên những gì đã học từ các trường hợp trước đó.
   * **Đề Xuất Phim:** Các mô hình đề xuất phim như Netflix hoặc YouTube cần liên tục cập nhập để phản ánh sở thích thay đổi của người dùng và các xu hướng mới trong nội dung, đồng thời duy trì khả năng đề xuất chính xác dựa trên lịch sử xem trước đó của người dùng.
   * **Robot Dọn Dẹp Tự Động:** Robot được huấn luyện để dọn dẹp một ngôi nhà có thể cần học cách xử lý các đồ vật mới hoặc bố trí nội thất thay đổi mà không quên cách xử lý các tác vụ đã học trước đó.

## **3.2 Test Production**

1. **Khái Niệm:**

Test Production là quá trình đánh giá mô hình học máy trong môi trường sản xuất thực tế, nơi mô hình sẽ được sử dụng. Điều này bao gồm việc kiểm tra hiệu suất, độ tin cậy, và khả năng mở rộng của mô hình.

1. **Cách Hoạt Động:**

* **A/B Testing:** So sánh hiệu suất của mô hình mới với mô hình hiện tại trong một môi trường kiểm soát.
* **Monitoring and Logging:** Theo dõi hiệu suất mô hình và ghi lại các vấn đề để nhanh chóng giải quyết.

1. **Ứng Dụng:**

* **Kiểm Thử Mô Hình Đề Xuất Sản Phẩm:** Trong một trang web bán lẻ, mô hình đề xuất sản phẩm mới được triển khai và kiểm thử để đảm bảo rằng nó cung cấp kết quả chính xác và tăng cường trải nghiệm mua sắm cho khách hàng.
* **Phát Hiện Giao Dịch Gian Lận:** Mô hình phát hiện gian lận được kiểm thử trong môi trường thực tế để đảm bảo rằng nó có thể phát hiện gian lận một cách chính xác mà không làm giảm hiệu suất xử lý giao dịch bình thường.
* **Hệ Thống Điều Khiển Giao Thông:** Mô hình dự đoán và điều kiển lưu lượng giao thông được kiểm thử trong môi trường thực tế để đảm bảo rằng nó có thể cải thiện hiệu quả giao thông mà không gây ra tắc nghẽn không mong muốn.

# Kết Luận

Trong xây dựng giải pháp học máy, sự kết hợp giữa các phương pháp Optimizer hiệu quả trong huấn luyện, khả năng thích nghi liên tục của Continual Learning, và độ tin cậy từ Test Production tạo nên một hệ thống mạnh mẽ và linh hoạt. Mô hình không chỉ cải thiện về mặt hiệu suất mà còn có khả năng thích ứng và phát triển trong môi trường thực tế, đáp ứng nhu cầu liên tục thay đổi của dữ liệu và người dùng.