# OPTIMIZER TRONG HUẤN LUYỆN MÔ HÌNH HỌC MÁY

***Link tham khảo:***

[***https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/***](https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/)

[***https://machinelearningcoban.com/2017/01/12/gradientdescent/***](https://machinelearningcoban.com/2017/01/12/gradientdescent/)

[***https://machinelearningcoban.com/2017/01/16/gradientdescent2/***](https://machinelearningcoban.com/2017/01/16/gradientdescent2/)

## Phương pháp Optimizer

* Thuật toán tối ưu (Optimizer) nhằm điều chỉnh các tham số trong quá trình huấn luyện để giảm giá trị của hàm mất mát.
* Thuật toán được sử dụng phổ biến trong xây dựng neural network để học các features (hay pattern) từ các dữ liệu bằng cách cập nhật lập đi lặp lại giá trị của các weights và biases nhằm tìm được 1 cặp giá trị weitght và bias phù hợp với modal.
* Các phương pháp Optimizer cơ bản: Stochastic Gradient Descent (SGD), Adam, RMSprop

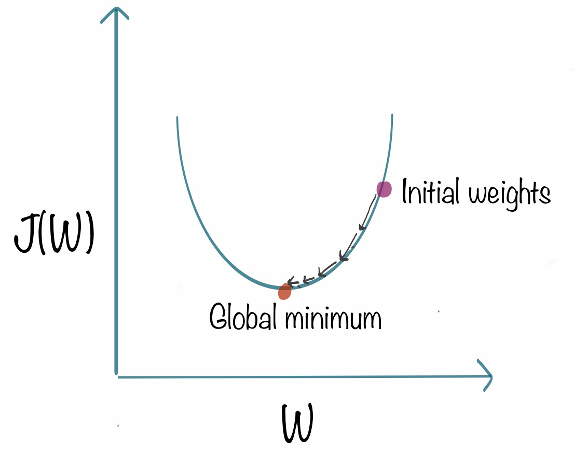
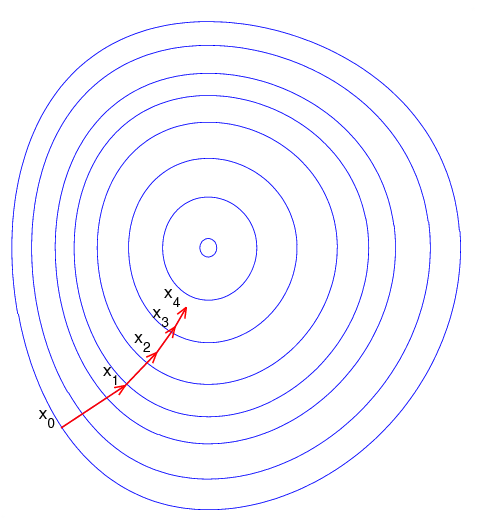
## Các phương pháp Optomizer

### Gradient Descent

* Là một thuật toán tối ưu tìm *local minimum* của một hàm số nào đó mà hàm số đó đạt giá trị nhỏ nhất khi đạo hàm bằng 0.
* Tuy nhiên, trong nhều trường hợp tính đạo hàm rất phức tạp hoặc không thể. Do đó, ta tiến hành tìm kiếm điểm gần với điểm cực tiểu nhất
* Thuật toán sẽ tiến hành theo các cách:
  + Khởi tạo một hệ số (coefficients), tính chi phí (cost) của hệ số. Sau đó tìm hệ số có cost value thấp hơn giá trị hiện tại. Di chuyển đến điểm có cost value thấp hơn và tiếp hành cập nhật giá trị coefficients.

Trong đó:

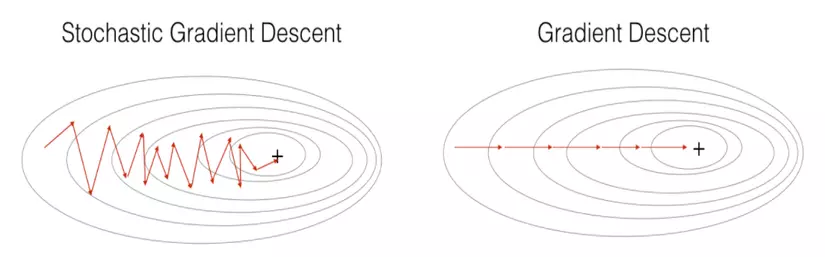
* + - là learning rate (tốc độ học)
    - là điểm ta tìm được sau vòng lặp thứ
  + Đối với hàm nhiều biến ta có công thức: . Trong đó:
    - là vecto thường được dùng để ký hiệu tập hợp các tham số của mô hình cần tối ưu.
    - là đạo hàm của hàm số tại một điểm bất kỳ.
  + Lặp đi lặp lại cho đến khi tìm được local minimum

* Ưu điểm:
  + Dễ hiểu và dễ triển khai
  + Hoạt động tốt cho các hàm lồi (convex function)
* Nhược điểm:
  + Tốn kém chi phí tính toán nếu kích thước của dữ liệu lớn
  + Learning rate quá lớn dẫn đến thuật toán không hội tụ, vì bước nhảy khá lớn nên giá trị thường rải xung quanh giá trị đích.
  + Learning rate quá nhỏ ảnh hưởng đến tốc độ training
  + Đối với hàm có 2 local minimum, trong đó có 1 điểm là global minimum thì tuỳ vào giá trị giá trị khởi tạo ban đầu sẽ cho ra 2 nghiệm cuối cùng khác nhau.

### Stochastic Gradient Descent

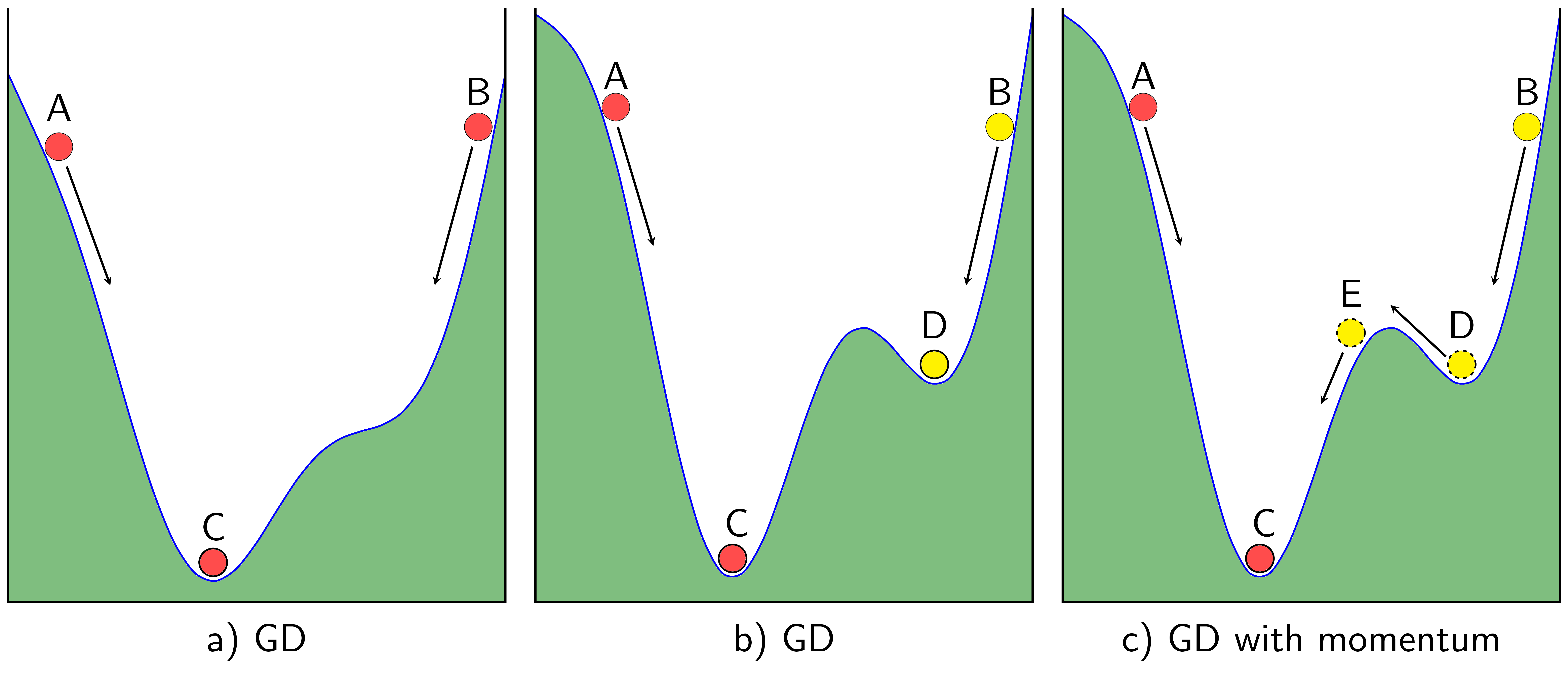
* Là một phần cải tiến của thuật toán GD. Thay vì lấy tất cả dữ liệu trong mỗi lần lặp, chúng ta sẽ lấy ngẫu nhiên các lô (batchs) dữ liệu. Điều đó có nghĩa là chúng ta sẽ lấy ít dữ liệu mẫu từ dữ liệu đầu vào.
* Ban đầu sẽ khởi tạo các tham số và learning rate . Sau đó trộn ngẫu nhiên dữ liệu ở mỗi lần lặp để tính mức tối thiểu xấp xỉ.
* Trong đó:
  + là hàm mất mát với chỉ một cặp dữ liệu (input; label) tương ứng với



* Chính vì do lựa chọn tập ngẫu nhiên trong bộ dữ liệu trong mỗi lần lặp, nên đường đi của thuật toán có dạng zic-zac và có nhiều nhiễu hơn so với thuật toán GD. Do đó, thuật toán SGD sử dụng số lần lặp nhiều hơn, đồng thời chi phí thời gian chạy cũng tăng lên theo số lần lặp. Nhưng khi tăng số lần lặp thì chi phí tính toán của thuật toán SGD vẫn thấp hơn so với thuật toán GD
* Ưu điểm:
  + Phù hợp với dữ liệu cập nhật liên tục (Online Learning)
  + Thích hợp cho các bộ dữ liệu lớn
  + Thời gian tính toán nhanh
* Nhược điểm:
  + Có nhiều độ nhiễu cao và không ổn định do chọn ngẫu nhiên các tập từ bộ dữ liệu
  + Không chắc chắn hội tụ tại global minimum

### Stochastic Gradient Descent With Momentum

* Để cải thiện bất lợi của thuật toán GD có thể không đến được đích do dữ liệu có 2 điểm local minimum.
* Nhằm tăng tốc độ hội tụ cho thuật toán và bỏ qua được local minimum bằng cách tính lượng thay đổi ở thời điểm t để cập nhật vị trí mới cho nghiệm.
* Dưới góc nhìn vật lý, đại lượng này tương tự như vận tốc ở thời điểm t, bây giờ vị trí của nghiệm sẽ có dạng . Phép ‘-‘ có nghĩa là di chuyển ngược lại với đạo hàm.



* Điều quan trọng trong thuật toán này là tính vận tốc sao cho vừa có thể biểu diễn được độ dốc và vận tốc trước đó:
* Trong đó:
  + là một siêu tham số Beta, thường được chọn có giá trị là 0.9
  + là vận tốc trước đó tại thời điểm t-1

* + là độ dốc của của điểm trước đó
  + là learning rate
* Sau đó tính vị trí của điểm:
* Ưu điểm:
  + Giải quyết được vấn đề 2 giá trị local minimum của thuật toán SG
  + Hiệu quả cho các bài toán thực tế, bài toán có nhiều local minimum
* Nhược điểm:
  + Khi giá trị tiến gần đến điểm đích, cần có thêm một khoảng thời gian để di chuyển qua lại điểm đích trước khi dừng hẳn do có đà trước đó

### Mini Batch Stochastic Gradient Descent

* Khác với SGD, thuật toán Mini Batch sử dụng một số lượng n dữ liệu (lớn hơn 1 và bé hơn N dữ liệu đầu vào rất nhiều).
* Giống với SGD, thuật toán bắt đầu mỗi epoch bằng việc trộn ngẫu nhiên dữ liệu và chia ra thành các mini-batch có n điểm dữ liệu. Sau mỗi lần cập nhật, thuật toán sẽ lấy ra 1 mini-batch để tính toán đạo hàm và cập nhật theo công thức sau:
* Trong đó:
  + là dữ liệu từ i đến i+n. Sau mỗi epoch dữ liệu này sẽ khác nhau vì được trộn lại. Giá trị n thường được chọn từ 50-100

A graph showing a blue line

Description automatically generated

* Ưu điểm:
  + Ổn định hơn so với thuật toán SGD
* Nhược điểm:
  + Siêu tham số mini-batch-size (giá trị thông thường là 32) cần phải được điều chỉnh để đạt được độ chính xác cần thiết.

### Adagrad (Adaptive Gradient Descent)

* Thuật toán thực hiện giảm dần độ dốc bằng cách thay đổi tốc độ học tập.
* Adagrad được cải thiện hơn bằng cách cho trọng số học tập chính xác dựa vào đầu vào trước nó để tự điều chỉnh learning rate theo hướng tối ưu nhất thay vì với một tỉ lệ học duy nhất cho tất cả các nút.
* Trong đó:
  + là vector độ dốc cho vị trí hiện tại
  + là hệ số tránh lỗi
  + là ma trận chéo chứa bình phương của đạo hàm vector tham số tại vòng lặp t
  + là learning rate và sẽ được cập nhật qua mỗi vòng lặp
* Ưu điểm:
  + Phù hợp cho các bài toán phân mảnh
  + Tỉ lệ học được tích hợp cho từng tham số
* Nhược điểm:
  + Tổng bình phương của ma trận đường chéo càng lớn khiến tốc độ học càng giảm.

### RMS Prop (Root Mean Square Propogation)

* Thuật toán sử dụng trung bình bình phương của gradient để chuẩn hoá gradient
* Thuật toán điều chỉnh độ dốc:
  + Giảm bước cho độ dốc: tránh hiện tượng phát nổ độ dốc (Exploding Gradient)
  + Tăng bước cho độ dốc: tránh biến mất độ dốc (Vanishing Gradient)
* RMSProp điều chỉnh learning rate và chọn một learning rate khác nhay cho mỗi tham số bằng công thức:
* Trong đó:
  + là tích luỹ phương sai của gradient trong quá khứ
  + là tham số suy giảm
  + là gradient của các tham số tại vòng lặp thứ t
  + là tham số đảm bảo kết quả xấp xỉ có ý nghĩa
* Ưu điểm:
  + Giảm ảnh hưởng của Gradient nhỏ
  + Phù hợp với các dữ liệu có Gradient biến động

### AdaDelta

* Thuật toán là một biến thể khác của thuật toán Adagrad và là một dạng của SGD dựa trên việc thay đổi learning rate trên mỗi chiều (dimention) để giải quyết hai vấn đề chính:
  + Learning rate giảm liên tục trong quá trình huấn luyện
  + Chọn thủ công global learning rate
* Thuật toán không có tham số learning rate. Thay vào đó, giúp điều chỉnh tốc độ học dựa trên việc giới hạn cửa sổ của gradient tích luỹ trong quá khứ, mà không tích luỹ tất cả độ dốc trong quá khứ. Bằng cách này, AdaDelta tiếp tục học hỏi ngay khi đã thực hiện nhiều lần cập nhật hoàn thành.
* So với thuật toán AdaGrad, ta không cần phải cài đặt learning rate ban đầu.
* Ưu điểm:
  + Khắc phục khuyết điểm của thuật toán AdaGrad bằng cách điều chỉnh tốc độ học

### Adam (Adaptive Moment Estimation)

* Thuật toán Adam được xem là sự kết hợp giữa RMS Prop và Stochastic Gradient Descent with Momentum.
* Adam được xem là một phương pháp tỉ lệ học thích ứng, bằng việc tính toán learning rate thích ứng cho từng tham số.
* Adam tính toán khoảng thời gian thứ nhất và thứ hai của độ dốc để điều chỉnh tỉ lệ học cho từng trọng số. Adam không chỉ lưu trữ trung bình bình phương các gradient trước đó như Adadelta mà còn lưu cả giá trị trung bình mô-men . Các giá trị và được tính bởi công thức:
* Trong đó:
  + là động lượng tại thời điểm t
  + là vận tốc tại thời điểm t
  + là tỉ lệ giảm mạnh mỗi chu kỳ cho động lượng và vận tốc tương ứng. Thường mang giá trị là là 0.9 và là 0.999.
  + là độ dốc tại thời điểm t
* Nếu khởi tạo và là các vector 0, các giá trị này có khuynh hướng nghiêng về 0, đặc biệt là khi và xấp xỉ bằng 1. Do vậy, để khắc phục, các giá trị này được ước lượng bằng cách:
* Sau đó cập nhật các trọng số theo công thức:
* Ưu điểm:
  + Tốn ít bộ nhớ
  + Thích hợp với dữ liệu có độ biến thiên không ổn định và dữ liệu phân mảnh
* Nhược điểm:
  + Có thể nhạy cảm với các siêu tham số

## Dữ liệu tương thích của các thuật toán

|  |  |
| --- | --- |
| Thuật toán | Thích hợp dữ liệu |
| Gradient Descent | Hàm lồi |
| Stochastic Gradient Descent | Dữ liệu cập nhật liên tục, bộ dữ liệu lớn |
| Momentum | Dữ liệu có nhiều local minimum |
| Mini-Batch | Tập dữ liệu lớn, hiệu suất cao |
| AdaGrad | Dữ liệu phân mảnh, dữ liệu không đồng nahát |
| RMS Prop | Dữ liệu có Gradient biến động mạnh |
| AdaDelta | Dữ liệu thưa  Dữ liệu không đồng nhất |
| Adam | Dữ liệu biến thiên không ổn định, dữ liệu phân mảnh  Dữ liệu đa dạng |

# CONTINUAL LEARNING VÀ TEST PRODUCTION

***Link tham khảo:***

[***github***](https://github.com/serodriguez68/designing-ml-systems-summary/blob/main/09-continual-learning-and-test-in-production.md#shadow-deployment)***,***

[***https://paperswithcode.com/task/continual-learning***](https://paperswithcode.com/task/continual-learning)***,***

[***https://deepchecks.com/how-to-test-machine-learning-models/***](https://deepchecks.com/how-to-test-machine-learning-models/)

## Continual Learning

* Continual learning, hay học liên tục, là quá trình học tập của mô hình mà trong đó mô hình không học từng dữ liệu riêng lẻ một cách cô lập, mà còn tích luỹ tri thức từ dữ liệu đã học trước đó để hổ trợ cho việc mô hình học tập dữ liệu mới. Điều này giúp cho mô hình phát triển và nâng cao hiệu suất theo thời gian.
* Mục tiêu của Contiunal Learning là sử dụng các tri thức đã học từ các tập dữ liệu trước đó sau đó hỗ trợ cho việc học dữ liệu mới
* Đặc trưng của Continual Learning:
  + Quá trình học tập liên tục và tăng dần (Continuous and incremental learning process): Tình trạng “quên đột ngột” (catastrophic forgetting) – Khi học tập một tập dữ liệu mới, mô hình sẽ bị thay đổi tham số học tập, điều này có thể gây ra ảnh hưởng đáng kể đối với độ chính xác của tập dữ liệu trước đó.
  + Tích luỹ kiến thức tính bằng Knowledge – Base (long-term memory)
  + Chuyển giao/ thích ứng dữ liệu mới:
    - Forward transfer: kết quả học tập trên tập dữ liệu cũ sẽ giúp cải thiện việc học của mô hình trên các dữ liệu mới.
    - Backward transfer: kết quả học trên tập dữ liệu mới sẽ giúp cải thiện các kết quả học trước đó.
* Tuy nhiên khi mô hình được cập nhật, chúng ta không thể sử dụng một cách bừa bãi mà phải thông qua một số quá trình kiểm thử để kiểm tra độ chính xác và ổn định của mô hình. Quá trình đó được gọi là “Test production”.

## Test Production

* Để kiểm tra và đánh giá mô hình đầy đủ trước khi đưa mô hình vào sử dụng, chúng ta cần phải thực hiện *pre-deployment offline* và *testing in production*.

### Pre-deployment offline

* Có hai phương pháp chính để đánh giá mô hình:
  + Sử dụng *test split* để so sánh với một mô hình cơ sở
  + Chạy backtests

#### **Test splits**: thường là dạng tĩnh (static) do đó nó là một thước đo tin cậy cho việc so sánh nhiều mô hình với nhau.

* **Backtesting**: sử dụng dữ liệu được gắn nhãn mà mô hình chưa học trong quá trình huấn luyện để kiểm tra hiệu suất và độ chính xác của mô hình
* Tuy nhiên, chạy backtest là không đủ để xác địch đưa mô hình vào sử dụng thực tế. Để chạy trong môi trường thực tế chúng ta cần phải xem xét thêm nhiều yếu tố hơn là chỉ mỗi hiệu suất liên quan đến nhãn. Ta cần phải theo dõi các yếu tố khác như tốc độ học tập, độ trễ, hành vi của người sử dụng với mô hình và tính chính xác của mô hình.

### Testing in production

#### Shadow Deployment

* Triển khai hai mô hình song song với nhau gọi là *challenger model* và *champion model*. Chuyển các yêu cầu được gửi đến cho cả hai model nhưng chỉ phục vụ dự đoán từ champion model. Ghi lại tất cả dự đoán của cả hai mô hình sau đó so sánh với nhau.
* Ưu điểm:
  + Các an toàn nhất để deploy model
  + Đơn giản để triển khai
* Nhược điểm:
  + Phương pháp này không được sử dụng khi đo hiệu suất của mô hình phụ thuộc vào việc cách người dùng tương tác với kết quả dự đoán.
  + Tốn kém chi phí do thực hiện dự đoán và ghi lại kết quả trên cả 2 mô hình
  + Khi mô hình shadow thực hiện dự đoán chậm hơn hoặc gặp lỗi thì ta cần phải tính toán các nguyên nhân có thể gây ra.

#### A/B Testing

* Tương tự như shadow deployment, ta triển khai cả 2 mô hình challenger model (model B) và champion model (model A). Tuy nhiên, ta sẽ chuyển một phần lưu lượng truy cập sang mô hình B. Dự đoán từ mô hình challenger sẽ được hiển thị cho người dùng. Sử dụng giám sát và phân tích cả hai mô hình để xem hiệu suất của mô hình challenger có cao hơn mô hình champion hay không.
* Một số lưu ý:
  + Trong một số trường hợp không phù hợp với việc chia lưu lượng cho cả hai mô hình trong cùng một lúc. Ta có thể chia theo thời gian thử nghiệm cho cả hai mô hình như một ngày sử dụng mô hình A, ngày tiếp theo sử dụng mô hình B.
  + Việc phân chia lưu lượng phải được thực hiện một cách ngẫu nhiên nếu không có thể dẫn đến kết quả test bị sai lệch.
  + Việc thử nghiệm phải được chạy đủ lâu để thu thập được đủ mẫu để đạt được độ tin cậy đáng kể.
  + Chúng ta có thể thực hiện nhiều hơn hai mô hình, chẳng hạn như A/B/C/D tests.
* Ưu điểm:
  + Thu thập được các người dùng tương tác với kết quả dự đoán trên cả hai mô hình.
  + Phương pháp đơn giản, có nhiều thư viện và tài liệu hỗ trợ.
  + Tiết kiệm chi phí hơn Shadow Deployment
* Nhược điểm:
  + Ít an toàn hơn phương pháp Shadow deployment.
  + Phải lựa chọn chấp nhận nhiều rủi ro (chuyển lưu lượng nhiều hơn tới mô hình B) và thu thập đủ mẫu để đánh giá độ đáng tin cậy của mô hình.
  + Thực hiện phải đảm bảo được tính ngẫu nhiên hoá, đủ thời gian chạy và phân tích để cho kết quả chính xác hơn.

#### Canary Release

* Triển khai challenger model và champion model song song nhưng bắt đầu bằng việc không chuyển lưu lượng nào cho mô hình challenger. Dần dần chuyển lưu lượng từ mô hình hiện tại sang mô hình challenger (canary). Giám sát các thông số hiệu suất của mô hình challenger, nếu ổn định ta tiếp tục chuyển lưu lượng cho đến khi toàn bộ lưu lượng được chuyển sang mô hình thách thức.
* Lưu ý:
  + Canary release có thể kết hợp với A/B testing để đo lường đáng tin cậy sự khác biệt về hiệu suất.
  + Canary release cũng có thể được chạy ở chế độ "YOLO" (You Only Look Once), trong đó bạn đánh giá sự khác biệt về hiệu suất một cách thủ công.
  + Một phiên bản khác của canary release là triển khai mô hình thách thức cho một thị trường nhỏ hơn trước, sau đó mở rộng sang tất cả các thị trường nếu mọi thứ trông tốt.
  + Nếu mô hình thách thức gặp vấn đề, bạn có thể chuyển lưu lượng trở lại cho mô hình hiện tại.
* Ưu điểm:
  + Dễ hiểu và dễ triển khai
  + Vì các dự đoán từ mô hình challenger được phục vụ, ta có thể sử dụng phương pháp này với các mô hình yêu cầu tương tác người dùng để thu thập hiệu suất.
  + So với shadow deployment, chi phí chạy thấp hơn. Một yêu cầu chỉ có một dự đoán.
  + Nếu kết hợp với A/B testing, nó cho phép bạn thay đổi độ lưu lượng mà mỗi mô hình đang nhận.
* Nhược điểm:
  + Không thể đánh giá rõ ràng sự chênh lệch giữa hai mô hình.
  + Nếu triển khaih không được giám sát cẩn thận, có thể xảy ra sự cố. Đây có thể xem là lựa chọn ít an toàn nhất, nhưng rất dễ phục hồi.

#### Interleaving Experiments

* Thay vì nhận được một kết quả dự đoán từ mô hình A hoặc B, ta có thể nhận được dự đoán xen kẽ của mô hình A và B. Sau đó ta tiến hành theo dõi hiệu suất của mỗi mô hình bằng cách dô lường sự lựa chọn của người dùng đối với mỗi dự đoán của mô hình.

A diagram of a group of people

Description automatically generated

* Ưu điểm:
  + Netflix đã tiến hành thực nghiệm và phát hiện ra việc triển khai phương pháp Interleaving Experiments sẽ tốt nhất và cho ra kết quả đáng tin cậy đối với kích thước mẫu dữ liệu nhỏ hơn đáng kể so với A/B testing.
* Nhược điểm:
  + Triển khai phức tạp
  + Tốn chi phí tính toán nhiều do phải lấy kết quả dự đoán của nhiều mô hình
  + Không hiệu quả cho các nhiệm vụ hồi quy

#### Bandits

* Bandits là một thuật toán giữ lại việc theo dõi hiệu suất hiện tại của mỗi biến mô hình và đưa ra lựa chọn đối với mỗi yêu cầu về việc sử dụng mô hình có hiệu suất tốt nhất (tức là khai thác kiến thức hiện tại) hoặc thử một trong các mô hình khác để thu thập thông tin về chúng (khám phá trong trường hợp một trong các mô hình khác có kết quả tốt hơn).
* Thuật toán quyết định việc lựa chọn mô hình nào thông qua một biến được thêm vào: chi phí cơ hội (opportunity cost).
* Có nhiều thuật toán bandits chẳng hạn như: epsilon-greedy, Thompson Sampling và Upper Confidence Bound (UCB).
* Ưu điểm:
  + Cần ít dữ liệu hơn so với A/B testing để xác định xem mô hình nào tốt hơn.
  + An toàn hơn so với A/B testing bởi vì nếu tồn tại một mô hình tệ và kém hiệu quả thì thuật toán sẽ ít chọn mô hình đó hơn. Thuật toán có sự hội tụ nhanh hơn nên ta có thể loại bỏ mô hình kém hiệu quả một cách nhanh chóng.
* Nhược điểm:
  + Khó triển khai hơn các loại mô hình khác.
  + Chỉ có thể sử dụng cho một số trường hợp cụ thể.
  + Không an toàn hơn so với Shadow deploy vì thuật toán nhận dữ liệu người dùng thực tế.