VIETNAM GENERAL CONFEDERATION OF LABOR
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY

PROJECT

INTRODUCTION TO MACHINE LEARNING

FINAL PROJECT

Authors: **Tran Minh Thuy – 521H0312**

Class**:** **21H50301**

Admission Course**:** **25**

Supervisor: **Prof. LE ANH CUONG**

HO CHI MINH CITY, 2023

VIETNAM GENERAL CONFEDERATION OF LABOR
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY

PROJECT

INTRODUCTION TO MACHINE LEARNING

FINAL PROJECT

Authors: **Tran Minh Thuy – 521H0312**

Class**:** **21H50301**

Admission Course**:** **25**

Supervisor: **Prof. LE ANH CUONG**

HO CHI MINH CITY, 2023

ACKNOWLEDGEMENTS

We are really grateful because we managed to complete our report assignment about Introduction To Machine Learning within the time given. We sincerely thank our lecturer, Mr. Le Anh Cuong for the guidance and encouragement in finishing this assignment and also teaching us in this course.

PROJECT COMPLETED AT TON DUC THANG UNIVERSITY

We hereby declare that this is our own project and is under the guidance of Mr. Le Anh Cuong. The research contents and results in this topic are honest and have not been published in any publication before. The data in the tables for analysis, comments and evaluation are collected by the author himself from different sources, clearly stated in the reference section.

In addition, the project also uses a number of comments, assessments as well as data of other authors, other agencies and organizations, with citations and source annotations.

If we find any fraud, we will take full responsibility for the content of our project. Ton Duc Thang University is not related to copyright and copyright violations caused by us (if any).

Ho Chi Minh City, 24th December , 2023

Authors

(signature and full name)

Tran Minh Thuy

LECTURER'S ASSESSMENT

Ho Chi Minh, date

(sign)

ABSTRACT

CONTENT

LIST OF ABBREVIATION

LIST OF FIGURE

Câu 1:

1. Tìm hiểu, so sánh các phương pháp Optimizer trong huấn luyện mô hình học máy.

1.1 Gradient Descent (GD):

- GD là một khái niệm cơ bản trong học sâu nhằm điều phối quá trình tối ưu hóa các mô hình phức tạp. Về cơ bản, nó là một kỹ thuật tối ưu hóa số nhằm mục đích giảm thiểu một hàm chi phí nhất định nhằm xác định các tham số mạng thần kinh lý tưởng như trọng số và độ lệch.
- Khi đào tạo các mô hình dựa trên mạng thần kinh, việc học xảy ra trong quá trình lan truyền ngược. GD là một thuật ngữ để tối ưu hóa độ lệch và trọng số theo hàm chi phí. Sự khác biệt giữa sản lượng dự kiến và thực tế được đánh giá bằng hàm chi phí.
- Giảm dần độ dốc (GD) là một thuật toán tối ưu hóa được sử dụng rộng rãi trong học sâu, được sử dụng để giảm thiểu hàm chi phí của các mô hình mạng thần kinh trong quá trình đào tạo. Các trọng số hoặc tham số của mô hình được điều chỉnh lặp đi lặp lại theo độ dốc âm của hàm chi phí cho đến khi đạt được giá trị tối thiểu của hàm chi phí.
- Giảm dần độ dốc là một phương pháp lặp được sử dụng trong các bài toán machine learning và deep learning (thường là các bài toán tối ưu lồi - Convex Optimization) với mục tiêu tìm ra một tập hợp các biến nội bộ (tham số bên trong) để tối ưu hóa một mô hình thuật toán tối ưu hóa. Vì thế:
    - Gradient: Độ dốc hoặc tốc độ giảm của độ dốc. Về mặt toán học, gradient của hàm là đạo hàm của hàm tương ứng với từng biến trong hàm. Đối với hàm một biến, chúng ta sử dụng khái niệm đạo hàm thay vì gradient.
    - Descent: Viết tắt của Descending, nghĩa là giảm dần.
- Giảm dần độ dốc có nhiều dạng, bao gồm giảm độ dốc ngẫu nhiên (SGD) và SDG lô nhỏ. Tuy nhiên, về cơ bản nó được thực hiện như thế này:
    - Khởi tạo các biến nội bộ.
    - Đánh giá mô hình bằng cách sử dụng các biến nội bộ và hàm mất mát.
    - Cập nhật các biến nội bộ để tìm điểm tối ưu.
    - Lặp lại bước 2 và 3 cho đến khi gặp điều kiện dừng.
    - Biểu thức cập nhật cho GD có thể được viết là:
    - Ưu điểm của phương pháp giảm độ dốc (GD) và các biến thể của nó:
        + Được sử dụng rộng rãi: GD và các biến thể của nó được sử dụng rộng rãi trong các bài toán tối ưu hóa và học máy vì chúng hiệu quả và dễ thực hiện.
        + Sự hội tụ: GD và các biến thể của nó có thể hội tụ đến mức tối thiểu toàn cầu hoặc mức tối thiểu cục bộ phù hợp của hàm chi phí, tùy thuộc vào vấn đề và biến thể được sử dụng.
        + Khả năng mở rộng: Nhiều biến thể của phương pháp GD có thể được song song hóa và chia tỷ lệ thành các tập dữ liệu lớn và mô hình nhiều chiều.
        + Tính linh hoạt: Các biến thể khác nhau của phương pháp GD cân bằng độ chính xác và tốc độ và có thể được tùy chỉnh để tối ưu hóa hiệu suất cho các vấn đề cụ thể.
    - Nhược điểm của phương pháp giảm độ dốc (GD) và các biến thể của nó:

+ Lựa chọn tốc độ học: Việc lựa chọn tốc độ học rất quan trọng đối với sự hội tụ trong quá trình giảm độ dốc và các biến thể của nó. Việc chọn tốc độ học quá lớn có thể gây ra biến động và vọt lố, còn việc chọn tốc độ học quá nhỏ có thể gây ra sự hội tụ chậm hoặc bị kẹt ở mức tối thiểu cục bộ.
+ Độ nhạy đối với việc khởi tạo: Độ dốc giảm dần và các biến thể của nó có thể bị ảnh hưởng bởi việc khởi tạo tham số mô hình, điều này có thể ảnh hưởng đến độ hội tụ và chất lượng giải pháp.
+ Tốn thời gian: Giảm dần độ dốc và các biến thể của nó có thể tốn thời gian, đặc biệt là khi xử lý các tập dữ liệu lớn và mô hình nhiều chiều. Tốc độ hội tụ cũng phụ thuộc vào biến thể được sử dụng và vấn đề cụ thể.
+ Tối ưu cục bộ: Độ dốc giảm dần và các biến thể của nó có thể hội tụ đến mức tối thiểu cục bộ thay vì mức tối thiểu toàn cầu của hàm chi phí, đặc biệt đối với các vấn đề không lồi. Điều này có thể ảnh hưởng đến chất lượng của giải pháp và các kỹ thuật như khởi tạo ngẫu nhiên và khởi động lại nhiều lần có thể được sử dụng để giảm bớt vấn đề này.

1.2 Stochastic Gradient Descent (SGD)

- Giảm dần độ dốc ngẫu nhiên (SGD) là một biến thể của thuật toán giảm độ dốc (GD) được sử dụng để tối ưu hóa các mô hình học máy. Điều này giải quyết sự thiếu hiệu quả tính toán của các kỹ thuật GD truyền thống khi xử lý lượng lớn dữ liệu trong các dự án học máy.
- Thay vì sử dụng toàn bộ tập dữ liệu trong mỗi lần lặp, SGD chỉ chọn một mẫu huấn luyện ngẫu nhiên duy nhất (hoặc lô nhỏ) để tính toán độ dốc và cập nhật các tham số mô hình. Lựa chọn ngẫu nhiên này đưa tính ngẫu nhiên vào quá trình tối ưu hóa, do đó có thuật ngữ "ngẫu nhiên" trong quá trình giảm độ dốc ngẫu nhiên. Ưu điểm của việc sử dụng SGD là hiệu quả tính toán, đặc biệt khi xử lý lượng lớn dữ liệu. Việc sử dụng một mẫu hoặc lô nhỏ giúp giảm đáng kể số lượng tính toán trên mỗi lần lặp so với phương pháp giảm độ dốc truyền thống vốn yêu cầu xử lý toàn bộ tập dữ liệu.
- SGD chỉ sử dụng một mẫu huấn luyện để tính gradient và cập nhật các tham số trong mỗi lần lặp. Điều này có thể nhanh hơn so với việc giảm độ dốc hàng loạt nhưng có thể gây ra nhiều nhiễu hơn trong các bản cập nhật.
- Ưu điểm của phương pháp Giảm dần độ dốc ngẫu nhiên (SGD):

+ Tốc độ: SGD nhanh hơn so với giảm độ dốc hàng loạt và các biến thể khác của giảm độ dốc hàng loạt vì nó chỉ sử dụng một phiên bản để cập nhật tham số.
+ Hiệu quả bộ nhớ: SGD cập nhật tất cả các tham số trên mỗi ví dụ huấn luyện, giúp tiết kiệm bộ nhớ và có thể xử lý các tập dữ liệu lớn không vừa bộ nhớ.
+ Tránh cực tiểu cục bộ: Cập nhật SGD ồn ào và có thể tránh cực tiểu cục bộ và hội tụ đến mức tối thiểu toàn cầu.

- Nhược điểm của phương pháp Giảm dần độ dốc ngẫu nhiên (SGD):

+ Cập nhật ồn ào: Cập nhật SGD ồn ào và có phương sai cao, điều này có thể khiến việc tối ưu hóa trở nên không ổn định hơn và gây ra biến động quanh mức tối thiểu.
+ Hội tụ chậm: SGD cập nhật từng tham số một mẫu huấn luyện tại một thời điểm, do đó có thể cần nhiều lần lặp hơn để hội tụ đến mức tối thiểu.
+ Độ nhạy với tốc độ học: SGD rất nhạy cảm với việc lựa chọn tốc độ học vì

sử dụng tốc độ học cao có thể khiến thuật toán vượt quá giá trị tối thiểu, trong khi sử dụng tốc độ học thấp có thể khiến thuật toán hội tụ chậm và trở nên quan trọng.

+ Độ chính xác kém: Do các bản cập nhật bị nhiễu nên SGD có thể không hội tụ về mức tối thiểu toàn cục chính xác, dẫn đến giải pháp dưới mức tối ưu. Điều này có thể được giảm thiểu bằng cách sử dụng các kỹ thuật như lập kế hoạch tốc độ học tập và cập nhật dựa trên động lượng.

1.3 Mini-batch Gradient Descent:

- Giảm độ dốc hàng loạt nhỏ là một kỹ thuật tối ưu hóa thường được sử dụng trong các mô hình học máy đào tạo. Nó là sự kết hợp giữa giảm độ dốc (GD) và giảm độ dốc ngẫu nhiên (SGD).
- Nghĩa là, một tập hợp con dữ liệu nhỏ hơn để đào tạo giảm dần độ dốc theo đợt nhỏ được chia thành các đợt nhỏ. GD lô nhỏ cập nhật trọng số mỗi khi nó đi qua dữ liệu như GD, mọi mẫu như SGD hoặc mỗi khi nó đi qua dữ liệu như GD.
- Quá trình giảm độ dốc theo lô nhỏ sử dụng một tập hợp các mẫu huấn luyện nhỏ để tính toán độ dốc và cập nhật các tham số ở mỗi lần lặp. Nó nhanh hơn so với giảm độ dốc hàng loạt và ít nhiễu hơn so với giảm độ dốc ngẫu nhiên, khiến nó trở thành một lựa chọn tốt để kết hợp giảm độ dốc hàng loạt và giảm độ dốc ngẫu nhiên.
- Minibatch GD tính toán độ dốc dựa trên một tập hợp nhỏ các điểm dữ liệu ngẫu nhiên được gọi là minibatch.
    - Ưu điểm của phương pháp Mini-batch GD:
        + Hiệu suất cao hơn: Thường nhanh hơn GD vì nó sử dụng thông tin độ dốc từ lượng nhỏ dữ liệu.
        + Ước tính độ dốc ổn định hơn: Tùy thuộc vào vấn đề, ước tính độ dốc từ lô nhỏ có thể ổn định hơn SGD và tốt hơn trong việc đạt mức tổn thất tối thiểu.
    - Nhược điểm của phương pháp Mini-batch GD:
        + Chọn kích thước lô nhỏ: Việc chọn kích thước lô nhỏ thích hợp có thể ảnh hưởng đến tốc độ đào tạo và độ ổn định hội tụ.
        + Yêu cầu quản lý bộ nhớ: Kích thước lô nhỏ lớn hơn có thể yêu cầu nhiều bộ nhớ hơn.

1.4 RMSprop:

- Tối ưu hóa Adam là một phương pháp giảm độ dốc ngẫu nhiên dựa trên ước tính thích ứng của khoảnh khắc thứ nhất và thứ hai.
- Theo Kingma và cộng sự, năm 2014, phương pháp này "hiệu quả về mặt tính toán, yêu cầu ít bộ nhớ hơn, không chia tỷ lệ theo đường chéo theo độ dốc và phù hợp với các vấn đề/tham số dữ liệu lớn".
- Trong biến thể này, tốc độ học cho từng tham số được điều chỉnh thích ứng dựa trên đường trung bình động của gradient bình phương. Điều này cho phép thuật toán hội tụ nhanh hơn trên các gradient nhiễu.
- Thuật toán RMSprop sử dụng đường trung bình động có trọng số theo cấp số nhân của gradient bình phương để cập nhật các tham số.
- Phương trình toán học của RMSprop là: Khởi tạo các tham số.
- Tỷ lệ học tập: α
- Tốc độ phân rã theo cấp số nhân của mức trung bình: γ
- Hằng số nhỏ cho sự ổn định số: ε

- Giá trị ban đầu của tham số: θ
- Khởi tạo độ dốc tích lũy (trung bình theo cấp số nhân): Độ dốc bình phương tích lũy của từng tham số: Et=0
- Lặp lại cho đến khi hội tụ hoặc lặp lại tối đa.
- Ưu điểm của phương pháp RMSprop:

    + Hiệu suất ổn định: RMSprop giúp ổn định quá trình đào tạo bằng cách điều chỉnh tốc độ học dựa trên kích thước gradient hiện tại. Điều này cho phép chúng tôi ước tính tốt hơn giá trị tổn thất tối thiểu.
    + Xử lý các vấn đề về tốc độ học không đồng đều: Tốc độ học không đồng đều trên các tham số có thể gây ra các vấn đề về đào tạo. RMSprop giảm bớt vấn đề này bằng cách điều chỉnh tốc độ học để phản ánh kích thước của gradient.
- Nhược điểm của phương pháp RMSprop:

    + Lựa chọn tham số: Để đảm bảo mô hình hoạt động tốt nhất, bạn cần điều chỉnh các siêu tham số RMSprop, chẳng hạn như hệ số quan trọng (giảm cân) và tham số beta.

    + Đồng bộ hóa kém: RMSprop có thể không đồng bộ hóa tốt trong một số trường hợp, đặc biệt khi áp dụng cho các mô hình phức tạp hoặc dữ liệu khác nhau.

1.5 Adagrad:

- Adagrad là một trình tối ưu hóa với tốc độ học theo tham số cụ thể, điều chỉnh theo tần suất cập nhật tham số trong quá trình đào tạo. Một tham số càng nhận được nhiều cập nhật thì càng nhận được ít cập nhật hơn.
- Trong biến thể này, tốc độ học cho từng tham số được điều chỉnh thích ứng dựa trên thông tin độ dốc trong quá khứ. Điều này cho phép cập nhật lớn hơn cho các tham số hiếm và cập nhật nhỏ hơn cho các tham số phổ biến.
- Adagrad là viết tắt của Trình tối ưu hóa độ dốc thích ứng. Có các trình tối ưu hóa như giảm độ dốc, giảm độ dốc ngẫu nhiên và SGD lô nhỏ, tất cả đều được sử dụng để giảm các hàm mất mát liên quan đến trọng lượng. Công thức cập nhật trọng số là:
- Ưu điểm của phương pháp Adagrad:

    + Tốc độ điều chỉnh học tập tự động (tốc độ học tập): Adagrad điều chỉnh tốc độ học tập cho từng tham số dựa trên tần suất xuất hiện của gradient. Điều này có nghĩa là tốc độ học của các tham số dốc giảm, tốc độ học ngược giảm và quá trình hội tụ trở nên ổn định.

    + Phù hợp nhất với dữ liệu thưa thớt: Adagrad thích ứng với tốc độ xảy ra các điểm kỳ dị, khiến nó phù hợp với dữ liệu thưa thớt.
- Nhược điểm của phương pháp Adagrad:
    + Bản chất dần dần của tốc độ học tập: Vấn đề với Adagrad là tốc độ học tập giảm dần theo thời gian. Điều này có thể khiến tốc độ học trở nên quá thấp sau một vài lần chạy chương trình huấn luyện, làm chậm quá trình hội tụ.
    + Xác suất vượt quá mục tiêu (target overshoot): Trong một số trường hợp, Adagrad có thể tối ưu hóa quá mức các tham số dốc, dẫn đến mục tiêu vượt quá mục tiêu và không thể đạt đến điểm tối thiểu mong muốn. Quản lý bộ nhớ: Adagrad cần lưu trữ lịch sử của từng tham số

nên bộ nhớ rất tốn kém khi làm việc với số lượng lớn tham số.

1.6 Adam:

- Tối ưu hóa Adam là một phương pháp giảm độ dốc ngẫu nhiên dựa trên ước tính thích ứng của khoảnh khắc thứ nhất và thứ hai.
- Theo Kingma và cộng sự, năm 2014, phương pháp này "hiệu quả về mặt tính toán, yêu cầu ít bộ nhớ hơn, không chia tỷ lệ theo đường chéo theo độ dốc và phù hợp với các vấn đề/tham số dữ liệu lớn".
- Adam là viết tắt của Ước tính thời điểm thích ứng và kết hợp các lợi ích của việc giảm dần dựa trên động lượng, Adagrad và RMSprop. Tốc độ học được điều chỉnh thích ứng cho từng tham số dựa trên đường trung bình động của gradient và gradient bình phương, dẫn đến độ hội tụ nhanh hơn và hiệu suất tốt hơn cho các bài toán tối ưu hóa không lồi. Đây là sự khác biệt giữa ước tính thời điểm đầu tiên, là mức trung bình giảm theo cấp số nhân của độ dốc trong quá khứ và ước tính thời điểm thứ hai, là mức trung bình giảm dần và tăng dần theo cấp số nhân theo bình phương của độ dốc trong quá khứ. hai. Ước tính khoảnh khắc đầu tiên được sử dụng để tính toán động lượng và ước tính khoảnh khắc thứ hai được sử dụng để chia tỷ lệ tốc độ học cho từng tham số. Đây là một trong những thuật toán tối ưu hóa deep learning phổ biến nhất.
- Ưu điểm của phương pháp Adam:
  + Hiệu suất cao: Adam thường không yêu cầu điều chỉnh nhiều siêu tham số và hoạt động tốt trên nhiều mô hình và dữ liệu.
  + Tự động điều chỉnh tốc độ học (Learning Rate): Adam tự động điều chỉnh tốc độ học cho từng thông số dựa trên độ dốc và mức trung bình của các độ dốc. Điều này làm cho tiến trình của chuỗi nhanh hơn và ổn định hơn.
  + Hiệu quả trên các tập dữ liệu lớn và thưa thớt: Adam thường xuyên hoạt động tốt trên các tập dữ liệu lớn và có thể tự điều chỉnh tốc độ học của mình dựa trên tần suất đặc trưng.
- Nhược điểm của phương pháp Adam:
  + Yêu cầu về bộ nhớ: Adam yêu cầu nhiều bộ nhớ hơn các kỹ thuật tối ưu hóa khác vì nó phải lưu trữ thông tin bổ sung như gradient bậc hai và bậc hai cho từng tham số.
  + Công suất dư thừa: Trong một số trường hợp, Adam có thể vượt quá mức tối thiểu âm và không đạt đến mức ổn định.
  + Yêu cầu tối ưu hóa siêu tham số: Mặc dù Adam tự điều chỉnh tốc độ học, nhưng các tối ưu hóa siêu tham số khác như Beta1 và Beta2 vẫn được yêu cầu để đạt được hiệu suất tốt nhất cho từng vấn đề cụ thể.

Câu 2:

2. Tìm hiểu về Continual Learning và Test Production khi xây dựng một giải pháp học máy để giải quyết một bài toán nào đó.

2.1Continual Learning

2.1.1 Khái niệm:

- Học liên tục (CL) tập trung vào phát triển các mô hình học các nhiệm vụ mới trong khi vẫn giữ lại thông tin từ các nhiệm vụ trước đó. CL là một lĩnh vực nghiên cứu quan trọng vì nó giải quyết các tình huống trong thế giới thực, nơi dữ liệu và nhiệm

vụ liên tục thay đổi và các mô hình cần thích ứng với những thay đổi này mà không quên kiến thức trước đó của chúng.

- Trong học máy truyền thống, một mô hình được đào tạo trên một tập dữ liệu cố định và được yêu cầu thực hiện một nhiệm vụ duy nhất. Tuy nhiên, cách tiếp cận này trở nên có vấn đề khi dữ liệu và nhiệm vụ thay đổi và trở nên linh hoạt, vì mô hình phải có khả năng thích ứng và học hỏi từ dữ liệu mới theo thời gian. Đây là lúc CL phát huy tác dụng, cho phép mô hình liên tục học hỏi và cải thiện mà không quên kiến thức trước đó.
- Một trong những thách thức lớn nhất của CL là vấn đề quên thảm họa. Trong bài toán này, một mô hình được huấn luyện trên nhiều nhiệm vụ phải ghi nhớ thông tin đã học được từ các nhiệm vụ trước đó khi tiếp xúc với dữ liệu mới. Nhiều kỹ thuật khác nhau đã được phát triển để giải quyết thách thức này, chẳng hạn như mạng chính quy hóa và mạng tăng cường bộ nhớ.
- Tóm lại, CL là một lĩnh vực quan trọng của học máy nhằm giải quyết thách thức về các mô hình đào tạo có thể liên tục học hỏi và thích ứng với các nhiệm vụ và dữ liệu mới mà không quên kiến thức trước đó. Việc sử dụng các kỹ thuật như mạng chính quy hóa và mở rộng bộ nhớ, cũng như thiết kế kiến trúc mô hình, đóng một vai trò quan trọng trong khả năng thích ứng của các mô hình CL.

2.1.2 Trường hợp sử dụng:

- Phát hiện bất thường – CL đặc biệt hữu ích trong các tình huống phát hiện bất thường trong đó việc phân phối dữ liệu thay đổi theo thời gian và các thuật toán học máy truyền thống có thể không hiệu quả. Mục tiêu của kiểu học liên tục này là liên tục theo dõi hành vi bình thường của hệ thống để tìm hiểu các cấp độ vận hành, quy trình hoặc luồng dữ liệu điển hình của nó và so sánh dữ liệu mới với hành vi bình thường đã học được. Điều này nhằm phát hiện những bất thường. Để làm được điều này, thuật toán học được đào tạo liên tục trên luồng dữ liệu để tìm hiểu hành vi bình thường của hệ thống. Khi có dữ liệu mới, nó sẽ được so sánh với hành vi bình thường đã học được và những sai lệch so với chuẩn mực được đánh dấu là bất thường. Các thuật toán học liên tục sau đó có thể được cập nhật để kết hợp dữ liệu mới, cho phép chúng thích ứng với những thay đổi trong hành vi bình thường theo thời gian.
- Cá nhân hóa – Một trường hợp sử dụng khác của CL là hệ thống đề xuất được cá nhân hóa. Nó nhằm mục đích cung cấp cho người dùng các đề xuất kịp thời và có tính tùy chỉnh cao. Bằng cách liên tục tìm hiểu về sở thích và hành vi của người dùng, hệ thống đề xuất có thể cải thiện khả năng đưa ra đề xuất chính xác và phù hợp. Để thực hiện được điều này, các thuật toán học liên tục dựa trên dữ liệu lịch sử tương tác của người dùng, bao gồm: B. Lịch sử mua hàng và truy vấn tìm kiếm. Họ được đào tạo để tìm hiểu sở thích và kiểu hành vi của bạn. Khi có dữ liệu mới, mô hình sẽ cập nhật hiểu biết về sở thích và hành vi của người dùng. Thông tin cập nhật này được sử dụng để cung cấp cho bạn các đề xuất chính xác và phù hợp hơn.
- Dự báo – CL cũng có thể được sử dụng trong lĩnh vực dự báo để liên tục cập nhật dự báo dựa trên dữ liệu mới khi có sẵn. Cách tiếp cận này cho phép các mô hình thích ứng với những thay đổi trong phân phối dữ liệu và duy trì độ chính xác theo thời gian, ngay cả khi xử lý dữ liệu phức tạp và động. Trong hệ thống dự báo, thuật toán học tập được đào tạo liên tục trên luồng dữ liệu lịch sử, chẳng hạn như dữ liệu tài chính hoặc dữ liệu chuỗi thời gian, để đưa ra dự đoán về các sự kiện trong tương lai. Khi có dữ liệu mới, mô hình sẽ cập nhật hiểu biết về các mối quan hệ trong dữ

liệu và sử dụng thông tin này để cải thiện dự đoán của mình.

2.1.3 Quá trình của Continual Learning (CL):

- CL là sự phát triển của mô hình học máy truyền thống. Do đó, nó bao gồm nhiều nguyên tắc mô hình hóa giống nhau như tiền xử lý, lựa chọn mô hình, điều chỉnh siêu tham số, đào tạo, triển khai và giám sát.
- Quá trình CL yêu cầu hai bước bổ sung: lấy mẫu dữ liệu và thực hiện chiến lược học tập liên tục. Các bước này cho phép mô hình học hỏi một cách hiệu quả từ các luồng dữ liệu mới dựa trên ngữ cảnh của ứng dụng và tác vụ dữ liệu của bạn.

2.1.4 Ưu điểm của Continual Learning (CL):

- Truy cập vào dữ liệu mới: Học liên tục cho phép các mô hình học máy tiếp tục học từ dữ liệu mới, giúp chúng cải thiện dần các đề xuất và dự đoán dựa trên thông tin mới.
- Tính linh hoạt: Thích ứng với môi trường thay đổi, dữ liệu mới và thậm chí các vấn đề khác mà không cần đào tạo lại.
- Tiết kiệm tài nguyên: Bằng cách sử dụng lại kiến thức đã học, việc học liên tục có thể giảm thiểu nhu cầu đào tạo lại toàn bộ mô hình từ đầu khi có dữ liệu mới.

2.1.5 Nhược điểm của Continual Learning (CL):

- Lãng phí thông tin cũ: Khi một mô hình học từ dữ liệu mới, thông tin cũ có thể bị lãng quên, điều này làm giảm hiệu suất và độ chính xác của mô hình đối với dữ liệu cũ.
- Hiện tượng "quên thảm khốc": Khi một mô hình tập trung vào việc học từ dữ liệu mới, nó có thể quên những kiến thức đã học trước đó và nhanh chóng quên đi những kỹ năng, dự đoán cũ.
- Khó khăn trong việc ổn định các mô hình: Học liên tục đòi hỏi sự cân bằng cẩn thận giữa việc học mới và bảo tồn kiến thức cũ để tránh các mô hình trở nên sai lệch hoặc tối ưu hóa quá mức cho dữ liệu mới.

2.2 Test Production

2.2.1 Khái Niệm

- Sản xuất thử nghiệm là giai đoạn bạn thử nghiệm mô hình AI của mình bằng dữ liệu thực trước khi triển khai. Mục tiêu là đánh giá độ chính xác, hiệu suất và độ ổn định của mô hình trong điều kiện thực tế.
- Các bước chính trong quá trình sản xuất thử nghiệm: thu thập dữ liệu thực, tạo bộ thử nghiệm, kiểm tra và đánh giá mô hình, xác định và khắc phục sự cố, đồng thời lặp lại quy trình cho đến khi đáp ứng yêu cầu.
- Các kỹ thuật phổ biến được sử dụng trong sản xuất thử nghiệm bao gồm thử nghiệm A/B, chế độ bóng tối và triển khai canary.

2.2.2 Các dạng test

- Có bốn loại thử nghiệm chính được sử dụng ở các thời điểm khác nhau trong chu kỳ phát triển:

1. **Kiểm tra đơn vị:** Kiểm tra các thành phần riêng lẻ, mỗi thành phần có một trách nhiệm (ví dụ: chức năng lọc danh sách).
2. **Kiểm thử tích hợp:** Kiểm tra chức năng kết hợp của từng thành phần riêng lẻ (chẳng hạn như xử lý dữ liệu).
3. **Kiểm tra hệ thống:** Kiểm tra thiết kế hệ thống của bạn để đảm bảo nó tạo ra kết quả mong đợi dựa trên đầu vào (đào tạo, suy luận, v.v.).
4. **Kiểm tra chấp nhận:** Kiểm tra để kiểm tra xem các yêu cầu có được đáp ứng hay không. Nó thường được gọi là Kiểm tra chấp nhận của người dùng (UAT).
5. **Kiểm tra hồi quy:** Kiểm tra các lỗi đã xảy ra trong quá khứ để đảm bảo rằng những thay đổi mới không khiến những lỗi đó xảy ra lần nữa.

=> Hệ thống ML có bản chất là xác suất nhưng chúng chứa nhiều thành phần xác định có thể được kiểm tra theo cách tương tự như các hệ thống phần mềm truyền thống. Sự khác biệt trong việc thử nghiệm hệ thống ML bắt đầu khi bạn chuyển từ thử nghiệm mã sang thử nghiệm dữ liệu và mô hình.

2.2.3 Chúng ta nên thử nghiệm như thế nào?

Khung chúng tôi sử dụng khi viết bài kiểm tra là phương pháp xác nhận hành động sắp xếp.

- Sắp xếp: Thiết lập các đầu vào khác nhau để thử nghiệm.
- Hành động: Áp dụng đầu vào cho thành phần bạn đang kiểm tra.
- Xác minh: Xác minh rằng bạn đã nhận được kết quả như mong đợi.

Python có một số công cụ như Unittest và Pytest giúp dễ dàng chạy thử nghiệm tuân thủ khung Xác nhận Hành động Sắp xếp. Những công cụ này có các tính năng tích hợp mạnh mẽ như tham số hóa và bộ lọc, cho phép bạn kiểm tra nhiều điều kiện trên quy mô lớn.

2.2.4 Chúng ta nên kiểm tra những gì?

Bạn nên kiểm tra những khía cạnh nào của đầu vào và đầu ra khi sắp xếp đầu vào và kiểm tra đầu ra dự kiến?

- Đầu vào: kiểu dữ liệu, định dạng, độ dài, trường hợp cạnh (tối thiểu/tối đa, nhỏ hơn/lớn hơn, v.v.)
- Đầu ra: kiểu dữ liệu, định dạng, ngoại lệ, đầu ra trung gian và cuối cùng

2.2.5 Thực hành.

Bất kể chúng ta sử dụng framework nào, điều quan trọng là phải gắn chặt việc thử nghiệm vào quá trình phát triển.

- Nguyên tử: Khi bạn tạo một hàm hoặc lớp, bạn muốn nó có một trách nhiệm duy nhất để có thể dễ dàng kiểm tra nó. Nếu không, bạn sẽ cần chia nó thành các thành phần chi tiết hơn.
- Cấu hình: Khi tạo một thành phần mới, bạn cần tạo các thử nghiệm để xác minh chức năng của nó. Đây là một cách tuyệt vời để đảm bảo độ tin cậy và phát hiện lỗi sớm.
- Tái sử dụng: Bạn cần duy trì một kho lưu trữ trung tâm nơi chức năng cốt lõi được kiểm tra tại nguồn và tái sử dụng trên nhiều dự án. Điều này giúp giảm đáng kể nỗ lực thử nghiệm cơ sở mã cho từng dự án mới.
- Hồi quy: Chúng tôi muốn tính đến các lỗi mới xảy ra trong quá trình kiểm tra hồi

quy và tránh lặp lại các lỗi tương tự trong tương lai.

- Phạm vi bao phủ: Chúng tôi muốn đảm bảo bao phủ 100% cơ sở mã của chúng tôi. Điều quan trọng là tính toán các bài kiểm tra theo từng dòng thay vì viết bài kiểm tra cho từng dòng mã.
- Tự động hóa: Tôi muốn chạy thử nghiệm tự động khi thực hiện thay đổi đối với cơ sở mã của mình, trong trường hợp tôi quên chạy chúng trước khi đẩy chúng vào kho lưu trữ.

REFERENCE

1. https://www.geeksforgeeks.org/intuition-behind-adagrad-optimizer/amp/
2. https://wiki.continualai.org/the-continualai-wiki/introduction-to-continual-learning
3. https://madewithml.com/courses/mlops/testing/#types-of-tests