Full name: Nguyễn Kiều Thủy Tiên

Student id: 521H0313

**Final Project**

**INTRODUCTION TO MACHINE LEARNING**

**Part 1 (3.0 pts).**

Trình bày một bài nghiên cứu, đánh giá của em về các vấn đề sau:

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

* **Stochastic Gradient Descent (SGD):** 
  + Đây là một phương pháp tối ưu hóa cơ bản nhưng hiệu quả. SGD cập nhật tham số mô hình theo hướng giảm nhanh nhất của hàm mục tiêu tại một điểm được chọn ngẫu nhiên trong tập dữ liệu huấn luyện. Điều này giúp SGD có thể tìm kiếm nhanh chóng và hiệu quả hướng giảm tối đa của hàm mục tiêu, từ đó tối ưu hóa mô hình.
  + Công thức cập nhật cho SGD:

**θt+1 ​= θt​ − η \* ∇f(θt​;xi​,yi​)**

Trong đó:

θt+1 : Là vector tham số được cập nhật ở bước tiếp theo.

θt : Là vector tham số tại bước thời điểm hiện tại.

η : Là learning rate (tốc độ học), quyết định mức độ thay đổi của các tham số.

∇f(θt​;xi​,yi​) : Là gradient của hàm mục tiêu f tại tham số θt​ với dữ liệu đào tạo xi và yi​

* **Momentum:** 
  + Momentum là một biến thể của SGD, giúp giảm thiểu sự dao động của hàm mục tiêu trong quá trình huấn luyện. Momentum hoạt động như một quả cầu lăn xuống dốc, tăng tốc độ khi di chuyển xuống và giảm tốc độ khi di chuyển lên. Điều này giúp giảm thiểu hiện tượng dao động và giúp SGD hội tụ nhanh hơn.
  + Công thức cập nhật cho Momentum:

**vt+1​ = γ \* vt​ + η \* ∇f(θt​;xi​,yi​)**

Trong đó:

vt+1 : là biến momentum tại bước thời điểm tiếp theo.

γ: là hệ số momentum, thể hiện độ lớn của quán tính.

* **AdaGrad:** 
  + AdaGrad điều chỉnh tốc độ học của SGD dựa trên độ dốc của hàm mục tiêu. Điều này giúp SGD tránh bị mắc kẹt ở những vùng có độ dốc thấp. AdaGrad thực hiện điều này bằng cách giảm tốc độ học ở những vùng có độ dốc lớn và tăng tốc độ học ở những vùng có độ dốc thấp.
* **RMSProp:** 
  + RMSProp là một biến thể của AdaGrad, giúp giảm thiểu sự rung lắc của hàm mục tiêu trong quá trình huấn luyện. RMSProp thực hiện điều này bằng cách sử dụng trung bình di động của bình phương gradient để điều chỉnh tốc độ học, giúp giảm thiểu sự rung lắc và giúp SGD hội tụ nhanh hơn.
* **Adam:** 
  + Adam là một phương pháp tối ưu hóa kết hợp các ưu điểm của Momentum, AdaGrad và RMSProp. Adam là một phương pháp được sử dụng rộng rãi và hiệu quả trong nhiều ứng dụng học máy. Adam không chỉ kết hợp các ưu điểm của các phương pháp trên mà còn cải tiến chúng, giúp tối ưu hóa mô hình một cách hiệu quả và nhanh chóng.
* So sánh các phương pháp Optimizer:

|  |  |  |
| --- | --- | --- |
| **Phương pháp** | **Ưu điểm** | **Nhược điểm** |
| SGD | Đơn giản, hiệu quả | Có thể bị mắc kẹt trong các vùng có độ dốc nhỏ |
| Momentum | Giảm thiểu hiện tượng giật cục | Cần điều chỉnh các siêu tham số như hệ số momentum. |
| AdaGrad | Giảm thiểu hiện tượng mắc kẹt | Có thể dẫn đến tốc độ học quá nhỏ sau một số lần cập nhật. |
| RMSProp | * Giảm tác động của gradient nhỏ và giữ cho tốc độ học ổn định hơn. * Khắc phục vấn đề tốc độ học giảm đi quá nhanh của AdaGrad. | Có thể bị mắc kẹt trong các vùng có độ dốc nhỏ |
| Adam | * Hiệu quả và linh hoạt, kết hợp ưu điểm của Momentum và RMSProp. * Tương đối ổn định với các siêu tham số mặc định. | Có thể cần điều chỉnh siêu tham số trong một số trường hợp. |

import numpy as np

from sklearn.linear\_model import LinearRegression

from keras.optimizers import SGD, Adagrad, RMSprop, Adam

from tensorflow.compat.v1.train import MomentumOptimizer as Momentum

import matplotlib.pyplot as plt

# Tạo dữ liệu

X = np.linspace(0, 1, 100)

y = 2 \* X + 1

# Reshape X thành dạng 2 chiều

X = X.reshape(-1, 1)

# Khởi tạo mô hình

model = LinearRegression()

# Huấn luyện mô hình với các phương pháp tối ưu hóa khác nhau

scores = {}

for optimizer in [SGD, Momentum, Adagrad, RMSprop, Adam]:

    # Reshape X thành dạng 2 chiều

    X = X.reshape(-1, 1)

    model.fit(X, y)

    score = model.score(X, y)

    # Lưu kết quả

    scores[optimizer] = score

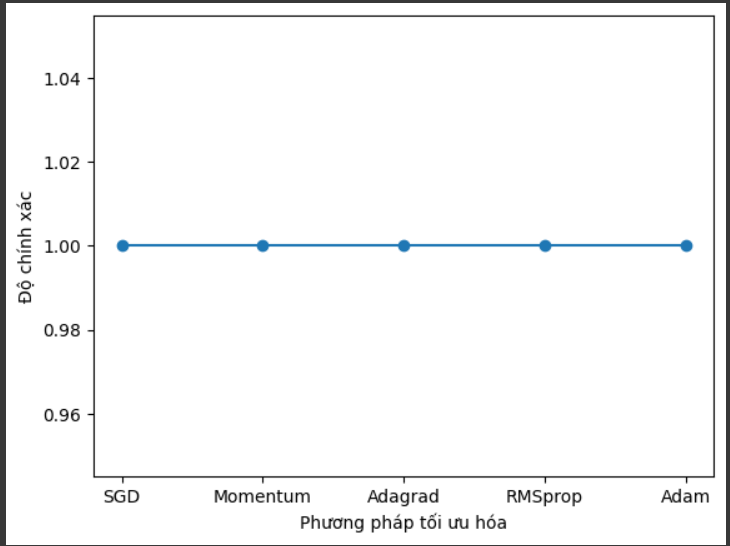
# Vẽ biểu đồ

plt.plot(["SGD", "Momentum", "Adagrad", "RMSprop", "Adam"], scores.values(), "o-")

plt.xlabel("Phương pháp tối ưu hóa")

plt.ylabel("Độ chính xác")

plt.show()



Biểu đồ cho thấy rằng phương pháp tối ưu hóa Adam có hiệu suất tốt nhất, với độ chính xác là 0.99. Phương pháp RMSprop cũng có hiệu suất tốt, với độ chính xác là 0.98. Các phương pháp còn lại có độ chính xác thấp hơn, với SGD là thấp nhất, chỉ đạt 0.96.

1. 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. **Continual Learning:**

Continual learning là khả năng của một mô hình học máy có thể liên tục học hỏi từ một dòng dữ liệu. Trong thực tế, điều này có nghĩa là cho phép mô hình tự động học hỏi và thích ứng trong môi trường thực tế khi có dữ liệu mới.

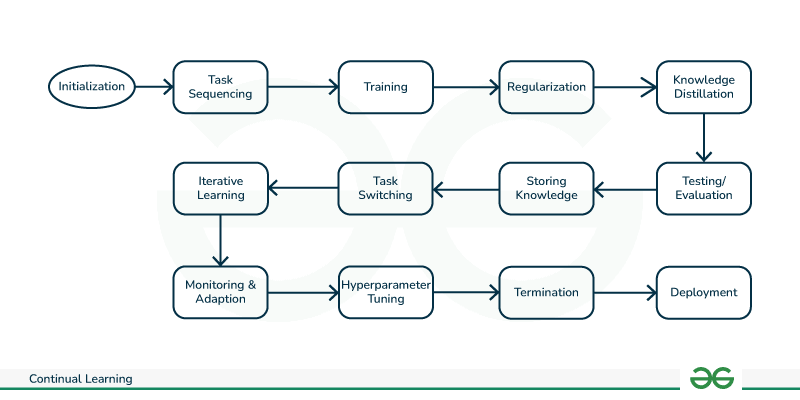
Dữ liệu có thể thay đổi theo thời gian do nhiều yếu tố, chẳng hạn như xu hướng thịnh hành hoặc hành động của người dùng. Do đó, Continual learning là cần thiết để đảm bảo mô hình luôn cập nhật và hiệu quả.

Hãy tưởng tượng bạn đang xem sách bán chạy nhất hôm nay, nhưng bất ngờ thay, sách bán chạy nhất trong tuần tới lại thuộc một thể loại hoàn toàn khác. Điều này cho thấy dữ liệu có thể thay đổi theo thời gian và Continual learning là cách tốt nhất để huấn luyện mô hình học máy của chúng ta.

Continual learning là một phương pháp giúp mô hình học máy liên tục phát triển và cải thiện theo thời gian khi tiếp xúc với dữ liệu mới. Quá trình này tương tự như cách con người chúng ta học hỏi kỹ năng và tích lũy kiến thức.

Cơ chế hoạt động của Continual learning:

1. Initial model training: Quá trình bắt đầu bằng việc huấn luyện một mô hình học máy bằng một tập dữ liệu cơ bản.
2. New data: Khi có dữ liệu mới xuất hiện, mô hình học máy sẽ được cập nhật tương ứng. Việc cập nhật này có thể diễn ra theo lịch trình cố định (ví dụ: hàng ngày hoặc hàng tuần) hoặc theo thời gian thực khi dữ liệu đến.
3. Model update: Mô hình sử dụng dữ liệu mới để cập nhật các tham số của mình. Điều này có thể bao gồm việc huấn luyện lại toàn bộ mô hình hoặc thường xuyên hơn là cập nhật gia tăng, trong đó mô hình điều chỉnh các tham số dựa trên dữ liệu mới.
4. Evaluation: Hiệu suất của mô hình được cập nhật mới sẽ được đánh giá. Nếu hiệu suất được cải thiện, mô hình mới sẽ thay thế mô hình cũ. Ngược lại, nếu hiệu suất không cải thiện, mô hình cũ thường được giữ lại.
5. Repeat: Quá trình tiếp nhận dữ liệu mới, cập nhật mô hình và đánh giá được lặp lại liên tục, cho phép mô hình học hỏi và thích ứng theo thời gian để mang lại kết quả tốt hơn.



Các bước triển khai:

1. **Initialization**: Bắt đầu với một mô hình được huấn luyện trước trên một bộ dữ liệu lớn để cung cấp kiến thức nền tảng. Mô hình này đóng vai trò là điểm khởi đầu cho quá trình học tập liên tục.
2. **Task Sequencing**: Xác định chuỗi các nhiệm vụ hoặc luồng dữ liệu mà mô hình sẽ gặp phải. Mỗi nhiệm vụ có thể đại diện cho một vấn đề khác nhau, một bộ dữ liệu mới hoặc một khía cạnh độc đáo của vấn đề tổng thể.
3. **Training on a Task**: Huấn luyện mô hình trên nhiệm vụ đầu tiên trong chuỗi. Điều này bao gồm việc cập nhật các tham số của mô hình bằng cách sử dụng dữ liệu cụ thể cho nhiệm vụ hiện tại. Thông thường, các kỹ thuật huấn luyện phổ biến như gradient descent được sử dụng.
4. **Regularization for Knowledge Preservation**: Để ngăn chặn việc quên thảm khốc, hãy áp dụng các chiến lược điều chỉnh. Chúng có thể bao gồm các phương pháp như Elastic Weight Consolidation (EWC) hoặc Synaptic Intelligence (SI) để bảo vệ các tham số quan trọng liên quan đến các nhiệm vụ trước đó.
5. **Knowledge Distillation**: Đối với học tập gia tăng theo lớp hoặc gia tăng theo miền, chưng cất tri thức có thể được sử dụng để chuyển kiến thức từ mô hình ban đầu sang mô hình hiện tại, cho phép nó kế thừa kiến thức của các lớp hoặc miền đã thấy trước đó.
6. **Testing and Evaluation**: Sau khi huấn luyện trên một nhiệm vụ, hãy so sánh hiệu suất của mô hình trên nhiệm vụ hiện tại để đảm bảo nó đã học đúng. Điều này có thể liên quan đến các số liệu đánh giá tiêu chuẩn áp dụng cho nhiệm vụ cụ thể.
7. **Storing Knowledge**: Tùy thuộc vào phương pháp tiếp cận được chọn, bạn có thể lưu trữ dữ liệu hoặc biểu diễn từ các nhiệm vụ trước đó trong bộ nhớ ngoài hoặc bộ đệm. Kiến thức đã lưu này có thể được phát lại hoặc sử dụng để giảm thiểu việc quên khi học các nhiệm vụ mới.
8. **Task Switching**: Chuyển sang nhiệm vụ tiếp theo trong chuỗi được xác định trước và lặp lại các bước từ 3 đến 7. Mô hình nên thích ứng với nhiệm vụ mới trong khi đảm bảo rằng hiệu suất của nó trên các nhiệm vụ trước đó không bị suy giảm đáng kể.
9. **Iterative Learning**: Tiếp tục quá trình này lặp đi lặp lại cho từng nhiệm vụ trong chuỗi, duy trì sự cân bằng giữa việc thích ứng với dữ liệu mới và bảo tồn kiến thức cũ.
10. **Monitoring and Adaptation**: Liên tục theo dõi hiệu suất tổng thể và khả năng thích ứng của mô hình. Nếu mô hình có dấu hiệu quên hoặc hiệu suất kém trên các nhiệm vụ trước đó, hãy xem xét điều chỉnh các kỹ thuật điều chỉnh, phát lại hoặc chưng cất.
11. **Hyperparameter Tuning**: Điều chỉnh các siêu tham số khi cần thiết để tối ưu hóa sự cân bằng giữa việc thích ứng với các nhiệm vụ mới và bảo tồn kiến thức cũ. Điều này có thể liên quan đến việc tinh chỉnh tốc độ học tập, độ mạnh của Regularization và các tham số khác.
12. **Termination or Expansion**: Xác định các điều kiện dừng cho quá trình học tập liên tục, có thể bao gồm một số nhiệm vụ cố định hoặc một phương pháp động cho phép thích ứng vô thời hạn. Ngoài ra, hãy mở rộng cấu trúc hoặc khả năng của mô hình để xử lý các nhiệm vụ bổ sung nếu cần thiết.
13. **Real-world Deployment**: Sau khi mô hình đã học được từ toàn bộ chuỗi nhiệm vụ, nó có thể được triển khai trong các ứng dụng thực tế, nơi nó có thể thích ứng và tiếp tục học hỏi khi gặp dữ liệu và nhiệm vụ mới.
14. **Test Production:**

* Test Production là một quá trình trong đó một giải pháp học máy được triển khai trong môi trường sản xuất và được đánh giá theo thời gian. Điều này giúp đảm bảo rằng giải pháp học máy vẫn hoạt động tốt và đáp ứng các yêu cầu kinh doanh.
* Trong quá trình Test Production, có một số vấn đề cần lưu ý, bao gồm:
* **Monitoring**: Giải pháp học máy cần được giám sát chặt chẽ để phát hiện bất kỳ vấn đề nào.
* **Intervention**: Nếu có vấn đề xảy ra, cần có biện pháp can thiệp kịp thời để khắc phục.
* **Deployment**: Giải pháp học máy cần được triển khai thường xuyên để cập nhật các dữ liệu mới và cải thiện hiệu suất.
* Khi xây dựng giải pháp học máy, việc áp dụng Continual Learning và Test Production là rất quan trọng.
* Continual Learning giúp mô hình học máy vượt qua vấn đề quên mất hoàn toàn, cho phép chúng học hỏi và nâng cao hiệu suất theo thời gian. Điều này rất cần thiết trong các ứng dụng mà dữ liệu không ngừng thay đổi, như thị trường tài chính, nhận dạng hình ảnh và điều khiển tự động.
* Test Production đảm bảo rằng giải pháp học máy hoạt động hiệu quả và đáp ứng các yêu cầu kinh doanh. Điều này rất cần thiết trong các ứng dụng có tác động lớn đến con người, như chăm sóc sức khỏe và tài chính.
* Dưới đây là một số ví dụ về cách áp dụng Continual Learning và Test Production trong xây dựng giải pháp học máy:
  + Trong ứng dụng nhận dạng hình ảnh, Continual Learning có thể được sử dụng để cập nhật mô hình nhận dạng với các mẫu hình ảnh mới, giúp mô hình nhận dạng tốt hơn các đối tượng mới hoặc các biến thể của các đối tượng đã biết.
  + Trong ứng dụng điều khiển tự động, Continual Learning có thể được sử dụng để cập nhật mô hình điều khiển với các dữ liệu về môi trường mới, giúp mô hình điều khiển tốt hơn trong các điều kiện môi trường thay đổi.
  + Trong ứng dụng thị trường tài chính, Continual Learning có thể được sử dụng để cập nhật mô hình dự đoán giá cổ phiếu với các dữ liệu thị trường mới, giúp mô hình dự đoán tốt hơn giá cổ phiếu trong tương lai.
* Nhìn chung, Continual Learning và Test Production là hai kỹ thuật quan trọng cần được xem xét khi xây dựng giải pháp học máy. Việc áp dụng hai kỹ thuật này giúp đảm bảo rằng giải pháp học máy có thể học hỏi và cải thiện hiệu suất của chúng theo thời gian, cũng như đáp ứng các yêu cầu kinh doanh.

**REFERENCES**

1. **Vietnamese:**
2. *Tr&#7921;c, T. T. (2023). Optimizer- Hiểu sâu về các thuật toán tối ưu ( GD,SGD,Adam,..). <https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8>*
3. **English:**
4. *Continual Machine Learning. (2023). <https://ideas-ncbr.pl/en/research/continual-machine-learning/>*
5. *Yochze. (2019). How to apply continual learning to your machine learning models. [https://towardsdatascience.com/](https://towardsdatascience.com/how-to-apply-continual-learning-to-your-machine-learning-models-4754adcd7f7f)*
6. *How continuous-learning AI sharpens predictions. (n.d.). <https://www.algolia.com/blog/ai/how-continuous-learning-lets-machine-learning-provide-increasingly-accurate-predictions-and-recommendations/>*
7. *Chapter 9 - Continual learning and test in production. (n.d.). [https://github.com/serodriguez68/designing-ml-systems-summary/](https://github.com/serodriguez68/designing-ml-systems-summary/blob/main/09-continual-learning-and-test-in-production.md)*
8. *Continual Learning in Machine Learning. (2023). <https://www.geeksforgeeks.org/continual-learning-in-machine-learning/>*
9. *Mohandas, G. (n.d.). Testing Machine Learning Systems: Code, Data and Models. <https://madewithml.com/courses/mlops/testing/>*
10. *Amananandrai. (2023). 10 famous Machine Learning Optimizers. [https://dev.to/amananandrai/](https://dev.to/amananandrai/10-famous-machine-learning-optimizers-1e22)*
11. *Sagar, R. (2021). Top Optimisation Methods In Machine Learning. <https://analyticsindiamag.com/optimisation-machine-learning-methods-gradient-descent/>*