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

* Optimizer trong huấn luyện mô hình học máy:
  + Optimizer là các thuật toán, phương pháp giúp mô hình cập nhật các tham số của nó dựa trên kết quả huấn luyện để cải thiện hiệu quả, độ chính xác của mô hình. Các Optimizer sẽ hoạt động dựa trên loss function của bài toán, cố gắng cập nhất các trọng số, bias để làm tối thiểu loss function.
  + Việc lựa chọn Optimizer sẽ phụ thuộc vào loss function hoặc activation function trong mạng neural network và cũng sẽ ảnh hưởng rất nhiều đến mô hình.
* Các phương pháp optimizer: một số Optimizer phổ biến như Gradient Descent, AdaGrad, RMSProp, Adam,…
  + Gradient Descent:
    - Gradient Descent là một optimizer phổ biến và đơn giản, hoạt động bằng cách tối thiểu loss function L của bài toán để tìm giá trị tốt nhất cho tham số trong bài. Vì mục tiêu là tối thiểu loss function L nên thông qua đạo hàm của L theo tham số wi bất kỳ ta có thể biết được hướng biến thiên của loss function theo wi, khi đó ta sẽ thay đổi giá trị wi một khoảng bằng , dấu trừ sẽ đảo chiều của đạo hàm để di chuyển dần đến điểm cực tiểu của hàm, learning rate γ sẽ là một số rất nhỏ, giá trị này sẽ quyết định độ lớn trong giá trị của mỗi lần cập nhật tham số. Thuật toán sẽ dừng lại khi tham số không thể cải thiện được nữa hoặc khi hết số vòng lặp.
    - Công thức
      * wi: tham số cần cập nhật
      * γ: learning rate
      * : đạo hàm của loss function (L) theo wi
    - Các bước thực hiện:
      * Khởi tạo giá trị ban đầu bất kỳ cho các tham số w
      * Tính đạo hàm của Loss Function theo từng w*­*i
      * Cập nhật từng wi theo công thức
  + Adaptive Gradient Algorithm (AdaGrad):
    - Là thuật toán mở rộng từ Gradient Descent, thay vì cố định một learning rate xuyên suốt cho tất cả các tham số thì ta sẽ có từng learning rate cho từng tham số và sẽ thay đổi xuyên suốt thuật toán. Learning rate sẽ được tính bằng cách chia một tham số step\_size cố định cho tổng bình phương của các gradient theo wi của cost function trong từng vòng lặp.
    - Công thức:
      * wi: tham số cần cập nhật
      * N: số thuộc tính của bộ dữ liệu
      * S­: tổng bình phương gradient theo từng w của L
      * : đạo hàm của loss function (L) theo wi
      * γi: learning rate của wi
      * step\_size: một giá trị cố định để tính learning rate
      * 1e-8: một số cực kỳ nhỏ để tránh trường hợp S = 0
    - Các bước thực hiện
      * Khởi tạo giá trị ban đầu bất kỳ cho các tham số wi
      * Tính gradient của L theo tất cả w
      * Tính tổng bình phương tất cả gradient S
      * Tính learning rate γi
      * Cập nhật từng wi theo công thức
  + Root Mean Squared Propagation (RMSProp):
    - RMSProp là phiên bản mở rộng từ AdaGrad nhằm khắc phục tốc độ học chậm của nó. Một tham số mới rho được gọi là decay rate sẽ được thêm vào để tính một giá trị momentum S mới để cập nhật learning rate. Momentum sẽ khiến cho trọng số của các gradient cũ giảm dần đi, giúp thuật toán điều chỉnh chủ yếu dựa trên các gradient gần đây.
    - Công thức:
      * wi: tham số cần cập nhật
      * rho: decay rate
      * Si: momentum
      * : đạo hàm của loss function (L) theo wi
      * γi: learning rate của wi
      * step\_size: một giá trị cố định để tính learning rate
      * 1e-8: một số cực kỳ nhỏ để tránh trường hợp Si = 0
    - Cách thực hiện:
      * Khởi tạo giá trị ban đầu bất kỳ cho các tham số wi
      * Tính gradient của L theo wi
      * Tính Si
      * Cập nhật learning rate γi
      * Cập nhật từng wi theo công thức
  + Adaptive Moment Estimation (Adam):
    - Là thuật toán tiếp tục được mở rộng từ AdaGrad và RMSProp, ta sẽ có 2 momentum m, v để điều chỉnh các learning rate thay vì 1 như 2 thuật toán trên. Momentum thứ 2 sẽ giúp normalize step\_size, khi step\_size quá lớn v sẽ giảm giá trị của nó và ngược lại, khi step\_size nhỏ thì v sẽ tăng giá trị của nó lên.
    - Công thức:
      * wi: tham số cần cập nhật
      * m: momentum thứ nhất
      * β1: decay rate của m
      * : đã được điều chỉnh bias
      * v: momentum thứ hai
      * β2: decay rate của v
      * : đã được điều chỉnh bias
      * : đạo hàm của loss function (L) theo wi
      * γi: learning rate của wi
      * step\_size: một giá trị cố định để tính learning rate
      * 1e-8: một số cực kỳ nhỏ để tránh trường hợp chia cho 0
    - Cách thực hiện:
      * Khởi tạo giá trị ban đầu bất kỳ cho các tham số wi
      * Tính gradient của L theo wi
      * Tính m và v
      * Điều chỉnh m và v thành và
      * Cập nhật learning rate γi
      * Cập nhật từng wi theo công thức
* So sánh

|  |  |  |
| --- | --- | --- |
| **Thuật toán** | **Ưu điểm** | **Nhược Điểm** |
| Gradient Descent | * Đơn giản, dễ thực hiện * Có thể áp dụng cho các hàm nào có thể đạo hàm * Tính toán hiệu quả với các bài toán kích thước nhỏ. * Phổ biến | * Có thể bị kẹt ở các điểm local minimal. * Dễ bị ảnh hưởng bởi nhiễu. * Bị ảnh hưởng bởi việc chọn learning rate. * Tốc độ chậm. |
| AdaGrad | * Ít ảnh hưởng bởi giá trị khởi tạo. * Dễ tìm được điểm cực tiểu. * Tự động điều chỉnh learning rate. * Tính toán hiệu quả hơn Gradient Descent nhờ sử dụng các learning rate khác nhau. | * Learning rate có thể giảm xuống giá trị rất nhỏ, dẫn tới dừng thuật toán sớm. * Learning rate tự điều chỉnh có thể giúp thoát khỏi local minima. |
| RMSProp | * Ổn định * Hội tụ nhanh * Khắc phục nhược điểm trong learning rate của AdaGrad * Khả năng mắc kẹt ở local minima được giảm thiểu. | * Dễ bị ảnh hưởng bởi các tham số. * Cần phải điều chỉnh một số tham số. |
| Adam | * Có các ưu điểm của AdaGrad và RMSProp * Tự động normalize learning rate, giúp tính toán hiệu quả hơn. * Kết hợp từ AdaGrad và RMSProp giúp thuật toán có khả năng thoát khỏi local minima. | * Dễ bị ảnh hưởng bởi các tham số. * Có khả năng dẫn tới overfitting. * Hội tụ kết quả tệ trong một số trường hợp. |

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.
   * Continual Learning
     + Đối với các mô hình học máy thông thường, các mô hình được huấn luyện với các bộ dữ liệu cố định để có thể thực hiện một nhiệm vụ nào đó trên các dự liệu mới. Continual Learning còn được gọi là Lifelong learning là ý tưởng xây dựng mô hình liên tục học hỏi về thế giới bên ngoài, liên tục cập nhật để có thể tự động phát triển các kiến thức, kỹ năng phức tạp, có thể thực hiện được nhiều nhiệm vụ mà không quên đi những kiến thức cũ. Một số ứng dụng của Continual learning như giúp xe tự lái nhận biết và thay đổi theo các yếu tố của môi trường, các hệ thống cá nhân hóa có thể thay đổi theo kế hoạch, phương pháp sinh hoạt, học tập, làm việc của từng người dùng,…
     + Một trong những vấn đề lớn nhất của Continual Learning là Catastrophic Forgetting, khi học một kiến thức, nhiệm vụ mới thì mô hình phải thay đổi đề phù hợp nhiệm vụ đề ra và có thể không phù hợp với nhiệm vụ cũ, giảm mức độ hiệu quả khi thực hiện các nhiệm vụ cũ. Nếu vượt qua được vấn đề này, các mô hình có thể giữ được các kiến thức cũ trong khi vẫn có khả năng học được các kiến thức mới, khi đó các kiến thức cũ và mới có khả năng bổ trợ lẫn nhau để thực hiện các nhiệm vụ.
     + Để khắc phục Catastrophic Forgetting các giải pháp Normalization được sử dụng, các phương pháp này giúp mô hình cân bằng giữa các kiến thức cũ và mới. Các phương pháp như Elastic Weight Consolidation sẽ giữ lại trong số quan trọng đối với các nhiệm vụ cũ để giữ khả năng thực hiện chúng, hoặc phương pháp Synaptic Intelligence sẽ bắt chước hoạt động của các synapse trong não, mỗi synapse sẽ lưu các thông tin liên quan đến một nhiệm vụ, từ đó mô hình có thể học các kiến thức mới và không quên đi kiến thức cũ.
   * Test Production: là một bước thiết yếu để đảm bảo hiệu quả của mô hình, bao gồm kiểm tra dữ liệu, kiểm tra mô hình và dự đoán. Quá trình cụ thể bao gồm:
     + Kiểm tra dữ liệu đầu vào, đảm bảo chất lượng và format của dữ liệu train giống với format của dữ liệu thực tế.
     + Kiểm tra các thuộc tính, để đảm bảo dữ liệu train và dự liệu thực tế có cùng thuộc tính.
     + Kiểm tra để đảm bảo các trọng số và output của model là các giá trị hợp lệ.
     + Kiểm tra hiệu quả của model trong quá trình train.
     + Kiểm tra hiệu quả của model với dữ liệu thật.
   * Khi xây dựng một giải pháp học máy, Continual learning sẽ giúpmô hình liên tục cập nhật dữ liệu, cải thiện liên tục theo thời gian, test production sẽ giám sát quá trình cập nhật và đảm bảo không có các lỗi phát sinh dẫn đến kết quả không mong muốn. Hai kỹ thuật kết hợp với nhau tạo ra một mô hình ổn định, cập nhật liên tục theo thời gian.