TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN CUỐI KÌ MÔN NHẬP MÔN HỌC MÁY

Người hướng dẫn: **PGS. TS. LÊ ANH CƯỜNG**

Người thực hiện: **NGUYỄN GIA MỸ – 521H0272**

VŨ BẢO AN – 521H0435

Lớp **: 21H50302 – 21H50203**

Khoá **: 25**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN CUỐI KÌ MÔN NHẬP MÔN HỌC MÁY

Người hướng dẫn: **PGS. TS. LÊ ANH CƯỜNG**
Người thực hiện: **NGUYỄN GIA MỸ**
VŨ BẢO AN
Lớp : **21H50302 – 21H50203**
Khoá : **25**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn sâu sắc đến thầy Lê Anh Cường vì những bài học của thầy vô cùng có giá trị để chúng em có thể hoàn thiện bài báo cáo này.

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Chúng tôi xin cam đoan đây là sản phẩm đồ án của riêng chúng tôi và được sự hướng dẫn của PGS. TS. Lê Anh Cường;. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày tháng năm

Tác giả

(ký tên và ghi rõ họ tên)

Nguyễn Gia Mỹ

Vũ Bảo An

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

TÓM TẮT

MỤC LỤC

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

CÁC KÝ HIỆU

CÁC CHỮ VIẾT TẮT

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH

DANH MỤC BẢNG

CHƯƠNG 1 – NGUYỄN GIA MỸ

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

1.1.1 Gradient Descent

Gradient descent là một trong những thuật toán tối ưu phổ biến trong huấn luyện mô hình học máy đơn giản và thuật toán này cũng có thể sử dụng trong nhiều mô hình học máy khác nhau.

Xét một ví dụ về Gradient Descent trong bài toán Linear Regression với 1 biến. Ta có mô hình như sau (1.1):

$$f_{w,b}(x) = w.x + b \quad (1.1)$$

Trong đó:
- w, b có thể được gọi là parameters/ variables/ coefficients/ weights, là những giá trị có thể thay đổi trong quá trình huấn luyện để cải thiện độ chính xác của một mô hình học máy.
- x gọi là feature, là biến đầu vào.

Như vậy, việc chọn giá trị w và b càng hợp lý thì hiệu suất của mô hình học máy sẽ càng cao vì giá trị của w và b được chọn sẽ tác động đến giá trị của $f_{w,b}$ dựa trên biến x đầu vào.

Vì w và b là số nên có rất nhiều giá trị được chọn cho w và b, để đánh giá mô hình Linear Regression có hiệu quả hay không, mọi người sử dụng hàm mất mát (cost function). Có rất nhiều công thức để tính toán độ hiểu quả của mô hình. Trong ví dụ này đề cập đến công thức của một cost function gọi là Mean Square Error (1.2):

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(yhat^{(i)} - y^{(i)})^2 \quad (1.2)$$

Trong đó:
- m: tổng số training examples.
- yhat: prediction/ giá trị đầu ra được dự đoán.
- y: target, giá trị đầu ra thực tế tương ứng với đầu vào x.

Tóm lại, hàm cost function này được dùng để kiểm tra xem độ chính xác của kết quả dự đoán bởi mô hình học máy so với kết quả thực tế. Từ đó ta có thể biết được mô

hình Linear Regression có hoạt động tốt hay không, hay giá trị w và b mà ta chọn có khiến kết quả dự đoán sát với thực tế hay không.

Như đã đề cập ở trên, vì w và b là những con số nên ta có thể chọn vô vàng cặp số (w, b) để đưa vào mô hình. Điều đó khiến cho việc chọn ra cặp số (w, b) phù hợp nhất để cho ra mô hình có thể dự đoán kết quả thật nhất là một việc khá khó, vì ta phải liên tục chọn các cặp số, tính toán cost function, sau đó chọn ra cặp số tối ưu nhất.

Thuật toán Gradient Descent ra đời là một giải pháp để ta có thể lựa chọn giá trị w và b tối ưu nhất một cách tự động mà không cần phải lặp lại các bước: chọn các cặp số (w, b), tính toán độ mất mát (bởi cost function), chọn cặp số (w, b) mà độ mất mát thấp nhất.

Tóm lại, Gradient Descent là một thuật toán tối ưu hóa dùng để tìm các trị w và b sao cho giá trị của cost function là nhỏ nhất. Dưới đây là công thức của Gradient Descent:

$$(1): w = w - \alpha \frac{d}{dw} J(w, b)$$

$$(2): b = b - \alpha \frac{d}{db} J(w, b)$$

Trong đó:

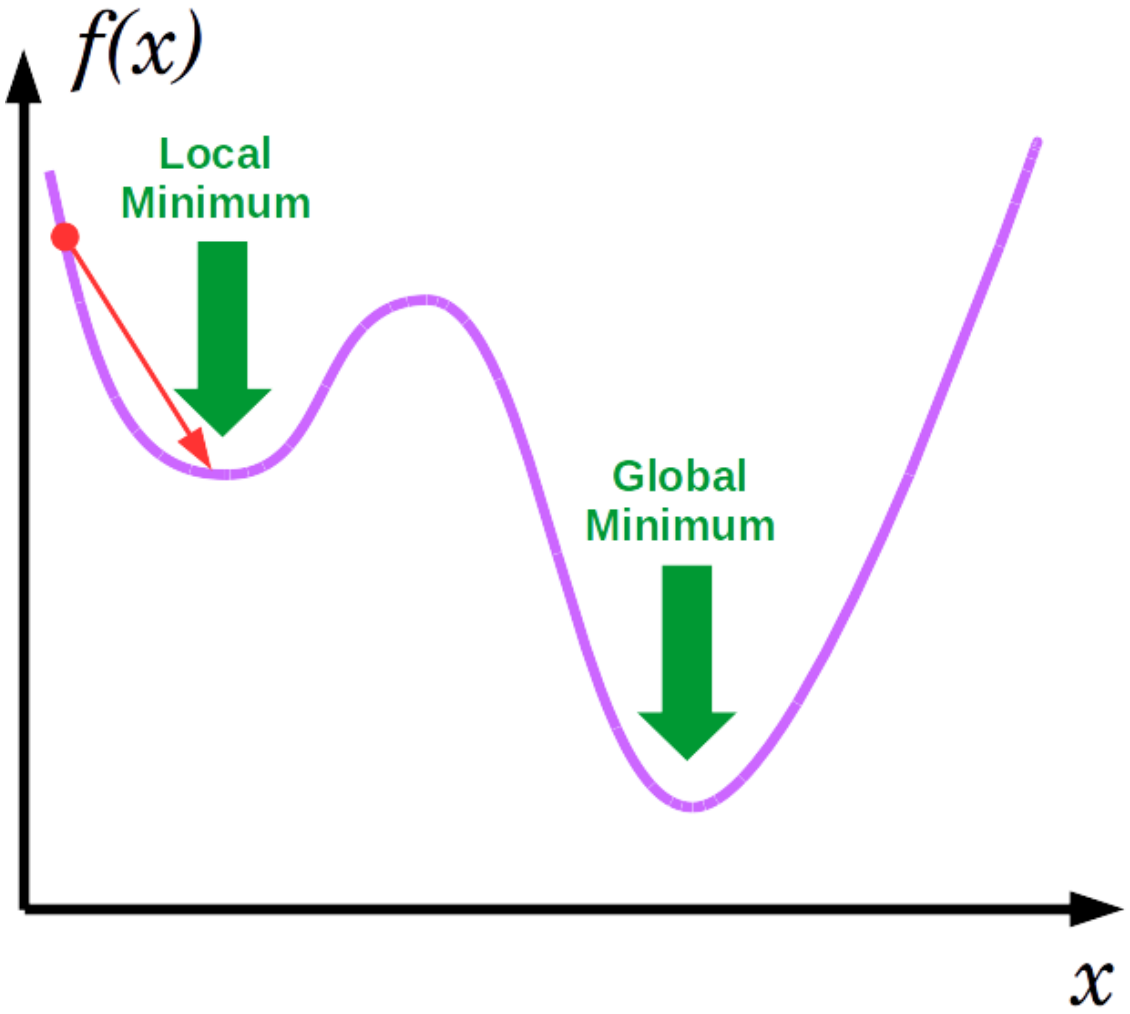
- α : learning rate, có giá trị trong khoảng [0, 1]. Dùng để điều chỉnh bước nhảy lớn hay nhỏ của đạo hàm $J(w, b)$.

Ta tuần tự thực hiện (1) và (2) cho đến khi $J(w, b)$ gần chạm đến cực tiểu (local minumum). Càng gần điểm cực tiểu, đạo hàm và bước nhảy có giá trị nhỏ hơn, tức hàm cost function gần đạt giá trị nhỏ nhất.

Trong huấn luyện mô hình mô hình học máy, Gradient Descent có thể được xem là một trong những thuật toán tối ưu dễ hiểu và dễ sử dụng nhất. Một số hạn chế của thuật toán Gradient Descent như:

- Hiệu suất của thuật toán này phụ thuộc rất lớn vào learning rate. Nếu như chọn learning rate quá nhỏ, đồng nghĩa với bước nhảy nhỏ và sẽ chạy khá lâu để có thể đạt đến/ gần đến cực tiểu. Nếu chọn learning rate quá lớn thì có thể sẽ nhảy qua khỏi điểm cực tiểu.

- Thuật toán có thể bị "mắc kẹt" ở điểm cực tiểu cục bộ vì đó là điểm nhỏ nhất mà nó đang tìm được.



Hình 1.1 Gradient Descent Local Minima

- Thuật toán Gradient Descent được đề cập cho đến đây được gọi là Batch Gradient Descent, nghĩa là mỗi lần lặp của thuật toán sử dụng hết các tập dữ liệu huấn luyện (traning example). Điều này dẫn đến một hạn chế tiếp theo của thuật toán này là khi tập dữ liệu huấn luyện quá lớn thì việc tính toán đạo hàm trên toàn bộ dữ liệu tại mỗi vòng lặp tốn rất nhiều thời gian.

1.1.2 Stochastic Gradient Descent

Stochastic Gradient Descent là một biến thể khác của Gradient Descent, ở mỗi vòng lặp, thay vì sử dụng hết tất cả các tập dữ liệu huấn luyện để tính toán thì Stochastic Gradient Descent ở mỗi lần lặp chỉ tính toán duy nhất một điểm dữ liệu huấn luyện. Mỗi lần duyệt qua tất cả các điểm dữ liệu được gọi là một epoch, tương ứng với một lần lặp của Batch Gradient Descent.

Stochastic Gradient Descent hoạt động hiệu quả trong mô hình học máy có rất nhiều điểm dữ liệu vì thuật toán này chỉ tính toán một điểm dữ liệu tại mỗi vòng lặp. Tuy nhiên việc tính toán này cũng mang lại hạn chế do việc chọn ngẫu nhiên một dữ liệu dễ gây hiện tượng "jitter" trong quá trình huấn luyện, đây là hiện tượng dao động giá trị của cost function tại mỗi vòng lặp. Thuật toán này cũng dễ bị "mắc kẹt" trong điểm cực tiểu cục bộ như Batch Gradient Descent.

1.1.3 Minibatch Stochastic Gradient Descent

Minibatch Stochastic Gradient Descent cũng là một biến thể khác của Gradient Descent, tuy nhiên thuật toán này không sử dụng hết tập dữ liệu huấn luyện để tính toán ở mỗi vòng lặp, cũng không sử dụng một điểm dữ liệu để tính toán, mà sử dụng một số điểm dữ liệu (k: batch size) để tính toán.

Là qui mô rất nhỏ của Batch Gradient Descent kết hợp với Stochastic Gradient Descent, Minibatch Stochastic Gradient Descent không mắc phải hiện tượng "jitter" trong quá trình huấn luyện giống như Stochastic Gradient Descent do thuật toán này sử dụng một lượng rất nhỏ dữ liệu dể tính toán. Tuy nhiên thuật toán này cũng dễ bị "mắc kẹt" trong điểm cực tiểu cục bộ như Batch Gradient Descent.

1.1.4 Adaptive Gradient Algorithm

Như đã đề cập ở trên, thuật toán Gradient descent bị phụ thuộc rất lớn vào learning rate. Tức là khi learning rate ta chọn quá nhỏ, không chỉ mất nhiều thời gian mà còn dễ bị "mắc kẹt" vào điểm cực tiểu cục bộ. Khi ta chọn learning rate quá lớn sẽ gây nên hiện tượng overshoot do bước nhảy nhanh.

Adagrad (Adaptive Gradient Algorithm) là thuật toán tối ưu hóa được dùng để tự động điều chỉnh learning rate dựa trên tổng độ dốc của tham số ở các lần lặp trước. Tại mỗi vòng lặp, ta tính dộ dốc của cost function theo mỗi tham số. Sau đó tính tổng bình phương giá trị của từng độ dốc của mỗi lần lặp và trước đó. Tiếp đến ta cập nhật learning rate: lấy căn bậc hai của tổng bình phương của từng độ dốc, kết quả được gọi là "trung bình độ dốc", lấy learning hiện tại chia cho "trung bình độ dốc". Cuối cùng ta cập nhật tham số giống như cách làm Gradient Descent thông thường.

Adagrad có thể tránh bị mắc kẹt trong một điểm cực tiểu cục bộ, nhưng đối với cost function có nhiều điểm cực tiểu thì thuật toán cũng có thể không thể tìm thấy điểm cực tiểu toàn cục. Thuật toán này cũng có thể bị hiện tượng "jitter" do sử dụng tổng bình phương của các độ dốc gần đây để điều chỉnh learning rate vì có sự dao động lớn giữa các bình phương.

1.1.5 Adaptive Moment Estimation (Adam)

Gradient Descent là một trong những thuật toán được sử dụng rộng rãi trong học máy và là cơ sở của nhiều thuật toán như Linear Regression, Logistic Regression, và ngay cả các phiên bản đầu tiên của Neural Network. Tuy nhiên hiện nay đã có những thuật toán tối ưu khác hiệu quả hơn Gradient Descent trong việc giảm giá trị của cost function. Thuật toán Adam nổi bật ở chỗ nếu nó phát hiện learning rate quá nhỏ và ta chỉ đi theo một hướng liên tục thì ta có thể tăng giá trị của learning rate. Ngược lại, nếu một tham số đang dao động liên tục thì ta có thể giảm learning rate để tham số không tiếp tục dao động.

Ưu điểm lớn nhất của thuật toán Adam là tự động điều chỉnh learning rate, điều này giúp cho huấn luyện mô hình trở nên hiệu quả hơn. Tuy nhiên việc lựa chọn learning rate ban đầu cũng rất quan trọng. Một nhước điểm của thuật toán này là tốn kém về bộ nhớ vì thuật toán này cần cần lưu trữ và cập nhật trạng thái cho mỗi tham số, điều này có thể đòi hỏi bộ nhớ lớn, đặc biệt khi mô hình có nhiều tham số.

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

Ngày nay ta có thể thấy được học máy được ứng dụng vào hầu hết tất cả các lĩnh vực, sản phẩm và dịch vụ, được người tiêu dùng sử dụng hằng ngày.

Một số ứng dụng học máy nổi tiếng có thể kể đến như:

- Trợ lý ảo Siri (Apple) hay Alexa (Google). Ta chỉ cần gọi "Hey Siri" và thực hiện các yêu cầu đối với các trợ lý ảo này, ví dụ như nói "Gọi cho A", Siri sẽ lập tức thực hiện cuộc gọi mà ta không cần phải thực hiện nhập số điện thoại hay vào danh bạ và nhấn gọi.
- Trong lĩnh vực giao thông có dự đoán giá tiền trong quá trình di chuyển. Thấy rõ nhất ở các ứng dụng đặt xe, đồ ăn công nghệ như Grab, Uber, v.v. Khi đặt xe, ứng dụng sẽ tự ước tính giá chuyến đi dựa trên số kilomet cũng như một số yếu tố khác. Có thể thấy được các ứng dụng đặt xe công nghệ đã chiếm phần lớn thị phần dịch vụ vận chuyển hành khách bởi giá cả rất hợp lý, thậm chí là rất rẻ. Một phần chi phí chi trả của khách hàng cho một chuyến đi có thể hợp lý như vậy là do ứng dụng đã hướng dẫn tài xế đi những đoạn đường tối ưu nhất để đến địa điểm một cách nhanh nhất. Và chúng sử dụng học máy để làm điều đó.

Như vậy, làm cách nào để một mô hình học máy có thể được đưa vào một sản phẩm trên thị trường biến đổi liên tục? Ta đều biết rằng học máy cần có tập dữ liệu huấn luyện lớn để đưa ra được các dự đoán sát với thực tế nhất. Lấy một ví dụ, vào ngày hôm

nay ứng dụng GrabClone được đưa ra thị trường, ứng dụng này là một ứng dụng công nghệ có sử dụng mô hình học máy để tính toán các đoạn đường ngắn nhất để tài xế có thể đưa khách hàng địa điểm khách hàng yêu cầu. Sử dụng mô hình này giúp cho chi phí mà một người khách hàng phải trả cho một chuyến đi giảm xuống, nhằm cạnh tranh với các ứng dụng khác và giúp tài xế có thể có thêm nhiều chuyến xe hơn trong ngày. Chị A có nhu cầu từ trường Đại học Tôn Đức Thắng đến Nhà thờ Đức Bà, ứng dụng đã hướng dẫn tài xế đi các đoạn đường X, Y, Z, H, K để có thể đến được Nhà thờ Đức Bà nhanh nhất. Một tuần sau, đoạn đường Y thực hiện bảo trì và không cho phép các phương tiện lưu thông. Như vậy, dữ liệu lúc này đã thay đổi, tuy nhiên trong dữ liệu huấn luyện lúc ban đầu thì Y là đoạn đường được cho phép lưu thông. Làm cách nào để có thể cho trong mô hình học máy rằng Y bây giờ đã không hợp lệ? Continual Learning đã ra đời từ đó.

Continual learning (Học liên tục) là việc cập nhật đều đặn các mô hình trong sản xuất học máy, cho phép mô hình không chỉ học từ dữ liệu mới mà còn phải giữ được kiến thức trước đó. Thách thức chính của Continual learning là đảm bảo rằng mô hình được cập nhật mà không làm hỏng hoặc làm giảm hiệu suất mặc dù mô hình vẫn đang trong môi trường sản xuất. [1] Đây có lẽ là điều khác biệt nhất giữa mô hình học máy trong nghiên cứu và trong sản phẩm ứng dụng học máy.

Để thấy được tầm quan trọng của Continual learning, hãy xét thêm một ví dụ. Sàn thương mại điện tử Shopee thực hiện đợt giảm giá lớn nhất trong năm vào ngày 12.12. Trong sự kiện như thế này hành vi tiêu dùng của người dùng sẽ có sự thay đổi mẽ và liên tục. Như vậy nếu mô hình học máy có thể đề xuất một cách hiệu quả, chẳng hạn như gợi ý các sản phẩm đang giảm giá nhiều nhất, các sản phẩm giảm giá nhiều nhất có liên quan đến đến sản phẩm có trong giỏ hàng của khách hàng, v.v thì chắc chắn doanh thu sẽ tăng lên rất lớn.

Trong môi trường doanh nghiệp, việc triển khai mô hình học máy học liên tục cũng là một thách thức lớn do dữ liệu từ người dùng cho các mô hình học máy trong doanh nghiệp đòi hỏi việc học liên tục, nghĩa là dữ liệu phải được truyền trực tiếp từ ứng dụng. Điều này khác với huấn luyện truyền thống, nơi dữ liệu được lưu trữ riêng biệt khỏi các mô hình học máy.

Về cơ bản, để tiếp cận học liên tục có 4 giai đoạn [2]:

- Giai đoạn 1: Huấn luyện lại thủ công, không trạng thái. Ở giai đoạn này chú trọng đến việc xây dựng mô hình học máy để giải quyết vấn đề cho doanh nghiệp. Do tập trung vào việc xây dựng mô hình mới nên việc cập nhật trở nên không quá quan trọng. Việc cập nhật chỉ diễn ra khi hiệu suất của mô hình đã giảm đến mức gây hại cho doanh nghiệp và đội ngũ nhân viên về học máy có thời gian để cập nhật nó. Một mô hình thường sẽ được cập nhất một quý một lần, một năm một lần hay một quý một lần. Đây là quá trình thủ công, một số người thực hiện thực hiện trích xuất dữ liệu, một số người thực hiện làm sạch dữ liệu và huấn luyện lại mô hình học máy dựa trên cả dữ liệu cũ và mới rồi xuất mô hình đã cập nhật sang dạng nhị phân. Một số nhân viên khác sẽ lấy mã nhị phân đó và triển khai mô hình.
- Giai đoạn 2: Huấn luyện tự động. Huấn luyện thủ công khá mất thời gian, ta có thể viết một kịch bản (script) để tự động hóa quá trình huấn luyện. Ở giai đoạn này ta cần viết các kịch bản tự động hóa, mô phỏng lại quy trình làm việc thủ công và cấu hình cơ sở hạ tầng để tự động lấy dữ liệu, làm sạch, gán nhãn cho dữ liệu trong quá trình huấn luyện lại.
- Giai đoạn 3: Huấn luyện tự động, có trạng thái. Huấn luyện tự đông có trạng thái là tiếp tục huấn luyện mô hình trên dữ liệu mới, không phải huấn luyện lại mô hình từ đầu.

- Giai đoạn 4: Học liên tục. Thay vì cập nhật mô hình dựa trên một lịch trình cố định, thì ta liên tục cập nhật mô hình mỗi khi phân phối dữ liệu thay đổi và hiệu suất mô hình giảm đáng kể.

Test in Production là triển khai mô hình vào sản phẩm thực tế, kiểm thử mô hình trong quy trình sản xuất và đánh giá hiệu suất mô hình dựa trên dữ liệu với dòng thời gian thực. Shadow Deployment là cách an toàn nhất để triển khai mô hình học máy do cách triển khai này rất an toàn. Đầu tiên ta triển khai một mô hình giống với mô hình hiện tại (tạm gọi là mô hình ứng viên), với mỗi yêu cầu thực thi mô hình để dự đoán (của người dùng), hai mô hình đều thực hiện dự đoán nhưng chỉ có dự đoán của mô hình hiện tại mới được gửi đến người dùng, dự đoán của mô hình ứng viên được lưu trữ và phục vụ cho việc phân tích. Như vậy cách triển khai này là cách triển khai an toàn, vì mục tiêu của nó là giữ cho trạng thái thực tế không bị ảnh hưởng bởi dự đoán của mô hình mới (mô hình ứng viên), cho đến khi mô hình ứng viên này được phân tích và xác định là tốt. [3]

CHƯƠNG 2 – VŨ BẢO AN

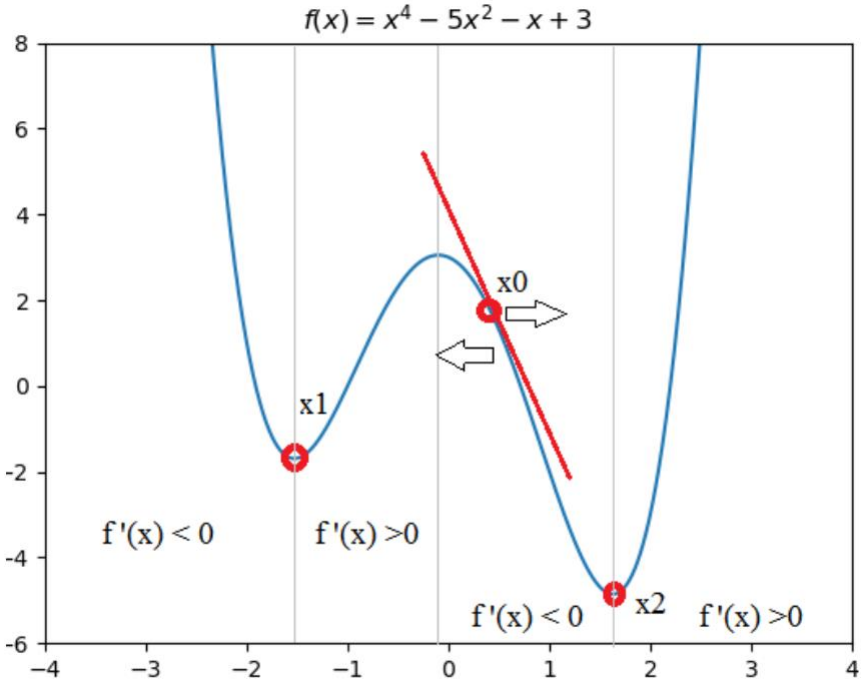
2.1 Tìm hiểu, so sánh các phương pháp Optimizer trong huấn luyện mô hình học máy

2.1.1 Batch Gradient Descent, Stochastic Gradient Descent, Minibatch Stochastic Gradient Descent

2.1.1.1 Batch Gradient Descent

Batch Gradient Descent (hay thường được gọi là Gradient Descent) là một thuật toán tối ưu phổ biến được sử dụng trong huấn luyện mô hình học máy. Thuật toán này có thể xem như là một bước nhảy nhỏ để tìm giá trị nhỏ nhất của một hàm số.

Cho một bài toán tối ưu hóa $f(x) = x^4 - 5x^2 - x + 3$. Tìm x sao cho hàm số này đạt giá trị tối thiểu. Đối với phương pháp truyền thống, ta có thể giải phương trình $f'(x) = 0$ để tìm tất cả các điểm cực tiểu rồi so sánh chúng để lấy điểm cực tiểu toàn cục như hình bên dưới:



Đối với mô hình tuyến tính Linear Regression, mô hình có thể như hàm số $f(x)$ trên. Thuật toán gradient descent là một thuật toán tối ưu hóa lặp lại để tìm giá trị nhỏ nhất của một hàm. Để tìm giá trị nhỏ nhất cục bộ của một hàm bằng cách sử dụng gradient descent, ta thực hiện các bước tỉ lệ với đối của độ dốc (hoặc độ dốc xấp xỉ) của hàm tại điểm hiện tại. Cụ thể hơn, thuật toán này sẽ lặp lại thay đổi giá trị của $\boldsymbol{x}$ ($\boldsymbol{x = x + \beta}$) sao cho trong mỗi lần lặp, ta muốn $\boldsymbol{f(x)}$ giảm đi và tiến gần đến giá trị nhỏ nhất. Một cách để đảm bảo rằng sự điều chỉnh $\boldsymbol{\beta}$ đối với x sẽ làm cho f(x) giảm là làm cho β bằng một phần của đối của độ dốc: $\boldsymbol{\beta = -\alpha f(x)}$ trong đó $\boldsymbol{\alpha}$ là một số dương và được gọi là tốc độ học.

2.1.1.2 Stochastic Gradient Descent

Stochastic Gradient Descent cũng là thuật toán tối ưu hóa phổ biến trong mô hình học máy. Thuật toán này đơn giản hơn Batch Gradient Descent, mặc dù có sai số nhưng lợi ích tính toán cao do tại một lần lặp, ta chỉ tính đạo hàm của hàm mất mát dựa trên một điểm dữ liệu ngẫu nhiên từ tập huấn luyện để cập nhật tham số của mô hình học máy.

2.1.1.3 Minibatch Stochastic Gradient Descent

Minibatch Stochastic Gradient Descent là sự kết hợp giữa Batch Gradient Descent và Stochastic Gradient Descent. Thay vì sử dụng một điểm dữ liệu như Stochastic Gradient Descent hay dùng hết tất cả dữ liệu như Gradient Descent trong một lần lặp, Minibatch Stochastic Gradient Descent sử dụng một lượng rất nhỏ các dữ liệu (so với tập dữ liệu) để cập nhật tham số cho mô hình học máy.

2.1.2 RMSprop (Root Mean Square Propagation)

RMSprop là một thuật toán tối ưu hóa lặp được sử dụng để huấn luyện các mô hình học máy, hoạt động bằng cách điều chỉnh tốc độ học (learning rate) của thuật toán tối ưu theo độ dốc của hàm mất mát. Thuật toán này được hoạt động như sau:

- Tính toán trung bình bình phương của các độ dốc của hàm mất mát gần đây.
- Nếu trung bình bình phương của các độ dốc gần đây nhỏ thì thuật toán sẽ giảm tốc độ học.

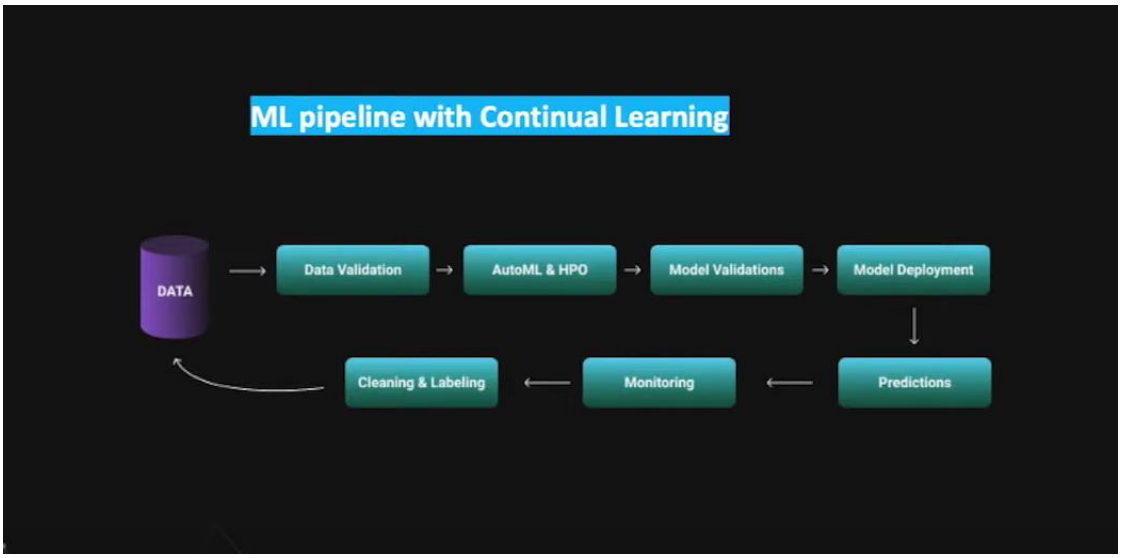
2.1.3 Ưu điểm và nhược điểm của các thuật toán

Thuật toán	Ưu điểm	Nhược điểm
Batch Gradient Descent	Dễ hiểu, dễ triển khai	Yêu cầu sử dụng nhiều bộ nhớ
		Không phù hợp với dữ liệu quá lơn vì độ phức tạp tính toán cao
Stochastic Gradient Descent	Hiệu suất tốt trên dữ liệu lớn do chỉ tính toán trên 1 điểm dữ liệu	Dao động nhiều do tính toán trên các số ngẫu nhiên
	Độ phức tạp tính toán không cao, không cần sử dụng nhiều bộ nhớ	
	Có thể tìm điểm cực tiểu cục bộ khác nhau	

Minibatch Stochastic Gradient Descent	Sự kết hợp của Batch Gradient Descent và Stochastic Gradient Descent	Vẫn cần sử dụng nhiều bộ nhớ nhiều hơn Stochastic Gradient Descent
	Hoạt động tốt trên tập dữ liệu lớn	Vẫn có sai số và dao động do kế thừa từ Stochastic Gradient Descent
RMSprop	Tự động điều chỉnh tốc độ học	Có thể bị dao động nhiều nếu kích thước được chọn quá nhỏ

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

Continual Learning (CL) cho phép mô hình học được nhiều tác vụ liên tiếp mà không quên đi kiến thức đã học được từ trước đó, trong đó dữ liệu của các tác vụ cũ không còn khả dụng trong quá trình huấn luyện các tác vụ mới. Về cơ bản đây là khả năng mô hình học liên tục từ luồng dữ liệu, cập nhật mô hình trong sản xuất để duy trì hiệu suất mà mức độ liên quan. Giả sử như một mô hình học máy như giao dịch sàn chứng khoán, khi có rất nhiều dữ liệu xuất hiện và ta muốn cập nhật mô hình của mình liên tục nhưng lại không muốn mô hình dừng hoạt động, ta có thể sử dụng continual learning nhằm đào tạo lại mô hình của mình bằng dữ liệu mới và sau đó triển khai phiên bản mới của mô hình.



Hình 2.2.1 Machine learning pipeline với Continual Learning

Bắt đầu với dữ liệu có thể là file csv, hình ảnh, hoặc bất cứ loại dữ liệu nào khác, sau đó xử lý trước dữ liệu để xác thực và xem thử có thể làm việc với dữ liệu. Tiếp theo, sử dụng AutoML và Hypochrometer optimization để huấn luyện mô hình và kiểm tra các loại mô hình khác nhau để nó có thể hoạt động tốt nhất trên dữ liệu của ta. Xác thực các mô hình so sánh với nhau, nhận được dự đoán vào mô hình sản xuất. Sau đó là quá trình theo dõi, rồi lấy dữ liệu mới từ dự đoán, gắn nhãn cho nó, sau đó đóng vòng lặp, bắt đầu lại và đào tạo mô hình.

Lý do cần Continual Learning: về cơ bản, mô hình luôn là điều quan trọng nhất trong dự liệu học máy, và trong khi dữ liệu đang thay đổi, mô hình cũng cần phải thay đổi nhanh chóng.

OpenAI đã áp dụng thành công việc học liên tục trong việc phát triển các mô hình như GPT-3, có thể học hỏi và thích ứng với các nhiệm vụ mới mà không quên kiến thức trước đó. Điều này đã cho phép tạo ra các hệ thống AI linh hoạt hơn có thể xử lý một loạt các nhiệm vụ và ứng dụng, chẳng hạn như hiểu ngôn ngữ tự nhiên, dịch, tóm tắt và trả lời câu hỏi. Một số kỹ thuật được sử dụng trong Continual Learning:

Bilevel continual learning: Phương pháp này kết hợp tối ưu hóa hai cấp độ với quản lý bộ nhớ kép để đạt được sự chuyển giao kiến thức hiệu quả giữa các nhiệm vụ và ngăn ngừa quên. [4]

Progressive neural networks: Cách tiếp cận này tận dụng cả dữ liệu được gắn nhãn và không được gắn nhãn để cải thiện tính khái quát của mô hình và giảm bớt tình trạng quên lãng thảm khốc. [4]

Incremental learning: Mô hình có thể được huấn luyện theo kiểu incremental, thêm vào đó một số lượng lớn các lớp hoặc mô-đun mới mà không ảnh hưởng đến kiến thức đã học từ các phần trước đó.

2.2.2 Test Production

Test production là quá trình kiểm tra một giải pháp học máy trong môi trường sản xuất. Điều này rất quan trọng để đảm bảo rằng giải pháp học máy hoạt động chính xác và đáp ứng các yêu cầu của người dùng.

Có nhiều cách khác nhau để thực hiện test production. Một cách tiếp cận phổ biến là sử dụng A/B testing. Trong A/B testing, hai phiên bản của giải pháp học máy được so sánh với nhau. Phiên bản nào hoạt động tốt hơn sẽ được triển khai trong môi trường sản xuất.

CHƯƠNG 3 – LINK GITHUB

https://github.com/my-nggia/machine-learning-final-project

TÀI LIỆU THAM KHẢO

Tiếng Việt

Tiếng Anh

[1]. Vincenzo Lomonaco. (2023). *Continual Learning for Production Systems*. Retrieved November 22 2023, from Medium website: Continual Learning for Production Systems | by Vincenzo Lomonaco | ContinualAI | Medium

[2]. Chip Huyen. (2023). *Real-time machine learning: challenges and solutions*. Retrieved November 22 2023, from huyenchip website: Real-time machine learning: challenges and solutions (huyenchip.com)

[3]. ML Collective. (2022). *Continual Learning and Test in Production - "Designing ML Systems" Reading Group*. Retrieved November 22 2023, from Youtube website: Continual Learning and Test in Production - "Designing ML Systems" Reading Group

[4] Continual Learning – Activeloop

Continual Learning | Continual Learning Definition, Continual Learning Use in ML (activeloop.ai)

PHỤ LỤC