TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CUỐI KÌ MÔN**

**NHẬP MÔN HỌC MÁY**

*Người hướng dẫn*: **Thầy Lê Anh Cường**

*Người thực hiện*: **TRẦN HỮU QUANG TRƯỜNG – 52100941**

Lớp **: 21050301**

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**



**BÁO CÁO CUỐI KÌ MÔN**

**NHẬP MÔN HỌC MÁY**

Người hướng dẫn: **Thầy Lê Anh Cường**

Người thực hiện: **TRẦN HỮU QUANG TRƯỜNG – 52100941**

Lớp **: 21050301**

Khoá  **: 25**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến Trường Đại học Tôn Đức Thắng đã đưa môn học Nhập môn học máy vào chương trình giảng dạy. Xin cảm ơn khoa Công nghệ thông tin đã cung cấp những tài liệu học tập đầy đủ, chi tiết.

Xin gửi lời cảm ơn chân thành đến giảng viên lí thuyết và thực hành bộ môn – thầy Lê Anh Cường, người đã tận tình giảng dạy em trong suốt quá trình tham gia môn học. Thầy đã tâm huyết truyền đạt những kiến thức khó nhằn một cách dễ hiểu và dễ tiếp thu. Em rất biết ơn những bài tập thầy giao đã giúp em củng cố kiến thức, những buổi chữa bài tập rất hữu ích đã giải đáp mọi thắc mắc của em. Những buổi học thực hành đã giúp em áp dụng được những kiến thức lí thuyết vào những bài toán thực tế thú vị và đã cho em thấy được tính ứng dụng rộng lớn của môn Nhập môn học máy trong ngành học của em nói riêng và trong cuộc sống nói chung.

Em xin trân trọng cảm ơn!

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi / chúng tôi và được sự hướng dẫn của Thầy Trần Bảo Tín;. 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 1 tháng 12 năm 2023*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Trần Hữu Quang Trường*

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

[LỜI CẢM ƠN 1](#_Toc154261850)

[TÓM TẮT 4](#_Toc154261851)

[MỤC LỤC 5](#_Toc154261852)

[DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 6](#_Toc154261853)

[CHƯƠNG 1 – CÁC PHƯƠNG PHÁP OPTIMIZER TRONG HUẤN LUYỆN MÔ HÌNH HỌC MÁY 8](#_Toc154261854)

[1.1 Khái quát về thuật toán tối ưu (Optimizer) 8](#_Toc154261855)

[1.1.1 Khái niệm 8](#_Toc154261856)

[1.1.2 Mục đích 8](#_Toc154261857)

[1.2 Các phương pháp Optimizer 9](#_Toc154261858)

[1.2.1 Gradient Descent 9](#_Toc154261859)

[1.2.2 Stochastic Gradient Descent (SGD) 14](#_Toc154261860)

[1.2.3 Momentum 17](#_Toc154261861)

[1.2.4 Adagrad 20](#_Toc154261862)

[1.2.5 RMSprop 23](#_Toc154261863)

[1.3 So sánh các phương pháp Optimizer 26](#_Toc154261864)

[CHƯƠNG 2 – CONTINUAL LEARNING VÀ TEST PRODUCTION 27](#_Toc154261865)

[2.1 Continual Learning 27](#_Toc154261866)

[2.1.1 Khái niệm 28](#_Toc154261867)

[2.2 Test Production 33](#_Toc154261868)

[2.2.1 Khái niệm 33](#_Toc154261869)

[2.2.2 Mục đích 33](#_Toc154261870)

[2.2.3 Phân loại 33](#_Toc154261871)

[2.2.4 Cách thức hoạt động 33](#_Toc154261872)

[2.2.5 Ví dụ minh họa 34](#_Toc154261873)

[2.2.6 Ưu và nhược điểm: 36](#_Toc154261874)

[TÀI LIỆU THAM KHẢO 38](#_Toc154261875)

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

CHƯƠNG 1 – CÁC PHƯƠNG PHÁP OPTIMIZER TRONG HUẤN LUYỆN MÔ HÌNH HỌC MÁY

1.1 Khái quát về thuật toán tối ưu (Optimizer)

1.1.1 Khái niệm

- Optimizer trong học máy và học sâu là một thuật toán hoặc phương pháp được sử dụng để cập nhật và điều chỉnh các tham số của mô hình (như trọng số và bias) nhằm giảm thiểu hàm mất mát (loss function). Hàm mất mát đánh giá mức độ chính xác của mô hình so với dữ liệu thực tế.

1.1.2 Mục đích

* Tối ưu hóa hàm mất mát
  + Định nghĩa: Hàm mất mát đo lường mức độ "sai lệch" giữa kết quả dự đoán của mô hình và giá trị thực tế.
  + Mục tiêu: Optimizers tìm cách giảm thiểu giá trị của hàm mất mát. Khi hàm mất mát giảm, điều này chỉ ra rằng mô hình đang học tốt hơn và cải thiện khả năng dự đoán của mình.
* Cập nhật trọng số và Bias trong mạng Nơ-ron.
  + Quá trình: Optimizers điều chỉnh trọng số và bias dựa trên gradient (đạo hàm) của hàm mất mát.
  + Mục tiêu: Mục tiêu là tìm ra bộ trọng số và bias tối ưu sao cho hàm mất mát đạt giá trị thấp nhất.
* Điều chỉnh tốc độ học: Tốc độ học quyết định mức độ "nhanh chóng" hoặc "chậm rãi" mà mô hình điều chỉnh trọng số của mình.Tuy nhiên,tốc độ học quá nhanh có thể dẫn đến việc "bỏ qua" điểm tối ưu, quá chậm có thể khiến quá trình học trở nên chậm chạp và không hiệu quả.
* Phòng Tránh Vấn Đề Như Overfitting và Local Minima:
  + Overfitting: Khi mô hình quá "khớp" với dữ liệu huấn luyện, nó có thể không hiệu quả trên dữ liệu mới.
  + Local Minima: Đây là điểm mà tại đó hàm mất mát đạt giá trị thấp nhưng không phải là giá trị thấp nhất có thể (không phải là global minimum).
* Thích ứng với đặc trưng của dữ liệu: Các optimizers như Adam hoặc RMSprop có thể hiệu quả hơn trong việc điều chỉnh trọng số khi đối mặt với dữ liệu phức tạp và không đồng đều.
* Hỗ trợ Học Sâu: Việc tối ưu hóa trở nên cực kỳ quan trọng trong các mô hình học sâu với nhiều lớp ẩn, nơi việc điều chỉnh trọng số ảnh hưởng đến hiệu suất tổng thể của mô hình.
* Tối ưu hóa tài nguyên tính toán: Một số optimizers được thiết kế để giảm thiểu tài nguyên tính toán cần thiết, điều này quan trọng đối với việc xử lý dữ liệu lớn và mô hình phức tạp.

1.2 Các phương pháp Optimizer

1.2.1 Gradient Descent

a) Khái niệm

* Gradient Descent là một thuật toán tối ưu hóa để tìm điểm tối thiểu của một hàm mất mát bằng cách điều chỉnh liên tục các tham số của mô hình (như trọng số và bias) theo hướng ngược lại với gradient (đạo hàm) của hàm mất mát đó.
* Gradient chỉ ra hướng tăng của hàm mất mát. Khi làm việc với Gradient Descent, chúng ta di chuyển ngược lại với gradient để giảm thiểu hàm mất mát.

b) Cách hoạt động

* Khởi tạo tham số: Thuật toán bắt đầu bằng cách khởi tạo các tham số của mô hình (như trọng số và bias) một cách ngẫu nhiên hoặc theo một quy tắc nào đó.
* Tính Gradient của Hàm Mất Mát.
  + Tại mỗi bước của thuật toán, gradient (đạo hàm) của hàm mất mát được tính tại điểm tham số hiện tại.
  + Gradient cung cấp thông tin về hướng và độ dốc của hàm mất mát, cho biết hướng mà hàm mất mát tăng lên nhanh nhất.
* Cập nhật tham số:
  + Các tham số sau đó được cập nhật theo hướng ngược lại với gradient. Điều này được thực hiện theo công thức sau:

A close up of words

Description automatically generated

* + Việc cập nhật này nhằm mục đích di chuyển các tham số đến vị trí mà tại đó hàm mất mát có giá trị thấp hơn.
* Lặp lại: Quá trình này được lặp lại nhiều lần (các epoch) cho đến khi đạt được điều kiện dừng, như sự thay đổi của hàm mất mát dưới một ngưỡng nhất định hoặc sau một số lượng epoch xác định.

c) Ví dụ

- Giả sử chúng ta có hàm mất mát f(x) = x2, và mục tiêu là tìm giá trị của x sao cho f(x) đạt giá trị tối thiểu.

**B1:** Khởi tạo x = 4

**A diagram of a curve

Description automatically generated**

**B2:** Tính Gradient

* Gradient: Đạo hàm của f(x) = x2 là f’(x) = 2x.
* Tại x = 4: Gradient là f’(4) = 2 x 4 = 8

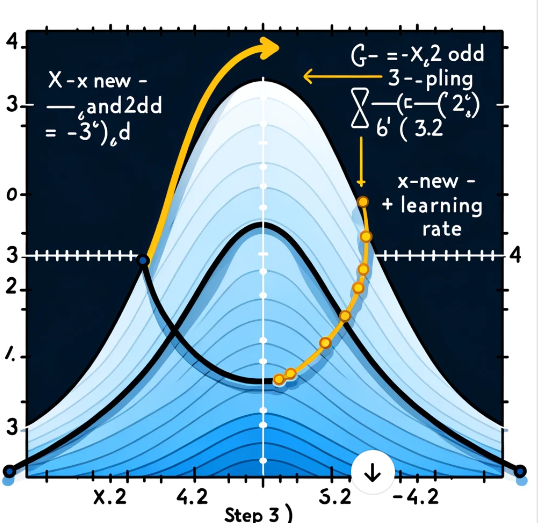
A graph of a function

Description automatically generated

Hình ảnh trên biểu diễn Bước 2 trong quy trình Gradient Descent. Trên đồ thị của hàm f(x) = x2, một mũi tên được vẽ từ điểm x = 4 và chỉ xuống dưới, đại diện cho gradient tại điểm này. Mũi tên chỉ ra hướng mà hàm mất mát giảm, minh họa việc tính toán gradient cho quá trình Gradient Descent.

**B3:** Cập nhật tham số:

* Learning Rate (α): Giả sử là 0.1.
* Cập nhật: xmới = xcũ – α × gradient.
* Tính toán: xmới = 4 – 0.1× 8 = 3.2



**B4:** Lặp lại

* Với mỗi lần lặp lại, điểm trên đồ thị sẽ tiếp tục di chuyển dần về phía giá trị tối thiểu của hàm mất mát. Điều này cho thấy cách Gradient Descent hoạt động để tìm ra bộ tham số tối ưu cho mô hình.

d) Ưu và nhược điểm:

* Ưu điểm:
  + Đơn Giản và Dễ Hiểu: Gradient Descent dễ hiểu và áp dụng, làm cho nó trở thành một công cụ mạnh mẽ, đặc biệt đối với những người mới bắt đầu trong lĩnh vực học máy.
  + Tính Linh Hoạt: Có thể áp dụng cho hầu hết các loại hàm mất mát, làm cho nó trở thành một lựa chọn linh hoạt trong nhiều ứng dụng khác nhau.
  + Hiệu Quả với Dữ Liệu Lớn: Đặc biệt hiệu quả khi làm việc với các bộ dữ liệu lớn, nhờ vào khả năng cập nhật trọng số một cách liên tục và dần dần.
* Nhược điểm:
  + Tốc Độ Hội Tụ: Có thể hội tụ chậm, đặc biệt là khi đối mặt với các hàm mất mát phức tạp hoặc không lồi.
  + Rủi Ro Mắc Kẹt tại Điểm Cực Tiểu Địa Phương: Đối với các hàm mất mát không lồi, Gradient Descent có thể mắc kẹt tại điểm cực tiểu địa phương thay vì tìm ra điểm cực tiểu toàn cục.
  + Chọn Tốc Độ Học (Learning Rate): Việc chọn tốc độ học không phù hợp có thể làm cho thuật toán không hội tụ hoặc hội tụ rất chậm.
  + Phụ Thuộc vào Điểm Khởi Đầu: Kết quả cuối cùng có thể phụ thuộc nhiều vào vị trí khởi đầu của thuật toán, đặc biệt trong các không gian có nhiều điểm cực trị.
  + Khó Khăn trong Việc Xử Lý Dữ Liệu Nhiều Chiều: Gradient Descent có thể trở nên kém hiệu quả khi làm việc với dữ liệu có số chiều rất lớn (vấn đề "nguyền rủa của chiều cao").

1.2.2 Stochastic Gradient Descent (SGD)

a) Khái niệm

* Stochastic Gradient Descent (SGD) là một phương pháp tối ưu hóa được sử dụng rộng rãi trong học máy và học sâu. Nó là một biến thể của Gradient Descent, phương pháp được thiết kế để tìm giá trị nhỏ nhất của một hàm mất mát, thường là trong quá trình huấn luyện một mô hình học máy.

b) Cách hoạt động

* Khởi Tạo Tham Số: Trước tiên, các tham số của mô hình (ví dụ, trọng số trong mạng nơ-ron) được khởi tạo, thường là một cách ngẫu nhiên.
* Chọn Mẫu Ngẫu Nhiên: Trong mỗi vòng lặp của quá trình huấn luyện, SGD chọn một điểm dữ liệu (hoặc một lô nhỏ các điểm dữ liệu) một cách ngẫu nhiên từ tập dữ liệu.
* Tính Gradient: Dựa trên điểm dữ liệu này, hàm mất mát (loss function) được tính toán. Sau đó, gradient của hàm mất mát theo các tham số mô hình được tính toán. Gradient này cung cấp thông tin về hướng và mức độ mà các tham số cần thay đổi để giảm giá trị của hàm mất mát.
* Cập Nhật Tham Số: Các tham số của mô hình sau đó được cập nhật bằng cách di chuyển chúng theo hướng ngược lại với gradient. Điều này làm giảm giá trị của hàm mất mát. Kích thước bước di chuyển trong không gian tham số được xác định bởi tốc độ học tập (learning rate).
* Lặp Lại: Quá trình này được lặp lại nhiều lần, mỗi lần với một mẫu dữ liệu hoặc lô dữ liệu ngẫu nhiên mới, cho đến khi hàm mất mát đạt đến một mức độ chấp nhận được hoặc quá trình huấn luyện đạt đến một số lượng vòng lặp đã xác định trước.

c) Ví dụ

**B1:** Khởi tạo tham số

* Giả sử mô hình hồi quy tuyến tính của chúng ta có dạng: y = wx + b, với y là cân nặng, x là chiều cao, w là trọng số, và b là độ chênh lệch.
* Khởi tạo w và b ngẫu nhiên, ví dụ w = 0.1, b = 0.0.

**B2:** Chọn mẫu ngẫu nhiên:

* Giả sử tập dữ liệu của chúng ta bao gồm nhiều cặp giá trị chiều cao và cân nặng.
* Chọn một cặp giá trị ngẫu nhiên, ví dụ x = 170 cm, y = 65 kg.

**B3:** Tính Gradient

* Hàm mất mát có thể là Mean Squared Error (MSE): 
* Tính toán dự đoán: 
* Tính gradient của L đối với w và b: A close up of words

  Description automatically generated

A math equation with numbers

Description automatically generated

**B4:** Cập nhật tham số

* Giả sử tốc độ học α = 0.01.
* Cập nhật w và b:

A group of black and blue letters

Description automatically generated

**B5:** Lặp lại

- Lặp lại quá trình từ Bước 2 đến Bước 4 với các mẫu ngẫu nhiên khác nhau cho đến khi hàm mất mát không còn giảm nữa hoặc đạt một số lượng vòng lặp nhất định.

d) Ưu và nhược điểm:

* Ưu điểm:
  + Hiệu Quả với Dữ Liệu Lớn: SGD hiệu quả hơn đáng kể so với Gradient Descent thông thường khi làm việc với các bộ dữ liệu lớn, do không cần tính toán gradient trên toàn bộ dữ liệu.
  + Hội Tụ Nhanh: Do cập nhật trọng số sau mỗi mẫu hoặc mini-batch, SGD có thể hội tụ nhanh hơn so với Gradient Descent thông thường.
  + Khả Năng Thoát Khỏi Điểm Cực Tiểu Địa Phương: Sự ngẫu nhiên trong việc chọn mẫu hoặc mini-batch giúp SGD có khả năng thoát khỏi các điểm cực tiểu địa phương, tiềm ẩn khả năng tìm được điểm cực tiểu toàn cục.
* Nhược điểm:
  + Dao Động trong Quá Trình Hội Tụ: Do sự ngẫu nhiên trong việc chọn mẫu, SGD thường có sự dao động nhiều hơn trong quá trình hội tụ so với Gradient Descent thông thường.
  + Khó Khăn trong Việc Chọn Tốc Độ Học: Tốc độ học (learning rate) cần được chọn cẩn thận. Nếu quá lớn, thuật toán có thể không hội tụ; nếu quá nhỏ, quá trình học sẽ chậm và có thể mắc kẹt ở điểm cực tiểu địa phương.
  + Cần Tinh Chỉnh Siêu Tham Số: Việc tinh chỉnh siêu tham số cho SGD (như kích thước mini-batch và tốc độ học) có thể phức tạp và yêu cầu thử nghiệm nhiều lần.
  + Khả Năng Vượt Qua Điểm Cực Tiểu Toàn Cục: Do tính ngẫu nhiên, có khả năng SGD sẽ "vượt qua" điểm cực tiểu toàn cục mà không nhận ra.

1.2.3 Momentum

a) Khái niệm

* Momentum trong bối cảnh của thuật toán tối ưu hóa, đặc biệt là trong học máy, là một kỹ thuật được thiết kế để giúp Gradient Descent hội tụ nhanh hơn và hiệu quả hơn. Khái niệm này được lấy cảm hứng từ vật lý, nơi "momentum" đại diện cho sự di chuyển của một vật thể.

b) Cách hoạt động

* Khởi tạo tham số:
  + Khởi tạo tham số mô hình θ (ví dụ, trọng số và bias trong mạng nơ-ron).
  + Khởi tạo vector velocity v ở giá trị không.
* Tính Gradient
  + Tính gradient của hàm mất mát ∇f(θ) tại tham số hiện tại θ. Gradient này cho biết hướng và độ dốc để cải thiện tham số.
* Cập nhật Velocity:
  + Cập nhật velocity dựa trên gradient hiện tại và velocity trước đó:

A close up of a text

Description automatically generated

* Cập nhật tham số:
  + Cập nhật tham số θ sử dụng velocity mới:

A mathematical equation with a number

Description automatically generated with medium confidence

* Lặp lại: Lặp lại các bước từ 2 đến 4 cho đến khi đạt điều kiện dừng (ví dụ: số lượng epoch nhất định, sự cải thiện hàm mất mát dưới ngưỡng cho phép, v.v.).

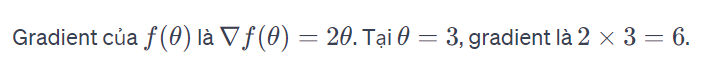
c) Ví dụ

Giả sử chúng ta có một hàm mất mát đơn giản: f(θ)=θ2 . Mục tiêu của chúng ta là tìm giá trị của θ sao cho f(θ) đạt giá trị nhỏ nhất.

**B1:** Khởi tạo tham số

* Khởi tạo θ với một giá trị nào đó, ví dụ θ = 3, và khởi tạo velocity v là 0.

**B2:** Tính Gradient



**B3:** Cập nhật Velocity

A number and numbers on a white background

Description automatically generated

**B4:** Cập nhật tham số

A number with a equal sign

Description automatically generated with medium confidence

**B5:** Lặp lại

Lặp lại quá trình từ Bước 2 đến Bước 4.

**Kết quả**: Qua mỗi lần lặp, θ sẽ dần tiến về 0, là điểm cực tiểu của hàm

f(θ) = θ2 . Sử dụng Momentum, quá trình tối ưu hóa sẽ diễn ra nhanh và mượt mà hơn so với Gradient Descent thông thường.

d) Ưu và nhược điểm

* Ưu điểm:
  + Tăng Tốc Hội Tụ: Momentum giúp tăng tốc quá trình học bằng cách thêm một thành phần động lượng, giúp cập nhật trọng số nhanh hơn so với thuần túy Gradient Descent.
  + Giảm Sự Nhạy Cảm Với Thung Lũng Hẹp: Nhờ vào động lượng, thuật toán có thể vượt qua các thung lũng hẹp trong không gian tối ưu mà không bị kẹt.
  + Hiệu Quả Trong Các Tình Huống Có Gradient Yếu: Động lượng giúp thuật toán không bị mắc kẹt tại các điểm cực tiểu địa phương khi gradient quá nhỏ.
  + Khắc Phục Dao Động: Thuật toán giảm thiểu dao động và tăng độ ổn định trong quá trình học.
* Nhược điểm:
  + Chọn Lựa Tham Số: Việc chọn lựa giá trị cho tham số động lượng đôi khi khó khăn và cần phải điều chỉnh một cách thận trọng.
  + Rủi Ro Vượt Quá Điểm Tối Ưu: Trong một số trường hợp, động lượng có thể gây ra hiện tượng "quá nhiệt" và vượt qua điểm tối ưu.
  + Không Phải Luôn Luôn Hiệu Quả: Trong một số tình huống nhất định, như với dữ liệu không ổn định hoặc phức tạp, Momentum có thể không mang lại lợi ích rõ rệt so với Gradient Descent thông thường.
  + Cần Thêm Bộ Nhớ: Do việc lưu trữ giá trị gradient trước đó, thuật toán này cần thêm bộ nhớ, có thể là một vấn đề với các mô hình lớn.

1.2.4 Adagrad

a) Đặc điểm

* Không giống như các thuật toán trước đó thì learning rate hầu như giống nhau trong quá trình training (learning rate là hằng số), Adagrad coi learning rate là 1 tham số. Tức là Adagrad sẽ cho learning rate biến thiên sau mỗi thời điểm t.

A math formula with square and square roots

Description automatically generated with medium confidence

Trong đó :

n : hằng số

gt : gradient tại thời điểm t

ϵ : hệ số tránh lỗi ( chia cho mẫu bằng 0)

G : là ma trận chéo mà mỗi phần tử trên đường chéo (i,i) là bình phương của đạo hàm vectơ tham số tại thời điểm t.

b) Cách hoạt động

* Khởi tạo tham số:
  + Khởi tạo tham số mô hình θ (ví dụ, trọng số và bias trong mạng nơ-ron).
  + Khởi tạo vector velocity v ở giá trị không.
* Tính Gradient
  + Tính gradient của hàm mất mát ∇f(θ) tại tham số hiện tại θ. Gradient này cho biết hướng và độ dốc để cải thiện tham số.
* Cập nhật Velocity:
  + Cập nhật velocity dựa trên gradient hiện tại và velocity trước đó:

A close up of a text

Description automatically generated

* Cập nhật tham số:
  + Cập nhật tham số θ sử dụng velocity mới:

A mathematical equation with a number

Description automatically generated with medium confidence

* Lặp lại: Lặp lại các bước từ 2 đến 4 cho đến khi đạt điều kiện dừng (ví dụ: số lượng epoch nhất định, sự cải thiện hàm mất mát dưới ngưỡng cho phép, v.v.).

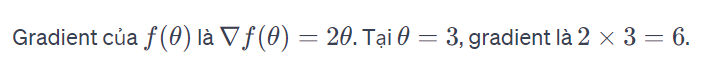
c) Ví dụ

Giả sử chúng ta có một hàm mất mát đơn giản: f(θ)=θ2 . Mục tiêu của chúng ta là tìm giá trị của θ sao cho f(θ) đạt giá trị nhỏ nhất.

**B1:** Khởi tạo tham số

* Khởi tạo θ với một giá trị nào đó, ví dụ θ = 3, và khởi tạo velocity v là 0.

**B2:** Tính Gradient



**B3:** Cập nhật Velocity

A number and numbers on a white background

Description automatically generated

**B4:** Cập nhật tham số

A number with a equal sign

Description automatically generated with medium confidence

**B5:** Lặp lại

Lặp lại quá trình từ Bước 2 đến Bước 4.

**Kết quả**: Qua mỗi lần lặp, θ sẽ dần tiến về 0, là điểm cực tiểu của hàm

f(θ) = θ2 . Sử dụng Momentum, quá trình tối ưu hóa sẽ diễn ra nhanh và mượt mà hơn so với Gradient Descent thông thường.

d) Ưu và nhược điểm

* Ưu điểm:
  + Tăng Tốc Hội Tụ: Momentum giúp tăng tốc quá trình học bằng cách thêm một thành phần động lượng, giúp cập nhật trọng số nhanh hơn so với thuần túy Gradient Descent.
  + Giảm Sự Nhạy Cảm Với Thung Lũng Hẹp: Nhờ vào động lượng, thuật toán có thể vượt qua các thung lũng hẹp trong không gian tối ưu mà không bị kẹt.
  + Hiệu Quả Trong Các Tình Huống Có Gradient Yếu: Động lượng giúp thuật toán không bị mắc kẹt tại các điểm cực tiểu địa phương khi gradient quá nhỏ.
  + Khắc Phục Dao Động: Thuật toán giảm thiểu dao động và tăng độ ổn định trong quá trình học.
* Nhược điểm:
  + Chọn Lựa Tham Số: Việc chọn lựa giá trị cho tham số động lượng đôi khi khó khăn và cần phải điều chỉnh một cách thận trọng.
  + Rủi Ro Vượt Quá Điểm Tối Ưu: Trong một số trường hợp, động lượng có thể gây ra hiện tượng "quá nhiệt" và vượt qua điểm tối ưu.
  + Không Phải Luôn Luôn Hiệu Quả: Trong một số tình huống nhất định, như với dữ liệu không ổn định hoặc phức tạp, Momentum có thể không mang lại lợi ích rõ rệt so với Gradient Descent thông thường.
  + Cần Thêm Bộ Nhớ: Do việc lưu trữ giá trị gradient trước đó, thuật toán này cần thêm bộ nhớ, có thể là một vấn đề với các mô hình lớn.

1.2.5 RMSprop

a) Đặc điểm

* RMSprop giải quyết vấn đề tỷ lệ học giảm dần của Adagrad bằng cách chia tỷ lệ học cho trung bình của bình phương gradient.

A math equations and formulas

Description automatically generated with medium confidence

b) Cách hoạt động

* Tính Gradient: Tương tự như các thuật toán tối ưu hóa khác, RMSprop tính toán gradient của hàm mất mát với mỗi tham số.
* Tích Lũy Gradient: Ao khác với Adagrad, RMSprop không tích lũy tất cả bình phương gradient từ khi bắt đầu. Thay vào đó, nó áp dụng một cửa sổ trượt để tích lũy gradient gần đây, sử dụng phép tính trung bình di động trọng số (exponential moving average). Điều này giúp nó tập trung nhiều hơn vào các thông tin gradient gần đây.
* Điều Chỉnh Tỷ Lệ Học: Tỷ lệ học được điều chỉnh cho mỗi tham số bằng cách chia tỷ lệ học cố định cho căn bậc hai của trung bình di động trọng số này (cộng thêm một số nhỏ để tránh chia cho 0).
* Cập Nhật Tham Số: Các tham số được cập nhật dựa trên tỷ lệ học đã điều chỉnh và gradient tương ứng.

c) Ví dụ

* Bài toán:
  + Diện tích nhà (m²): 50, 60, 70
  + Giá nhà (ngàn USD): 300, 350, 400
* Khởi tạo:
  + Tham số mô hình (trọng số w và độ chệch b): w = 0, b = 0
  + Tỷ lệ học cố định: 0.01
  + Biến tích lũy cho RMSprop: v\_w = 0, v\_b = 0
  + Hệ số suy giảm: 0.9
* Quá trình huấn luyện
  + B1: Tính Gradient:
    - Tính gradient của hàm mất mát (ví dụ: sai số bình phương) đối với w và b.
    - Giả sử gradient tại bước đầu tiên là dw = -20, db = -5.
  + B2: Cập Nhật Biến Tích Lũy:
    - v\_w = 0.9 \* v\_w + 0.1 \* (dw)^2
    - v\_b = 0.9 \* v\_b + 0.1 \* (db)^2
    - Thay số vào, v\_w = 0 + 0.1 \* (-20)^2 = 40, v\_b = 0 + 0.1 \* (-5)^2 = 2.5.
  + Điều Chỉnh Tỷ Lệ Học:
    - Tỷ lệ học cho w: learning\_rate / sqrt(v\_w + epsilon) (epsilon là một số nhỏ để tránh chia cho 0, giả sử epsilon = 1e-8).
    - Tương tự cho b.
    - Thay số vào, tỷ lệ học mới cho w là 0.01 / sqrt(40 + 1e-8) ≈ 0.00158.
  + Cập Nhật Tham Số:
    - w = w - learning\_rate \* dw
    - b = b - learning\_rate \* db
    - Thay số vào, w = 0 - 0.00158 \* (-20) ≈ 0.0316, b = 0 - 0.01 \* (-5) = 0.05.
  + Lặp Lại Quá Trình: Quá trình này được lặp lại với dữ liệu mới, cập nhật v\_w, v\_b, và các tham số w, b tương ứng.

d) Ưu và nhược điểm

* Ưu điểm:
  + Điều Chỉnh Tự Động Tỷ Lệ Học: RMSprop tự động điều chỉnh tỷ lệ học dựa trên gradient gần đây, giúp quá trình học được ổn định và hiệu quả hơn so với các phương pháp cố định tỷ lệ học.
  + Hiệu Quả với Gradient Phức Tạp: RMSprop hoạt động tốt trong các tình huống có gradient phức tạp hoặc không đồng đều, như trong các mô hình mạng nơ-ron hồi quy (RNN).
  + Tránh Vấn Đề Tỷ Lệ Học Giảm Quá Nhanh: So với Adagrad, RMSprop không để cho tỷ lệ học giảm quá mạnh do nó chỉ tính toán trung bình di động của gradient, giúp tránh được việc tỷ lệ học tiến đến gần 0.
  + Phù Hợp Cho Các Bài Toán Khó: RMSprop thường hoạt động tốt trong các bài toán khó, nơi mà các thuật toán tối ưu hóa thông thường khó đạt được kết quả tốt.
* Nhược điểm:
  + Cần Chọn Lựa Tham Số Cẩn Thận: Việc chọn hệ số suy giảm và tỷ lệ học ban đầu cần được thực hiện một cách cẩn thận, vì chúng có ảnh hưởng lớn đến hiệu suất của thuật toán.
  + Không Có Đảm Bảo Toàn Cục Tối Ưu: Như mọi thuật toán tối ưu hóa dựa trên gradient, RMSprop không đảm bảo tìm được điểm toàn cục tối ưu, đặc biệt trong không gian tham số có nhiều cực trị.
  + Có Thể Yêu Cầu Nhiều Bộ Nhớ Hơn: Việc lưu trữ trung bình di động của gradient cho mỗi tham số có thể đòi hỏi nhiều bộ nhớ hơn, đặc biệt trong các mô hình có số lượng tham số lớn
  + Tinh Chỉnh Cần Thiết: Trong một số trường hợp, việc tinh chỉnh tham số có thể cần thiết để đạt được hiệu suất tối ưu, điều này có thể tốn thời gian và nỗ lực.

1.3 So sánh các phương pháp Optimizer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tiêu Chí** | **Gradient Descent** | **Stochastic Gradient Descent** | **Momentum** | **Adagrad** | **RMSprop** |
| **Cơ Chế Cơ Bản** | Cập nhật tham số dựa trên gradient của hàm mất mát trên toàn bộ tập dữ liệu. | Cập nhật tham số dựa trên gradient của hàm mất mát trên mỗi mẫu dữ liệu. | Dựa trên SGD, thêm thành phần động lượng để tăng tốc độ học. | Dựa trên SGD, tự động điều chỉnh tỷ lệ học dựa trên lịch sử gradient. | Cải tiến của Adagrad, điều chỉnh tỷ lệ học dựa trên trung bình di động của gradient bình phương. |
| **Ưu Điểm** | Đơn giản, dễ hiểu; hiệu quả với tập dữ liệu nhỏ. | Nhanh chóng và hiệu quả với tập dữ liệu lớn; hỗ trợ học trực tuyến. | Giảm dao động, học nhanh hơn SGD. | Hiệu quả với dữ liệu thưa thớt; tự điều chỉnh tỷ lệ học. | Giải quyết vấn đề giảm tốc độ học của Adagrad; hiệu quả với gradient phức tạp. |
| **Nhược Điểm** | Tốn thời gian và tài nguyên với tập dữ liệu lớn; dễ mắc kẹt tại cực tiểu địa phương. | Dao động mạnh trong quá trình học; dễ mắc kẹt tại cực tiểu địa phương. | Cần chọn lựa tham số động lượng. | Tỷ lệ học có thể giảm quá nhanh; cần tinh chỉnh tham số. | Cần chọn lựa tham số suy giảm và tỷ lệ học ban đầu. |
| **Ứng Dụng** | Mô hình nhỏ và tập dữ liệu không quá lớn. | Mô hình lớn và tập dữ liệu lớn; học trực tuyến. | Bài toán cần tối ưu hóa nhanh. | Bài toán với dữ liệu thưa thớt. | Bài toán học sâu với gradient phức tạp. |

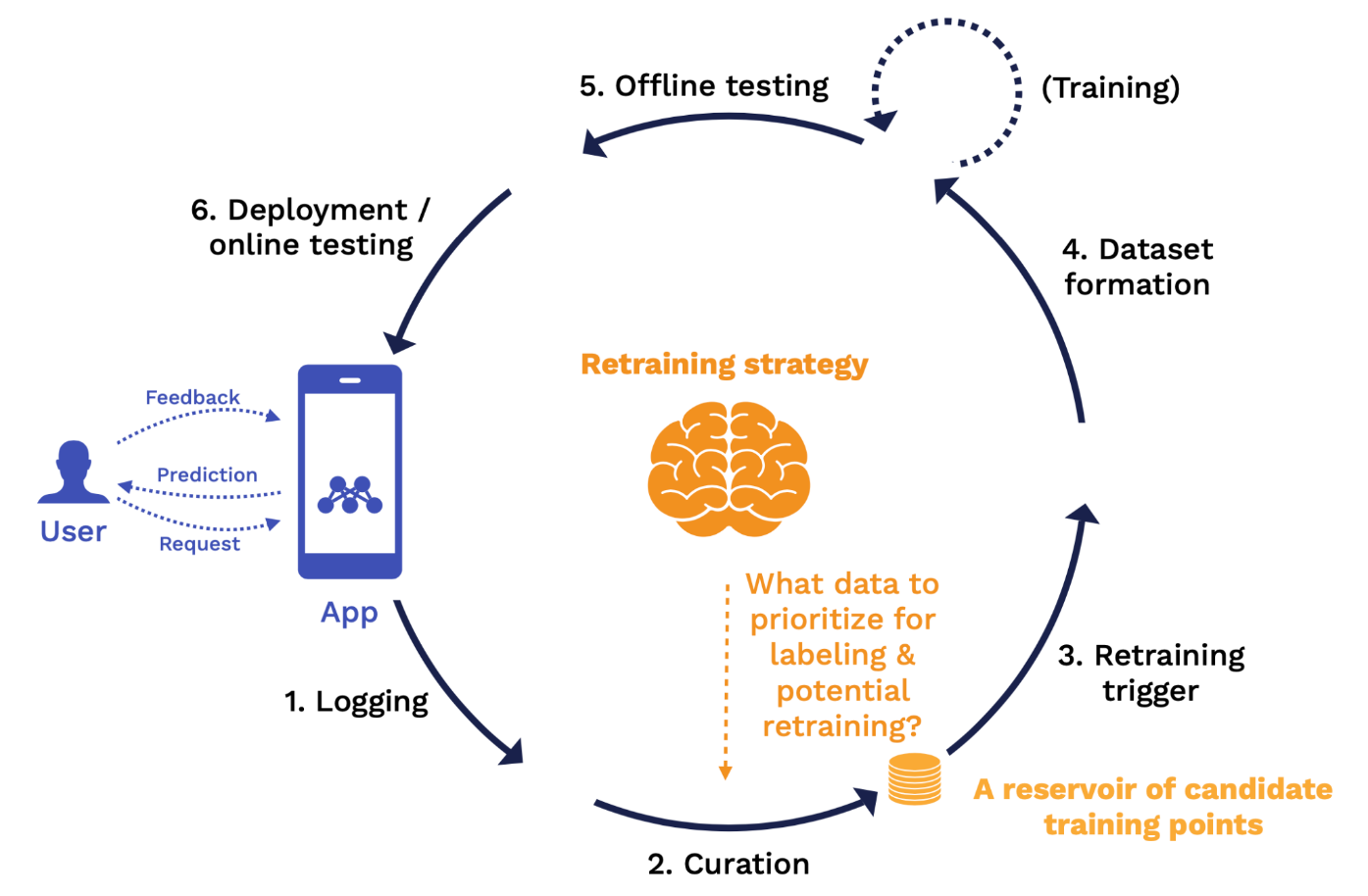
CHƯƠNG 2 – CONTINUAL LEARNING VÀ TEST PRODUCTION

2.1 Continual Learning

2.1.1 Khái niệm

* Continual Learning là quá trình mà trong đó một mô hình học máy tiếp tục học và phát triển sau khi được triển khai. Mục tiêu là để mô hình có thể thích nghi với dữ liệu mới và thay đổi môi trường mà không quên những gì đã học trước đó.
* Trong môi trường thực tế, dữ liệu thường xuyên thay đổi và phát triển. Việc đảm bảo rằng mô hình có thể thích ứng với những thay đổi này là rất quan trọng để duy trì hiệu suất và độ chính xác.

2.1.2 Cách thức hoạt động



* Thu thập dữ liệu di động:
  + Dữ Liệu Động: Trong môi trường thực, dữ liệu liên tục thay đổi và phát triển. Một mô hình Continual Learning cần có khả năng xử lý và học từ dữ liệu mới này.
  + Luồng Dữ Liệu: Thay vì học từ một tập dữ liệu cố định, mô hình liên tục nhận dữ liệu mới và cập nhật kiến thức.
* Học Từ Dữ Liệu Mới:
  + Tích Lũy Kiến Thức: Mô hình phải có khả năng tích lũy kiến thức từ dữ liệu mới mà không quên những gì đã học.
  + Adaptation: Mô hình thích nghi với các xu hướng và mẫu mới trong dữ liệu.
* Ngăn Chặn Quên Catastrophic:
  + Catastrophic Forgetting: Đây là thách thức chính của Continual Learning, nơi mô hình quên đi kiến thức cũ khi tiếp xúc với kiến thức mới.
  + Phương Pháp Ngăn Chặn: Sử dụng các kỹ thuật như Regularization, Rehearsal (luyện tập lại dữ liệu cũ), và Architectural Approaches (thiết kế mô hình đặc biệt) để giữ kiến thức cũ.
* Cập Nhật Mô Hình:
  + Online Learning: Mô hình có khả năng học trực tuyến từ mỗi mẫu dữ liệu mới hoặc từ batch nhỏ của dữ liệu mới.
  + Model Plasticity and Stability: Cân bằng giữa khả năng thích nghi với dữ liệu mới (plasticity) và giữ ổn định kiến thức cũ (stability).
* Đánh Giá và Tối Ưu Hóa:
  + Continuous Evaluation: Đánh giá mô hình liên tục để đảm bảo hiệu suất và tính chính xác.
  + Feedback Loop: Sử dụng phản hồi từ môi trường thực tế để tối ưu hóa và cải thiện mô hình.

2.1.3 Ví dụ minh họa

* Bối cảnh: Giả sử chúng ta có một trang thương mại điện tử với các dữ liệu sau:
  + Số Lượng Người Dùng: 10.000 người dùng.
  + Số Lượng Sản Phẩm: 1.000 sản phẩm.
  + Hành Vi Người Dùng: Mỗi ngày, trang web ghi nhận khoảng 5.000 lượt xem sản phẩm và 500 giao dịch mua hàng.
* Mục Tiêu:
  + Tối ưu hóa đề xuất sản phẩm để tăng tỷ lệ chuyển đổi mua hàng.
  + Thích ứng với sở thích thay đổi của người dùng và xu hướng thị trường.
* Cách thức hoạt động:
  + Thu thập dữ liệu hằng ngày:
    - Ngày 1: 5.000 lượt xem sản phẩm, 500 giao dịch mua hàng.
    - Ngày 2: Dữ liệu mới với 5.100 lượt xem và 520 giao dịch.
  + Cập nhật mô hình đề xuất:
    - Sử dụng dữ liệu từ Ngày 1 để huấn luyện mô hình ban đầu.
    - Vào Ngày 2, cập nhật mô hình bằng cách học từ dữ liệu mới (5.100 lượt xem và 520 giao dịch) mà không quên những gì đã học từ Ngày 1.
  + Ngăn chặn quên Catastrophic: Áp dụng các kỹ thuật như Rehearsal (luyện tập lại) hoặc Elastic Weight Consolidation để mô hình không quên dữ liệu cũ.
  + Đánh giá và tối ưu hóa:
    - Theo dõi metrices như tỷ lệ chuyển đổi mua hàng, độ chính xác đề xuất.
    - Ngày 3: Tỷ lệ chuyển đổi tăng từ 10% lên 12% do đề xuất được cải thiện.
  + Kết quả : Mô hình học liên tục và thích nghi với sở thích người dùng và xu hướng mới. Tăng tỷ lệ chuyển đổi và sự hài lòng của người dùng thông qua đề xuất cá nhân hóa chính xác hơn.

Code minh họa:

A computer screen with colorful text

Description automatically generated

A computer screen shot of a program code

Description automatically generated

A screen shot of a computer program

Description automatically generated

Kết quả:

A screenshot of a computer

Description automatically generated

2.2 Test Production

2.2.1 Khái niệm

- "Test Production" trong ngữ cảnh của học máy và phát triển phần mềm thường ám chỉ quá trình kiểm thử và đánh giá mô hình hoặc ứng dụng trong một môi trường sản xuất thực tế, nơi nó tương tác với dữ liệu thực và người dùng thực. Đây là một bước quan trọng để đảm bảo rằng ứng dụng hoặc mô hình hoạt động đúng như mong đợi khi được phát hành.

2.2.2 Mục đích

- Đánh Giá Hiệu Suất Thực Tế: Kiểm tra xem mô hình hoặc ứng dụng có hoạt động tốt trong môi trường thực tế hay không.

- Phát Hiện Lỗi và Vấn Đề: Tìm ra các lỗi hoặc vấn đề không được phát hiện trong quá trình phát triển hoặc kiểm thử ban đầu.

2.2.3 Phân loại

- A/B Testing: Phân chia người dùng thành các nhóm để kiểm thử các phiên bản khác nhau của một sản phẩm.

- Canary Testing: Phát hành tính năng mới hoặc mô hình mới cho một nhóm người dùng nhỏ trước khi triển khai rộng rãi.

- Blue/Green Deployment: Chuyển đổi giữa hai môi trường (blue và green) để kiểm thử và triển khai một cách an toàn.

2.2.4 Cách thức hoạt động

A diagram of a test production

Description automatically generated

- Triển Khai Mô Hình hoặc Ứng Dụng: Đưa mô hình hoặc ứng dụng vào môi trường sản xuất.

- Theo Dõi và Thu Thập Dữ Liệu: Sử dụng các công cụ để theo dõi hiệu suất và thu thập phản hồi từ người dùng.

- Phân Tích và Đánh Giá: Phân tích dữ liệu thu thập được để đánh giá hiệu suất và tìm kiếm lỗi.

2.2.5 Ví dụ minh họa

* Đầu tiên, chúng ta cần một mô hình học máy đơn giản. Giả sử chúng ta đang làm việc với một mô hình hồi quy tuyến tính để dự đoán giá nhà. Dưới đây là một ví dụ cơ bản:

A screen shot of a computer program

Description automatically generated

* Tiếp theo, chúng ta sẽ cần một cách để theo dõi hoạt động của mô hình trong môi trường sản xuất. Ở đây, chúng ta sẽ sử dụng logging để ghi nhật ký:

A screen shot of a computer program

Description automatically generated

* Cuối cùng, chúng ta giả định rằng mô hình này sẽ được tải và sử dụng trong môi trường sản xuất để dự đoán giá nhà dựa trên các đặc điểm mới:

A screen shot of a computer code

Description automatically generated

2.2.6 Ưu và nhược điểm:

* Ưu điểm:
  + Các chương trình Beta nơi mà khách hàng cung cấp các phản hồi sớm về các tính năng mới và trải nghiệm người dùng.
  + Ngăn chặn các thảm họa với kiểm thử phục hồi và khả năng phục hồi tốt hơn. Ứng dụng có thể phục hồi từ các sự kiện mong đợi (hỗn loạn) hoặc không mong đợi mà không mất chức năng và dữ liệu.
  + Thiết kế và xây dựng 1 quy trình khắc phục thảm họa sẽ giúp giải phóng các hỗn loạn trong môi trường pre-production trước khi thực hiện trong môi trường live production.
  + Bạn đang kiểm thử với dữ liệu production. (thật khó để mô phỏng theo lưu lượng và dữ liệu trên môi trường production, dẫn tới khó có thể phát hiện ra mọi tình huống có thể xảy ra để kiểm thử).
  + Nó sẽ loại bỏ rủi ro của việc phát triển thường xuyên trên môi trường production khi được thực hiện hàng ngày, trong khi bạn giám sát hiệu suất ứng dụng trong thời gian thực với các công cụ như New Relic. (đại loại là nếu như có sự kiểm thử thường xuyên trên môi trường product thì việc phát hiện ra lỗi trước khi mà khách hàng báo lỗi là cao hơn)
* Nhược điểm:
  + Không có kế hoạch dự phòng cho trường hợp ứng dụng có nguy cơ mất dữ liệu
  + Không có kế hoạch rollback khi release 1 bản mới.
  + Để lộ ra những lỗ hổng tiềm năng
  + Không thể phục hồi sau những sự hỗn loạn bất ngờ
  + Thời gian kiểm thử gây ra trải nghiệm không tốt cho người dùng

TÀI LIỆU THAM KHẢO

**Tiếng Việt**

[1]

<https://vi.wikipedia.org/wiki/H%E1%BB%8Dc_c%C3%B3_gi%C3%A1m_s%C3%A1t>

[2]

<https://machinelearningcoban.com/general/2017/02/06/featureengineering/>

**Tiếng Anh**