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

DỰ ÁN CUỐI KỲ

NHẬP MÔN HỌC MÁY

Người hướng dẫn: **TS LÊ ANH CƯỜNG**

Người thực hiện: **LA NGUYỄN QUỐC THỊNH – 521H0513**

Lớp **: 21H50302**

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

DỰ ÁN CUỐI KỲ

NHẬP MÔN HỌC MÁY

Người hướng dẫn: **TS LÊ ANH CƯỜNG**

Người thực hiện: **LA NGUYỄN QUỐC THỊNH – 521H0513**

Lớp **: 21H50302**

Khoá **: 25**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Em xin cảm ơn thầy Lê Anh Cường đã đồng hành cùng em trong suốt Học kỳ I năm 2023-2024. Những kiến thức giảng dạy mà thầy đã truyền tải cho em trong suốt quá trình học tập đã giúp cho em có được những kiến thức cần thiết để có thể qua được bài đồ án cuối kỳ của môn.

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 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)

La Nguyễn Quốc Thịnh

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

Nội dung trong bài sẽ bao gồm các yêu cầu của đề tài trong phần 1 của bài Đồ án cuối kỳ môn “Nhập môn học máy” bao gồm việc tìm hiểu về các các tối ưu một mô hình học náy và các phương pháp học khi xây dựng một mô hình hiệu quả.

MỤC LỤC

Contents

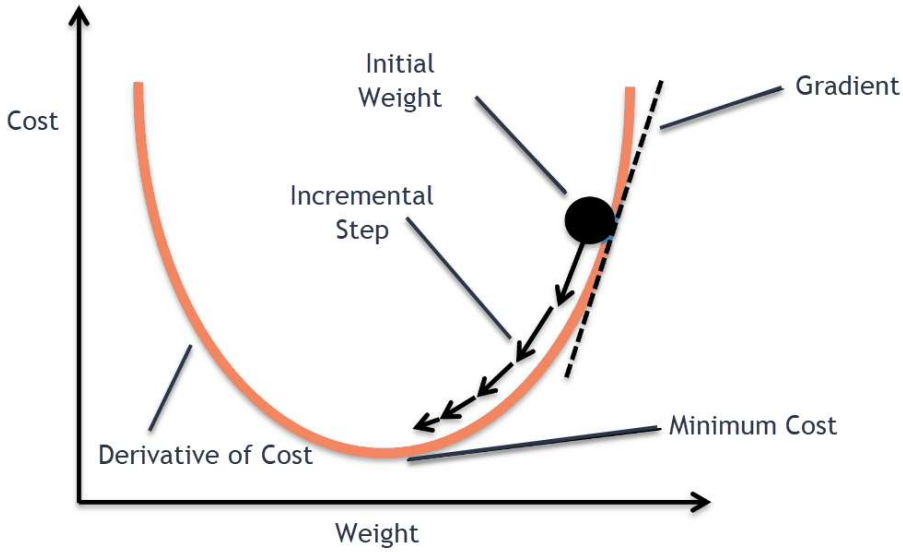
CHƯƠNG 1 – CÁC PHƯƠNG PHÁP OPTIMIZER TRONG MÔ HÌNH HỌC MÁY

Tổng quan: Có rất nhiều phương pháp Optimizer trong việc xây dựng một mô hình học máy hoặc một mô hình Neural Network, tuy nhiên trong bài báo cáo này sẽ nhắc đến 6 loại phổ biến đó là: Gradient Descent (GD), Stochastic Gradient Descent (SGD), Momentum, Adagrad, RMSprop, Adam.

1.1 Gradient Descent (GD):

- Hàm mất mát – Loss function được biết đến là một hàm số dùng để tính toán sai lệch giữa kết quả dự đoán một mô hình học máy và kết quả thực tế hoặc trên tận dữ liệu, một mô hình học máy được xem là một mô hình lí tưởng là khi loss function bằng 0. Tuy nhiên với những tập dữ liệu đa biến thì điều này gần như là bất khả thi, nên thay vào đó chúng ta thường tìm giá trị nhỏ nhất của hàm mất mát này.

- Gradient Descent là khái niệm tính đạo hàm của hàm mất mát theo trọng số của mỗi tập đầu vào, điều chỉnh tham số sau mỗi vòng lặp để đảm bảo lossfuction đạt giá trị cực tiểu.

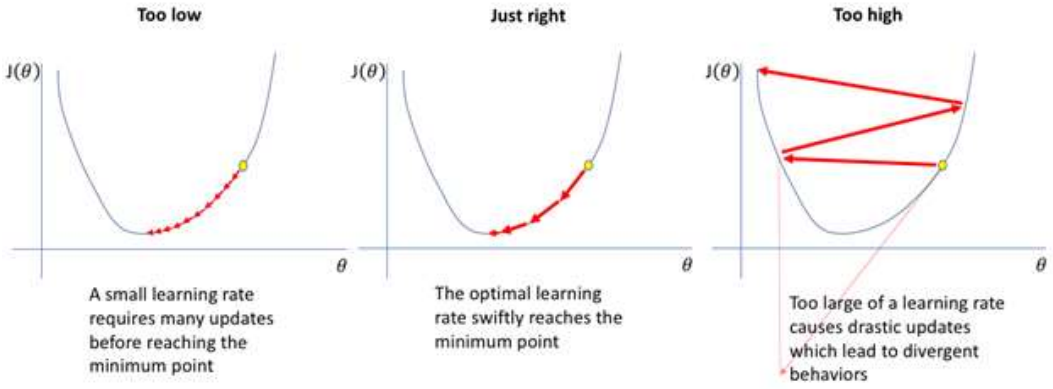


Bước khởi tạo	Khởi tạo tham số	---------	Lạc vào đâu đó trên núi
Bước 1	Tính đạo hàm của hàm loss theo các biến	---------	Quan sát các vị trí xung quanh
Bước 2	Cập nhật các tham số mới	---------	Di chuyển đến vị trí đã chọn
Bước 3	Lặp lại bước 1 và 2 đến khi thỏa điều kiện dừng	---------	Lặp lại bước 1 và 2 đến khi xuống đến thung lũng

- Công thức cho mỗi lần cập nhật trọng số:

$$w_i = w_{i-1} - \alpha * \frac{d(loss)}{d(w)}$$

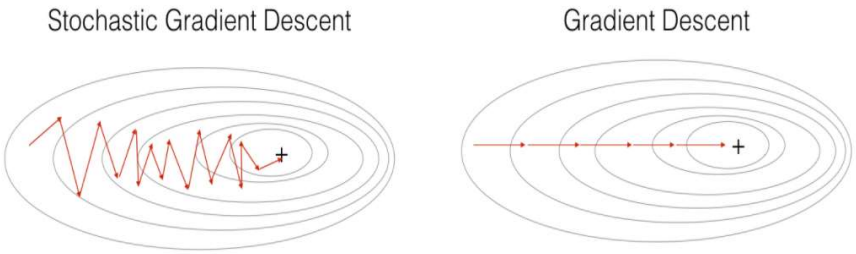
- Với α là learning rate, giá trị độ lớn cho mỗi lần di chuyển (có thể được gọi là bước nhảy). Nếu learning rate quá nhỏ thì khi này thuật toán sẽ mất rất nhiều thời gian để có thể hội tụ đến giá trị nhỏ nhất (tốn nhiều bước nhảy để có thể di chuyển đến điểm cần đến) và nếu learning rate quá to thì sẽ khiến thuật toán đi qua điểm hội tụ và không thể đến được điểm cực tiểu.



- Ưu điểm:
    - Đơn giản, dễ hiểu. Thuật toán đã giải quyết được vấn đề tối ưu bằng cách cập nhật trọng số sau mỗi lần lặp.
- Nhược điểm:
    - Phụ thuộc vào nghiệm ngẫu nhiên được khoier tạo ban đầu.
    - Hàm bị phụ thuộc vào learrning rate.

1.2 Scochastic Gradient Descent (SGD):

- SGD là một dạng biến thể của GD. Với thuật toán GD ta sẽ cập nhật toàn bộ trọng số của tập dữ liệu sau mỗi vòng lặp (epoch) thì SGD sẽ cập nhật lần lượt trọng số cho mỗi điểm dữ liệu có trong tập dữ liệu đầu vào.



-Ví dụ tập dữ liệu đầu vào của mô hình gồm 5 dữ liệu vào thì trong 1 epoch, SGD sẽ lần lượt cập nhật 5 trọng số cho từng điểm dữ liệu và tiếp tục lặp lại cho đến khi hết số lượng vòng lặp (thường là 10).

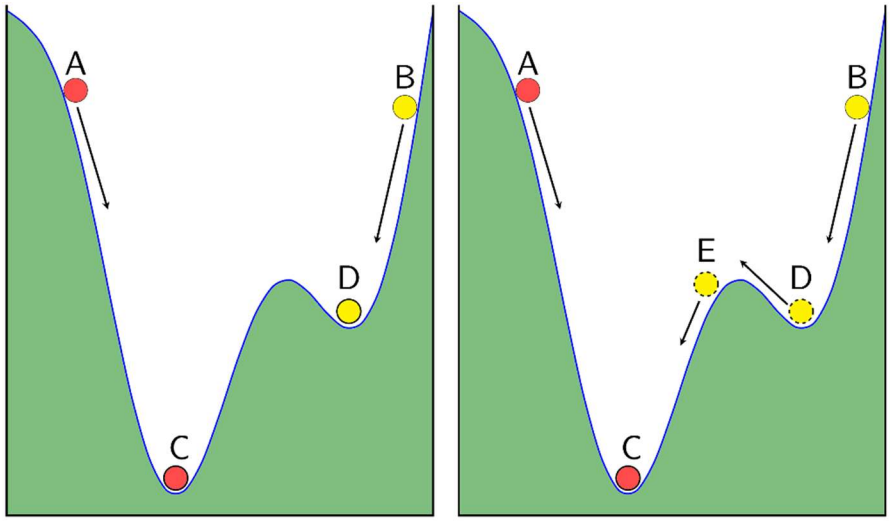
- Điều này cùng nghĩa với việc thời gian chạy 1 epoch sẽ lâu hơn so với thuật toán GD, tuy nhiên SGD sẽ hội tụ rất nhanh chỉ sau vài epoch. Công thức cập nhật trọng số của SGD hoạt động tương tự như GD nhưng thực hiện trên từng điểm dữ liệu.

- Ưu điểm:
    - Giải quyết được với các tập dữ liệu lớn mà GD không làm được, thuật toán SGD đến ngày nay vẫn được sử dụng rộng rãi.
- Nhược điểm:
    - Chưa giải quyết được 2 nhược điểm của bài toán GD: bị phụ thuộc vào learning rate và dữ liệu ban đầu.

1.3 Momentum:

- Momentum là một biến thể của Gradient Descent. Momentum được tạo ra nhằm giảm thời gian hội tụ của thuật toán GD. Ý tưởng chính của thuật toán này ta sẽ sử dụng động lượng dựa trên epoch trước đó để giảm bớt tác động của một độ dốc nhỏ và dao động.

- Để dễ hiểu hơn thì ta có thể nhìn dưới góc độ vật lý.



b) GD c) GD with momentum

- Tưởng tượng nếu ta thả 2 viên bi tại 2 điểm khác nhau thì với thuật toán Gradient Descent thì viên bi đỏ sẽ dừng ở điển C và viên bi vàng sẽ dừng ở điểm D trong khi điểm D không phải là điểm cực tiểu của thuật toán mà ta mong muốn. Dưới sự tác động của cơ năng thì viên bi vàng sẽ được tiếp thêm động lượng, vượt qua đỉnh E và lăn xuống X, đây chính là điều mà thuật toán Gradient Descent Momentum đang thực hiện, không những vậy dưới tác động của động năng thì viên bi sẽ lăn xuống điểm cực tiểu nhanh hơn.

- Công thức của GDM:

$$x_{new} = x_{old} - (gamma * v + \propto \frac{d(loss)}{d(w)})$$

- Trong đó
    - X_{new} là tọa độ mới
    - X_{old} là tọa độ cũ
    - Gamma là một parameter, thường là 0.9
- Ưu điểm:
    - Thuật toán tối ưu giải quyết được vấn đề GD không tiến được đến điểm global minimum mà chỉ dừng lại ở local minimun.
- Nhược điểm:
    - Khi gần đến điểm đích cần mất nhiều thời gian dao động qua lại trước khi dừng hẳn (bảo toàn cơ năng) nên phải tốn một khoản thời gian.

1.4 Adagrad:

- Adagrad là một thuật toán dựa trên gradient descent. Khác với GD, thuật toán này sẽ điều chỉnh tham số learning_rate cho từng tham số của mô hình. Ý tưởng của Adagrad là cập nhật tốc độ học của mỗi tham số dựa trên tổng bình phương của gradient đã tích lũy trước đó.

- Công thức của Adagrad:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \varepsilon}} \cdot g_{t,i}$$

$$g_{t,i} = \nabla_{\theta} J\left(\theta_{t,i}\right)$$

- Theo quy tắc cập nhật, Adagrad điều chỉnh tốc độ học η tại bước t tương ứng với trọng số θi xác định dựa trên các đạo hàm đã tính được theo θi. Mẫu số là chuẩn L2 (L2 norm) của ma trận đường chéo Gt trong đó phần tử i i, là tổng bình phương của các gradient tương ứng với θi tính đến bước t. ε là một số dương khá nhỏ nhằm tránh trường hợp mẫu số bằng 0.

- Ưu điểm:
  - Tránh điều chỉnh learning rate bằng tay, chỉ cần đề learning rate bằng 0.01 thì thuật toán sẽ tự động điều chỉnh.
- Nhược điểm:
  - Điểm yếu của thuật toán này nằm ở chỗ tổng bình phương biến thiên sẽ lớn dần theo thời gian cho đến khi nó làm tốc độ học cực kì nhỏ, điều này có thể làm cho quá trình học bị đóng băng.

1.5 RMSprop:

- RMSprop tương tự như Adagrad, tuy nhiên thuật toán này sẽ giải quyết vấn đề tỷ lệ học của Adagrad bằng cách chia tỷ lệ học cho trung bình của phương trình Gradient.

$$E[g^2]_t = 0{,}9E[g^2]_{t\text{-}1} + 0{,}1g_t^2$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

- Ưu điểm:
  - Giải quyết vấn đề cập nhật trọng số của Adagrad.
- Nhược điểm:
  - Có kết quả đạt được chỉ là local minimum chứ không phải global minimum như Momentum, thông thường người ta sẽ kết hợp cả 2 thuật toán cho ra một thuật toán tối ưu.

1.6 Adam:

- Thuật toán Adam chính là thuật toán tối ưu, được kết hợp bởi GD Momentum và RMSprop, thuật toán tự động điều chỉnh learning rate cho từng tham số và theo dõi các momentum trong quá trình tối ưu hóa.

- Trong khi momentum giống như một quả cầu lao xuống dốc, thì Adam lại giống như một quả cầu rất nặng và có ma sát (friction), nhờ vậy nó dễ dàng vượt qua local minimum và đạt tới điểm tối ưu nhất (flat minimum)

- Nó đạt được hiệu ứng Heavy Ball with Friction (HBF) nhờ vào hệ số (mt/ sqrt(vt))

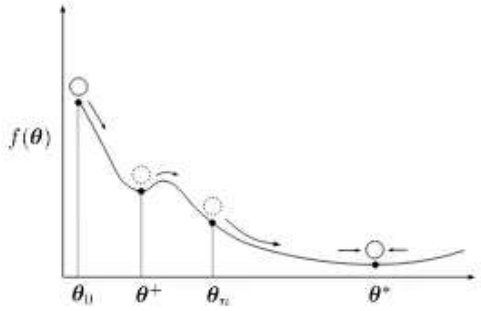


Figure 2: Heavy Ball with Friction, where the ball with mass overshoots the local minimum θ^+ and settles at the flat minimum θ^*.

- Công thức tối ưu:

$$
\begin{aligned}
\boldsymbol{g}_n &\longleftarrow \nabla f(\boldsymbol{\theta}_{n-1}) \\
\boldsymbol{m}_n &\longleftarrow (\beta_1/(1-\beta_1^n))\,\boldsymbol{m}_{n-1} + ((1-\beta_1)/(1-\beta_1^n))\,\boldsymbol{g}_n \\
\boldsymbol{v}_n &\longleftarrow (\beta_2/(1-\beta_2^n))\,\boldsymbol{v}_{n-1} + ((1-\beta_2)/(1-\beta_2^n))\,\boldsymbol{g}_n \odot \boldsymbol{g}_n \\
\boldsymbol{\theta}_n &\longleftarrow \boldsymbol{\theta}_{n-1} - a\,\boldsymbol{m}_n/(\sqrt{\boldsymbol{v}_n} + \epsilon)\,,
\end{aligned}
$$

CHƯƠNG 2 – TÌM HIỂU VỀ CONTINUAL LEARNING VÀ TEST PRODUCTION KHI XÂY DỰNG MỘT BÀI TOÁN MÔ HÌNH HỌC MÁY.

1.1 Continual Learning:

- Continual Learning còn được biết đến với cái tên là lifelong learning hoặc incremental learning, là một phương pháp học máy mà mô hình được cập nhật liên tục khi có dữ liệu mới, mà không cần phải training model lại từ đầu. Phương pháp này hữu ích với những trường hợp khi dữ liệu thay đổi theo thời gian hoặc khi mô hình cần học các dữ liệu mới.

1.1.1 Online Learning

- Là một hình thức của continuous learning trong đó mô hình được cập nhật ngay khi có dữ liệu mới đến, không cần phải giữ lại toàn bộ dữ liệu đã sử dụng trước đó.

1.1.2 Transfer Learning

- Đây là phương thức vô cùng phổ biến và hữu ích nếu có ít data. Cách áp dụng rất đơn giản. Sử dụng model đã được huấn luyện trên 1 dataset rất lớn và có nét tương đồng với dataset đó (ImageNet, COCO, ...). Sau đó thay thế 1 vài layer cuối của model đó bằng các layer mới, và sau đó tiếp tục huấn luyện mô hình mới này với dataset. Trong quá trình này thì chỉ có các layer mới được gắn thêm vào mới được update thông qua backpropagation.

1.1.3 Fine – tuning:

- Gần tương tự như Transfer learning, nhưng số lượng các layer được update sẽ linh động hơn. Có thể 1 vài hoặc thậm chí toàn bộ các layer sẽ được update. Ngoài ra những layer cuối cùng có thể sẽ không bị thay đổi bởi các layer mới

1.1.4 Domain Adaptation

- Quá trình điều chỉnh mô hình để làm cho nó hiệu quả trên dữ liệu từ một miền mới mà nó chưa được huấn luyện trước đó.

1.1.5 Incremental Learning

- Mô hình được cập nhật với dữ liệu mới và không cần phải giữ lại toàn bộ dữ liệu đã sử dụng trước đó.

1.1.6 Retraining

- Một phương pháp trong đó mô hình được đào tạo lại trên tập dữ liệu mới, thường là cộng với tập dữ liệu đã được sử dụng trước đó.

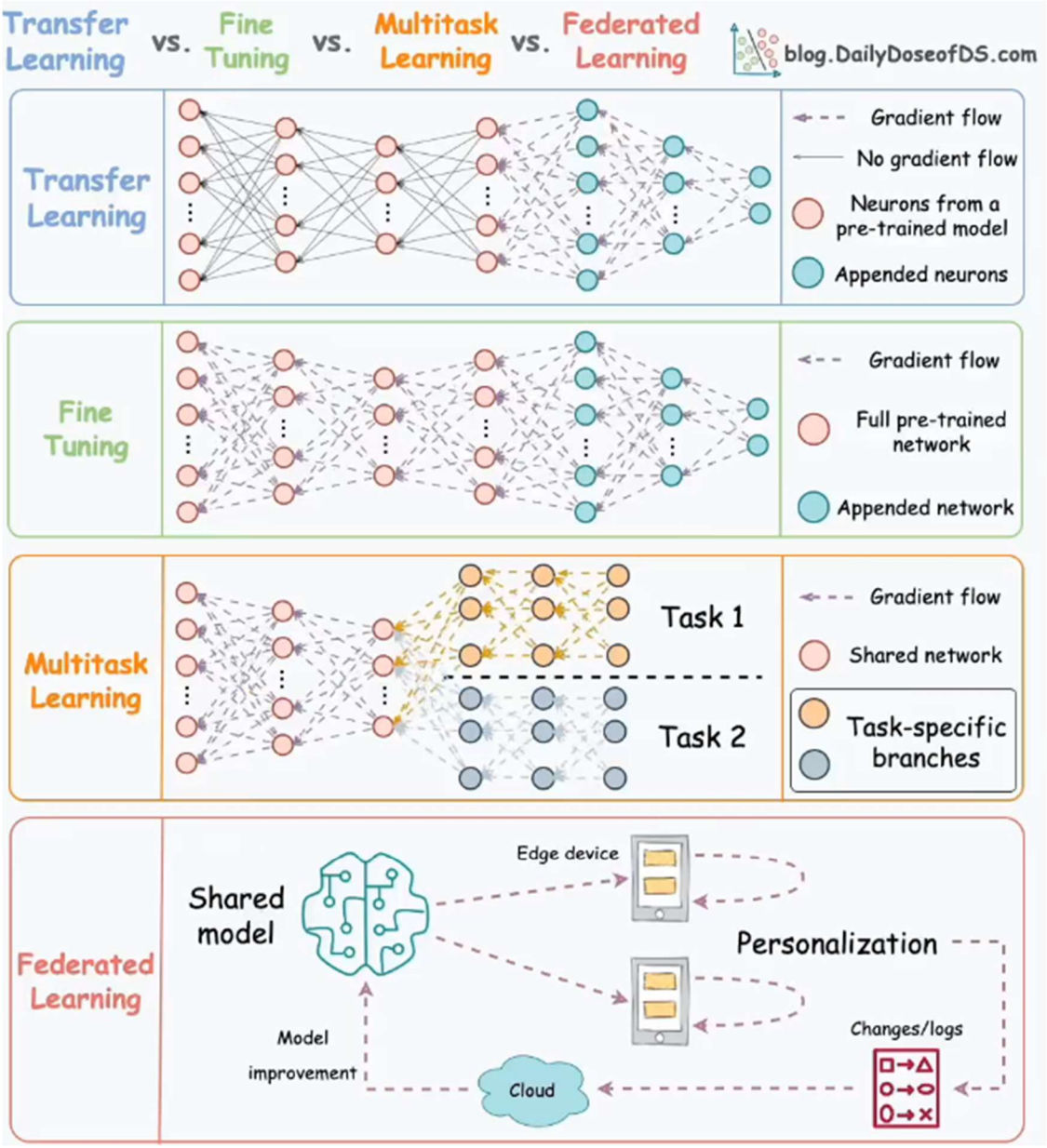
1.1.7 Catastrophic

- Hiện tượng mà mô hình quên hoặc làm mất đi khả năng học được từ dữ liệu trước đó khi đào tạo trên dữ liệu mới.

1.1.8 Federated Learning:

- Đây là cách tiếp cận phi tập trung hóa, trong đó thì training data sẽ nằm ở thiết bị người dùng (e.g. điện thoại di động, ...). Sau đó, thay vì dữ liệu được gửi tới server thì model sẽ được gửi tới thiết bị người dùng và được huấn luyện tại đó. Cuối cùng, chỉ có các update là sẽ được thu thập và gửi ngược về cho server.

- Lưu ý là cách thức này vẫn có nhiều hạn chế và hạn chế lớn nhất là vấn đề phần cứng của máy người dùng, để mỗi một model có thể chạy độc lập trên mọt thiết bị thì đòi hỏi cần có tài nguyên đủ nhiều và đủ mạnh để có thể thực hiện việc huấn luyện các mô hình (Thường là sẽ tận dụng Vram trên GPU của thiết bị).


Transfer Learning
vs.
Fine Tuning
vs.
Multitask Learning
vs.
Federated Learning
blog.DailyDoseofDS.com
Transfer Learning
Gradient flow
No gradient flow
Neurons from a pre-trained model
Appended neurons
Fine Tuning
Gradient flow
Full pre-trained network
Appended network
Multitask Learning
Task 1
Task 2
Gradient flow
Shared network
Task-specific branches
Federated Learning
Shared model
Edge device
Personalization
Changes/logs
Model improvement
Cloud

1.2 Test Production:

- Test in Production (TiP) trong machine learning là một chiến lược để thử nghiệm và triển khai mô hình trực tiếp trong môi trường sản xuất thay vì chỉ thử nghiệm trong môi trường phát triển hoặc thử nghiệm. Điều này giúp giảm khoảng cách giauwx quá trình phát triển mô hình và trển khai thực tế, đảm bảo mô hình có thể hoạt động hiệu quả trong môi trường thực tế.

- Tuy nhiên quá trình triển khai phương phái TiP có rất nhiều vấn đề cần phải được quan tâm:

- **Training-serving skew**: Dữ liệu đầu vào giữa quá trình huấn luyện và quá trình phục vụ khác nhau. Có hai loại skew quan trọng: Schema skew và Feature skew. Để giải quyết vấn đề này, bạn nên sử dụng cùng một schema để xác minh dữ liệu huấn luyện và dữ liệu phục vụ. Ngoài ra, bạn cũng nên theo dõi số lượng tính năng bị skew và tỷ lệ các ví dụ bị skew cho mỗi tính năng.
- **Model Age**: Nếu dữ liệu phục vụ thay đổi theo thời gian nhưng mô hình của bạn không được huấn luyện lại thường xuyên, thì bạn sẽ thấy sự suy giảm về chất lượng mô hình. Để giải quyết vấn đề này, bạn nên theo dõi thời gian kể từ khi mô hình được huấn luyện lại trên dữ liệu mới và đặt ngưỡng tuổi cho cảnh báo. Ngoài ra, bạn cũng nên theo dõi tuổi của mô hình trong suốt pipeline để bắt các pipeline stalls.
- **Numerically Stable Model Weights and Outputs**: Trong quá trình huấn luyện mô hình, trọng số và đầu ra của lớp của bạn không nên là NaN hoặc Inf. Để giải quyết vấn đề này, bạn nên viết các bài kiểm tra để kiểm tra các giá trị NaN và Inf của trọng số và đầu ra của lớp của bạn. Ngoài ra, bạn cũng nên kiểm tra rằng hơn một nửa các đầu ra của một lớp không phải là số không.

- **Model Performance**: Bạn nên theo dõi hiệu suất của mô hình bằng cách theo dõi các phiên bản mã, mô hình và dữ liệu. Theo dõi này cho phép bạn xác định chính xác nguyên nhân cho bất kỳ sự suy giảm hiệu suất nào. Bạn nên kiểm tra số bước huấn luyện mỗi giây cho một phiên bản mô hình mới so với phiên bản trước đó và so sánh với ngưỡng cố định. Bạn cũng nên theo dõi thời gian phản hồi API và theo dõi phân vị của chúng. Bạn nên theo dõi số lượng truy vấn được trả lời mỗi giây.