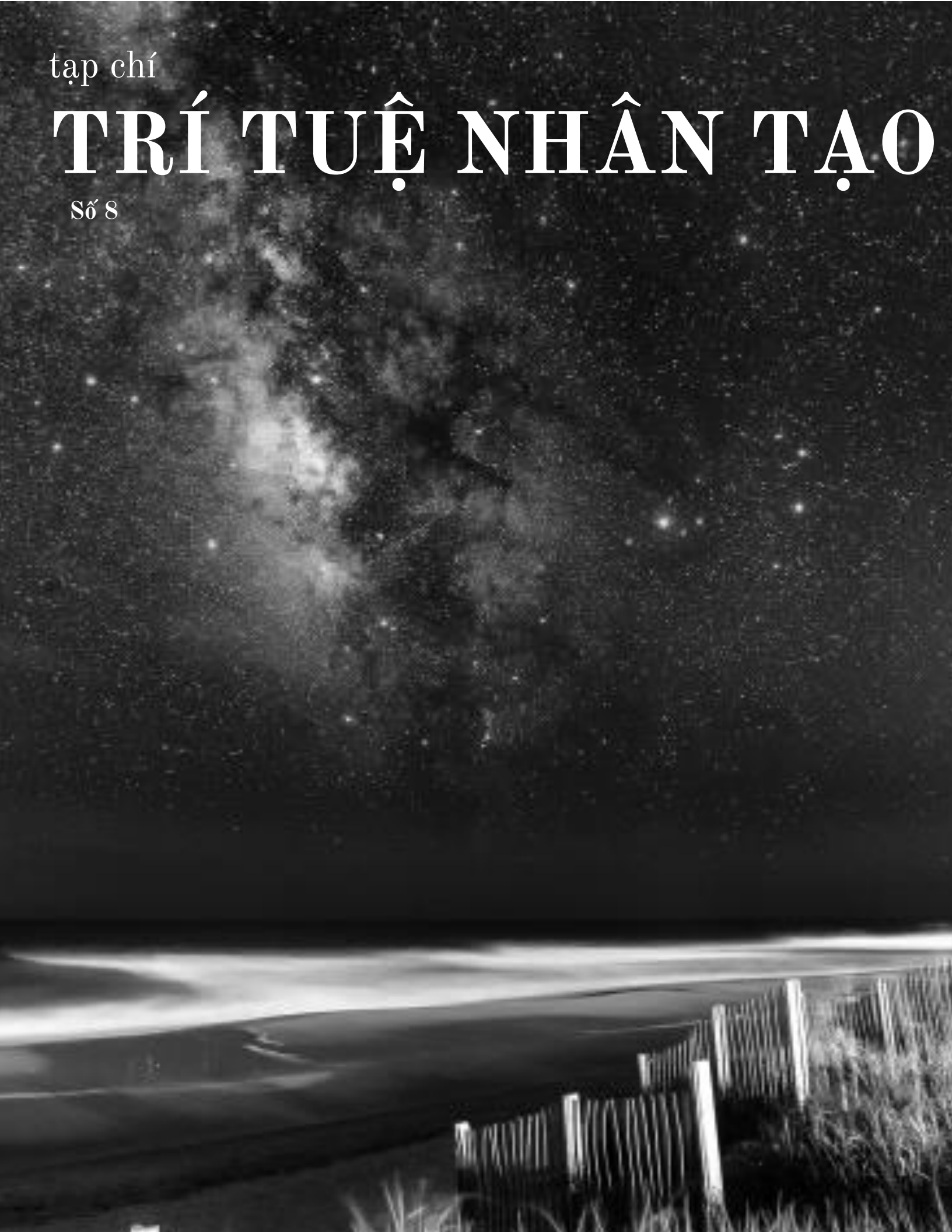


tạp chí

TRÍ TUỆ NHÂN TẠO

Số 8



Tạp chí Trí tuệ Nhân tạo

Số 8, phát hành ngày 04/04/2018

Hà Nội

Website: www.tapchiai.com

Email: tapchiai@gmail.com

Facebook: @tapchiai

Twitter: @tapchiai

Mọi đóng góp về chất lượng, nội dung tạp chí xin gửi về địa chỉ email tapchiai@gmail.com hoặc qua các mạng xã hội nêu trên.

Ban Biên Tập xin chân thành cảm ơn!!!

Contributors

Mr. Linh Nguyễn

Mr. Hà Đỗ

Ms. Lưu Bích Hồng

Mục Lục

Góc nhìn chuyên gia	2
25 năm của Machine Learning trong tài chính	
Machine Learning tutorial.....	8
Tìm hiểu sâu hơn về multi-collinearity và giới thiệu về ridge regression	
Special Article.....	17
A brief introduction on Learning To Learn.	

Góc nhìn chuyên gia

25 năm của Machine Learning trong tài chính

Dịch từ [link](#), theo bài đăng của tác giả Robert Hillman trên LinkedIn cá nhân ngày 26 tháng 10 2015. Tác giả là Giám đốc Đầu tư (CIO) tại Neuron Advisers LLP, là một nhà kinh tế và cũng là trader. Ông đang tạo ra các mô hình dựa trên các agent (agent based model) đã được hiệu chỉnh để cung cấp các hệ thống trade, dự báo và rủi ro theo thời gian thực.

Thời kỳ đầu những năm 1990 đã có rất nhiều kỳ vọng và phấn khích xung quanh tiềm năng của việc ứng dụng Machine Learning và các kỹ thuật trí tuệ nhân tạo vào ngành tài chính. Sau đây là phần giới thiệu của một cuốn sách vào năm 1993:

"Có hai sự phát triển đặc biệt quan trọng đã xảy ra vào khoảng năm 1980; cả hai đều được kích hoạt bởi sức mạnh của các máy tính cho phép ghi lại các chuỗi thời gian (time series) dài hơn, cho phép áp dụng các thuật toán phức tạp hơn, và các kết quả của các thuật toán này có thể tương tác trực quan. Sự phát triển đầu tiên, tái tạo lại state-space bằng cách nhúng thời gian trễ (time-delay embedding), dựa trên các ý tưởng từ topo vi phân (differential topology) và các hệ thống động (dynamical systems) nhằm cung cấp một kỹ thuật giúp xác định một chuỗi thời gian được tạo ra bởi các phương trình xác định (deterministic) và, nếu có, để hiểu cấu trúc hình học nằm ẩn bên dưới các hành vi quan sát được. Sự phát triển thứ hai là sự nổi lên của lĩnh vực machine learning, điển hình là neural network, có thể khám phá một cách không gian rộng lớn của các mô hình tiềm năng. Với sự chuyển đổi về hướng trí tuệ nhân tạo từ các rule – based methods sang các phương pháp dựa vào dữ liệu (data – driven methods), lĩnh vực này đã sẵn sàng để áp dụng chính nó vào time series và ngược lại chính các time series, bây giờ đã được ghi lại với số điểm dữ liệu ngày càng nhiều hơn, đã sẵn sàng để được phân tích bằng các kỹ thuật machine-learning mà đòi hỏi bộ dữ liệu tương đối lớn."

Trích dẫn này nằm trong phần giới thiệu của cuốn sách "The Future of Time Series" - Tương lai của Chuỗi thời gian" viết bởi Neil A. Gershenfeld và Andreas S. Weigend vào năm 1993, báo cáo về một cuộc thi của Viện Santa Fe để khám phá sự đa dạng của phương pháp machine learning mới và các phương pháp dự báo phi truyền thống khác đã phát triển vào cuối những năm 80 và 90. Trong thập kỷ tiếp theo, nhiều nhà nghiên cứu vẫn tiếp tục theo đuổi chương trình nghiên cứu này.

Chuyển tới năm 2015, hiện tại **cộng đồng đầu tư dường như lại trở nên rộn rã một lần nữa khi nói về machine learning và trí tuệ nhân tạo như đó là cuộc cách mạng tiếp theo trong quản lý đầu tư!** Vậy thì những chuyện gì đã xảy ra trong khoảng giữa đó, **không phải chúng ta đã làm tất cả những điều này từ 25 năm trước rồi sao?**

Vâng, những gì xảy ra đó là: trái với những hứa hẹn ban đầu, **machine learning và AI đã không trở thành một nhân tố thay đổi cuộc chơi trong thế giới tài chính và kinh tế.** Điều này không chỉ dừng lại ở lĩnh vực tài chính mà còn nhiều lĩnh vực khác như nhận dạng hình ảnh và giọng nói đã quay lại dùng những cách tiếp cận đơn giản hơn, và neural network - công cụ được đánh giá rất cao thời đó, đã bị trôi dạt ra khỏi tiềm thức của cộng đồng. Cách giải thích đơn giản nhất cho điều này, ít nhất là trong ngành tài chính, là các mạng neuron đã không mang lại những cải thiện hữu hình trong việc hiểu hoặc dự đoán. Hầu hết các nhà kinh tế đều không có chút ngạc nhiên nào và có thể được diễn giải là "Các bạn đã mong đợi điều gì chứ? Chúng tôi đã nói rằng các thị trường tương đối hiệu quả và không thể dự đoán được mà. Việc xây dựng các công cụ phức tạp hơn là hoàn toàn vô nghĩa ". Ngay cả đối với các tác vụ đơn giản hơn như nhận diện hình ảnh, người ta nhanh chóng khám phá ra rằng mặc dù các mạng neuron (và các kỹ thuật liên quan khác) có thể giải quyết được các vấn đề về lý

thuyết nhưng trên thực tế thì rất khó để huấn luyện chúng. Thay vào đó, hầu hết các mô hình và dự báo trong tài chính tiếp tục thông qua một quá trình xây dựng mô hình bởi nhà nghiên cứu, tiếp sau đó là các quá trình chủ yếu thủ công trong việc lựa chọn và ước lượng mô hình. Dù vậy, tư tưởng về lợi ích của *việc học tương đồng* (learning analogy) không hoàn toàn biến mất, và người ta có thể cho rằng việc các kỹ thuật Bayesian trở nên phổ biến trong hai thập niên tiếp đó phần nào là câu trả lời cho vấn đề này.

Machine Learning năm 2015 – có phải là một thủ thuật marketing?

Vậy có gì đang xáo trộn cả lên vào năm 2015 này? Ngay từ đầu cần phải thừa nhận rằng ngành công nghiệp đầu tư, đặc biệt là ngành tài chính định lượng, đã ngày càng trở nên có *tính hàng hoá* (commoditized) và cạnh tranh hơn trong thập kỷ qua. Khi các khoản phí (management fee –nd) giảm xuống, các nhà cung cấp dịch vụ quản lý đầu tư đã tìm cách để chứng minh rằng họ khác biệt với các đồng nghiệp của họ. Các nhà quản lý đang chia thành hai khu vực, một nhóm là những người nói rằng chúng tôi sẽ tiếp tục làm những gì mà chúng tôi đã làm và bạn có thể yên tâm rằng các nghiên cứu đang được tiến hành của chúng tôi sẽ giúp chúng tôi có thể cạnh tranh. Và nhóm thứ hai: những người tuyên bố sẽ làm một cái gì đó thật khác biệt so với đám đông. Một người hoài nghi có thể cho rằng ML và AI đang được một số người trong nhóm thứ hai này sử dụng như một thủ thuật marketing.

Các phát triển kỹ thuật

Nhưng nếu để chủ nghĩa hoài nghi này sang một bên thì có hai điều đã xảy ra. Thứ nhất, sức mạnh tính toán của máy đã tăng theo cấp số mũ. Thứ hai, không phải ai cũng đã từ bỏ machine learning và một số đã làm việc thầm lặng để cải thiện các thuật toán và kỹ thuật thực tiễn đã cho phép machine learning giải quyết một loạt các vấn đề mới. Ví dụ như Deep Learning, một trong những thuật ngữ gây sốt thời điểm này, phù hợp với câu chuyện này một cách hoàn hảo. Với Deep Learning, nhờ một số đột phá lớn trong kỹ thuật khiến cho các mạng này làm những gì chúng ta muốn, cuối cùng người ta đã bắt đầu sử dụng các deep neural network (mạng neuron nhiều lớp) mà chúng ta đã biết là rất hấp dẫn trong những năm 80 và 90. Một phần đột phá này là nhờ sức mạnh xử lý đã tăng lên theo luật của Moore. Nhưng một phần lớn trong số đó là việc con người đã học được rằng họ cần đóng vai trò lớn hơn trong quá trình đào tạo các hệ thống này. Mặc dù kiến trúc mô hình cơ bản bên dưới không thay đổi một cách triệt để, nhưng cách mà các nhà nghiên cứu tiếp cận chúng đã thay đổi khá nhiều. Vì vậy, điều này đã dẫn chúng ta đến một tình huống gần như nghịch lý khi nó đồng thời đúng: đó là các máy móc có thể làm nhiều việc hơn, và ngày càng tốt hơn so với trước đây, nhưng con người cũng tham gia sâu vào quá trình đào tạo hơn trước đây rất nhiều, và chắc là sâu hơn cả những gì mà chính họ đã từng mong đợi hoặc hy vọng. Machine learning đã hoạt động, nhưng nó thực sự được hưởng lợi từ chính con người.

Big data - Dữ liệu lớn thật sự mới mẻ và cần machine learning

Hai lời giải thích cho sự lạc quan vừa rồi đây này nghe vẫn còn giống như những gì mọi người đã nói vào những năm 90 (xem trích dẫn mở đầu), nhưng có một sự phát triển khác ủng hộ ý tưởng rằng lần này thực sự khác biệt, đó là big data.

Việc số hóa ồ ạt và tăng trưởng của Internet đã tạo ra một sự gia tăng lớn trong tính có sẵn và phổ biến của dữ liệu. Tuy nhiên, dữ liệu lớn mang lại những thách thức lớn và việc này làm tăng nhu cầu về các kỹ thuật lọc ra những thứ tốt từ những bộ dữ liệu khổng lồ không liên quan. Và đây là một nửa những gì luôn luôn xảy ra xung quanh machine learning. Một phần của machine learning là về việc cho phép các thuật toán mô hình hóa dữ liệu bằng cách tự phát triển chúng theo thời gian. Nhưng

một định nghĩa theo khía cạnh khác của machine learning là khả năng cho các máy móc khám phá các mối quan hệ và mối liên hệ trong các bộ dữ liệu có thể đã bị các nhà nghiên cứu che giấu hoặc thậm chí chưa được biết đến (thường được biết đến như là "unsupervised learning - học không giám sát"). Khía cạnh này của machine learning cần rất nhiều dữ liệu theo định nghĩa, bởi vì toàn bộ ý tưởng là máy tính chỉ có dữ liệu mà thôi. Vì vậy, sự tăng trưởng về big data và các ứng dụng có nguồn gốc từ nó (như chẩn đoán y tế, nghiên cứu di truyền, mô hình mua sắm qua mạng) đang kích thích những cải tiến trong machine learning.

Tính phản ánh (reflexivity) của thị trường hạn chế các cơ hội kiếm lời

Vậy đâu là điều dẫn chúng ta đến machine learning và tài chính? Nhiều lý do cho sự thất vọng trước đó đã không biến mất, và những người ủng hộ cho machine learning trong quản lý đầu tư hiện nay chắc chắn sẽ cố gắng nghiên cứu kỹ càng lịch sử vài thập kỷ qua trước khi rơi vào cái bẫy đã biết. Điểm chính là chúng ta có những lý do đáng kể để tin rằng, trong hầu hết thời gian, thị trường gần như khá hiệu quả, ít nhất theo nghĩa thống kê, và sẽ cực kỳ khó khăn để tạo ra các lợi nhuận không rủi ro (consistent risk-free profits) bằng cách dự báo xu hướng giá cả. **Đó là bởi vì việc khai thác thông tin trong dự báo (trading) của nhà nghiên cứu / trader sẽ ảnh hưởng đến giá trong tương lai.** Tính phản ánh của thị trường làm cho vấn đề học trong đầu tư về cơ bản là khác với việc học để xác định một chiếc xe đạp hay xe máy trong việc nhận dạng hình ảnh.

Nhiều dữ liệu hơn không nhất thiết là nhiều thông tin hay cơ hội hơn

Một vấn đề thứ hai đối với các ứng dụng trong lĩnh vực tài chính là việc các dữ liệu lịch sử mà việc học của máy móc có thể dựa vào là cực kỳ hạn chế. Mặc dù người ta có thể, và khá phổ biến hiện nay, tạo các mẫu con (sub-sample) với lịch sử dữ liệu chi tiết đến micro giây; việc này tạo ra những thứ trông giống như chuỗi dữ liệu thời gian không giới hạn (unlimited time-series of data). Nhưng cần phải rõ ràng rằng điều này không nhất thiết tăng được khả năng phát hiện các features hoặc tạo ra các cơ hội trading có thể được khai thác theo thời gian mà vẫn hữu ích cho chúng ta. Một người sử dụng machine learning để xác định mô hình hành vi của người dùng website cho phép họ dự đoán được nơi người dùng có nhiều khả năng nhấp chuột tiếp theo có thể sẽ kiếm được tiền từ hàng triệu người dùng khác gần như ngay lập tức và liên tục. Các cơ hội lợi nhuận liên ngành (cross-sectional) và các cơ hội có sự lặp lại về lợi nhuận trong các ứng dụng này là khổng lồ. Nhưng hầu hết các cơ hội lợi nhuận trong đầu tư đòi hỏi phải có thời gian. Trading có chi phí ban đầu và các giá cũng cần phải di chuyển đủ một khoảng cách nhất định trước khi có thể kiếm được lợi nhuận.

Đầu tư cũng bị giới hạn bởi vì cơ hội liên ngành là khá nhỏ. Ví dụ, bất cứ ai muốn áp dụng machine learning vào các giao dịch futures phải nhận thấy rằng trong bất kỳ khoảng thời gian nào có một tập hợp các danh mục hữu hạn (giới hạn bởi các sản phẩm) có thể được xây dựng và trong thực tế, người quản lý chỉ có thể giữ một danh mục tại một thời điểm (thậm chí các nhà quản lý đa chiến lược (multi-strategy) có thể *nắm giữ một danh mục đầu tư các vị thế duy nhất* (a single net portfolio of positions) tại bất kỳ thời điểm nào). Hơn nữa, hiện nay người ta ngày càng nhận ra rằng trong bất kỳ khoảng thời gian nào, chỉ có một vài yếu tố rủi ro rõ rệt nằm ẩn dưới sự thay đổi của giá cả tài sản, thì việc chúng bị chớp lấy cùng lúc cũng có nghĩa là cơ hội hiệu quả thậm chí còn bị giới hạn hơn rất nhiều các cơ hội mà tổ hợp chiến thuật khác có thể mang lại.

Vì vậy, sự phản ánh nội tại của thị trường, lịch sử dữ liệu ngắn và cơ hội giới hạn tạo ra những trở ngại lớn hạn chế khả năng của machine learning trong việc tạo ra những cải thiện rõ nét so với các kỹ thuật khác. Nhưng có một lĩnh vực trong đó machine learning có thể đóng vai trò rộng hơn trong việc nâng cao hiểu biết của chúng ta về thị trường và hành vi của nhà đầu tư, và đó là giao diện giữa

machine learning và nghiên cứu về cách các cá nhân học cách thu thập, xử lý và hành động dựa trên thông tin.

Agent based economic model trong những tiên đoán năm 1990 về thị trường ngày nay

Khá im lặng nhưng rõ ràng đã có những sự thay đổi mô hình diễn ra trong ngành kinh tế trong suốt 25 năm qua. Ngày nay việc nghiên cứu làm thế nào các cá nhân hình thành kỳ vọng của họ và đưa ra quyết định là hoàn toàn chấp nhận được. Có vẻ kỳ quặc rằng điều này không bao giờ xảy ra, nhưng trong nhiều năm, đối với các vấn đề lý luận lý thuyết và thuận tiện về toán học, thực tế là các agent hành động theo các cách không phù hợp với *lý thuyết tối ưu hóa lựa chọn hợp lý* (rational utility maximising choice theory) phần lớn bị bỏ qua. Nhưng ngày nay việc học chính nó đã trở thành một phần của chương trình nghiên cứu về kinh tế học chủ đạo, và điều này đang diễn ra ở trong cả nghiên cứu lý thuyết và nghiên cứu thực nghiệm. Nghiên cứu các thuật toán rõ ràng là cũng liên quan chặt chẽ.

Liên quan đến nghiên cứu về việc học là một chùm các nghiên cứu dựa trên việc xây dựng các mô hình agent nhân tạo (trong máy tính) cung cấp một phương pháp thay thế để nghiên cứu các hiện tượng kinh tế. Các nhà khoa học đã mô phỏng sự tương tác của nhiều trader trong thị trường chứng khoán nhân tạo để có được cái nhìn sâu sắc về cách mà biến động giá như bong bóng và sụp đổ thị trường (crashes) có thể được tạo ra. Các "traders" trong các hệ thống này thực sự chỉ là các thuật toán, và trong các phương pháp tiếp cận tiên tiến hơn, các thuật toán học cách trade theo thời gian. Nói cách khác, một số nhà nghiên cứu đang sử dụng các mô hình agent nhân tạo mà đến lượt nó lại dựa vào machine learning như một phương tiện để hiểu sâu hơn về các phương pháp và hiệu quả của việc học của con người. Điều thú vị ngày hôm nay là với việc tăng cường sử dụng các thuật toán và đầu tư dựa trên luật (rules-based investing) ngày càng tăng, thế giới thực đã tiến đến gần giống với các mô hình agent nhân tạo của cách đây 25 năm. Giống như khoa học viễn tưởng đôi khi dự đoán tương lai, những mô hình kinh tế hư cấu từ những năm 80 và 90 là những dự đoán tốt về các thị trường thuật toán ngày nay. Giống như chính machine learning, những ý tưởng này đã tồn tại được khoảng một thời gian dài, nhưng cũng giống như machine learning, lĩnh vực này có thể là để tìm một phiên bản mới của thế giới, như các nhà nghiên cứu đã khai thác dữ liệu lớn để hiệu chỉnh các mô hình nhân tạo này đang đưa chúng đến gần thế giới thực.

Tạo các kịch bản rủi ro trong các môi trường chiến lược

Một ứng dụng của những kỹ thuật này có thể ảnh hưởng đến thế giới quản lý đầu tư là việc đánh giá rủi ro. Một cách tiêu chuẩn để vượt qua trở ngại về lịch sử dữ liệu ngắn trong tài chính là tạo ra các dữ liệu nhân tạo để xây dựng và đánh giá các chiến lược đầu tư. Nguyên tắc này đã được thiết lập từ lâu, và tính toán VaR (value-at-risk, một thước đo rủi ro trong tài chính - nd) và stress-test (một phương pháp quản trị rủi ro - nd) thường dựa vào ý tưởng cơ bản này. Người ta có thể tạo các kịch bản rủi ro trong tương lai bằng cách ngẫu nhiên nối các mẫu ngắn của các dữ liệu trong quá khứ vào với nhau. Người ta có thể hỏi vậy thì cái gì có thể là những rủi ro đối với chiến lược của tôi nếu cuộc khủng hoảng LTCM (long term capital management) vào năm 1998 kéo theo ngay sau đó là một giai đoạn sụt giảm bất ngờ năm 2008 mặc dù sự kết hợp các sự kiện này không bao giờ xảy ra trong lịch sử.

Nhưng việc tạo ra các kịch bản từ cùng một lượng dữ liệu lịch sử hạn chế rõ ràng là rất hạn chế, do đó, một cách tiếp cận khác là xây dựng các mô hình nhân tạo nhân tạo có thể tạo ra các dữ liệu giả tạo, nhưng thực tế có thể hy vọng bù đắp cho việc thiếu lịch sử. Theo một nghĩa nào đó người ta có thể thấy đây là một cách tiếp cận mô phỏng chuyến bay (flight-simulator). Trên thực tế, phi công không

thể (và cũng không muốn) gặp tình huống hạ cánh xuống đất nhiều lần, nhưng trong một mô phỏng máy tính, họ có thể đối mặt với những rủi ro và học cách ứng xử mà không thực sự đe dọa cuộc sống của bất kỳ ai. Một chuyến bay mô phỏng có thể tạo ra những rủi ro mà có thể đã không bao giờ thực sự xảy ra, ví dụ như bay vào đám mây bụi núi lửa ở giữa một cơn bão sắp tới. Mô hình rủi ro nhân tạo có thể tạo ra kịch bản này mặc dù nó chưa bao giờ xảy ra trong lịch sử.

Trên thực tế, **chính xác là khả năng sử dụng các hệ thống dựa trên machine learning để tạo ra dữ liệu chứ không chỉ phân tích nó**, đã thúc đẩy một số tiến bộ công nghệ then chốt đang củng cố những tiến bộ trong deep learning, thứ đang tạo ra những cường điệu trong tài chính ngày nay. Một trong những chuyên gia hàng đầu thế giới về machine learning Geoffrey Hinton đã phát triển ý tưởng này và tóm lược nó ngắn gọn trong tiêu đề của một bài báo: "Để nhận diện hình dạng, trước tiên hãy học cách tạo ra hình ảnh" (Hinton, 2007).

Các bài học từ việc machine learning có thể khiến cho những mô phỏng rủi ro trở nên sống động hơn, đặc biệt trong lĩnh vực kinh tế, nơi mà hành động của các agent tác động trực tiếp đến môi trường mà họ đang tham gia vào. Trong những trường hợp này, tốt nhất nên nghĩ đến việc học cá nhân (và hành động) trong môi trường bao gồm nhiều cá nhân khác cũng học về môi trường. Nghĩ về cuộc thi sắc đẹp nổi tiếng của Keynes¹, nhưng trong một bối cảnh năng động. Đầu vào quan trọng từ machine learning là chiến lược trading có thể được kiểm tra theo kịch bản phản ánh chiến lược, trong đó kịch bản rủi ro sẽ phát triển nội sinh để đáp ứng chiến lược trading. Các nhà nghiên cứu và các nhà quản lý đang sử dụng các kỹ thuật agent nhân tạo này để nghiên cứu các thuật toán high frequency trading có thể là nguồn gây mất ổn định trong các thị trường như thế nào trong sách của Hayes và cộng sự (2012).

Sự cân bằng giữa các thuật toán và sự giám sát của con người - một xung đột nội tại

Tuy là dù có thể hoặc không thể cải thiện hiệu suất cho các nhà đầu tư thông qua việc sử dụng machine learning, thì chắc chắn sẽ có những thách thức ngày càng tăng đối với các nhà lập pháp, nhà quản lý và nhà đầu tư. Lấy ví dụ vấn đề gai góc là cái gì tạo nên một "lỗi giao dịch". Các nhà đầu tư ngày nay thường cố gắng để đảm bảo các nhà quản lý bồi thường cho họ về bất kỳ thiệt hại gây ra bởi lỗi. Nhưng lỗi là gì khi hệ thống trading đang theo một AI lấy cảm hứng từ cách tiếp cận machine learning? Các giao dịch thua lỗ sẽ được thực hiện. Hệ thống có thể hành xử theo cách mà người quản lý và chủ đầu tư không lường trước ngay từ ban đầu. Máy này có thể không chỉ trade với tốc độ nhanh hơn con người có thể giám sát (như các hệ thống hiện tại đã làm), mà còn tạo ra *các thuật toán* mới với tốc độ nhanh hơn con người có thể giám sát. Tất cả những khả năng này có thể được lập luận để xác định chính xác những lợi ích chính của phương pháp tiếp cận machine learning, do đó có lẽ sẽ có khó khăn nội tại trong việc cân bằng giữa machine learning và sự giám sát của con người.

Hệ thống điều khiển bằng máy cũng cần phải nhận thức được các hạn chế về quy định và làm việc với chúng. Trong các lĩnh vực khác, người ta nghĩ rằng nó có thể chấp nhận được bằng cách "nhúng ngón chân vào nước", nghĩa đen là khi người ta cố gắng tìm hiểu nhiệt độ nước. Nhưng cái gì là một hành động tương đương với một thuật toán trading bằng machine learning? Một thuật toán khám phá có thể muốn gửi lệnh cho thị trường để kiểm tra tính thanh khoản hoặc khả năng phục hồi của thị trường. Sự phản ứng của thị trường có thể khiến nó hủy bỏ lệnh. Có thể có việc algo không có ý định cam kết trade đủ số tiền được đăng, giống như người bơi không cam kết sẽ lặn xuống khi họ nhúng

¹ John Maynard Keynes – nhà kinh tế học nổi tiếng người Anh, người cho rằng thị trường không hoàn hảo và ủng hộ sự can thiệp của chính phủ vào thị trường

ngón chân của họ vào trong nước. Một khi thị trường đã trả lời, các algo có thể phát hiện ra cơ hội giao dịch ở phía bên kia của thị trường. Thế thì đó có phải là giả mạo không?

Và làm thế nào máy móc phản ánh được một thứ tổng quát hơn như hành vi đạo đức làm nền tảng cho các khung quy định trên toàn thế giới? Các máy tính theo lý thuyết là tuyệt vời khi tìm kiếm các giải pháp bằng cách phát triển các quy tắc hành vi trong một tập hợp các quy tắc đặt trước hoặc một khuôn khổ được xác định rõ ràng - gần như bằng cách xây dựng chính xác những gì ML được thiết kế để làm. Nhưng nếu các giải pháp được tìm thấy được cho là quá con người nằm trong các quy tắc nhưng trái với tinh thần của các quy tắc? Những tình huống này nảy sinh trong thế giới thực, và con người có thể hiểu được sự khác biệt giữa hai thứ đó, mặc dù đôi khi cần có một tòa án để giúp họ đi cùng. Có những thách thức thực sự ở đây, và điều có thể dự đoán được là những vấn đề này chắc chắn sẽ là nguyên nhân gây ra một số sự chú ý giữa các nhà quản lý rủi ro, phía compliance và phía regulators, theo Paddrik (2013). Sự phức tạp tính toán của máy tính cũng không giúp ích gì cho tình hình, và do đó ý thức của tôi là một số giao thức và các nguyên tắc rộng lớn sẽ cần được xác định khá nhanh để giải quyết vấn đề sắp tới.

Nói tóm lại, mặc dù trong một số trường hợp, việc chúng ta đã đến được đây trước khi có một số yếu tố mới có nghĩa là một số vấn đề nóng đã được đảm bảo. Một phần là do những thách thức và cơ hội tạo ra bởi các dữ liệu lớn. Nhưng tôi nghĩ là tác động cuối cùng sâu sắc hơn sẽ đến từ một tiến bộ chung hơn trong hiểu biết của chúng ta về hành vi cá nhân thông qua việc sử dụng các phương pháp tiếp cận mang tính tính toán hơn để nghiên cứu, trong đó machine learning sẽ đóng một vai trò quan trọng. Chúng ta sẽ bị buộc phải xem xét các vấn đề pháp lý, đạo đức và thậm chí cả triết học theo cách đó.

Tài liệu tham khảo

1. Neil A. Gershenfeld and Andreas S. Weigend, 'The Future of Time Series' (1993), <http://www.santafe.edu/media/workingpapers/93-08-053.pdf>
2. Hinton, G. E., 'To recognize shapes, first learn to generate images', (2007) in P. Cisek, T. Drew and J. Kalaska (Eds.) Computational Neuroscience: Theoretical Insights into Brain Function. Elsevier. <http://www.cs.toronto.edu/~hinton/>
3. Hayes, Roy and Paddrik, Mark E. and Todd, Andrew and Yang, Steve Y. and Beling, Peter and Scherer, William, Agent Based Model of the E-Mini S&P 500 Future: Application for Policy Making (2012). Proceedings of the 2012 Winter Simulation Conference. Available at SSRN:<http://ssrn.com/abstract=2112139> or <http://dx.doi.org/10.2139/ssrn.2112139>
4. Paddrik, Mark E., Assessing Financial Markets Through System Complexity Management (December 2013). Available at SSRN:<http://ssrn.com/abstract=2400998> or <http://dx.doi.org/10.2139/ssrn.2400998>

Machine Learning tutorial

Tìm hiểu sâu hơn về multi-collinearity và giới thiệu về ridge regression

Trong số trước, chúng ta đã làm quen với một vấn đề khá nghiêm trọng của mô hình regression: vấn đề multi-collinearity. Nguyên nhân và hậu quả của vấn đề này tới mô hình regression cũng đã được chỉ rõ: *Khi có hiện tượng phụ thuộc tuyến tính (hoàn hảo hoặc không hoàn hảo) giữa các cột của ma trận input, mô hình regression hoàn toàn không ổn định và overfit toàn bộ vào dữ liệu mà nó học được.*

Trong số này, chúng ta sẽ làm rõ hơn cơ chế của hiện tượng này và giới thiệu một vài phương pháp khắc phục cổ điển.

Tới giờ chắc hẳn bạn đọc đã rất quen thuộc với công thức của β của regression:²

$$\hat{\beta} = (X^T X)^{-1} X^T Y \quad (1)$$

hay

$$\hat{\beta} = I X^T Y$$

với $I = (X^T X)^{-1}$, information matrix, nghịch đảo của ma trận correlation Σ ,

Từ công thức này chúng ta thấy, khi xuất hiện 1 sự biến động nhỏ (perturbation) trong dữ liệu (input X hoặc output Y):

- Với perturbation trong X:

$$\hat{\beta}' = I(X + \epsilon)^T Y = I X^T Y + I \epsilon^T Y = \hat{\beta} + I(\Delta X)^T Y$$

$$\text{Hay } \Delta\beta = \hat{\beta}' - \hat{\beta} = I(\Delta X)^T Y$$

- Tương tự với perturbation trong Y:

$$\hat{\beta}' = I X^T (Y + \epsilon) = I X^T Y + I X^T \epsilon = \hat{\beta} + I X^T \epsilon$$

$$\text{Hay } \Delta\beta = \hat{\beta}' - \hat{\beta} = I X^T \Delta Y$$

, biến động này sẽ được khuếch đại thông qua ma trận I để trở thành biến động trong hệ số của mô hình.

Chúng ta có thể thấy, ma trận I ở đây đóng vai trò như một tác nhân **khuếch đại sai số** hệ số β

Các sai số xuất hiện trong input X hay output Y được hấp thụ vào hệ số β . Điều này là đặc biệt nghiêm trọng khi sự khuếch đại này là lớn như trong trường hợp multi-collinearity.

Như trong ví dụ mở đầu ở số trước, khi input (cân nặng) và output (chiều cao) biến động tương ứng $\pm 1.5\%$ và $\pm 0.5\%$, hệ số β bị biến động lần lượt là **57.6%, -38.8% và -132.5%**.

² Theo kinh nghiệm của tác giả, câu hỏi về công thức của β là một trong những câu hay được hỏi nhất (cùng với các vấn đề liên quan như multi-collinearity) đối với các sinh viên mới ra trường cho các vị trí data scientist, data analyst, quantitative analyst, etc.

Trong số trước chúng ta đã đo lường sự khuếch đại này bằng cách tính toán khoảng cách từ $\hat{\beta}$ tới β : $L_1^2 = (\hat{\beta} - \beta)^T (\hat{\beta} - \beta)$. Chúng ta có $E(L_1^2) = \sigma^2 \text{trace}(I) = \sigma^2 \sum_{j=1}^p 1/\lambda_j$, với λ_j là các eigenvalue của Σ .

Từ ý nghĩa hình học của phép nhân ma trận (hộp 1) chúng ta thấy, nếu tất cả các eigenvalue của ma trận I là nhỏ, tương ứng với eigenvalue của ma trận Σ là lớn thì sự khuếch đại sai số là nhỏ. Ngược lại, nếu tồn tại một eigenvalue của ma trận I rất lớn, **eigenvalue của ma trận Σ tương ứng rất nhỏ, chúng ta có hiện tượng khuếch đại sai số nghiêm trọng**. Lúc này mô hình regression sẽ hấp thụ sai số trong dữ liệu và **sẽ overfit vào bộ dữ liệu này**.

Ký hiệu $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p \geq 0$ ³ các eigenvalue của ma trận correlation Σ , chúng ta định nghĩa $\kappa(\Sigma) = \frac{|\lambda_1|}{|\lambda_p|} = \frac{\lambda_1}{\lambda_p}$, condition number của ma trận Σ . Lúc này chúng ta thấy, nếu $\lambda_p \rightarrow 0$, tương ứng với $\kappa(\Sigma) = \kappa(I) \rightarrow \infty$, hiện tượng overfit vào sai số dữ liệu sẽ xảy ra. Lúc này chúng ta nói ma trận Σ và ma trận I là ill-conditioned⁴.

Từ đây chúng ta thấy tầm quan trọng của ma trận correlation của input cũng như các eigenvalue của nó. Nhưng điều này thật sự ám chỉ điều gì? Hay nói cách khác, trong trường hợp nào thì λ_p là nhỏ đến mức nguy hiểm và mối quan hệ của nó với multi-collinearity là gì? Từ đó, liệu có thể suy ra các cách giải quyết có thể áp dụng với trường hợp này?

Hộp 1: Nhắc lại về trị riêng và vector riêng của một ma trận

Trị riêng (Eigenvalue): số λ được gọi là eigenvalue của ma trận A khi và chỉ khi ma trận $A - \lambda I$ là suy biến:

$$\det(A - \lambda I) = 0, \text{ với } I \text{ là ma trận đơn vị.}$$

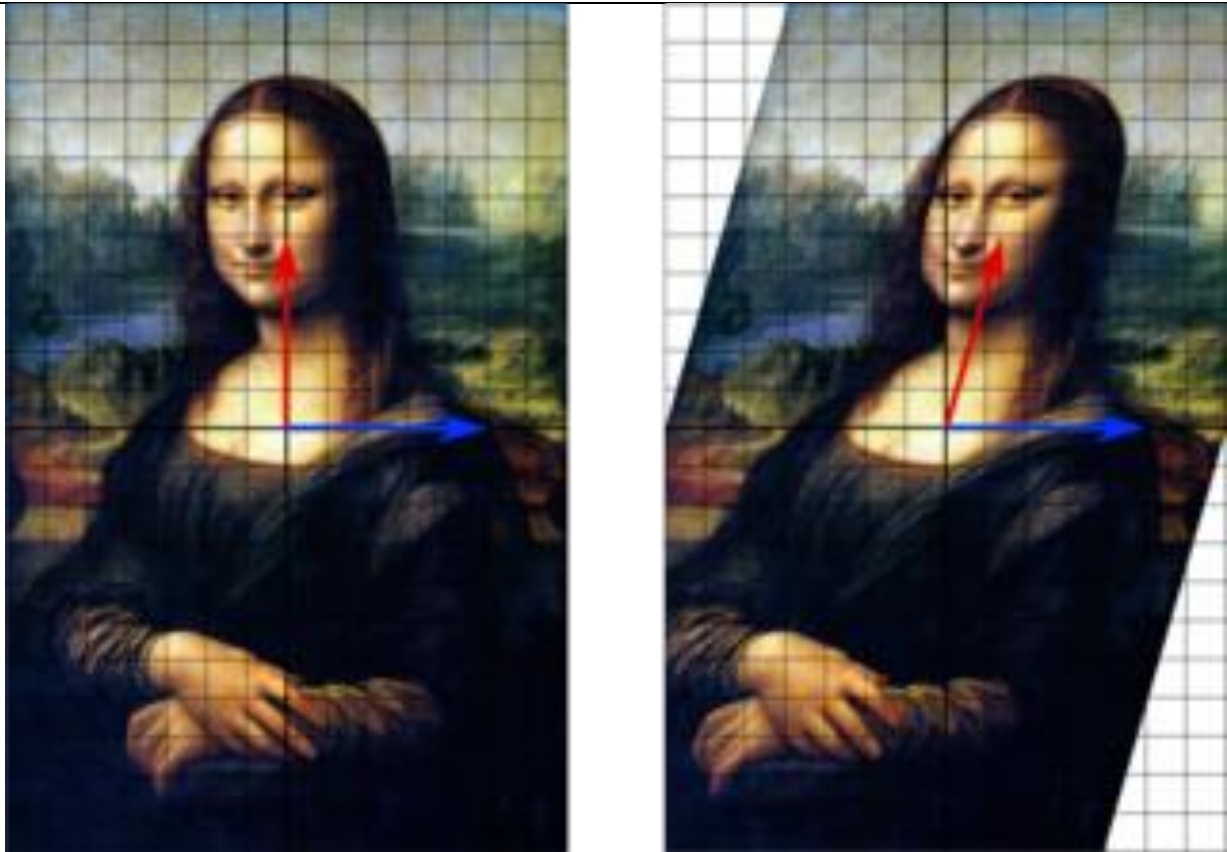
Vector riêng (Eigenvector): vector x được gọi là eigenvector tương ứng của eigenvalue λ khi và chỉ khi:

$$Ax = \lambda x$$

Như chúng ta đã biết, ý nghĩa hình học của phép nhân ma trận là một phép biến hình. Từ 2 định nghĩa trên chúng ta thấy: Eigenvector của một ma trận không bị biến đổi phương sau phép biến hình. Hay nói cách khác, nó đặc trưng cho phương của phép biến hình.

³ Do ma trận Σ là positive semi-definive (nửa xác định dương) nên tất cả eigenvalue của nó đều không âm

⁴ Một số rule of thumb nói rằng khi condition number này lớn hơn 30 chúng ta hiện tượng multi-collinearity trong input. Đôi khi threshold này cũng được khuyến nghị ở mức thấp hơn như 15 hay 20.



Hình 1: Phép nhân ma trận và phép biến hình⁵

Như trong hình trên, chúng ta thấy vector màu xanh không bị đổi phương (cũng như chiều và độ lớn), do vậy nó là eigenvector của phép chiếu. Phép chiếu tương ứng tịnh tiến mọi điểm ảnh sang phải.

Eigenvalue cho chúng ta biết chiều và độ lớn của phép biến hình. $\lambda = 1$ ứng với phép biến hình giữ nguyên đơn vị ban đầu (như trong hình minh họa). $\lambda = -2$ ứng với phép biến hình đảo ngược chiều tăng gấp đôi đơn vị ban đầu.

Chúng tôi mời bạn đọc xem thêm minh họa tương tác về eigenvalue và eigenvector ở [đây](#)

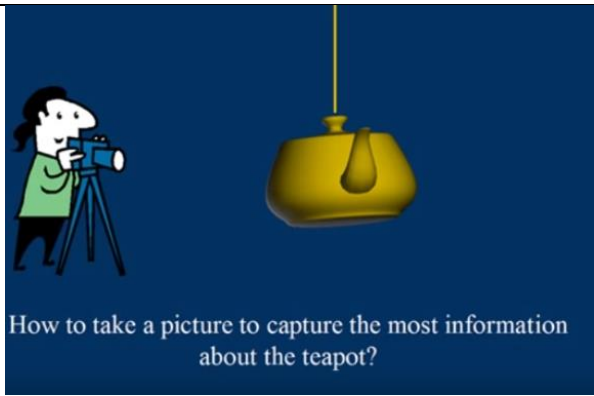
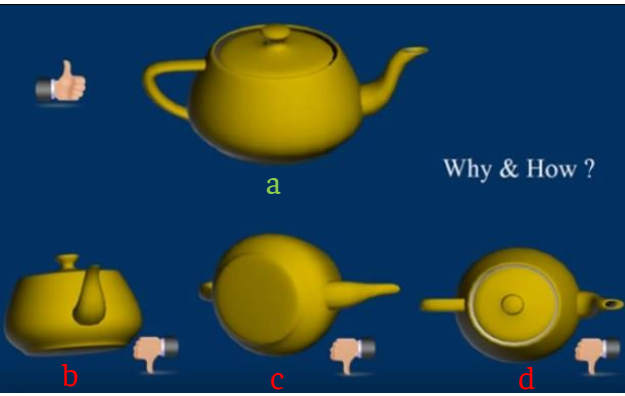
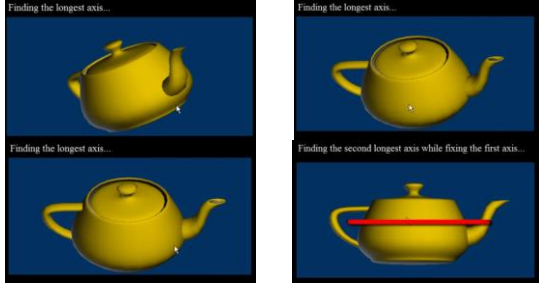
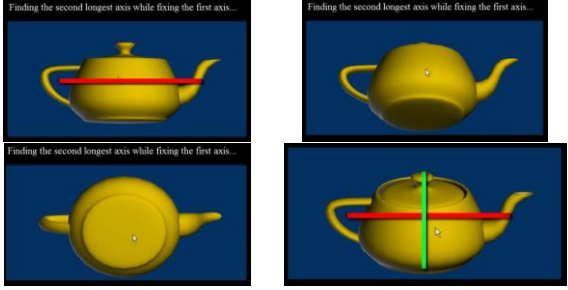
Ngoài ra chúng ta có :

- $\det(A) = \prod \lambda_i$, $\text{trace}(A) = \sum \lambda_i = \sum A_{jj}$. Tích của các eigenvalue chính là định thức của ma trận. Tổng của các eigenvalue chính là tổng đường chéo của ma trận hay vết của ma trận.
- $\text{eigenvalue}(A^{-1}) = \frac{1}{\text{eigenvalue}(A)}$, eigenvalue của ma trận nghịch đảo bằng nghịch đảo của eigenvalue.

⁵ https://en.wikipedia.org/wiki/Eigenvalues_and_eigenvectors

PCA và linear regression, một thuật toán unsupervised để hiểu rõ hơn một thuật toán supervised⁶

Một ví dụ mở đầu:⁷

 <p>Hình 2: Ví dụ minh họa - Đặt vấn đề</p> <p>Giả sử chúng ta muốn chụp hình một ấm trà (vật thể trong không gian 3 chiều). Vậy làm sao chúng ta có thể lưu giữ được nhiều nhất thông tin lên một bức ảnh (không gian 2 chiều)</p>	 <p>Hình 3: Ví dụ minh họa - Đâu mới là giải pháp và tại sao?</p> <p>Vậy đâu mới là hình chúng ta nên chụp? Rõ ràng hình a cho chúng ta thấy nhiều chi tiết hình ảnh nhất về ấm trà từ đó giúp chúng ta dễ dàng nhận ra được vật thể. Các hình còn lại chứa ít chi tiết về ấm trà hơn, do vậy không tốt bằng. Vì sao vậy?</p>
 <p>Hình 4: Ví dụ minh họa - Đi tìm giải pháp: đi tìm trục thông tin lớn nhất⁸</p> <p>Trước hết, để tìm ra được hình a, chúng ta sẽ xoay vật thể để tìm ra góc chụp giữ được cạnh dài nhất của vật thể. Vì là trục dài nhất nên trục này sẽ giữ lại được nhiều thông tin nhất về vật thể.</p>	 <p>Hình 5: Ví dụ minh họa - đi tìm giải pháp: đi tìm trục thông tin thứ 2</p> <p>Sau đó, giữ nguyên trục đã tìm được, chúng ta xoay tròn vật thể xung quanh trục này cho tới khi tìm được góc chụp giữa được cạnh dài tiếp theo của vật thể. Từ đó chúng ta tìm ra hình a.</p>

Hai trục tìm được ở trên gọi là thành phần chính thứ nhất và thứ hai (first and second principle component)⁹.

Dưới đây là hình minh họa của 2 trục chính (principle component - PC) cho không gian 2

⁶ Chúng tôi xin phép được giới thiệu ngắn gọn ý nghĩa và cố gắng tránh mọi vấn đề toán học liên quan tới kỹ thuật PCA.

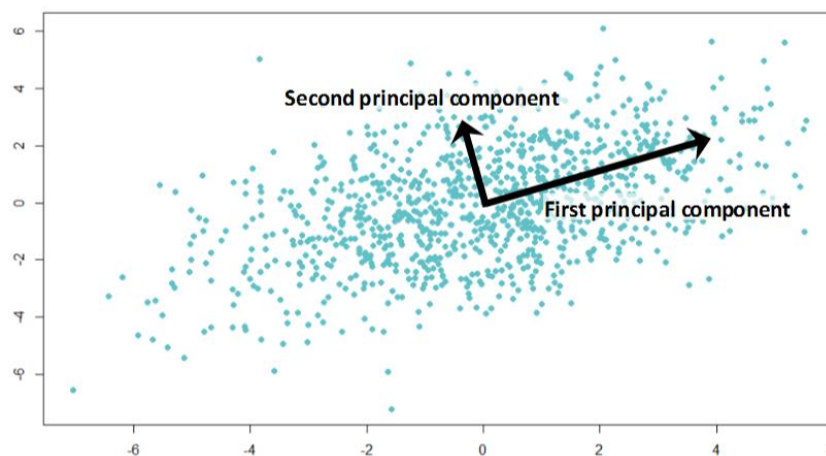
⁷ Chúng tôi mời bạn đọc theo dõi video ở [đây](#)

⁸ Xem từ trái qua phải và từ trên xuống dưới.

⁹ Xin chú ý rằng vì chiều không gian của dữ liệu là 3 nên chúng ta vẫn còn thành phần chính thứ 3 (vuông góc với và chứa ít thông tin hơn 2 thành phần chính minh họa ở trên). Tuy nhiên do không gian của bức ảnh chụp là 2 chiều nên chúng ta không thể hiện được thành phần này lên bức ảnh. Do vậy chúng ta ưu tiên giữ lại 2 thành phần chính chứa nhiều thông tin hơn.

chiều. Phương của 2 trục này vuông góc với nhau và đồng nhất với phương giải thích được nhiều nhất hình dáng của đám mây dữ liệu. Chiều dài của 2 vector chính là 2 eigenvalue của ma trận covariance matrix. Do vậy chúng ta thấy, eigenvalue đo lường thông tin về sự biến thiên chung (covariance) của đám mây dữ liệu.

Phương pháp để tìm ra các trục này gọi là phương pháp phân tích thành phần chính (Principal Component Analysis – PCA).



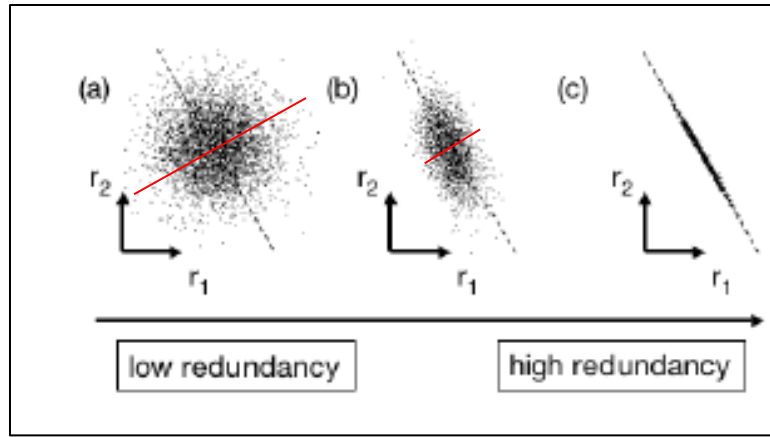
Hình 6: PCA với không gian 2 chiều

Quay trở lại với đám mây dữ liệu input X với p chiều¹⁰. Nếu ma trận correlation Σ của X không suy biến, lúc này chúng ta sẽ có p eigenvalue của ma trận Σ . Ký hiệu chúng là $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p > 0$, tương tự như trên chúng ta có như sau: λ_j ứng với trục dài thứ j của đám mây dữ liệu và đại diện cho thông tin giữ lại được dọc theo trục này.

Do vậy, các eigenvalue có giá trị nhỏ hơn sẽ tương ứng với ít thông tin hơn được chứa đựng dọc theo chiều biến thiên này của dữ liệu. Hình 7 dưới đây minh họa cụ thể điều này: với hình 7a, chúng ta có eigenvalue λ_2 tương đối lớn, do vậy thông tin chứa đựng theo trục PC2 là tương đối quan trọng. Ở hình 7b, λ_2 nhỏ hơn tương đối so với λ_1 , thông tin chứa đựng ở chiều PC2 này kém quan trọng tương đối so với PC1. Ở hình 7c, $\lambda_2 \approx 0$, **đám mây dữ liệu hầu như được fit bởi một đường thẳng**, chúng ta có **hiện tượng near collinearity khi X_2 có thể được mô tả bởi một đường thẳng tuyến tính của X_1** . Lúc này, chúng ta có thể coi thông tin được chứa đựng ở PC2 chỉ đơn thuần là các nhiễu (noise). Chúng ta cũng nói đám mây dữ liệu 7c chứa các thông tin dư thừa do X_2 có thể được xem như là clone của X_1 (hoặc ngược lại). Do vậy chúng ta chỉ cần sử dụng 1 trong 2 biến là chúng ta đã có thông tin của toàn bộ đám mây dữ liệu này. Trong trường hợp cực trị, khi $\lambda_2 = 0$, chúng ta có perfect collinearity: toàn bộ dữ liệu nằm trên một đường thẳng (chính là trục PC1) và không có biến thiên (hay thông tin) gì ở trục PC2.

Từ đây chúng ta thấy rõ được cơ chế khuếch tán sai số của ma trận thông tin I đối với linear regression:

¹⁰ Đã thực hiện chuẩn hóa (scaling)



Hình 7: Ý nghĩa của eigenvalue trong dữ liệu

Khi xuất hiện hiện tượng near multi-collinearity trong input X : có ít nhất một cột dữ liệu X_j (hay một chiều không gian của X) là một tổ hợp tuyến tính không hoàn hảo của các cột dữ liệu còn lại (chiều không gian còn lại): $X_j = X^{-j}\delta + \xi$ với ξ là nhiễu. Lúc này, trong số các eigenvalue của ma trận correlation Σ của X , có một eigenvalue $\lambda_k \approx 0$ ứng với trục thông tin này của nhiễu. Khi đó perturbation xuất hiện trong input hoặc output sẽ được khuếch đại qua phép nhân ma trận I , hay đúng hơn là qua nhân tử eigenvalue $\frac{1}{\lambda_k}$ có giá trị rất lớn của I . Lúc này hệ số β của mô hình regression hoàn toàn bị ảnh hưởng bởi i) nhiễu/sai số xuất hiện trong multi-collinearity, ii) sai số xuất hiện một cách bình thường trong dữ liệu.

Do vậy để khắc phục điều này, chúng ta cần loại bỏ sai số xuất hiện trong multi-collinearity.

Một số cách khắc phục hiện tượng multi-collinearity

Ở đây tạm thời chúng ta bỏ qua các cách phát hiện hiện tượng multi-collinearity. Một số phương pháp thông dụng đã được nhắc qua ở các phần trên như: tính toán định thức của ma trận correlation $\det(\Sigma)$, tính toán condition number hay eigenvalue, tính toán R-squared của partial regression. Một số phương pháp khác cũng thường được trình bày ở các sách tiêu chuẩn về econometrics, statistics hay machine learning như tính toán hệ số VIF, CVIF, etc. Chúng tôi mời bạn đọc tham khảo thêm [hướng dẫn sử dụng](#) của package R mctest. Package này implement một lượng khá lớn các phương pháp để phát hiện hiện tượng này. Khi nào có dịp chúng tôi sẽ xin quay lại vấn đề này sau.

Theo như cơ chế khuếch đại ở trên, cách đơn giản nhất để khắc phục hiện tượng này là xác định và loại bỏ các biến X_j bị multi-collinearity. Tuy nhiên phương pháp này tỏ ra ít hiệu quả khi số chiều của dữ liệu lớn do chúng ta phải thực hiện khá nhiều mô hình regression chéo (partial regression) để xác định chính xác.

Cách thứ 2 để khắc phục hiện tượng trên là thực hiện khống chế trực tiếp vào ma trận Σ hay I , hay nói đúng hơn là eigenvalue của 2 ma trận này. Phương pháp này gọi là phương pháp shrinkage, một ứng dụng trực tiếp từ phương pháp tổng quát regularization mà chúng tôi đã nhắc tới trong số trước. Cụ thể hơn, ứng dụng của phương pháp shrinkage (hay regularization) để giải quyết multi-collinearity cho mô hình linear regression là Ridge Regression, được đề xuất bởi Hoerl. A.E. năm 1962. Đây là một phương pháp khá cổ xưa nhưng rất hiệu quả.

Ridge Regression và các tính chất

Hoerl. A.E. [1] đã đề xuất: thay vì sử dụng công thức (1), chúng ta sử dụng công thức ước lượng sau của β để kiểm soát sự phóng đại và sự mất ổn định của $\hat{\beta}$:

$$\hat{\beta}^{ridge} = \hat{\beta}^* = (X^T X + kI_p)^{-1} X^T Y, k \geq 0, I_p \text{ là ma trận đơn vị}$$

$$\text{Hay: } \hat{\beta}^* = W X^T Y, \text{ với } W = (X^T X + kI_p)^{-1}$$

$$\text{Và: } \hat{\beta}^* = [I_p + kI]^{-1} \hat{\beta} = Z\hat{\beta}$$

Nói cách khác, từ công thức trên chúng ta $\hat{\beta}^*$ là một biased estimator của β , nó là một estimator của $Z(k)\beta$.

Chúng ta có thể chứng minh được mối quan hệ của W với I và Σ như sau. Gọi $\delta_i(W)$ là eigenvalue của W , chúng ta có:

$$\delta_i(W) = \frac{1}{(\lambda_i + k)}$$

Xin chú ý rằng chúng ta có eigenvalue tương ứng của I là $\frac{1}{\lambda_i}$. Do vậy chúng ta đang cộng thêm vào mẫu số của eigenvalue của ma trận I một lượng tương ứng là k (hệ số regularization). Từ đó, chúng ta đang tác động vào eigenvalue, đặc biệt là eigenvalue tương ứng của nhiễu trong hiện tượng multicollinearity, từ đó giúp làm giảm ảnh hưởng của nhiễu này. Tuy nhiên khi hệ số k tác động vào tất cả các eigenvalue cùng một lúc, chúng ta có một sự đánh đổi về tính chính xác và tính ổn định. Một hệ số k đủ lớn sẽ lọc bỏ được tín hiệu nhiễu nhưng cũng có thể tác động đủ mạnh để bóp méo các thông tin ở các trục không gian tốt của dữ liệu. Ngược lại một k đủ nhỏ để không có tác động tới các thông tin tốt có thể sẽ chưa đủ lớn để loại bỏ tác động của nhiễu. Ngoài ra, sự xuất hiện của hệ số k trong công thức của $\hat{\beta}^{ridge}$ khiến cho shrinked correlation matrix $X^T X + kI_p$ có thể khả nghịch hay cả khi ma trận correlation suy biến (perfect multicollinearity).

Chúng ta cũng đo lường khoảng cách từ $\hat{\beta}^*$ tới β : $L_1^2(k) = (\hat{\beta}^* - \beta)^T (\hat{\beta}^* - \beta)$. Ta có:

$$\begin{aligned} E[L_1^2(k)] &= \sigma^2 \sum_{i=1}^p \frac{\lambda_i}{(\lambda_i + k)^2} + k^2 \beta^T (\Sigma + kI_p)^{-2} \beta \\ &= \text{Var}(\hat{\beta}^*) + (Z\beta - \beta)^T (Z\beta - \beta) \end{aligned}$$

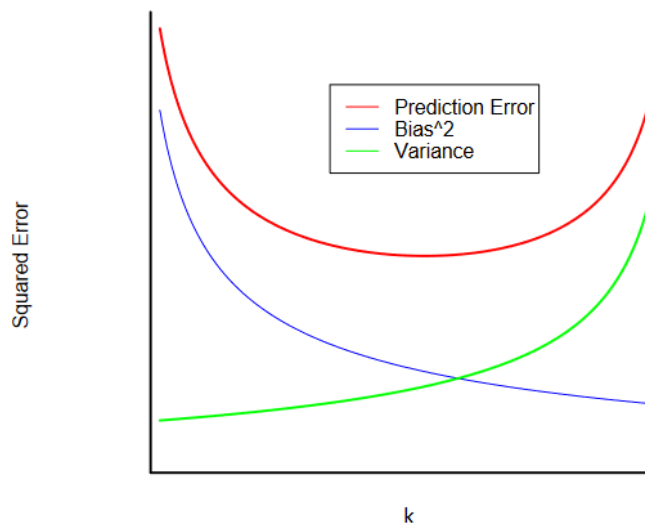
$$= \text{variance của ridge estimator} + \text{bình phương khoảng cách của } Z\beta \text{ và } \beta \text{ (hay } bias^2)^{11}$$

Hay

$$Error^2 = Var + bias^2$$

Đây chính là công thức thể hiện biased-variance tradeoff nổi tiếng. Xin chú ý rằng chúng ta có thể chứng minh được rằng với $\forall k > 0, \text{Var}(\hat{\beta}^*) < \text{Var}(\hat{\beta})$. Điều này có nghĩa là Ridge Regression luôn shrink hệ số $\hat{\beta}^*$ khiến nó có variance nhỏ hơn, từ đó làm tăng tính ổn định của mô hình. Tuy nhiên mặt khác, một hệ số k lớn sẽ làm cho $\hat{\beta}^*$ ước lượng được cách xa khỏi trị số β thật của mô hình.

¹¹ Phần này được gọi là bias do chúng ta cần ước lượng β nhưng ridge regression lại ước lượng $Z\beta$.



Hình 8: Bias-Variance Tradeoff

Chúng tôi mời bạn đọc tham khảo thêm [1], về một số các tính chất khác của ridge regression cũng như các chứng minh toán học.

Ridge regression với Regularization và Penalization

Vấn đề Ridge Regression có thể được viết lại thành vấn đề optimization như sau:

$$\text{minimize } L(\beta) = \|Y - X\beta\|_2^2 \text{ dưới điều kiện } \|\beta\|_2^2 \leq t$$

Điều này ám chỉ chúng ta đang kiểm soát (regulate) các hệ số β của mô hình bằng cách đặt chặn trên, không để chúng bị khuếch đại tự do qua nhiễu của multi-collinearity.

Thông qua phương pháp Lagrange Multiplier (nhân tử Lagrange), bài toán trên có thể được viết thành:

$$\text{minimize } L_{\text{ridge}}(\beta, k) = \|Y - X\beta\|_2^2 + k\|\beta\|_2^2$$

Phần $k\|\beta\|_2^2$ được gọi là penalization part, do nó trừng phạt hệ số β trong loss function. Lúc này thuật toán phải tìm cách cân bằng giữa error và hệ số của mô hình một cách tự động, từ đó kiểm soát trị số của β , không để nó lớn một cách tùy ý. Chúng ta nhận thấy trong penalization part, norm được sử dụng là L_2 norm. Do vậy ridge regression còn được gọi là L_2 -penalty hay L_2 -regularization linear regression. Giải vấn đề nói trên bằng phương pháp đạo hàm, chúng ta sẽ thu được công thức explicite của $\hat{\beta}^*$ như đã chỉ ra ở phần trên.

Trên đây chúng ta đã xem xét một vấn đề nghiêm trọng của regression: multi-collinearity. Các nguyên nhân và hệ quả của nó, cũng như tác động của phương pháp khắc phục ridge regression cũng được giới thiệu về mặt lý thuyết. Tuy nhiên chúng ta cũng chưa chỉ ra làm thế nào để chọn ra hệ số regularization tốt nhất, khi mà chúng ta phải đối diện với biased-variance tradeoff: Một sự hi sinh nhỏ về biasness có thể cải thiện chất lượng mô hình, nhưng một sự shrinkage quá đà cũng hoàn toàn có thể làm hỏng mô hình. Những vấn đề này chúng tôi mời bạn đọc theo dõi ở các số tiếp theo.

Bibliography

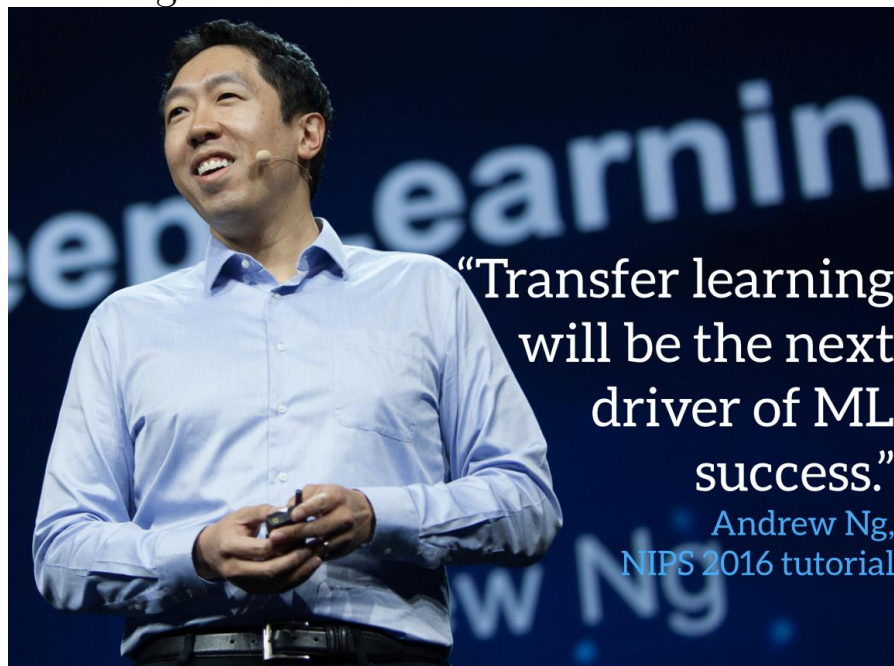
[1]. HOERL, A. E. and KENNARD, R. W. , "Ridge Regression: Biased Estimation for Nonorthogonal Problems," 1970.

A brief introduction on Learning To Learn

Trong những năm qua, lĩnh vực Trí tuệ Nhân tạo nói chung và Machine Learning nói riêng đã đạt được rất nhiều thành công nổi bật, đặc biệt là khi có sự xuất hiện của Deep Learning. Tuy vậy, vẫn tồn tại một vấn đề rất lớn mà các nhà nghiên cứu trong lĩnh vực Trí tuệ Nhân tạo phải đối mặt, đó là các thuật toán Machine Learning đòi hỏi một lượng dữ liệu dán nhãn cực lớn, Deep Learning còn đòi hỏi một lượng dữ liệu lớn hơn nhiều lần. Dữ liệu dán nhãn không luôn luôn có sẵn đối với mọi tác vụ, và chi phí để có thể thu thập một lượng dữ liệu dán nhãn đủ để có thể training các thuật toán một cách tốt nhất là rất đắt đỏ.

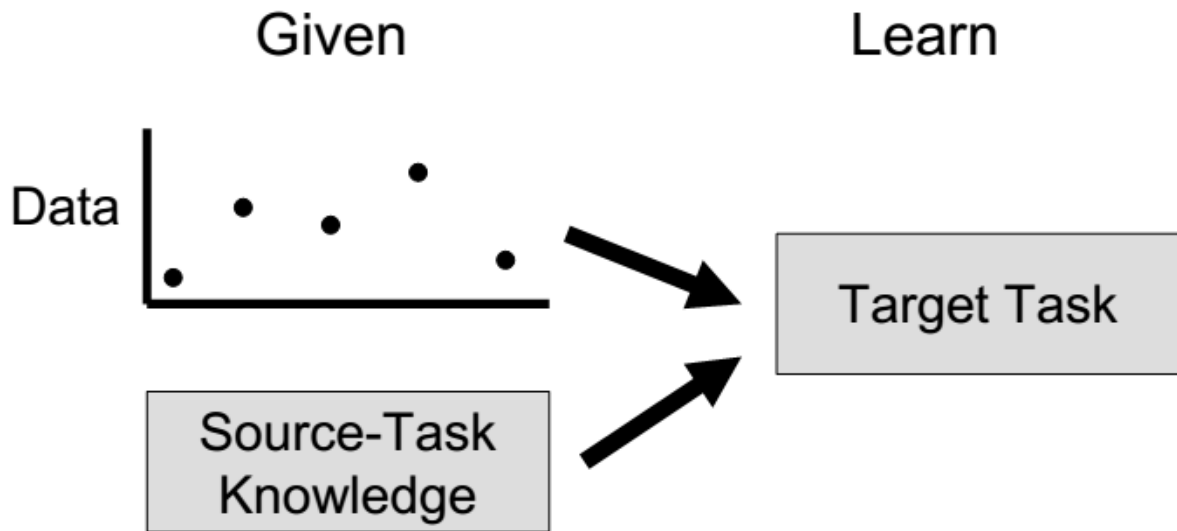
Vì thế mà đã có nhiều nghiên cứu được thực hiện và một số phương pháp được đề xuất để giải quyết (một phần) vấn đề nan giải này. Trong bài viết này, chúng ta sẽ bàn về những phương pháp, thường được gọi là “learning to learn”, phổ biến hiện nay.

1. Transfer Learning



Trước tiên, hãy nói về Transfer Learning, một khái niệm dường như phổ biến, được nhiều người biết đến hơn. Trong hội thảo NIPS 2016, Andrew Ng đã nhận định rằng Transfer Learning, sau Supervise Learning, sẽ là động lực chính để định hướng cho sự phát triển của Machine Learning trong tương lai.

Khái niệm Transfer Learning được nhắc đến lần đầu tiên tại workshop có tên là “Learning to learn”, bàn về những phương pháp Machine Learning cho phép lưu giữ và tái sử dụng những thông tin thuật toán đã học được, trong hội thảo NIPS-95. Kể từ đó, Transfer Learning thu hút ngày càng nhiều sự chú ý. Ý tưởng chung của Transfer Learning là chia sẻ các kiến thức mà hệ thống AI đã nhận được từ quá trình training của tác vụ/domain gốc (source) sang cho một tác vụ /domain mục tiêu (target).



Hình 1: Minh họa về nguyên lý hoạt động của Transfer Learning. Nguồn: Lisa Torrey và Jude Shavlik.

Khái niệm Transfer Learning

Giả sử, chúng ta có một *feature space* \mathcal{X} và một phân phối xác suất $P(X)$ với $X = \{x_1, x_2, \dots, x_n\} \in \mathcal{X}$. Chúng ta có một domain \mathcal{D} như sau:

$$\mathcal{D} = \{\mathcal{X}, P(X)\}$$

Chúng ta cũng sẽ có một label space \mathcal{Y} và một mô hình ước lượng $f(\cdot)$ chưa xác định. Chúng ta có một tác vụ \mathcal{T} như sau:

$$\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$$

Như thường lệ, mô hình ước lượng $f(\cdot)$ có thể được xác định thông qua quá trình training dựa trên dữ liệu training $\{x_i, y_i\}$ với $x_i \in \mathcal{X}$ và $y_i \in \mathcal{Y}$.

Hãy xem xét hai domain:

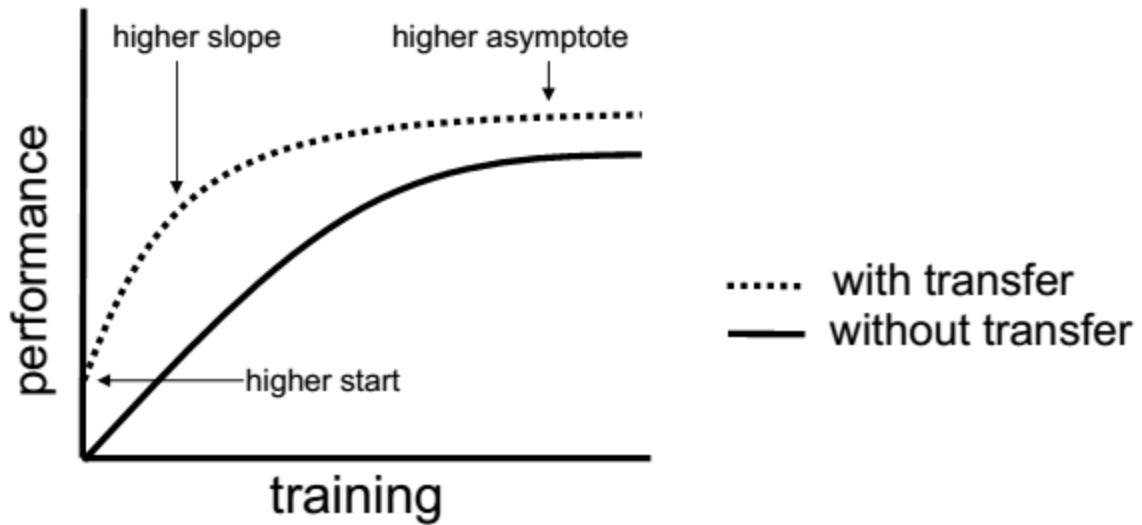
- Domain gốc: $\mathcal{D}_S = \{\mathcal{X}_S, P_S(X)\}$ với $X = \{x_1, x_2, \dots, x_n\} \in \mathcal{X}_S$
- Domain mục tiêu: $\mathcal{D}_T = \{\mathcal{X}_T, P_T(X)\}$ với $X = \{x_1, x_2, \dots, x_n\} \in \mathcal{X}_T$

Và hai tác vụ tương ứng:

- Tác vụ gốc: $\mathcal{T}_S = \{\mathcal{Y}_S, f_S(\cdot)\}$ với $f_S(\cdot) \rightarrow y_i \in \mathcal{Y}_S$
- Tác vụ mục tiêu: $\mathcal{T}_T = \{\mathcal{Y}_T, f_T(\cdot)\}$ với $f_T(\cdot) \rightarrow y_i \in \mathcal{Y}_T$

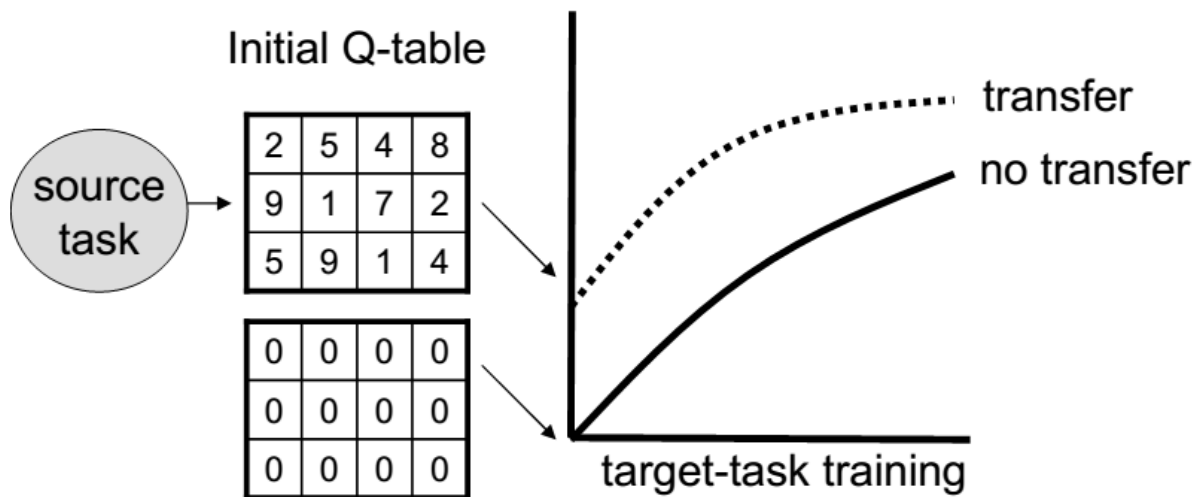
Từ đó, chúng ta sẽ có một khái niệm về transfer learning như sau:

Khái niệm: Giả sử ta có một domain gốc \mathcal{D}_S với tác vụ \mathcal{T}_S và một domain mục tiêu \mathcal{D}_T với tác vụ \mathcal{T}_T , **transfer learning** có mục tiêu cải thiện quá trình học hỏi của hàm ước lượng của tác vụ mục tiêu $f_T(\cdot)$ trong domain \mathcal{D}_T bằng cách sử dụng những kiến thức trong domain \mathcal{D}_S và tác vụ \mathcal{T}_S , với $\mathcal{D}_S \neq \mathcal{D}_T$ và $\mathcal{T}_S \neq \mathcal{T}_T$.



Hình 2: Transfer Learning có mục tiêu cải thiện/nâng cao hiệu quả training của tác vụ/domain mục tiêu. Nguồn: Lisa Torrey và Jude Shavlik.

Việc “transfer” thông tin từ tác vụ/domain gốc sang tác vụ/domain mục tiêu, đơn giản là sử dụng bộ parameter đã được tinh chỉnh (optimized parameters) trong tác vụ/domain gốc như là bộ parameter khởi tạo (initial parameters) của tác vụ/domain mục tiêu.



Hình 3: Phương thức “parameter transfer” giữa các tác vụ/domain. Nguồn: Lisa Torrey và Jude Shavlik.

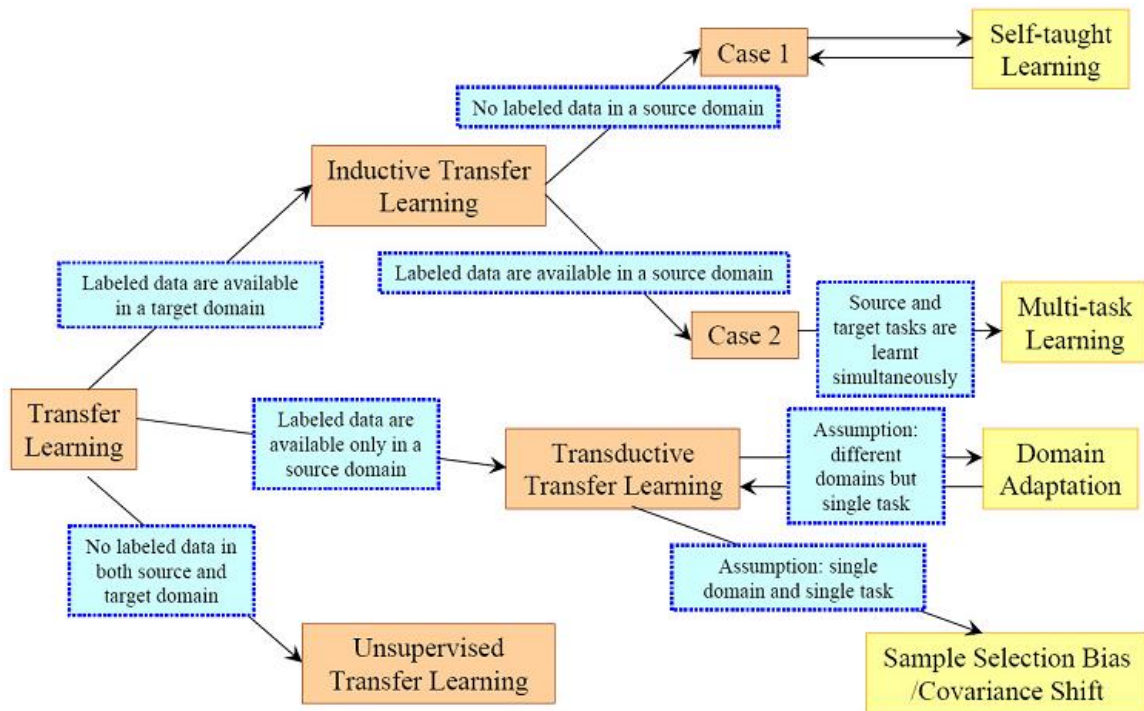
Phân loại Transfer Learning

Dựa vào mối quan hệ giữa tác vụ/domain gốc và mục tiêu cũng như độ sẵn có của dữ liệu của chúng mà chúng ta có thể chia nhỏ transfer learning thành các nhóm nhỏ khác nhau, mà chúng ta có thể gọi là các setting.

Nếu tác vụ mục tiêu khác biệt với tác vụ gốc, bỏ qua mối liên hệ giữa các domain, chúng ta sẽ có *inductive transfer learning*. Nếu tác vụ gốc và tác vụ mục tiêu là giống nhau, nhưng hai domain là khác nhau, chúng ta sẽ có nhóm *transductive transfer learning*. Trong hai nhóm transfer learning đầu tiên này, các tác vụ thường là regression và classification.

Cuối cùng, chúng ta sẽ có nhóm *unsupervised transfer learning*. Tương tự như trong inductive transfer learning, tác vụ gốc và mục tiêu trong unsupervised transfer learning là khác nhau, nhưng với

unsupervised transfer learning, chúng ta sẽ tập trung vào xử lý các tác vụ unsupervised learning như clustering, dimensionality reduction và density estimation.



Hình 4: Phân loại các phương pháp transfer learning. Nguồn: Pan and Yang (2010).

Các nghiên cứu về Transfer Learning thường xoay quanh ba vấn đề chính “What to transfer?”, “When to transfer?” và “How to transfer?”.

Mối quan hệ giữa tác vụ/domain gốc và mục tiêu có ảnh hưởng rất nhiều tới hiệu quả của quá trình transfer learning. Khái niệm “khác” được nhắc đến trong cách phân loại ở trên được hiểu là “khác biệt nhưng có liên quan” (different but related). Tức những tác vụ/domain tham gia trong quá trình transfer learning cần có một sự liên quan (relatedness) nhất định, bởi nếu chúng không liên quan tới nhau, sẽ không có thông tin gì để có thể transfer. Còn nếu hai tác vụ quá liên quan đến nhau, việc cố tình (brutal-force) thực hiện transfer learning có thể dẫn tới hiện tượng *negative transfer* (Rosenstein *et al.*). Đây chính là vấn đề “When to transfer?” trong Transfer Learning. Dù việc hạn chế negative transfer rất quan trọng nhưng hiện tại còn rất ít nghiên cứu về vấn đề này, mà chủ yếu về vấn đề “What to transfer?” và “How to transfer?”. Các nghiên cứu của Ben-David và Schulle, Bakker và Heskes đã thực hiện việc phân tích sự liên hệ giữa các tác vụ và các kỹ thuật clustering tác vụ có thể sẽ cung cấp một số thông tin cho việc tránh hiện tượng negative transfer.

Vấn đề “What to transfer?” đề cập đến việc phần nào của kiến thức học được trong tác vụ/domain nên được transfer. Có những kiến thức mang tính đặc thù đối với từng tác vụ, nhưng có những kiến thức lại giống nhau giữa nhiều tác vụ và có thể chia sẻ giữa chúng để nâng cao hiệu quả của tác vụ/domain mục tiêu. Nội dung vấn đề này chúng ta có thể nhóm các phương pháp transfer learning vào bốn nhóm chính: (1) là phương pháp mà trong đó coi một phần dữ liệu của domain gốc có thể được sử dụng lại bằng cách *re-weighting*, thường được nhắc đến với cái tên *instance-based transfer learning* (hoặc instance transfer). (2) là phương pháp *feature representation transfer*. Trong phương pháp này, chỉ có những kiến thức hữu dụng được chia sẻ giữa các tác vụ bằng cách “encode” chúng vào các feature representation. Phương pháp thứ (3) là *parameter transfer*, với giả định là các tác vụ chia sẻ một số

trọng số hoặc phân phối tiên lượng¹² của các hyper-parameter trong mô hình. Khi đó, các kiến thức được chia sẻ sẽ được “encode” trong các trọng số hoặc thông tin tiên lượng¹³ (Hình 3). Phương pháp cuối cùng (4) được nhắc đến với cái tên *relational knowledge transfer*, sử dụng trong quá trình transfer learning của các domain có mối liên quan, với giả định là các dữ liệu trong domain gốc có mối quan hệ tương tự với các dữ liệu trong domain mục tiêu.

Sau khi xác định được “What to transfer?”, việc tiếp theo sẽ là xác định thuật toán Machine Learning sẽ được sử dụng cho quá trình “learning”, và đây chính là vấn đề “How to transfer?”

Để tìm hiểu sâu thêm về transfer learning, các bạn có thể tham khảo các nghiên cứu sau:

1. A Survey on Transfer Learning. Pan, S. J. và Yang, Q. (2010).
2. Transfer Learning. Torrey, L. và Shavlik, J. (2009).
3. Transfer Learning for Speech and Language Processing. Wang, D. và Zheng, T. F. (2015).

2. Multitask Learning

Trong phần phân loại Transfer Learning, chúng ta có thể thấy Multitask Learning được phân chia như là một nhóm nhỏ trong Transfer Learning. Tuy vậy, không phải nhà khoa học nào cũng đồng ý với sự phân biệt này, một số coi Multitask Learning và Transfer Learning là những khái niệm tương đồng. Sự khác biệt lớn nhất ở đây là trong Transfer Learning, có sự phân biệt giữa tác vụ/domain gốc và tác vụ/domain mục tiêu. Mà trong đó, tác vụ/domain mục tiêu có vai trò quan trọng hơn (Yang, Q. and Zhang, Y.). Trong Multitask Learning, các tác vụ/domain có vai trò bình đẳng hơn.

Khái niệm: Giả sử ta có m tác vụ $\{T_i\}_{i=1}^m$, mà trong đó tất cả các tác vụ hoặc một phần của chúng có liên quan đến nhau. Multitask learning có mục tiêu cải thiện/nâng cao hiệu quả quá trình training của mô hình của tác vụ T_i bằng cách sử dụng kiến thức có trong m tác vụ.

Phần lớn các nghiên cứu về Multitask Transfer Learning (MTL) (khoảng 90%) tập trung vào các tác vụ supervised learning. Trong nhóm những tác vụ này, tác vụ T_i thường đi kèm với tập dữ liệu D_i bao gồm n_i training samples $D_i = \{x_j^i, y_j^i\}_{j=1}^{n_i}$. Ta có $\mathbb{X}^i = (x_1^i, \dots, x_{n_i}^i)$ là ma trận dữ liệu training của tác vụ T_i . Nếu các tác vụ khác nhau có cùng một tập dữ liệu training, MTL trở thành multi-label learning hay multi-output regression. Trong một thiết lập tổng quát của MTL, có ít nhất hai trong số các ma trận \mathbb{X}^i không giống nhau.

Phân loại MTL

Nếu các tác vụ khác nhau trong MTL có cùng feature space, MTL sẽ được phân loại vào nhóm *homogeneous feature MTL*, còn nếu không, chúng sẽ được phân loại vào nhóm *heterogeneous feature MTL*. Thông thường, nếu không có lý giải cụ thể, phần lớn MTL sẽ rơi vào nhóm homogeneous-feature MTL do chúng chiếm đại đa số (khoảng 99%) các nghiên cứu về MTL.

Trong một cách phân loại khác, *heterogeneous MTL* là dạng MTL có thể chứa những tác vụ thuộc nhiều dạng thuật toán khác nhau như supervised learning, unsupervised learning, semisupervised learning, reinforcement learning, multi-view learning và graphical models. Còn trong *homogeneous MTL*, các tác vụ nằm trong một dạng thuật toán duy nhất, 90% nghiên cứu MTL thuộc nhóm này.

¹² Tiếng Anh: Prior distribution

¹³ Tiếng Anh: Priors

Nói một cách dễ hiểu homogeneous feature MTL và heterogeneous feature MTL phân loại dựa trên feature representations còn heterogeneous MTL và homogeneous MTL phân loại dựa trên loại thuật toán tham gia trong MTL.

Cũng giống như Transfer Learning, các nghiên cứu về MTL tập trung vào hai vấn đề là “What to share?” và “How to share?” (dùng share thay vì transfer là do các tác vụ trong MTL có vai trò bình đẳng). Thông thường, trong MTL có thể “share” ba yếu tố là feature, instance và parameter. Features MTL chia sẻ những features chung giữa các tác vụ với nhau. Instance MTL xác định những thông tin trong tác vụ này có lợi cho những tác vụ khác và chia sẻ chúng. Parameter MTL chia sẻ parameter của một tác vụ để hỗ trợ quá trình training trong các tác vụ khác dưới một dạng nào đó, ví dụ như regularization.

Sau khi xác định xong vấn đề “What to share?”, chúng ta sẽ xác định vấn đề “How to share?” dựa vào những gì đã biết về “What to share?”. Zhang, Z. và Yang, Q. đã viết cụ thể về vấn đề này trong nghiên cứu của họ.

Các tài liệu tham khảo dành cho Multitask Learning:

1. Zhang, Z. và Yang, Q. (2017). *A Survey on Multi-Task Learning*.
2. Caruana, R. (1997). *Multitask Learning*. Doctor of Philosophy Thesis. School of Computer Science, Carnegie Mellon University.
3. Ruder, S. (2017). *An Overview of Multi-Task Learning in Deep Neural Networks*. arXiv:1706.05098v1.
4. Godwin, J. (2016). Multi-Task Learning in Tensorflow. Available at <https://goo.gl/swNBwh>.

3. Meta Learning

Meta Learning là một thuật ngữ xuất phát từ lĩnh vực Tâm lý Xã hội với định nghĩa gốc là “the process by which learners become aware of and increasingly in control of habits of perception, inquiry, learning, and growth that they have internalized”. Nói một cách ngắn gọn, meta learning được dùng để chỉ việc con người nhận thức và làm chủ được quá trình tiếp thu kiến thức của mình.

Thuật ngữ Meta Learning bắt đầu xuất hiện trong các nghiên cứu Machine Learning từ thập niên 90, nhưng ý tưởng về Meta Learning đã được khởi xướng từ giữa thập niên 70. Meta Learning trong Machine Learning có ý nghĩa tương tự như trong lĩnh vực Tâm lý Xã hội. Nếu những thuật toán Machine Learning thông thường “học” từ những kinh nghiệm rút ra trong một tác vụ cụ thể. Meta Learning hoạt động ở mức độ cao hơn và “học” từ những kinh nghiệm tổng hợp từ nhiều tác vụ khác nhau.

Một khái niệm về Meta Learning được Lemke et al. đề xuất như sau:

Khái niệm: Một hệ thống Meta Learning phải bao gồm một hệ thống con có thể thích nghi bằng các kinh nghiệm được rút ra từ quá trình training trên một tập dữ liệu trước đó hoặc từ những tác vụ hoặc domain khác nhau.

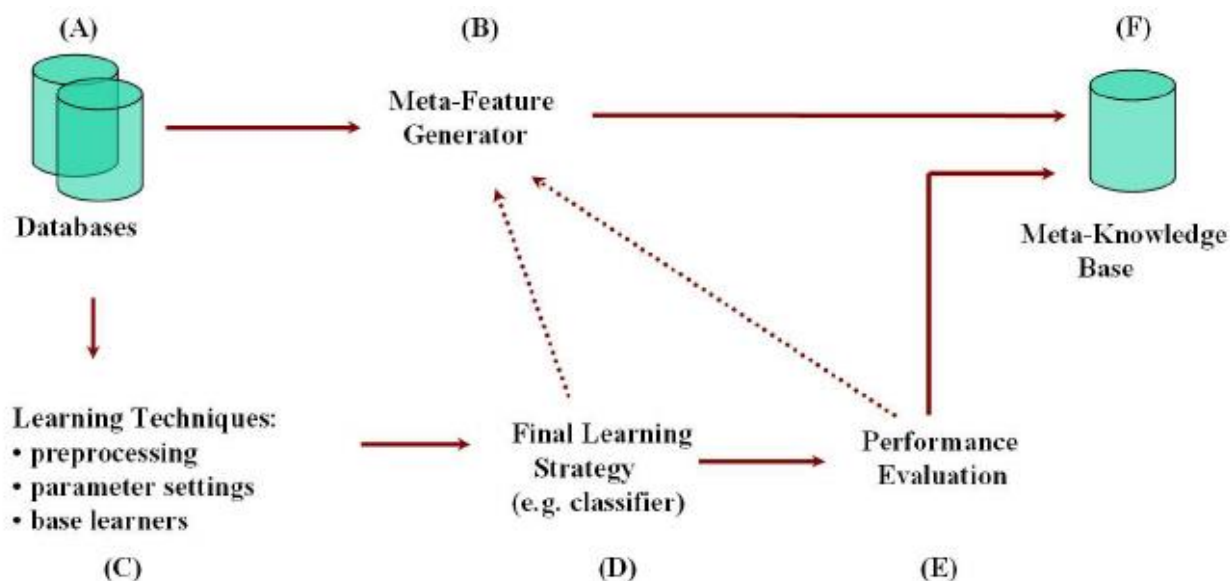
Một thuật toán Machine Learning thông thường sẽ được train trên một tập dữ liệu D được chia làm hai phần D_{train} và D_{test} của tác vụ \mathcal{T} . Trong Meta Learning, thay vì train trên một tác vụ duy nhất, meta-learner (hệ thống con được nhắc đến trong định nghĩa ở trên) sẽ được train trên tập dữ liệu meta-set \mathbb{D} chứa nhiều tập dữ liệu của các tác vụ thông thường $D_i \in \mathbb{D}$ với D_i là tập dữ liệu của tác vụ \mathcal{T}_i .

Trong Meta Learning, tập dữ liệu meta-set \mathbb{D} được chia làm ba tập con (\mathbb{D}_{train} , $\mathbb{D}_{validation}$ và \mathbb{D}_{test}). Tập \mathbb{D}_{train} sẽ được dùng để train meta-learner, hệ thống con sẽ nhận dữ liệu đầu vào D_{train}^* và tạo ra một classifier cho tác vụ \mathcal{T}^* tương ứng. Tập $\mathbb{D}_{validation}$ được sử dụng để thực hiện hyper-parameter selection cho meta-learner và cuối cùng đánh giá hiệu quả hoạt động của thuật toán trên tập \mathbb{D}_{test} .



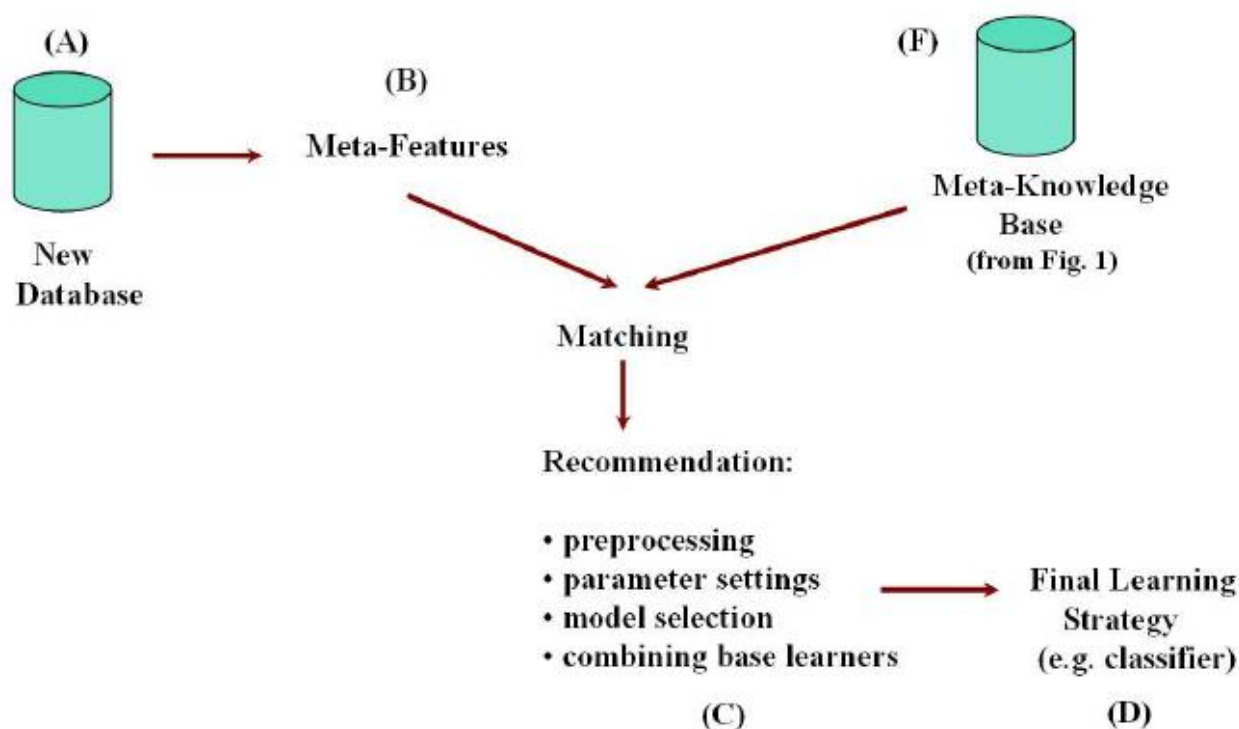
Hình 5: Cơ chế hoạt động chung của Meta Learning. Nguồn: Chelsea Finn.

Như vậy, chúng ta có thể thấy một hệ thống Meta Learning sẽ bao gồm hai thành phần hay hai quá trình hoạt động. Thành phần đầu tiên, thường gọi là *Knowledge-Acquisition*, là thành phần được sử dụng để train meta-learner. Trong quá trình này, hệ thống Meta Learning sẽ rà soát các tập dữ liệu một cách độc lập để trích xuất ra các đặc điểm đặc thù của dữ liệu, còn gọi là meta-features. Mục tiêu của quá trình này là tìm kiếm những thông tin có thể sử dụng để khái quát hóa ra bên ngoài domain của một tác vụ cụ thể. Sau quá trình này, hệ thống sẽ rút ra được một “chiến lược học tập” dưới dạng một classifier hoặc tổ hợp nhiều classifier cũng như các con số thống kê về hiệu quả hoạt động của các tác vụ cụ thể. Những thông tin này cùng với những meta-feature nói trên sẽ tạo thành một *meta-knowledge base*, là sản phẩm cuối cùng và quan trọng nhất của toàn bộ quá trình Knowledge-Acquisition và cũng là đại diện cho toàn bộ kiến thức hệ thống tổng hợp được qua tất cả các tác vụ.



Hình 6: Mô tả hoạt động của quá trình Knowledge-Acquisition. Nguồn: Brazdil et al.

Quá trình hoạt động thứ hai của Meta Learning được gọi là *Advisory*. Trong quá trình này, những dữ liệu meta-knowledge được thu thập trong quá trình acquisition sẽ được hệ thống sử dụng để khám phá các meta-feature của những dữ liệu mới. Những meta-feature này sẽ được so sánh với dữ liệu meta-knowledge base, nếu phát hiện ra sự liên hệ giữa chúng, meta-knowledge base sẽ đưa ra những gợi ý về chiến lược học tập phù hợp.



Hình 7: Mô tả hoạt động của quá trình Advisory. Nguồn: Brazdil et al.

Một số tài liệu tham khảo về Meta-learning:

1. Giraud-Carrier, C. (2008). Metalearning—a tutorial.
2. Finn, C. (2017). Learning to Learn.

Kết luận

Learning to Learn vẫn còn là một lĩnh vực rất mới trong Trí tuệ Nhân tạo. Ba phương pháp tiếp cận được nêu trong bài viết này chỉ là những đại diện nổi bật nhất của Learning to Learn, ngoài ra, còn rất nhiều phương pháp tiếp cận khác cũng đang được nghiên cứu. Dù vậy, tất cả chúng đều hoạt động dựa trên nguyên tắc cơ bản là chia sẻ những thông tin giữa các tác vụ/domain với nhau để tận dụng thông tin và bổ sung cho sự thiếu hụt dữ liệu.

Trong tương lai, vẫn sẽ còn cần rất nhiều nghiên cứu để có thể giải quyết những vấn đề còn tồn tại, như *negative learning* là một ví dụ. Dù vậy, Learning to Learn vẫn là một hướng đi đầy hứa hẹn để thúc đẩy lĩnh vực Machine Learning nói riêng và Trí tuệ Nhân tạo nói chung.

TÀI LIỆU THAM KHẢO

1. Torrey, L. and Shavlik, J. (2009). *Transfer Learning*.
2. Pan, S. J. and Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*. 22(10). pp. 1345–1359.
3. Y. Bengio (2012). Deep learning of representations for unsupervised and transfer learning, In *ICML Unsupervised and Transfer Learning*.
4. Zhang, Z. và Yang, Q. (2017). *A Survey on Multi-Task Learning*. arXiv:1707.08114v1
5. Rosenstein. M. T., Marx, Z. and Kaelbling, L. P. (2005). To transfer or not to transfer. In *NIPS-05 Workshop on Inductive Transfer: 10 Years Later*.
6. Bakker, B. and Heskes, T. (2003). Task clustering and gating for bayesian multitask learning, *Journal of Machine Learning Reserch*. 4, pp. 83–99.
7. Ben-David, S. and Schuller, R. (2003). Exploiting task relatedness for multiple task learning. In *Proceedings of the Sixteenth Annual Conference on Learning Theory*. San Francisco: Morgan Kaufmann. pp. 825–830.
8. Zhang, Z. và Yang, Q. (2017). *A Survey on Multi-Task Learning*.
9. Caruana, R. (1997). *Multitask Learning*. Doctor of Philosophy Thesis. School of Computer Science, Carnegie Mellon University.
10. Giraud-Carrier, C. (2008). Metalearning—a tutorial. In *Proceedings of the 7th international conference on machine learning and applications*. San Mateo: Morgan Kaufmann.
11. Finn, C. (2017). Learning to Learn. Berkeley Artificial Intelligence Research. [online] Available at: <http://bair.berkeley.edu/blog/2017/07/18/learning-to-learn/>