tạp chí

TRÍ TUỆ NHÂN TẠO

Số 2

Tạp chí Trí tuệ Nhân tạo

Số 2, phát hành ngày 03/10/2017

Hà Nội

Website: www.tapchiai.com

Email: tapchiai@gmail.com

Facebook: @tapchiai

Twitter: @tapchiai

GitHub: @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. Hà Đỗ

Ms. Lưu Bích Hồng

Cover Image: Bức tượng Alan Turing do nghệ sĩ Anh quốc Stephen Kettle thực hiện và đặt tại Công viên Blectchy. Ảnh bởi Matthew Oldfield.

Mục Lục

Góc nhìn chuyên gia

HƯỚNG DẪN CHỌN KHÓA HỌC VỀ MACHINE LEARNING VÀ DATA SCIENCE HIỆU QUẢ TRONG 15 PHÚT

Lời khuyên dành cho các chuyên gia trẻ không trong lĩnh vực Khoa học máy tính (Computer Science) muốn tìm hiểu và đóng góp vào lĩnh vực Data Science/Machine learning, lấy từ kinh nghiệm cá nhân của tác giả Tirthajyoti Sarkar.

Động lực

Trong một buổi lễ tốt nghiệp gần đây, Bill Gates đã tuyên bố rằng trí tuệ nhân tạo (AI), năng lượng và khoa học sinh học là ba lựa chọn nghề nghiệp thú vị và đáng để lựa chọn nhất.

Tôi không thể đồng ý nhiều hơn.

Tôi tin rằng một số câu hỏi quan trọng nhất của thế hệ chúng ta liên quan đến sự bền vững, sản xuất và phân phối năng lượng, vận chuyển, tiếp cận với các tiện ích cơ bản của cuộc sống vv, phụ thuộc vào cách chúng ta có thể kết hợp thông minh thế nào hai nhánh kiến thức đầu tiên mà Gates đề cập đến.

> *Nói cách khác, thế giới của điện tử (mà ngành công nghiệp bán dẫn là một phần trung tâm của thế giới đó), phải làm nhiều hơn để nắm lấy đầy đủ các thành quả của công nghệ thông tin và các phát triển mới trong trí tuệ nhân tạo (AI) hoặc khoa học dữ liệu (Data science).*

Tôi muốn học, nhưng bắt đầu từ đâu?

Tôi là một chuyên gia bán dẫn với 8 năm kinh nghiệm sau tiến sỹ trong một công ty công nghệ hàng đầu. Tôi tự hào vì thực tế là tôi đang làm việc trong lĩnh vực thiết kế điện tử, trực tiếp góp phần vào ngành năng lượng: Tôi phát triển các thiết bị bán dẫn. Chúng được chế tạo để mang điện năng hiệu quả và đáng tin cậy, và chúng cung cấp năng lượng cho tất cả mọi thứ từ cảm biến nhỏ bé bên trong chiếc điện thoại thông minh của bạn cho đến các máy móc, thiết bị công nghiệp lớn đang xử lý thức ăn hoặc vải để tiêu dùng hàng ngày.

Do đó, một cách tự nhiên, tôi muốn học và áp dụng các kỹ thuật khoa học dữ liệu hiện đại và machine learning để cải tiến thiết kế, độ tin cậy và vận hành các thiết bị và hệ thống này.

Nhưng tôi không phải là một sinh viên khoa học máy tính. Tôi còn không biết những "linked list" từ "heap". "Support vector machines", cách đây vài tháng, với tôi nghe như một thiết bị đặc biệt để hỗ trợ người khuyết tật. Và từ khóa duy nhất của AI mà tôi nhớ được (từ kiến thức năm thứ nhất của tôi) là 'first order predicate calculus', một phần còn lại của cái gọi là 'AI cũ' hay cách tiếp cận theo hướng knowledge-engineer, thay vì phương pháp tiếp cận dựa trên machine learning mới hơn.

> *Tôi phải bắt đầu từ một nơi nào đó để tìm hiểu những điều cơ bản và sau đó nghiên cứu sâu hơn theo hướng của tôi. Rõ ràng là nên lựa chọn các khóa học trực tuyến MOOC (Massive Open Online Courses). Tôi vẫn còn đang trong giai đoạn học tập nhưng tôi tin rằng ít nhất tôi đã thu thập được một số kinh nghiệm tốt trong việc chọn đúng MOOC cho con đường này. Trong bài này, tôi muốn chia sẻ những hiểu biết của mình về khía cạnh đó.*

Biết điểm mạnh và điểm yếu của bạn

> *[...]Bạn nên biết rõ điểm mạnh, điểm yếu và xu hướng kỹ thuật của mình trước khi bắt đầu quá trình học qua MOOC.*

Bởi vì thời gian và năng lượng là có hạn và bạn không thể lãng phí nguồn lực quý giá của mình vào một thứ mà bạn không thực hành trong công việc hiện tại hoặc công việc trong tương lai. Và tôi giả sử rằng bạn muốn theo một con đường học tập gần như miễn phí, tức là theo các MOOCs thay vì thanh toán cho các chứng chỉ. Tôi nói 'gần như' vì cuối bài viết này, tôi có liệt kê vài MOOCs mà tôi nghĩ bạn nên trả tiền để có các chứng nhận của nó. Và, với cá nhân tôi, tôi đã phải trả cho một số khóa học Udemy bởi vì Udemy không bao giờ miễn phí nhưng bạn có thể mua chúng với giá của một bữa trưa khi đang có khuyến mãi.

Những gì bạn có thể và không thể học hỏi từ MOOCs

Trong hình bên, tôi muốn cho bạn thấy những gì là có thể và những điều không thể của học MOOCs, những gì bạn có thể hy vọng học được thông qua việc tự học và thực hành và những gì cần phải học được trong công việc hoặc tư duy nào phải được gieo trồng nuôi dưỡng bất kể nghề của bạn là gì. Những hình tròn lớn bao gồm các kỹ năng cốt lõi mà ta có thể nghiên cứu để kết hợp vào lĩnh vực khoa học dữ liệu/machine learning từ một nền tảng không phải là Computer Science. Xin lưu ý rằng ngay cả khi bạn đang trong lĩnh vực công nghệ thông tin (CNTT), bạn có thể có một con đường học tập khác xa vì CNTT truyền thống đang bị phá vỡ bởi những lĩnh vực mới này và các kỹ năng cốt lõi và thực tiễn thường khác nhau.

Tôi nhận thấy lĩnh vực khoa học dữ liệu có tính dân chủ hơn nhiều lĩnh vực chuyên môn khác (ví dụ như công việc của tôi trong lĩnh vực công nghệ bán dẫn), nơi mà rào cản gia nhập thấp và chỉ cần làm việc chăm chỉ và nhiệt huyết, ai cũng có thể đạt được các kỹ năng cần thiết. Đối với cá nhân tôi, tôi không có ham muốn cháy bỏng trong lĩnh vực này, thay vào đó, tôi chỉ có một niềm đam mê được mượn các kết quả của nó để áp dụng cho lĩnh vực chuyên môn của tôi. Tuy nhiên, mục tiêu cuối cùng không ảnh hưởng đến con đường học tập ban đầu mà người ta phải đi qua. Vì vậy, bạn có thể muốn làm gì cũng được, kỹ sư dữ liệu, hoặc nhà phân tích kinh doanh, hoặc nhà khoa học machine learning hoặc một chuyên gia về visualization - các lĩnh vực và các lựa chọn đều rộng mở. Và nếu mục đích của bạn giống như của tôi - ở trong lĩnh vực chuyên môn hiện tại và áp dụng các kỹ thuật mới được học - bạn cũng sẽ ổn thôi.

Bạn có thể bắt đầu với những điều cơ bản thực sự, đừng xấu hổ

Tôi bắt đầu với những thứ thực sự cơ bản - học Python trên Codeacademy. Xét mọi trường hợp thì bạn không thể tìm cách đơn giản hơn, và nó hiệu quả. Tôi đã có ác cảm với lập trình nhưng giao diện đơn giản, vui nhộn và đúng tốc độ của khóa học miễn phí của Codeacademy thực sự thích hợp để kích thích tôi đủ để tiếp tục. Tôi cũng có thể chọn một khóa học Java hoặc C ++ trên Coursera hoặc Datacamp hoặc Udacity nhưng một số bài đọc và nghiên cứu đã nói rằng Python là sự lựa chọn tối ưu để cân bằng giữa phức tạp và tiện ích học tập (đặc biệt là khoa học dữ liệu) và tôi quyết định tin tưởng vào các nhận định này.

Sau một thời gian, bạn muốn học kiến thức sâu hơn (nhưng ở tốc độ nhẹ nhàng)

Khóa học ở Codeacademy là một sự bắt đầu đơn giản. Tôi có rất nhiều lựa chọn từ các khóa MOOC trực tuyến và tôi đã đăng ký nhiều khóa học cùng một lúc. Tuy nhiên, sau khi tham gia lớp Coursera trong vài ngày, tôi nhận ra rằng mình chưa sẵn sàng để học Python từ một giáo sư! Do vậy, tôi tìm một khóa học được giảng dạy bởi một giảng viên nhiệt tình, người dành thời gian cung cấp các khái niệm một cách chi tiết về những công cụ thiết yếu khác như Git và Jupyter notebook, và có một sự cân bằng giữa các khái niệm cơ bản và các chủ đề nâng cao trong chương trình giảng dạy. Và người phù hợp nhất chính là Jose Marcial Portilla. Ông cung cấp nhiều khóa học trên Udemy và là một trong những giảng viên được đánh giá cao nhất và tích cực. Tôi đã đăng ký và hoàn thành khóa học Bootcamp Python - một khóa nhập môn tuyệt vời cho Python với tốc độ phù hợp, có chiều sâu và nghiêm ngặt. Tôi đề xuất khóa học này cho người mới học mặc dù bạn phải bỏ ra 10 đô la (các khóa học Udemy thường không miễn phí và giá thường là $ 190 hoặc $ 200 nhưng bạn luôn có thể chờ vài

ngày đến khi có khuyến mãi và đăng ký với 10 đô la hoặc 15 đô la).

Điều quan trọng là phải tập trung vào Data science

Bước tiếp theo rất quan trọng đối với tôi. Tôi có thể đã đi lạc hướng và cố gắng nghiên cứu bất cứ điều gì và tất cả mọi thứ tôi có thể trên Python. Đặc biệt, phần object-oriented (lập trình hướng đối tượng) và các class dễ dàng kéo bạn vào một cuộc hành trình dài và gian khổ. Vậy nên, bây giờ, đừng lấy gì từ không gian chính của vũ trụ Python, bạn vẫn có thể thực hành học machine learning và data science (khoa học dữ liệu) tốt mà không cần có khả năng định nghĩa các class (lớp) và method (phương thức) của riêng bạn bằng Python. Một trong những lý do cơ bản của sự phổ biến ngày càng tăng của Python như là ngôn ngữ chính được lựa chọn cho data science, là sự sẵn có của số lượng lớn các thư viện, các lớp (class) và methods chất lượng cao, được viết bởi các chuyên gia và được đánh giá, chỉ cần đợi để tải xuống trong các package và bung ra để tích hợp vào mã của bạn.

> *Do đó, điều quan trọng bây giờ là phải nhanh chóng học các thư viện và các methods được sử dụng rộng rãi nhất cho khoa học dữ liệu - NumPy, Pandas, và Matplotlib.*

Các nội dung trên được giới thiệu với một khóa học nhỏ gọn từ edX. Mặc dù hầu hết các khóa học về edX là từ các trường đại học, khá nghiêm ngặt (và dài), nhưng có một số ít các khóa học ngắn hạn và nhiều thực hành, ít lý thuyết hơn được cung cấp bởi các công ty công nghệ như Microsoft. Một trong số đó là Microsoft Professional Program in Data Science . Bạn có thể đăng ký các khóa học theo chương trình này như bạn muốn. Tuy nhiên, tôi chỉ học các khóa học sau đây (và tôi sẽ trở lại các khóa học khác trong phần tới)

- Data Science Orientation: Thảo luận về cuộc sống hàng ngày của một nhà khoa học dữ liệu điển hình và nhấn mạnh đến những kỹ năng cốt lõi mà người ta mong đợi cùng với việc giới thiệu cơ bản các chủ đề quan trọng.
- Introduction to Python for Data Science: Dạy các khái niệm cơ bản về Python - các cấu trúc dữ liệu, các vòng lặp, các hàm, và sau đó giới thiệu NumPy, Matplotlib và Pandas.
- Introduction to Data Analysis using Excel: Dạy các hàm phân tích dữ liệu cơ bản và nâng cao, vẽ, và các công cụ với Excel (ví dụ như pivot table, power pivot, và solver plug-in)
- Introduction to R for Data Science: Giới thiệu về cú pháp R, các loại dữ liệu, các phép toán vector và ma trận, các factors, các hàm, data frame khung dữ liệu và đồ họa với ggplot2.

Mặc dù các khóa học này trình bày các tài liệu một cách đơn giản và chỉ bao gồm hầu hết các ví dụ cơ bản nhưng chúng đã đủ để kết nối tôi với Data science.

Tôi chuyển sang học R một cách chi tiết một thời gian

Khóa học cuối cùng đã làm cho tôi nhận ra vài điều quan trọng: (a) thống kê và đại số tuyến tính là cốt lõi của data science, (b) tôi không biết/đã quên mất điều đó, và (c) R phù hợp một cách tự nhiên với việc tôi muốn làm với bộ dữ liệu của tôi - vài MB dữ liệu được tạo ra bởi thí nghiệm controlled wafer fab hoặc mô phỏng TCAD, dễ dàng cho phân tích suy luận cơ bản.

Điều này buộc tôi phải tìm kiếm một khóa học nhập môn R thật vững và một lần nữa, còn ai tốt hơn Jose Portilla! Tôi đã đăng ký lớp học "Data Science and Machine Learning Bootcamp with R". Đây là một khóa mua một tặng một miễn phí vì khóa học đã bao gồm các yếu tố cần thiết của R trong nửa đầu và chuyển sang giảng dạy các khái niệm cơ bản của machine learning (tất cả các khái niệm quan trọng đều có đầy đủ). Không giống như khóa học của edX Microsoft sử dụng môi trường lab-based trên máy chủ, khóa học này đã bao gồm việc cài đặt và thiết lập R Studio và các gói cần thiết, giới thiệu cho tôi kaggle và đưa ra yêu cầu cần thiết để tránh việc học sinh học thụ động (chỉ xem video MOOC) cho một người không sợ dữ liệu. Nội dung của nó tuân theo từng chương của cuốn sách tuyệt vời"Introduction to Statistical Learning in R" (ISLR) của Gareth James, Daniela Witten, Trevor Hastie và Robert Tibshirani.

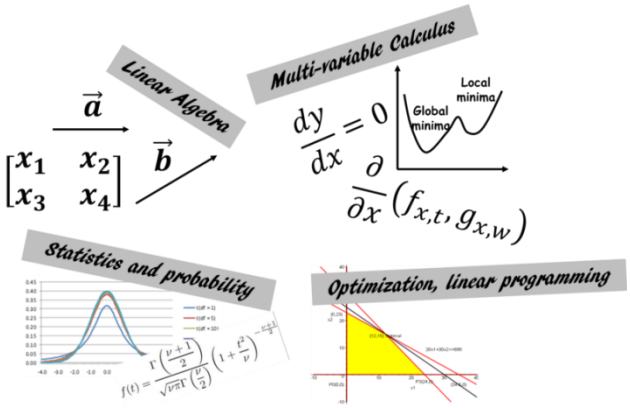
> *Nếu bạn chỉ được phép đọc một cuốn sách trong suốt cuộc đời để học machines learning, hãy chọn cuốn sách này và đọc tất cả các chương, đừng bỏ qua chương nào. Dù vậy, không có neural network hay deep learning trong cuốn sách này đâu, thế nên, ...*

Được trang bị các tài liệu khóa học, sách ISLR, và thực hành các tập dữ liệu ngẫu nhiên được tải xuống từ kaggle hoặc thậm chí dữ liệu sử dụng

điện của PG&E, tôi không còn ngại viết các đoạn mã nhỏ thực sự có thể mô hình cái gì đó thú vị và hữu ích. Tôi đã phân tích một số dữ liệu về tội phạm tại Hoa Kỳ, tại sao một thử nghiệm lớn được thiết kế có thể dẫn đến một tương quan giả mạo và thậm chí mức sử dụng điện của căn hộ tôi trong 3 tháng qua. Tôi cũng thành công khi sử dụng R để xây dựng mô hình tiên đoán dựa trên một số bộ dữ liệu thực tế từ công việc của tôi. Tính thống kê/ chức năng của ngôn ngữ và việc ước tính của khoảng tin cậy (p-values hoặc z-score) cho nhiều mô hình (hồi quy hoặc phân loại) thực sự giúp một người học mới đạt được vài thành quả dễ dàng trong lĩnh vực thống kê mô hình hóa.

Học thêm càng nhiều về toán học cơ bản càng tốt

Khía cạnh học tập này không thể nhấn mạnh quá nhiều - đặc biệt là đối với những sinh viên không phải trong ngành khoa học máy tính và các kỹ sư CNTT, những người không làm việc liên quan đến toán học nhiều năm. Tôi thậm chí đã viết một bài trên medium về những kiến thức toán học nào là cần thiết phải có cho machine learning và data science.



Đối với việc này tôi đã chọn vài khóa học từ Cousera và edX. Rất ít khóa có chiều sâu và đầy đủ, sau đây là các khóa ấy:

- Statistical Thinking for Data Science and Analytics (Columbia Univ.): Khóa thống kê cơ sở của Đại học Columbia trong Chương trình Data Science Executive certificate program của họ trên edX. Khóa học rất cốt lõi và đưa ra các bài tập theo cấu trúc tốt.
- Computational Probability and Inference (MIT): Đây là một khóa học khó của MIT, nó bao gồm các chủ đề nâng cao và rất sâu như các mô hình Bayesian và các mô hình graphical.
- Statistics with R Specialization (Duke Univ.): Đây là khóa học 5 khóa chuyên môn từ Đại học Duke (khóa cuối cùng là một dự án, bạn có thể bỏ qua), để bạn nắm vững cơ bản về thống kê và thực hành bài tập lập trình. Tôi đề xuất nó vì nó có sự cân bằng giữa độ khó và sự nghiêm ngặt.
- LAFF: Linear Algebra—Foundations to Frontiers (UT Austin): Đây là một khóa học tuyệt vời cung cấp kiến thức nền về đại số tuyến tính (cùng với thảo luận sâu về hiệu năng tính toán trong đại số tuyến tính) mà bạn phải thử. Khóa học được cung cấp bởi Đại học Texas, Austin trên edX. Hãy tin tưởng tôi khi tôi nói, sau khi tham gia khóa học này, bạn sẽ không bao giờ muốn đảo ngược một ma trận để giải quyết một hệ phương trình tuyến tính, ngay cả khi đó là cám dỗ và dễ hiểu nhưng bạn sẽ cố gắng tìm một phân tích QR factorization hoặc Cholesky decomposition để giảm sự phức tạp trong tính toán.
- Optimization Methods in Business Analytics (MIT): Đây là khóa học về optimization/ operation research methods trong phân tích kinh doanh của MIT. Tôi đã đăng ký vì đây là khóa học đánh giá cao duy nhất trên edX mà tôi có thể tìm thấy về linear and dynamic programming techniques. Tôi tin rằng việc học về những kỹ thuật này có thể vô cùng hữu ích vì vấn đề tối ưu hóa xuất hiện trong hầu hết các thuật toán machine.

Xin lưu ý rằng tôi đã không tìm kiếm và đăng ký bất kỳ lớp toán nào vì tôi thấy kiến thức toán trong đại học là đủ cho việc học machine learning hoặc data science và thực hành. Nếu bạn đang thiếu kiến thức ở các vấn đề đó, hãy tìm kiếm một khóa học tốt.

Machine Learning - những cá tính khác nhau làm cho nó trở thành một câu chuyện đầy màu sắc

Một cách nào đó, tôi đã hoàn thành khóa học được xem là tiên phong của tất cả các MOOCs- khóa học về machine learning của Andrew Ng trên Coursera. Tôi đoán đã có rất nhiều bài báo về nó và do đó, tôi sẽ không lãng phí thêm thời gian của bạn mô tả khóa học này. Chỉ cần lấy nó, làm tất cả bài tập về nhà và lập trình, học cách nghĩ theo vectorized codes cho tất cả các thuật toán machine learning

chính mà bạn biết, và lưu các ghi chú để sẵn sàng tham khảo cho công việc trong tương lai của bạn.

Oh, nhân tiện, nếu bạn muốn học MATLAB từ đầu (bạn sẽ cần phải viết code MATLAB cho khóa học này, không phải R hoặc Python), bạn có thể xem khóa Giới thiệu về Lập trình với MATLAB Introduction to Programming with MATLAB.

Bây giờ, tôi muốn nói về tính cách.

> *Tôi đã tham gia nhiều khóa học về machine learning và khía cạnh tôi thích nhất là nhận ra cách xử lý với cùng một chủ đề cơ bản khác nhau theo cá tính và thế giới quan của các giảng viên :) Đây là một trải nghiệm rất hấp dẫn.*

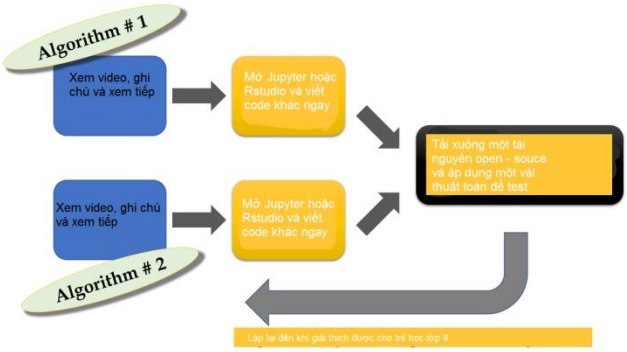
Tôi liệt kê dưới đây những khóa học MOOC về machine learning mà tôi đã học:

- Machine Learning (Stanford Univ.): Khóa học nổi tiếng của Andrew Ng. tôi có nói về nó trong đoạn trên.
- Machine Learning Specialization (Univ. of Washington): Khóa này có một phong cách khác với khóa của Andrew Ng. Emily Fox và Carlos Guestrin trình bày các khái niệm từ quan điểm của một nhà thống kê và của một người ứng dụng. Tôi không thể cài đặt gói Python mà công ty Carlos cung cấp bằng free licence nhưng khóa học này vẫn rất giá trị nếu hoàn thành các bài lý thuyết của nó. Các bằng chứng và thảo luận về một số khái niệm cơ bản như bias - variance trade-off (sự đánh đổi giữa bias và variance), cost computation, và sự so sánh giữa analytic với hướng tiếp cận số học trong việc tối thiểu cost function, được trình bày tự nhiên và cẩn thận hơn cả các chuyên đề của Giáo sư Ng.
- Machine Learning for Data Science and Analytics (Columbia Univ.): Khóa học này có một giáo trình không bình thường cho một khóa học về machine learning chung bằng cách dành nửa đầu cho các bài giảng về thuật toán thông thường. Nó bao gồm các thuật toán phân loại, tìm kiếm, đồ thị và các thuật toán lên lịch. Không có nhiều thảo luận riêng về cách các thuật toán này được sử dụng chính xác trong các vấn đề machine learning nhưng nghiên cứu về chúng cho bạn một ý tưởng về các kiến thức CNTT cần thiết để thấy được những vấn đề khoa học dữ liệu trên quy mô lớn được giải quyết như thế nào. Hãy nghĩ đến O(n^3) bất cứ khi nào bạn sắp nhân ma trận hoặc nghĩ đến O(nlog (n)) bất cứ khi nào bạn sắp xếp một danh sách. Bạn có thể không sử dụng kiến thức này trong công việc hàng ngày của bạn, nhưng hiểu biết về những điểm quan trọng trong computation process chắc chắn mở rộng tầm nhìn thế giới của bạn về vấn đề đang bàn đến.
- Data Science: Data to Insights (MIT xPro 6 weeks online course): Đây là một trong số rất ít các khóa học trả tiền tôi đã đăng ký. Khóa này không có trên trang edx cho công chúng mặc dù nó sử dụng edX để phân phối nội dung. Khóa học 6 tuần có cấu trúc tốt và đầy đủ các nội dung thú vị, mở ra thế giới rộng của khoa học dữ liệu và machine learning cho những người không có kinh nghiệm. Các case studies rất thú vị nhưng có độ khó hợp lý và tốn thời gian để lập trình. Các bài giảng rất hấp dẫn với minh họa từ những case studies điển hình này. Phần tôi đặc biệt yêu thích là recommendation system - hệ thống khuyến nghị. Tôi thực sự bắt đầu nhìn màn hình Netflix trên máy tính xách tay của tôi theo ma trận kể sau khi học lớp này!
- Neural Networks for Machine Learning (Univ. of Toronto): Đây là một khóa học bị đánh giá thấp trên Coursera, ngay cả với khi thầy dạy là Jeff Hinton, nhà tiên phong của mạng nơ-ron. Tôi thấy rằng khóa deep learning của Andrew Ng sẽ trực tiếp cạnh tranh với khóa học này và tôi sẽ không ngạc nhiên nếu Coursera loại bỏ khóa này trong trong tương lai gần. Tuy nhiên, khi nó vẫn còn ở đó, chúng ta có thể học nó và tập trung vào vấn đề này, ngay cả khi chỉ để đánh giá lịch sử phát triển của deep networks.
- Deep Learning Specialization (deeplearning.ai): Đây là khóa mới nhất trong tất cả nhưng nó vẫn thuộc trong lĩnh vực của Andrew Ng, và do đó sẽ rất hứa hẹn :) Tôi đã hoàn thành khóa học thứ 2 và giờ đã đến khóa thứ 3. Chắc chắn bạn nên xem xét hoàn thành loạt bài này nếu bạn muốn biết các xu hướng mới nhất trong deep learning. Ngay cả khi các bài tập lập trình trông rất khó và bạn muốn lập trình bằng tay một deep network (bạn có thể cho rằng luôn có những gói mã nguồn mở xuất sắc như TensorFlow, Keras, Theranos để xử lý), nhưng bắt buộc cần có sự hiểu biết sâu sắc về các khái niệm thiết yếu như regularization, exploding gradient, hyperparameter tuning, batch normalization , vv để sử dụng có hiệu quả các framework deep learning trình độ cao.

Học tập được dân chủ hóa - hãy tận dụng nó

Với sự xuất hiện của MOOCs, các platform lập trình mã nguồn mở, các công cụ cộng tác và lưu trữ trên cloud miễn phí hầu như không giới hạn, học tập đã được dân chủ hóa, trở nên phổ biến, và phổ cập cho tất cả. Kể cả bạn không phải là chuyên gia về khoa học dữ liệu/machine learning nhưng bạn muốn học nó, hãy viết một số code để tăng năng suất làm việc, cố gắng nâng cao kỹ năng nghề nghiệp, hoặc chỉ để cho vui, hãy bắt đầu học tập nó. Đây là một số ý kiến của tôi:

- Nếu bạn là một nhà khoa học dữ liệu: Đừng để bất kỳ một người nào được xem là chuyên gia làm nhụt chí bạn bằng cách nói "MOOCs là trò trẻ con, bạn sẽ không học được khoa học dữ liệu thực sự như thế". Thực tế là bạn đang cố gắng nghiên cứu khoa học dữ liệu bằng cách ghi danh vào MOOC có nghĩa là : (a) bạn đang giải quyết các vấn đề với dữ liệu trong nghề nghiệp của mình và (b) bạn muốn học một cách khoa học, có cấu trúc để khai thác giá trị tối đa từ dữ liệu của bạn và đưa ra các câu hỏi thông minh xung quanh dữ liệu đó. Điều đó có nghĩa là bạn đã là một nhà khoa học dữ liệu. Nếu vẫn không thấy thuyết phục, hãy đọc blog của Brandon Rohrer, một trong những nhà khoa học dữ liệu được ngưỡng mộ và đầy cảm hứng nhất mà tôi biết.
- Bạn không phải trả một số tiền lớn cho việc học này: Tôi biết rằng tôi liệt kê rất nhiều khóa học và chúng có thể trông rất đắt đối với bạn. Nhưng, may mắn thay, hầu hết (nếu không phải tất cả), có thể được ghi danh miễn phí. Các khóa học edX luôn được tự do ghi danh và thường không có bất kỳ hạn chế nào về nội dung khóa học, tức là bạn có thể xem, thực hành, gửi tất cả các bài tập có tính điểm (không giống như Coursera, cho phép bạn xem tất cả các video nhưng ẩn các tài liệu có tính điểm). Nếu bạn nghĩ rằng một số giấy chứng nhận sẽ có giá trị trên CV của bạn, bạn luôn có thể trả tiền cho nó ở giữa khóa học sau khi bạn đã hoàn thành một số video và đánh giá nó có xứng đáng bỏ tiền không.
- Thực hành, code, và xây dựng những thứ khác để bổ sung cho các kiến thức từ việc học trực tuyến của bạn: Có một thuật toán 'học trực tuyến' trong ngữ cảnh của machine learning. Trong kỹ thuật này, thay vì xử lý một ma trận đầy đủ của hàng triệu điểm dữ liệu, thuật toán này làm việc với vài điểm dữ liệu mới nhất và cập nhật các dự đoán. Bạn cũng có thể làm việc ở chế độ này. Vấn đề dừng / đỗ lại ở đâu luôn luôn là một vấn đề hấp dẫn và nó cũng áp dụng cho việc học. Chúng ta luôn tự hỏi có bao nhiêu để nghiên cứu và tiếp thu trước khi xây dựng những thứ khác. Điều đó ngăn chặn việc học tập và bắt đầu học. Đừng ngần ngại, đừng trì hoãn. Tìm hiểu một khái niệm và kiểm tra nó bằng các đoạn code đơn giản. Làm việc với thủ thuật mới nhất hoặc kỹ thuật mà bạn đã xem video, đừng chờ đợi để thành thạo về toàn bộ chủ đề. Bạn sẽ ngạc nhiên bởi cách mà 20 dòng mã đơn giản có thể cho bạn những kiến thức thực hành vững chắc (và làm cho bạn đổ mồ hôi) về các khái niệm phức tạp nhất mà bạn đã học được thông qua xem video đó.



- Có rất nhiều dữ liệu trên mạng: Bạn cũng sẽ ngạc nhiên trước xem có bao nhiêu nguồn dữ liệu miễn phí phong phú trên mạng. Đừng đi đến Kaggle, hãy thử một cái gì đó khác cho vui. Hãy thử cổng dữ liệu data.gov hoặc của Liên hợp quốc. Tới repository của UCI Machine learning. Bạn muốn cảm thấy phiêu lưu hơn? Hãy tải dữ liệu về các quốc gia khác nhau từ CIA và thử tất cả các cách biểu diễn mới nhất mà bạn đã học được trong bài giảng mới nhất của Matplotlib hoặc ggplot2? Nếu không có gì khác, tải dữ liệu sử dụng điện của chính bạn từ nhà cung cấp năng lượng của bạn và phân tích nếu bạn có thể tiết kiệm vài đô la nếu bạn bật AC hoặc máy rửa chén vào một thời điểm khác.

Bài viết của tác giả **Tirthajyoti Sarkar** được đăng trên Blog cá nhân ngày 26-09-2017 tại địa chỉ: https://goo.gl/PmRmZT.

Dịch bởi **Lưu Bích Hồng**

Machine Learning tutorial

1. Machine Learning Basics

Một thuật toán ML thường bao gồm 4 thành phần cơ bản nhất là mô hình (model), cost function, thuật toán tối ưu hóa và dữ liệu. Mục tiêu cốt lõi của ML là kết hợp 4 thành phần này một cách tối ưu nhất để thuật toán có thể thực hiện một tác vụ với hiệu quả như con người hoặc hơn con người.

Để dễ hiểu hơn chúng ta hãy đặt ra một bài toán thực tế $\mathcal{O}$. Giả sử chúng ta muốn dự đoán mức lương của một người dựa trên n yếu tố liên quan như độ tuổi, trình độ học vấn, số năm kinh nghiệm, etc,. Chúng ta có một tập dữ liệu bao gồm m trường dữ liệu của m người bao gồm các yếu tố kể trên và mức lương tương đương.

1.1. Dữ liệu

Đối với vài toán $\mathcal{O}$ các điểm dữ liệu liên quan như độ tuổi, trình độ học vấn, số năm kinh nghiệm etc, là các biến độc lập (independent) của thuật toán ML. Những biến này được biểu diễn dưới dạng một vector $\mathbf{x}$, với $\mathbf{x} = (x_1, x_2, \cdots, x_n) \in \mathbb{R}^n$. Mỗi vector $\mathbf{x}$ đại diện cho một trường dữ liệu của một người trong bài toán $\mathcal{O}$ (hay một observation trong bài toán tổng quát) với $x_1, x_2, ..., x_n$ là các biến độc lập đại diện cho mỗi yếu tố liên quan (features).

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad (1)$$

Một vector sẽ được thể hiện dưới dạng một cột trong công thức (1). Một tập dữ liệu chứa m trường dữ liệu của n biến độc lập sẽ được biểu diễn dưới dạng một ma trận $\mathbf{X}$ bao gồm n dòng và m cột:

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,m} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,m} \end{bmatrix} \quad (2)$$

Trong đó $x_{i,j}$ với $1 \le i \le n$ và $1 \le j \le m$ là biến độc lập thứ i của trường dữ liệu thứ j trong bài toán $\mathcal{O}$.

Chúng ta có y là biến phụ thuộc, trong bài toán $\mathcal{O}$ sẽ là mức lương của một người. Với một tập dữ liệu với m trường dữ liệu ta sẽ có một vector $\mathbf{y}$:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \quad (3)$$

Trong đó y_j với $1 \le j \le m$ là biến phụ thuộc thứ j và là mức lương của người thứ i trong bài toán $\mathcal{O}$.

1.2. Mô hình

Mô hình (model) trong một thuật toán ML là một phương trình toán học (một hàm) cho phép thể hiện mối quan hệ giữa dữ liệu đầu vào và dữ liệu đầu ra. Mục tiêu cuối cùng của các thuật toán ML là tìm ra một mô hình hay một hàm f có thể ước lượng một cách chính xác nhất biến số đầu ra với một trường dữ liệu đầu vào cho trước.

$$\mathbf{y} = f(\mathbf{x}, \theta) \quad (4)$$

Với θ là vector của các trọng số (coefficients, parameters, weights) của các biến dữ liệu đầu vào $\mathbf{x}$.

Ban đầu, thuật toán sẽ khởi tạo một hàm f bằng cách chọn các trọng số một cách bất kỳ hoặc theo một nguyên tắc nào đó sao cho tối ưu nhất với bài toán. Sau đó thuật toán sẽ sử dụng các biến dữ liệu đầu vào $\mathbf{x}_i$ của tập dữ liệu huấn luyện (training data) để ước lượng biến số đầu ra $\hat{y}_i$. Giữa giá trị ước lượng $\hat{y}_i$ và giá trị thực của biến đầu ra y_i tương ứng sẽ có sự sai lệch. Để hàm f có thể ước lượng một cách chính xác nhất các giá trị y trong tương lai thì thuật toán cần tìm ra các trọng số θ sao cho có thể tối thiểu hóa sai lệch giữa giá trị ước lượng $\hat{y}_i$ và giá trị thực y_i.

1.3. Cost function

Cost function hay Loss function trong một số tài liệu là hàm toán để tính ra sai lệch giữa giá trị ước lượng $\hat{y}_i$ và giá trị thực y_i tương ứng với mỗi trường dữ liệu $\mathbf{x}_i$. Trong một số tài liệu Cost function và Loss function có ý nghĩa như nhau, nhưng trong một số tài liệu khác Loss function dùng để chỉ hàm tính sai lệch giữa ŷ và y của từng trường dữ liệu còn Cost function là hàm tính trung bình cộng của tất cả các loss function. Trong loạt bài của mình, để có thể diễn giải một cách cụ thể hơn, mình sẽ coi cost function và loss function là hai khái niệm tách biệt.

Hàm loss function của một thuật toán ML không có một dạng tổng quát mà hoàn toàn dựa trên sự lựa chọn của người phát triển thuật toán và dựa trên bài toán mà thuật toán muốn giải quyết. Hàm loss function:

$$\mathcal{L}(\hat{y}_j, y_j) \quad (5)$$

Với m trường dữ liệu ta sẽ có hàm cost function như sau:

$$\mathcal{J}(\theta) = \frac{1}{m}\sum_{j=1}^{m} \mathcal{L}(\hat{y}_j, y_j) \quad (6)$$

Hàm cost function (6) được gọi là hàm cost function của trọng số θ, tức là thuật toán sẽ tìm ra các trọng số θ sao cho hàm $\mathcal{J}$ đạt giá trị tối thiểu.

1.4. Thuật toán tối ưu hóa— Optimization algorithms

Về mặt toán học, tối ưu hóa là một bộ môn toán khá rộng với nhiều phương pháp tối ưu hóa khác nhau. Các thuật toán tối ưu hóa là các phương pháp toán học nhằm tìm ra điểm tối ưu (cực tiểu hoặc cực đại) của một phương trình toán học.

Trong machine learning, các thuật toán tối ưu hóa được dùng để tìm ra điểm tối thiểu của cost function $\mathcal{J}$ và từ đó tìm ra các trọng số θ tương ứng. Convex và Gradient Descent là những thuật toán được sử dụng phổ biến nhất trong các thuật toán machine learning.

Trên đây là những thành phần chính của các thuật toán Machine Learning. Từng thành phần sẽ được phân tích và giới thiệu kỹ khi đi vào từng thuật toán cụ thể trong các phần sau của mục Machine Learning Tutorial.

1.5. Supervised Learning và Unsupervised Learning

Supervised Learning và Unsupervised Learning là hai phân nhóm cơ bản của Machine Learning, không kể đến Reinforcement Learning là một phương pháp tiếp cận hoàn toàn khác. Phần lớn các thuật toán Machine Learning phổ biến đều nằm trong hai phân nhóm này.

Trong các thuật toán Supervised Learning, (các) trường dữ liệu đầu vào x_i liên kết với các dữ liệu đầu ra y_i. Thuật toán Supervised Learning sẽ tìm một phương trình phù hợp nhất để liên kết dữ liệu đầu vào và dữ liệu đầu ra. Bài toán $\mathcal{O}$ là một trong những ví dụ về Supervised Learning.

Ngược lại, trong các thuật toán Unsupervised Learning chỉ có (các) trường dữ liệu đầu vào x_i và không có dữ liệu đầu ra liên kết y_i để giám sát (supervise) - vì vậy các thuật toán này được gọi là Unsupervised.

1.6. Dữ liệu định tính và định lượng

Các dữ liệu trong Machine Learning có thể được phân loại thành dữ liệu định tính và dữ liệu định lượng. Dữ liệu định lượng là dữ liệu dưới dạng số như độ tuổi, khoảng cách, mức lương, etc,. Trong khi đó dữ liệu định tính trong Machine Learning là những dữ liệu dạng phân nhóm như giới tính, loại bệnh, trường đại học theo học, etc,.

Trong các bài toán Regression, dữ liệu thường là dữ liệu định lượng còn trong các bài toán Classification, dữ liệu thường là dữ liệu định tính. Tuy nhiên sự phân biệt này không mang tính tuyệt đối.

2. Linear Regression

Regression và Classification là hai bài toán cơ bản và quan trọng nhất trong Supervised Learning. Trong đó, Classification là bài toán phân loại dữ liệu còn Regression (hồi quy) là bài toán tìm mối liên hệ giữa các dữ liệu. Trong bài đầu tiên, chúng tôi sẽ giới thiệu những vấn đề cơ bản về dạng quan hệ dữ liệu đơn giản nhất: tuyến tính (linear).

Một phương trình tuyến tính tổng quát có dạng:

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n \quad (7)$$

Phương trình (7) biểu diễn mối quan hệ tuyến tính của n biến độc lập x_i và biến phụ thuộc y. Trong Simple Linear Regression, phương trình (7) đơn giản hóa chỉ còn:

$$y = \beta_0 + \beta_1 x \quad (8)$$

Trong phương trình (8):

- Tham số β_1 được gọi là the slope of the line hay ngắn gọn là slope. Đây là một trong những định lượng quan trọng nhất trong linear regression. $\beta_1 \approx 0$ cho thấy không có mối quan hệ giữa hai biến x và y. $\beta_1 > 0$ cho thấy mối quan hệ đồng

biến còn $\beta_1 < 0$ cho thấy mối quan hệ nghịch biến.
- Tham số β_0 được gọi là the intercept of the line hay ngắn gọn là intercept.

2.1. Simple Linear Regression

Xét bài toán $\mathcal{K}$ như sau: Trường Đại học Kinh tế Quốc dân năm 2017 có 4000 sinh viên mới. Bằng cách nào đó, chúng ta thu thập được dữ liệu về điểm tổng kết cấp 3 và (tổng) điểm thi đại học của 500 sinh viên. Yêu cầu đặt ra là tìm ra mối liên hệ giữa hai dữ liệu này để có thể đưa ra dự báo điểm thi đại học dựa trên điểm trung bình tổng kết cấp 3.

Trước tiên chúng ra cần giả định mối liên hệ giữa điểm tổng kết cấp 3 và điểm thi đại học có mối quan hệ tuyến tính – ***linear relationship***. Bởi chỉ khi hai dữ liệu có mối quan hệ này thì mới có thể sử dụng linear regression để giải quyết bài toán.

Với giả định trên, điểm thi đại học y_i của mỗi sinh viên có thể được biểu diễn bằng một phương trình của điểm tổng kết cấp 3 x_i có dạng:

$$y_i = \beta_0 + \beta_1 x_i \quad (8)$$

Khi thay mỗi biến độc lập x_i vào phương trình (8), chúng ta sẽ nhận được một giá trị ước lượng $\hat{y}_i$. Do phương trình (8) không thể biểu diễn một cách hoàn toàn chính xác mối quan hệ giữa x_i và y_i mà sẽ có một sai lệch nhỏ e_i.

$$e_i = y_i - \hat{y}_i \quad (9)$$

e_i được gọi là ***residual error***.

Phương trình (8) được gọi là tối ưu với tập dữ liệu của bài toán $\mathcal{K}$ khi các ước lượng $\hat{y}_i$ có residual error nhỏ nhất có thể hay nói cách khác, cần tìm $\hat{\beta}_0$ và $\hat{\beta}_1$ để phương trình:

$$\mathcal{L} = \sum_{i=1}^{n} {e_i}^2 \quad (10)$$

$$\Leftrightarrow \mathcal{L} = \sum_{i=1}^{n} \left(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i\right)^2 \quad (11)$$

Đạt tối thiểu. Phương trình (11) được gọi là sum of square residual (SSR). Quá trình tìm giá trị $\hat{\beta}_0$ và $\hat{\beta}_1$ để phương trình (11) tối thiểu gọi là least squares estimation. Ta sẽ có:

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \quad (12)$$

Và:

$$\hat{\beta}_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} \quad (13)$$

Phương trình (9) và (10) trong machine learning đóng vai trò lần lượt là lost function và cost function.

2.2. Error và Residual

Phương trình:

$$y = \hat{\beta}_0 + \hat{\beta}_1 x \quad (14)$$

Với $\hat{\beta}_0$ và $\hat{\beta}_1$ được tính như công thức (12) và (13) là tối ưu với tập dữ liệu của bài toán $\mathcal{K}$. Khi đó đồ thị của phương trình (14) thường được gọi là fitted line (hay fitted surface với nhiều biến độc lập x_i). Tuy nhiên phương trình (14) chỉ tối ưu đối với dữ liệu của 500 sinh viên được thu thập. Nếu chúng ta thực hiện lấy dữ liệu của 500 sinh viên khác, các giá trị $\hat{\beta}_0$ và $\hat{\beta}_1$ sẽ khác đi. Giả sử, bằng cách nào đó, chúng ta thu thập dữ liệu của toàn bộ tổng thể 4000 sinh viên (điều này là có thể trong thực tế vì tổng thể nhỏ và đã biết, nhưng hiếm khi có một tổng thể như vậy). Khi đó:

$$y = \beta_0 + \beta_1 x \quad (15)$$

sẽ là fitted line của tổng thể hay còn được gọi là population regression line. Khi đó $\hat{\beta}_0$ là sample intercept và là ước lượng của population intercept β_0 và $\hat{\beta}_1$ là sample slope và là ước lượng của population slope β_1.

μ được gọi là trung bình (mean) của tổng thể, khi đó statistical error (disturbance) ε_i là sự sai lệch giữa giá trị quan sát thực tế và mean:

$$\varepsilon_i = y_i - \mu$$

Trong Linear Regression, hai khái niệm statistical error và residual error rất dễ bị nhầm lẫn. Hai khái niệm này có điểm tương đồng nhưng về bản chất là khác nhau và cần phân biệt rõ.

Trong Linear Regression, ε_i được giả định là phân phối chuẩn: $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. Đây là giả định thứ hai của Linear Regression.

Giả định thứ 3 của Linear Regression là ε_i là độc lập (Independent). Giả định thứ 4 là ε_i là như nhau tại mọi biến độc lập x_i (Equal variances).

2.3. Variance và Coefficient of Determination

Theo giả định của linear regression, residual error ε_i phân phối chuẩn: $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. ***Variance*** của residual error σ^2 sẽ cho thấy mức độ sai lệch của giá trị quan sát thực tế y xung quanh fitted line. Nếu σ^2 nhỏ, các tham số β_0, β_1 là đáng tin cậy và do đó, các kết quả ước lượng từ phương trình (15) là đáng tin cậy.

Tuy nhiên hiếm khi chúng ta biết được fitted line của tổng thể, do đó chỉ có thể ước lượng phương sai của residual error của tổng thể σ^2 từ mẫu dữ liệu đã có. Một trong những cách ước lượng phương sai residual error của tổng thể là ***mean square error***:

$$\sigma^2 = S^2 = \frac{Sum\ of\ squared\ error}{n - number\ of\ parameter} \quad (16)$$

$$= \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n-2} \quad (17)$$

Trong đó S là ***residual standard error*** với $S = \sqrt{\sigma^2}$.

Coefficient of Determination r^2 là giá trị phản ánh tỷ lệ biến phụ thuộc y có thể được giải thích bởi biến độc lập x bằng phương trình model đã chọn (trong bài toán $\mathcal{K}$ là phương trình (8)). Coefficient of Determination được tính theo công thức sau:

$$r^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST} \quad (18)$$

Trong đó SSE là sum of squared error, SSR là sum of squared regression và SST là sum of squared total. Phương trình (18) tương đương:

$$r^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \quad (19)$$

Phương trình (19) được gọi là Ordinary R-squared. Ngoài ra, còn một dạng r^2 nữa là Adjusted R-squared:

$$r^2_{adj} = 1 - \left(\frac{n-1}{n-p}\right)\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \quad (20)$$

Với p là số lượng tham số trong phương trình model – trong bài toán $\mathcal{K}$ thì $p = 2$.

Variance σ^2 và Coefficient of Determination r^2 là hai giá trị thường được sử dụng để đánh giá độ tối ưu (fitted) của phương trình model đã lựa chọn. Variance σ^2 hay thường được gọi là MSE (và root MSE cho residual standard error) ước lượng độ biến thiên của residual – unfit – của phương trình. r^2 đánh giá khả năng dự đoán – fit – của phương trình model.

2.4. Mutiple Linear Regression

Sau khi hiểu về Simple Linear Regression, chúng ta sẽ chuyển sang Multiple Linear Regression. Từ Multiple ở đây có nghĩa rằng phương trình model sẽ có hai hoặc nhiều hơn hai biến độc lập. Hầu hết những thông tin chúng ta đã tìm hiểu về Simple Linear Regression cũng đúng với Multiple Linear Regression.

Phương trình model cho tổng thể của Multiple Linear Regression có dạng:

$$y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \cdots + \beta_n x_{i,n} + \varepsilon_i \quad (21)$$

Phương trình (21) biểu diễn mối quan hệ tuyến tính giữa n biến độc lập x và biến phục thuộc y (giả định giữa chúng có mối quan hệ tuyến tính). Residual error ε_i cũng tuân theo 3 giả định còn lại của Linear Regression: độc lập (Independent), phân phối chuẩn (Normal distributed), như nhau tại mọi điểm dữ liệu (Equal variances).

Trong Machine Learning, Multiple Linear Regression sẽ phải giải quyết những bài toán với số lượng rất lớn, sẽ là hiệu quả hơn nếu sử dụng vector và matrix để biểu diễn phương trình (21). Với một tập dữ liệu gồm m trường dữ liệu, các thành phần của phương trình (21) được biểu diễn dưới dạng vector và matrix như sau:

Biến phụ thuộc

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \quad (22)$$

Biến độc lập

$$X = \begin{bmatrix} 1 & x_{1,1} & x_{2,1} & \cdots & x_{m,1} \\ 1 & x_{1,2} & x_{2,2} & \cdots & x_{m,2} \\ 1 & x_{1,3} & x_{2,3} & \cdots & x_{m,3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1,n} & x_{2,n} & \cdots & x_{m,n} \end{bmatrix} \quad (23)$$

Tham số

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix} \quad (24)$$

Residual error

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_m \end{bmatrix} \quad (25)$$

Khi đó, phương trình (21) sẽ trở thành:

$$Y = \boldsymbol{\beta} X + \boldsymbol{\varepsilon} \quad (26)$$

Tiếp tục sử dụng phương trình sum squared error:

$$\begin{aligned} \mathcal{L} &= \sum_{i=1}^{n} e_i^2 \\ &= \sum_{i=1}^{n} (Y - \beta X)^2 \\ &= (Y - \boldsymbol{\beta} X)^{\mathbf{T}} (Y - \boldsymbol{\beta} X) \quad (27) \end{aligned}$$

Kết quả của least squares estimation sẽ cho ta giá trị ước lượng của $\boldsymbol{\beta}$ là $\widehat{\boldsymbol{\beta}}$:

$$\widehat{\boldsymbol{\beta}} = (X^{\mathrm{T}} X)^{-1} X^{\mathrm{T}} Y \quad (28)$$

Tương tự như Simple Linear Regression, variance σ^2 và Coefficient of Determination r^2 được sử dụng để đo lường sự tối ưu của phương trình với tham số $\widehat{\boldsymbol{\beta}}$ đã được tính trong phương trình (28).

Cũng giống như Simple Linear Regression, variance σ^2 có thể được ước lượng bởi công thức:

$$\sigma^2 = S^2 = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n - k} \quad (29)$$

$$= \frac{\boldsymbol{e}^{\mathrm{T}} \boldsymbol{e}}{n - k} \quad (30)$$

Với $\boldsymbol{e}$ là residual error dưới dạng matrix, $\boldsymbol{e}^{\mathrm{T}}$ là transpose matrix của $\boldsymbol{e}$ và k là số lượng tham số trong phương trình model.

Coefficient of Determination r^2 được ước lượng theo phương trình (18).

$$r^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$$

$$= \frac{\boldsymbol{e}^{\mathrm{T}} \boldsymbol{e}}{Y^{\mathrm{T}} N Y} \quad (31)$$

Trong đó N là matrix được định nghĩa như sau:

$$N = I - \frac{1}{n} i^{\mathrm{T}} i \quad (32)$$

Với I là identity matrix, i là matrix $n \times 1$ của n số 1 và i^{T} là transpose matrix của i.

Trong các số tiếp theo, Machine Learning tutorial sẽ tiếp tục giới thiệu về Regression và các phương pháp để tính toán ra tham số β của phương trình model./

Python trong Machine Learning

Trong mục Python trong Machine Learning, chúng tôi sẽ giới thiệu những kiến thức lập trình Python, những thư viện Python được sử dụng để lập trình phát triển và nghiên cứu Machine Learning và Trí tuệ Nhân tạo. Các tài liệu về Python cơ bản đã có rất nhiều trên mạng, các bạn có thể tìm hiểu thêm. Mục Python trong Machine Learning sẽ đề cập chủ yếu đến những kiến thức liên quan tới lập trình phát triển Machine Learning.

Vì sao lại dùng Python

Python được sử dụng rất nhiều bởi các nhà khoa học dữ liệu, lập trình viên phát triển Machine Learning và Trí tuệ Nhân tạo. Có rất nhiều lý do cho sự yêu thích này, nhưng đơn giản là 3 cái dễ: dễ học, dễ sử dụng và dễ tích hợp.

Tất nhiên bạn vẫn có thể sử dụng các ngôn ngữ khác để phát triển Machine Learning và Trí tuệ Nhân tạo, tuy nhiên đó chỉ là việc nên làm khi bạn đã nắm vững kiến thức về Machine Learning, bởi vì hầu hết các tài liệu hay khóa học về Machine Learning hay Deep Learning hiện nay đều sử dụng Python làm ngôn ngữ lập trình chính thức.

Python được download miễn phí tại www.python.org. Ngoài ra, các bạn còn có thể sử dụng Jupyter Notebook thay thế. Đây là ứng dụng nền tảng web cho phép bạn lập trình và thực thi nhiều ngôn ngữ lập trình, trong đó có Python. Trong bài viết đầu tiên, chúng tôi sẽ điểm lại những kiến thức Python cơ bản cần nắm chắc để thực hiện lập trình với Machine Learning.

1. Dữ liệu

Python hỗ trợ hầu hết các kiểu dữ liệu cơ bản như các ngôn ngữ lập trình khác như *boolean* (bool), *long, integers* (int), *floating number* (float), *complex, string, bytes*, và dữ liệu dạng *mảng* (array). Chúng ta sẽ đi sâu vào các loại dữ liệu dạng *mảng* vì vai trò của chúng trong machine learning và vì những kiểu dữ liệu cơ bản không có gì khác biệt so với các ngôn ngữ lập trình khác. Dữ liệu dạng mảng được chia làm 4 dạng chính là ***immutable table*** hay ***tuple***, ***dynamic tables*** hay ***list***, ***dictionnary*** hay ***dict*** và ***set***. Ngoài ra chúng ta sẽ đề cập đến string vì cách Python xử lý dữ liệu chuỗi có khác so với ngôn ngữ khác.

1.2. List

List là một danh sách các giá trị, có tính chất giống như mảng. Mỗi giá trị trong list được đánh chỉ mục, giá trị đầu tiên có chỉ mục là 0, và cứ như thế.

```
List = ['tap chi',1,5.,2+0j]
// list được khởi tạo với dấu []
// hoặc bằng hàm list
a = list('tap chi', 1, 5., 2+0j)
```

Chỉ mục và sliced của list:

```
letters = ['a','b','c','d','e']
print(letters[2])
// print c
print(letters[-1])
// print e
print(letters[0:2])
// print ['a', 'b', 'c']
```

List trong Python là mutable, tức bạn có thể thêm giá trị, bớt giá trị hay thay đổi giá trị trong một list của Python.

```
number = [1,2,3]
// add more value to list
number.append(4) // hoặc
number +=[4]
// number = [1,2,3,4]
// add a list to another list
number.extend([6,7])
// number = [1,2,3,4,6,7]
// insert a value at a given index
number.insert(4,5)
// number = [1,2,3,4,5,6,7]
// remove a value from list
number.remove(6)
// number = [1,2,3,4,5,7]
// xóa giá trị ở chỉ mục đã cho
number.pop([4])
// number = [1,2,3,4,7]
// nếu không có giá trị chỉ mục, hàm
sẽ xóa giá trị cuối cùng
number.pop()
//number = [1,2,3,4]
//mở rộng list
number = number*2
// number = [1,2,1,2]
```

Trong Python, một list nằm trong một list sẽ được gọi là nested list:

```
// nested list
nlist = ['a','b',[1,2]]
```

Nested list là cách thể hiện matrix trong Python:

```
// matrix 3x4
matrix = [[1,2,3,4],[5,6,7,8],[9,10,
11,12]]
// matrix = [[1,2,3,4]
             [5,6,7,8]
             [9,10,11,12]]
```

List comprehension là một phương pháp xác để tạo list, với mỗi thành phần của list là kết quả của các tính toán dựa trên các thành phần của một chuỗi khác:

```
// list comprehension
list = [x**2 for x in range(5)]
// list = [0,1,4,9,16,25]
list = [(x,y) for x in [1,3,5] for y
in [3,5,7] if x!=y]
//  list  =  [(1,3),(1,5),(1,7),(3,
5),(3,7),(5,7)]
```

Ngoài ra, list còn có các hàm dựng sẵn `index()` – trả về chỉ mục của giá trị, `count()` – trả về số lần giá trị đã cho xuất hiện trong list, `sort()` – sắp xếp giá trị trong list và `reverse()` – đảo ngược thứ tự giá trị trong list.

1.2. Tuple

Tuple cũng giống như list nhưng tuple chiếm ít bộ nhớ hơn và nhanh hơn. Các giá trị trong tuple được khởi tạo ban đầu sẽ giữ nguyên cho tới cuối chương trình. Nested tuple có thể được dùng như keys-values cho dictionary.

```
// tuple được khai báo với dấu ()
tuple = ('tap tri',23,0.5)
// hoặc với hàm tuple
tuple = tuple('tri tue',4,0.6)
// hoặc chỉ với giá trị
tuple = 'nhan tao', 3, 9
// tuple có 1 giá trị được khởi tạo
tuple = (1,)
```

Tuy bạn không thế thay đổi các giá trị đã khai báo trong tuple nhưng bạn có thể thêm các giá trị mới vào tuple như sau:

```
letters = ('a','b','c')
letters += ('d',)
// letters = ('a', 'b', 'c', 'd')
number = (1,2)
multi_number = number*3
//multi_number = (1,2,1,2,1,2)
```

Tuple là một object đơn giản trong Python với chỉ 2 hàm dựng sẵn là `index()` – trả về chỉ mục của giá trị và `count()` – trả về số lần xuất hiện của giá trị trong tuple. Tuple cũng có thể được sliced như list. Ngoài ra tuple còn có dùng để gán giá trị cho biến, được gọi là tuple unpacking:

```
tuple = (1,2,3)
x,y,z = tuple
// hoặc
(x,y,z) = (1,2,3)
// x = 1, y = 2, z = 3
```

Tuple là immutable nhưng không hoàn toàn, nếu thành phần của tuple là mutable thì chúng vẫn có thể được chỉnh sửa:

```
tuple = (1,2,[3,4])
tuple[2][0] = 10
// tuple = (1,2,[10,4])
```

1.3. Dictionary

Dict cũng giống như list nhưng mỗi giá trị sẽ có key và value tương ứng. Bạn cũng có thể thêm, bớt và chỉnh sửa các giá trị như list:

```
// dict được khai báo với dấu {}
dict = {'name':'tapchiai','issue':2}
// hoặc với hàm dict
dict = dict('name':'tapchiai',
'issue':2)
// hoặc từ tuple với hàm dict
dict = dict([('name','tapchiai'),
('issue',2)])
// hoặc từ tuple với hàm fromkeys
seq = ('name','issue')
dict2 = dict.fromkeys(seq)
// dict2 =
{'name':None,'issue':None}
dict2 = dict.fromkeys(seq,10)
// dict2 = {'name':10,'issue':10}
```

Dict có một số hàm dựng sẵn khác với list và tuple:

```
dict = {'type':'tap chi'}
// tạo thành phần mới với giá trị
cho sẵn (None nếu không có)
dict.setdefault('name','tapchiai')
// dict = {'type':'tap chi', 'name':
'tapchiai'}
// xóa mọi thành phần trong dict
dict.clear()
// dict = {}
// kết hợp 2 dict
d1 = {'a':1}
d2 = {'a':2,'b':3}
d1.update(d2)
// d1 = {'a':2,'b':3}
```

Các hàm để truy vấn thông tin trong dict:

```
dict = {'type':'tap chi','name':
'tapchiai'}
print(dict.items())
// print [('type','tap chi'),
('name', 'tapchiai')]
print(dict.keys())
// print ['type','name']
print(dict.values())
// print ['tap chi','tapchiai']
// tìm một key cụ thể trong dict
dict.has_key('type')
// print True
// trả về value với key cho trước
dict.get('type')
// print 'tap chi'
//trả về value với key cho trước và
xóa cặp key-value đó
dict.pop('type')
// print 'tap chi'
// và dict = {'name':'tapchiai'}
// trả về một cặp key-value bất kỳ
vì dict không có thứ tự rồi xóa
chúng
dict.popitem()
```

1.4. Set

Set cũng là danh sách các giá trị giống như list nhưng các giá trị không được đánh chỉ mục. Các giá trị trong một set bắt buộc có tính chất hashable, do đó bạn không thể thay đổi các giá trị trong set, tức immutable.

```
Set = ['tri','tue','nhan','tao']
// set được khởi tạo với cặp dấu []
// hoặc với hàm set
set =
set(['tri','tue','nhan','tao'])
//Từ Python 2.7 về sau, set có thể
được khởi tạo với dấu {}
Set = {'tri','tue','nhan','tao'}
```

Set cho phép sử dụng các phép toán để biến đổi và kết hợp các set:

```
x = set([1,2,3,4])
y = set([3,4,5,6])
// Kết hợp set
x|y // {1,2,3,4,5,6}
// Giá trị trùng giữa các set
x&y // {3,4}
// hoặc
z = x.intersection(y) // {3,4}
// Lấy giá trị không trùng lặp
x ^ y // {1,2,5,6}
// hoặc
x.symmetric_difference(y)
// {1,2,5,6}
```

```
// Kiểm tra subset
x < y // False
// hoặc
z.issubset(x) // True
// hoặc
z <= y // True
// Lấy giá trị x không thuộc y
x - y // {1,2}
// hoặc
x.difference(y) // {1,2}
```

1.2. String

String trong Python được khai báo với cặt dấu ngoặc đơn hoặc ngoặc kép:

```
str_1 = 'tap chi tri tue nhan tao'
str_2 = "machine learning"
```

Trong Python, string được xử lý như là một list của các ký tự. Do đó:

```
str_1 = 'tap chi tri tue nhan tao'
print(str_1[0])
// print t
print(str_1[0:3])
// print tap
print(str_1[:7])
// print tap chi
print(str_1[8:])
// print tri tue nhan tao
```

Các kỹ thuật xử lý string cơ bản:

```
firstname = "Harry"
lastname = "Potter"
// Add string
print(firstname + ' ' + lastname)
// print Harry Potter
// Multiply string
print(firstname * 3)
// print Harray Harry Harry
```

```
// Search string in string
print('ar' in firstname)
// print True
// Split string
fullname = "Albus Percival Wulfric
Brian Dumbledore"
print(fullname.split(' ').[0])
//print Albus
print(fullname.split(' ').[-1])
// print Dumbledore
```

Mutability là khả năng thay đổi giá trị sau khởi tạo của dữ liệu. List và Dictionary là mutable tức có thể thêm, bớt và chỉnh sửa dữ liệu trong mảng. Set và Tuple là immutable.

Order, List và Tuple sẽ giữ nguyên thứ tự giá trị như thứ tự khởi tạo, còn Set và Dictionary thì không.

Duplication, List và Tuple cho phép lặp lại các giá trị trong mảng trong khi trong Set và Dictionary mỗi giá trị (key trong trường hợp của Dictionary) là duy nhất.

2. Câu lệnh và hàm cơ bản

*2.1. Câu lệnh **if***

```
x = int(raw_input("Nhập số:"))
if x < 0:
      print "Số âm"
elseif x == 0:
      print "Số 0"
else:
      print "Số dương"
```

*2.2. Hàm **range()***

Hàm `range()` là một hàm dựng sẵn trong Python cho phép khởi tạo một chuỗi số. Hàm range có 3 tham số: điểm khởi đầu (mặc định là 0), điểm kết thúc và bước nhảy (mặc định là 1).

```
range(5)
// [0,1,2,3,4]
range(2,5)
// [2,3,4]
range(70,100,10)
// [70,80,90]
```

Điểm kết thúc sẽ không nằm trong chuỗi giá trị được tạo ra. Các bước nhảy cũng có thể đặt là giá trị âm.

*2.3. Câu lệnh **for***

Câu lệnh `for` được dùng để lặp qua một chuỗi các dữ liệu.

```
seq = ['tri','tue','nhan','tao']
for s in seq:
      print s
// tri
// tue
// nhan
// tao
// kết hợp với hàm range()
for i in range(5):
      print i*2
// 0
// 2
// 4
// 6
// 8
// kết hợp với chuỗi và hàm range()
for i in range(len(seq)):
      print seq[i]
// tri
// tue
// nhan
// tao
```

```
seq = ['tri','tue','nhan','tao']
for s in seq:
      print s
// tri
// tue
// nhan
// tao
// kết hợp với hàm range()
for i in range(5):
      print i*2
// 0
// 2
// 4
// 6
// 8
// kết hợp với chuỗi và hàm range()
for i in range(len(seq)):
      print seq[i]
// tri
// tue
// nhan
// tao
```

Điểm đặc biệt của vòng lặp `for` trong Python là nó có thể đi cùng lệnh `else` như lệnh `if`. Tuy nhiên cách Python thực hiện lệnh else trong vòng lặp `for` có sự khác biệt.

```
n = [1]
if n:
      print "True"
else:
      print "False"
// False
for i in n:
      print "True"
else:
      print "False"
// True
// False
```

Nói ngắn gọn, dòng lệnh sau lệnh `else` của vòng lặp `for` sẽ được thực hiện khi vòng lặp hoàn thành. Khi đó tất cả các dòng lệnh trong vòng lặp `for` vẫn được thực hiện. Trừ khi vòng lặp `for` có câu lệnh `break`.

```
x = 3
seq = [1,2,3,4]
for i in seq:
      if x == i:
            print “ting”
else:
      print ‘tong’
// ting
// tong
for i in seq:
      if x == i:
            print “ting”
            break
else:
      print ‘tong’
// ting
```

Câu lệnh `continue` cho phép bỏ qua các dòng lệnh phía sau lệnh `continue` và cho phép vòng lặp tiếp tục với vòng lặp kế tiếp.

```
seq = [1,2,3,4,5,6]
for i in seq:
      if x == 4:
            continue
      print i
// 1
// 2
// 3
// 5
// 6
```

Câu lệnh pass cho phép bỏ qua một dòng lệnh cần có nhưng không có tác dụng trong việc thi hành mã.

```
seq = [1,2,3,4]
for i in seq:
      if x == 2:
            pass
      print i
// 1
// 2
// 3
// 4
```

Nhìn có vẻ câu lệnh pass hoàn toàn vô dụng nhưng trong thực tế pass có thể cho phép bạn lờ đi các exception hoặc lỗi vô hại của mã nguồn mà không ảnh hưởng gì đến quá trình thực thi code.

2.4. Hàm vô danh Lambda

Hàm `lambda` là một hàm rất đặc biệt trong Python cho phép tạo ra các hàm vô danh ngay trong quá trình chạy code

```
def tinh_cong(x):
      return x+1
print tinh_cong(3)
// 4
g = lambda x: x+1
print g(3)
// 4
```

`lambda` thể hiển rõ tính hữu dụng khi sử dụng với các hàm như `filter()`,`map()` hay `reduce()`:

```
seq = [6,7,8,9,10,11,12]
s = filter(lambda x: x%3 == 0,seq)
print s
// [6,9,12]
m = map(lambda x: x-1,seq)
print m
// [5,6,7,8,9,10,11]
n = reduce(lambda x,y: x+y,seq)
print n
// 63
```

3. Thư viện

Vì Python là một ngôn ngữ dòng lệnh nên khi bạn tắt Python shell, toàn bộ những gì bạn đã thực hiện sẽ biến mất. Vì vậy Python cho phép bạn lưu các mã nguồn của mình thành một file .py rồi sau đó load lại mã nguồn để sử dụng. Những files đó được gọi là module.

Để load một module vào phiên làm việc hiện tại, bạn dùng câu lệnh import:

```
import mod
```

Các bạn cũng có thể load một số hàm/câu lệnh nhất định bằng cách:

```
from mod import func
// import hàm func từ module mod
from mod import *
// import tất cả các hàm trừ những hàm bắt đầu với gạch dưới __
```

Một tập hợp có cấu trúc các module tạo thành các package hay library. Một library là một module nhưng một module chưa chắc là một library.

Trong Machine Learning, bạn cần biết 3 thư viện dựng sẵn của Python là math, random và statistics và ít nhất 5 thư viện khác là NumPy, SciPy, Pandas, matplotlib, scikit-learn.

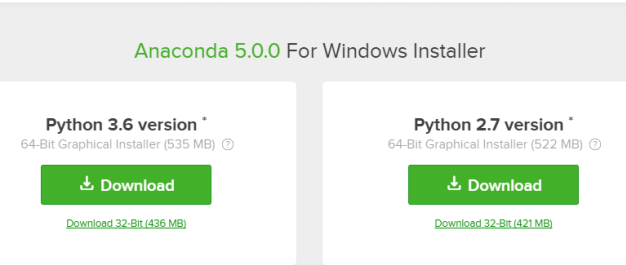
Trong các số sau, Python trong Machine Learning sẽ giới thiệu sâu về những thư viện này tới bạn đọc./

Jupyter Notebook

Như là một cách thuận tiện nhất để cài đặt và thực hành Python, Jupiter Notebook sẽ được sử dụng làm công cụ minh họa thực hành các kiến thức Python trong các số tiếp theo của Tạp chí Trí tuệ Nhân tạo.

Để cài đặt và sử dụng Jupyter Notebook, bạn truy cập www.jupyter.org để download và sử dụng. Bạn sẽ có hai lựa chọn để cài đặt Jupyter Notebook, (1) tải và cài đặt Anaconda, đây là cách đơn giản và được Jupyter Notebook khuyến khích sử dụng,(2) tải và cài đặt Jupyter Notebook thông qua Python's package manager, pip. Hướng dẫn này sẽ sử dụng Anaconda làm công cụ cài đặt.

Bạn truy cập đường dẫn download Anaconda từ trang của Jupyter Notebook.

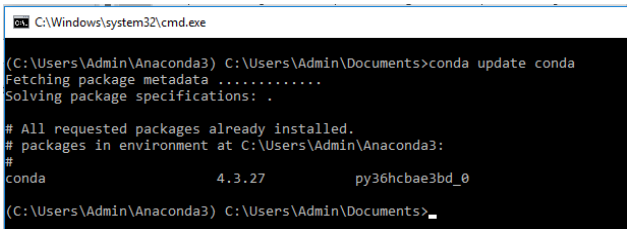


Cho tới thời điểm bài viết này được viết, Anaconda đang hỗ trợ hai phiên bản Python chính là Python 3.6 và Python 2.7. Bạn hãy lựa chọn theo nhu cầu sử dụng, nhưng những ví dụ trong Tạp chí Trí tuệ Nhân tạo sẽ sử dụng Python 3.6. Khi có những vấn đề khác biệt giữa hai phiên bản, chúng tôi sẽ chỉ rõ để các bạn có thể nắm được.

Sau khi cài đặt xong Anaconda, bạn hãy chạy chương trình Anaconda Promt. Sẽ mất một vài giây để chương trình có thể khởi động và hiện ra cửa sổ promt. Tiếp theo chúng ta sẽ update `conda`, là package manager của Anaconda bằng dòng lệnh:

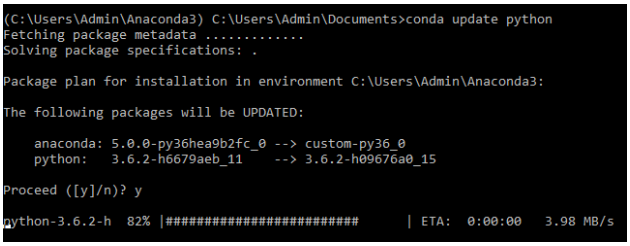
```
conda updata conda
```

Chương trình sẽ tự động kiểm tra các bản cập nhật, nếu có, bạn chọn yes để cập nhật phiên bản `conda` mới.



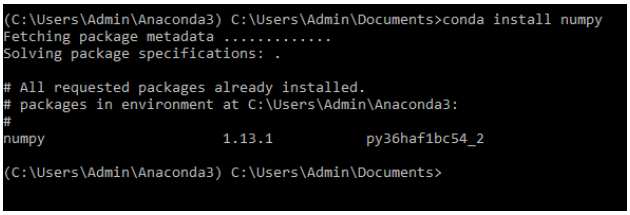
Các bạn thực hiện tương tự để cập nhật `Python`:

```
conda update Python
```



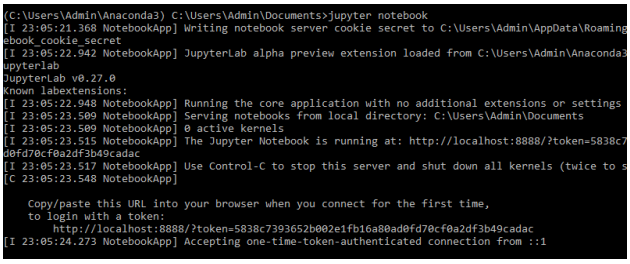
Như vậy là bạn đã cài đặt xong Anaconda và Python để sử dụng. Tuy nhiên, bạn sẽ muốn cài đặt thêm một số thư viện Python để sử dụng về sau. Bạn có thể sử dụng dòng lệnh `conda install` để cài đặt các thư viện bạn muốn:

```
conda install numpy
```

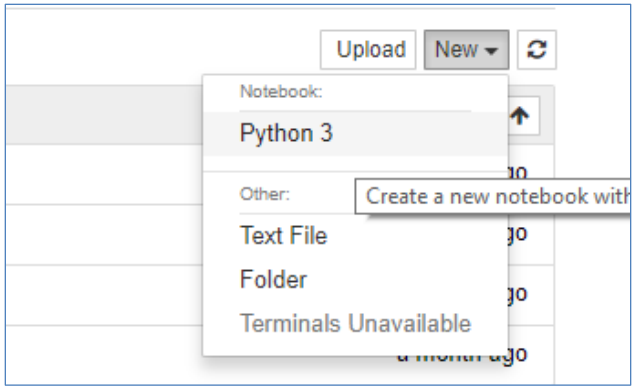


Bây giờ, bạn đã sẵn sàng để sử dụng Jupyter Notebook. Hãy gõ vào Anaconda Promt câu lệnh:

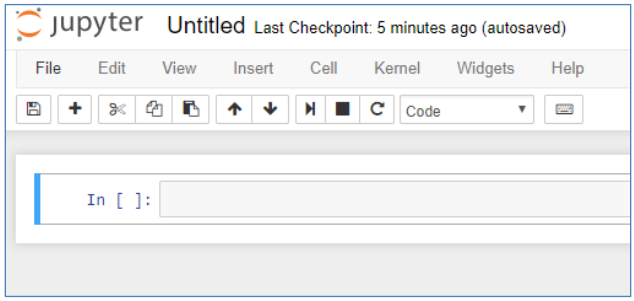
```
jupyter notebook
```



Sau vài giây, chương trình sẽ mở ra một trình duyệt mới dưới dạng danh mục các thư mục và file. Để tạo một jupyter notebook mới, bạn chọn New ⇒ Python 3:



Trình duyệt sẽ mở ra một cửa sổ mới với dạng dòng lệnh như sau:

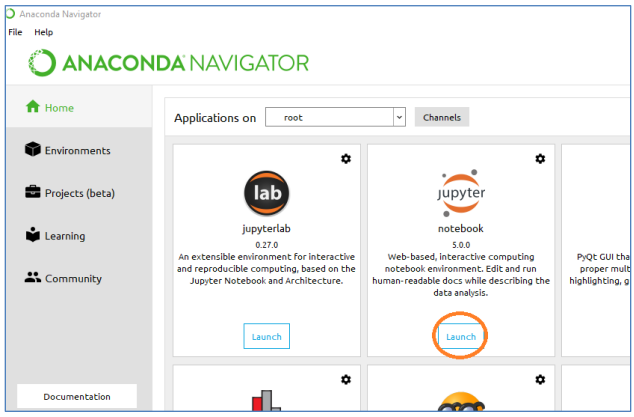


Từ đây, bạn có thể thực hiện các dòng lệnh Python với một giao diện thân thiện hơn, thao tác với file Python như trong trình soạn thảo văn bản WYSIWYG.

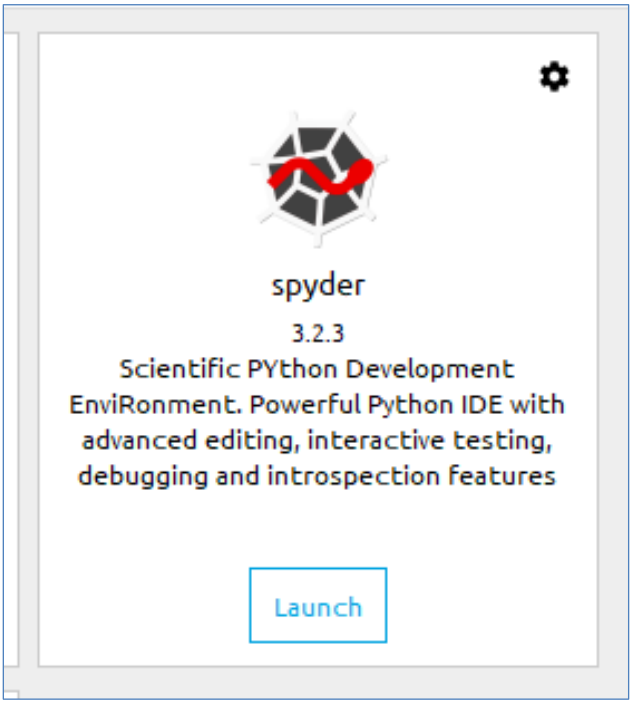
Ngoài ra, Anaconda cũng có thể thực thi ngôn ngữ R, rất thuận lợi cho các bạn làm khoa học dữ liệu đơn thuần. Để cài đặt gói ngôn ngữ R, bạn gõ dòng lệnh sau vào Anaconda Promt:

```
conda install -c r r-essentials
```

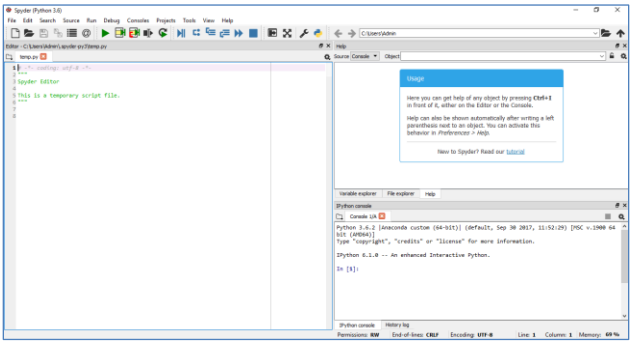
Bạn cũng có thể mở Jupyter Notebook theo một cách khác qua Anaconda Navigator:



Khi đó, Anaconda Navigator cũng mở lên một cửa sổ trình duyệt tương tự như khi mở bằng Anaconda Promt. Ngoài ra, khi sử dụng Anaconda Navigator, bạn có thể truy cập vào spyder, một Python framework khác tương tự như Jupyter Notebook nhưng cho phép tương tác nhiều hơn:



Giao diện của spyder như sau:



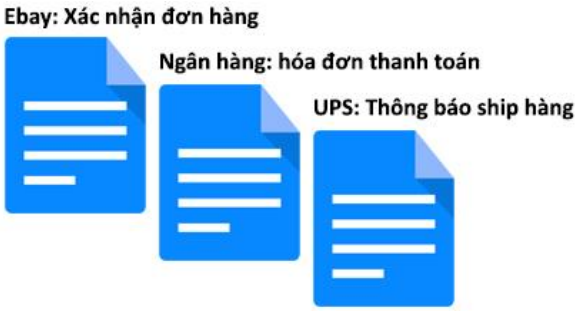
Tuy nhiên, cho tới giờ thì Jupyter Notebook là đủ để cho các bạn thực hành các kiến thức về Python.

2 Pages Paper

Bài toán phân loại email của người dùng thành các nhóm như spam/non-spam hay thành các danh mục như primary/social/promotion mà Google đang thực hiện đã được giải quyết rất tốt. Tiến xa hơn một bước, các nhà cung cấp dịch vụ email muốn dự đoán email người dùng sắp nhận được trong tương lai. Tại hội nghị của Ủy ban Quốc tế World Wide Web năm 2017, một đề tài nghiên cứu mang tên "Email Category Prediction" (Zhang *et al.*, 2017) trình bày về việc sử dụng Machine Learning để dự đoán loại email người dùng sắp nhận được trong tương lai.

1. Bài toán

Ý tưởng chính của đề tài dựa trên việc phần lớn email người dùng nhận được là các email tự động gửi từ các mạng xã hội hay trang thương mại điện tử. Những email này có thể nhóm vào những danh mục (cateogry khác nhau như 'giới thiệu sản phẩm', 'xác nhận đơn hàng', 'hóa đơn thanh toán'. Một số kết quả nghiên cứu trước đó cho thấy những nhóm email này có quan hệ nhân quả (causality relation) (Ailon *et al.*, 2013). Mục tiêu của nghiên cứu của Zhang *et al.* là dựa vào việc phân tích mối quan hệ giữa các email đã nhận được để dự báo loại email người dùng sẽ nhận được trong tương lai.



Hình 1: Ví dụ chuỗi email có quan hệ nhân quả

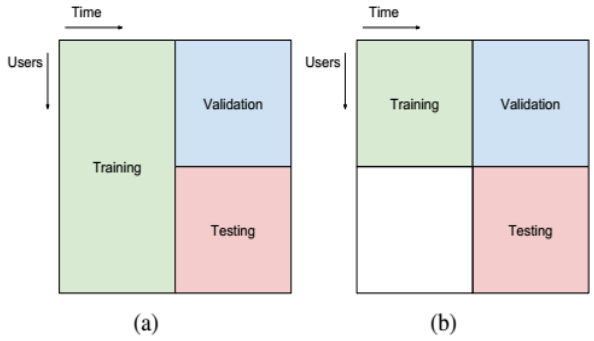
Đề tài này kế thừa và phát triển từ nghiên cứu của Gamzu *et al.* được công bố năm 2015 về việc dự đoán mẫu email (template) mà người dùng sẽ nhận được trong tương lai dựa vào mối quan hệ nhân quả giữa các template.

Bài toán của Zhang *et al.* được cho là đơn giản hơn khi mà các danh mục (categories) là tập hợp của các template có cùng nội dung cơ bản.

2. Dữ liệu

Nghiên cứu sử dụng bộ dữ liệu gồm 2,562,361 email tự động mà 102,743 người dùng nhận được trong khoảng thời gian 90 ngày. Những email này đã được phân loại vào 17 danh mục phù hợp với nội dung thư sử dụng thuật toán phân loại của Google. Tất cả các dữ liệu đều được ẩn danh và chỉ sử dụng hai dữ liệu chính là nhãn thời gian (chỉ thời điểm người dùng nhận email) và danh mục thư.

Đối với mỗi hòm thư, dữ liệu được chia thành hai tập dữ liệu huấn luyện và thử nghiệm ở mốc 45 ngày, với tập huấn luyện là toàn bộ email ở 45 ngày đầu và tập thử nghiệm là các email ở 45 ngày sau.



Hình 2: Phương án phân chia tập dữ liệu

Mỗi email sẽ được gán cho một chỉ mục bước thời gian t được sắp xếp theo thứ tự tăng dần. Các danh mục cũng được gán cho một ID số $\mathcal{A}$ đại diện cho danh mục đó với $\mathcal{A} = \{1,2,\cdots,n\}$. Danh mục của một email cụ thể được ký hiệu là $a \in \mathcal{A}$. Một email được nhận vào thời điểm t sẽ có danh mục $a^{(t)}$ và nhãn thời gian $m^{(t)}$.

Tại mỗi thời điểm dự đoán t, thuật toán sẽ trả về phân phối xác xuất cho mỗi danh mục a, xác xuất mà người dùng sẽ nhận được một email thuộc danh mục a trong một khung thời gian $(m^{(t)}, m^{(t)} + \Delta]$[1]. Khung thời gian Δ có thể là 1 ngày, 2 ngày, etc; trong quá trình tính toán của đề tài, Δ được đặt là 3 ngày. Vì thế kết quả dự đoán tại thời điểm t sẽ là một vector n thành phần $\mathbf{y}^{(t)}$ với thành phần thứ a là likelihood của danh mục a.

[1] Đọc là "lớn hơn $m^{(t)}$ và nhỏ hơn hoặc bằng $m^{(t)} + \Delta$"

3. Thuật toán

Thử nghiệm được thực hiện với 3 thuật toán khác nhau với mục tiêu so sánh kết quả thử nghiệm. Trong đó *k*-dependent Markov chains được sử dụng như cơ sở để so sánh. Đây cũng chính là thuật toán mà Gamzu *et al.* sử dụng để mô hình hóa bài toán trong nghiên cứu hồi năm 2015.

k-dependent Markov chains là phương pháp tính toán xác xuất một sự kiện xảy ra sau một chuỗi k sự kiện liên tiếp. Cho một danh sách $\{a^{(1)}, \cdots, a^{(n)}\}$ của n email mà người dùng nhận được, ta sẽ có các cặp trạng thái:

$$\left\{\left[\left(a^{(i)}, \cdots, a^{(i+k-1)}\right), \left(a^{(i+1)}, \cdots, a^{(i+k)}\right)\right] \forall i \in \{1, \cdots, n-k\}\right\}$$

Từ các cặp trạng thái này có thể xây dựng một "directed graph", trỏ từ trạng thái gốc tới xác xuất của các trạng thái mục tiêu.

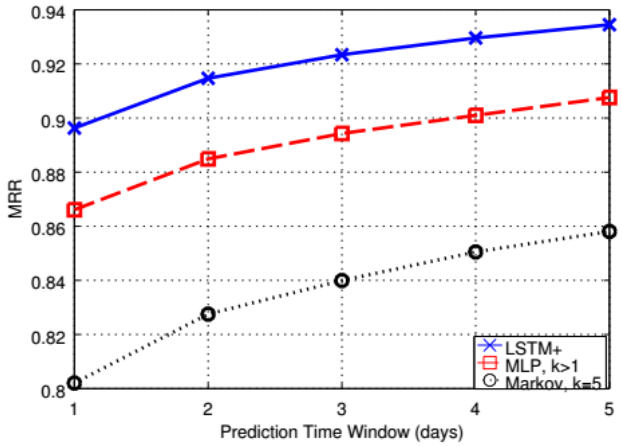
Tiếp theo, nghiên cứu sử dụng hai loại thuật toán neural network: một thuật toán feedforward neural network là multilayer perceptrons (MLP) và một thuật toán recurrent neural network là long short-term memory (LSTM). Với mỗi thuật toán, nghiên cứu thử nghiệm với nhiều thiết lập khác và hyper-parameters khác nhau.

Cả hai thuật toán đều sử dụng activation function là hàm rectified linear units (ReLU) và Adam stochastic optimization làm thuật toán tối ưu hóa. MLP được thực hiện với nhiều thiết lập khác nhau về số lượng layer và số lượng neuron trong một layer. Tuy nhiên, thuật toán LSTM, như tên của nó, còn có "memory" để lưu trữ các thông tin từ những dữ liệu quan sát trước đó. Do đó, việc tính toán ra kết quả cuối cùng còn có sự tham gia của các thông tin trong "memory" của thuật toán. Cơ chế lưu trữ "memory" trong nghiên cứu này được mô tả trong một nghiên cứu gần đây của Zaremba *et al.*

Cuối cùng là phương pháp đánh giá hiệu quả của các thuật toán. Zhang *et al.* sử dụng Mean Reciprocal Rank (MRR), một thuật toán phổ biến trong việc xếp hạng sự chính xác của các thuật toán dự báo, làm thước đo. Xếp hạng tốt nhất của một dự báo được xác định là vị trí của danh mục đầu tiên được dự báo chính xác khi xếp hạng theo thứ tự xác suất giảm dần. Cụ thể, nếu danh mục có xác suất xuất hiện cao nhất thực sự xuất hiện trong khung thời gian cho sẵn, xếp hạng tốt nhất sẽ là 1.

4. Kết quả

Kết quả thử nghiệm cho thấy cả hai thuật toán neural network đều cho ra kết quả tốt hơn k-dependent Markov chains. Trong hai thuật toán neural network, LSTM cho kết quả nhỉnh hơn một chút so với perceptrons. Trong các điều kiện thiết lập tốt nhất mà thử nghiệm thực hiện, kết quả cho về mức dự đoán thành công ở 0.8737 – tức trong 87,37% các dự đoán, người dùng sẽ nhận được một email thuộc danh mục đã được dự đoán có xác xuất xuất hiện cao nhất trong vòng 3 ngày.



Hình 3: So sánh kết quả thử nghiệm của ba thuật toán

Tài liệu liên quan

Zhang, A., Garcia-Pueyo, L., Wendt, J. B., Najork, M. and Broder, A, (2017). Email Category Prediction. In: *International World Wide Web Conference Committee.* [online] Available at: https://goo.gl/UdeK62

Ailon N., Karnin Z. S., Liberty E., and Maarek Y. (2013). Threading machine generated email. In*: 6th ACM International Conference on Web Search and Data Mining.*

Gamzu I., Karnin Z., Maarek Y., and Wajc D. (2015). You will getmail! Predicting the arrival of future email. In: *24th International Conference on World Wide Web.*

Zaremba W., Sutskever I., and Vinyals O.(2014). *Recurrent neural network regularization.* arXiv preprint:1409.2329.