TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN MÔN HỌC
Tìm hiểu và áp dụng Machine Learning vào bài toán cụ thể

Lê Võ Minh Đạt
Dat.lvm160921@sis.hust.edu.vn

Ngành Công nghệ thông tin
Chuyên ngành An toàn thông tin

Giảng viên hướng dẫn: TS. Trần Hải Anh

Chữ ký của GVHD

Bộ môn: Truyền thông và Mạng máy tính
Viện: Công nghệ thông tin và truyền thống

HÀ NỘI, 6/2020

ĐỀ TÀI TỐT NGHIỆP

Biểu mẫu của Đề tài/khóa luận tốt nghiệp theo qui định của viện, tuy nhiên cần đảm bảo giáo viên giao đề tài ký và ghi rõ họ và tên.

Trường hợp có 2 giáo viên hướng dẫn thì sẽ cùng ký tên.

Giáo viên hướng dẫn
Ký và ghi rõ họ tên

Lời cảm ơn

Với sự giúp đỡ khích lệ của phía nhà trường, thầy cô, gia đình, bạn bè trong khoa và sự nỗ lực của bản thân. Cuối cùng thì đồ án môn học "Tìm hiểu và áp dụng Machine Learning vào bài toán cụ thể".

Đầu tiên, con xin gửi lời cảm ơn đến cha mẹ, những người luôn ủng hộ quan tâm theo dõi cũng như tạo điều kiện tốt cho con hoàn thành nhiệm vụ. Em xin gửi lời cảm ơn đến thầy cô của Viện Công nghệ Thông tin và Truyền thông cũng như các thầy, cô của Trường Đại học Bách khoa Hà Nội đã chỉ bảo và cung cấp nguồn tri thức vô cùng quý giá để em có đầy đủ kiến thức hoàn thành đề tài này. Đặc biệt, em xin cảm ơn TS. Trần Hải Anh, bộ môn Truyền thông Mạng máy tính, người đã giúp đỡ, hướng dẫn, chỉ bảo tận tình cho em trong những lúc khó khăn cũng như trong quá trình thực hiện đề tài này. Xin cảm ơn các bạn bè đã giúp đỡ, khích lệ em trong quá trình học tập thực hiện đề tài này. Xin chân thành cảm ơn

Tóm tắt nội dung đồ án

Các vấn đề cần thực hiện trong đồ án là: tìm hiểu về Machine Learning, sau đó áp dụng kiến thức để giải bài toán KDD Cup 1999. Vì mới bắt đầu nghiên cứu về Machine Learning, em đã chọn sử dụng thuật toán phổ biến và đơn giản là Logistic Regression. Sau đó, em sử dụng Visual Studio Code và ngôn ngữ python, với thư viện scikit-learn.

Kết quả, em đã giải được bài toán. Định hướng tiếp theo của em là thu thập các dữ liệu thật và dựa vào dữ liệu đã huấn luyện để nhận biết được tấn công trong mạng. Qua đồ án môn học, em đã có các kiến thức cơ bản về Machine Learning, các thuật ngữ, thuật toán cơ bản về lĩnh vực này. Em mong muốn sẽ nghiên cứu thêm và phát triển một chương trình Machine Learning tốt, phục vụ cho mọi người hoặc các tổ chức.

MỤC LỤC

DANH MỤC HÌNH VẼ

CHƯƠNG 1. GIỚI THIỆU

1.1 Giới thiệu bài toán

Sử dụng bộ dữ liệu của cuộc thi KDD Cup 1999, áp dụng thuật toán và đánh giá mức độ chính xác của thuật toán.

1.2 Quá trình nghiên cứu và giải quyết bài toán

Các quá trình mà em đã thực hiện để giải quyết bài toán trên:

- Tìm hiểu về Machine Learning.
- Tìm hiểu về thuật toán cơ bản Linear Regression.
- Tìm hiểu về thuật toán Gradient Descent.
- Xác định đây là bài toán phân loại, thuộc nhóm Supervised Learning.
- Tìm hiểu về python, thư viện hỗ trợ làm việc hiệu quả về Machine Learning.
- Giải quyết bài toán bằng thuật toán Logistic Regression.

Em xin được trình bày chi tiết về kết quả nghiên cứu và giải quyết bài toán qua các chương:

Chương 2: Machine Learning

Chương 3: Gradient Descent

Chương 4: Logistic Regression

Chương 5: Giải quyết bài toán

CHƯƠNG 2. MACHINE LEARNING

2.1 Định nghĩa

“Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed” - Arthur Samuel (1959).

Năm 1959, Arthur Samuel lần đầu đưa ra định nghĩa về Machine Learning: “Machine Learning là một lĩnh vực nghiên cứu giúp cho máy tính có khả năng học mà không cần đến việc lập trình cụ thể”.

“Well-posed Learning Problem: A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.” - Tom Mitchell (1998).

Năm 1998, Tom Mitchell đưa ra định nghĩa chính thức hơn về các thuật toán được nghiên cứu trong lĩnh vực Machine Learning: “Một chương trình máy tính được cho là học từ kinh nghiệm E đối với tác vụ T và một phép đo hiệu suất P, nếu nó hiệu quả trên T, được đo lường bởi P, và được cải thiện với kinh nghiệm E”.

Định nghĩa này thiên về việc khái quát hoạt động cơ bản của Machine Learning, về việc thế nào thì một chương trình máy tính được xem là Machine Learning.

Để làm rõ hơn về định nghĩa này, ta lấy ví dụ: Xác định một email có phải Spam hay không thông qua việc đánh dấu Spam của người dùng.

Tác vụ T: xác định email là spam hay email thường.

Kinh nghiệm E: việc đánh dấu mail spam của người dùng.

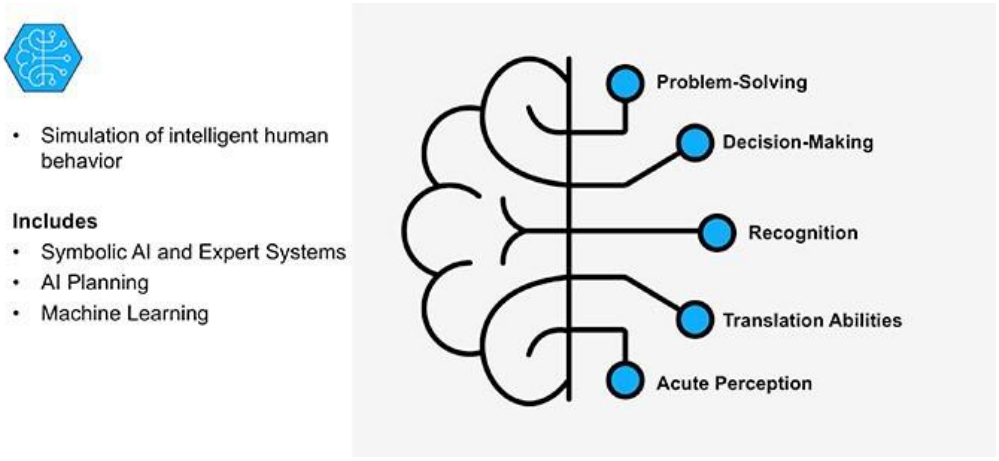
Hiệu suất P: phần trăm xác định đúng là email spam.

2.2 AI, Machine Learning và Deep Learning.

2.2.1 Artificial Intelligence

AI (Artificial Intelligence) hay là trí tuệ nhân tạo đã xuất hiện trong trí tưởng tượng của chúng ta từ rất lâu, và nó lần đầu được trở thành một ngành nghiên cứu tại một hội nghị diễn ra ở Dartmouth College vào năm 1956.

Theo nghĩa rộng nhất, người ta định nghĩa AI là hệ thống có suy nghĩ, giác quan, lý trí giống như hoặc hơn con người, hay còn được gọi là “General AI”. Nó là tham vọng, mục tiêu to lớn của chúng ta.



Hình 2.1 Các khả năng General AI cần có

Nhưng qua nhiều thập kỷ nghiên cứu, những gì chúng ta có thể làm chính là "Narrow AI" – các công nghệ có thể thực hiện các công việc chuyên biệt với khả năng ngang bằng, hoặc hơn con người, ví dụ như chế độ tự lái trên các xe điện Tesla hoặc khả năng nhận diện khuôn mặt trên Facebook.

Phía trên là một số ví dụ về Narrow AI trong thực tế. Những công nghệ này thể hiện một số khía cạnh của trí tuệ con người, nhưng làm thế nào, trí thông minh này ở đâu mà có, đó là nhờ phương tiện mang tên Machine Learning.

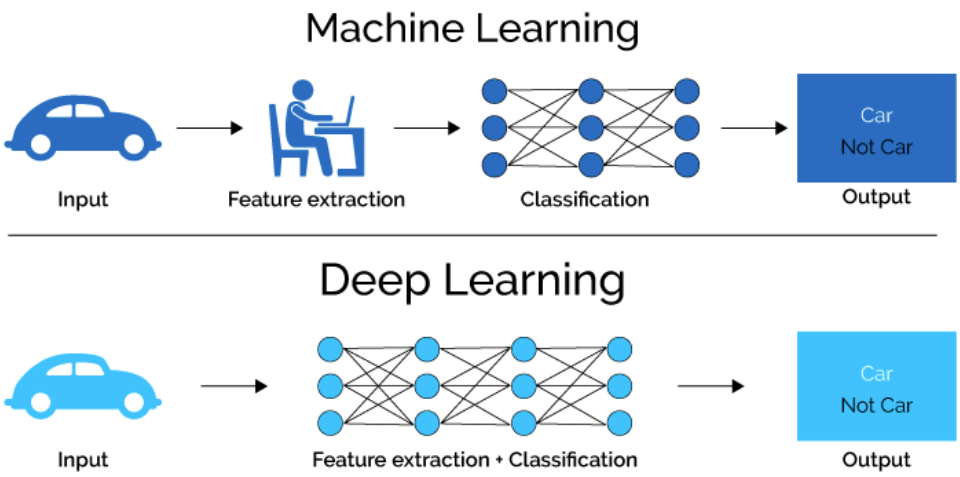
2.2.2 Machine Learning

Hình 2.2: Sử dụng ML để nhận biết Spam mail

Ở mức cơ bản nhất, Machine Learning là sử dụng thuật toán để xử lý dữ liệu, học từ nó, và cuối cùng là đưa ra quyết định hoặc dự đoán về một việc gì đó.

Machine Learning được nghĩ đến từ những ngày đầu nghiên cứu về AI, và các thuật toán như decision trees, regression analysis, Bayesian networks là các cách tiếp cận trong một khoảng thời gian. Nhưng trên thực tế, cách tiếp cận này không mang lại sự đột phá, nó không thể tạo ra một "General AI", hoặc thậm chí là "Narrow AI". Nó quá mỏng manh và dễ dàng phát sinh lỗi. Cho đến gần đây, một thuật toán tốt và thời điểm tốt đã tạo ra sự khác biệt.

2.2.3 Deep learning

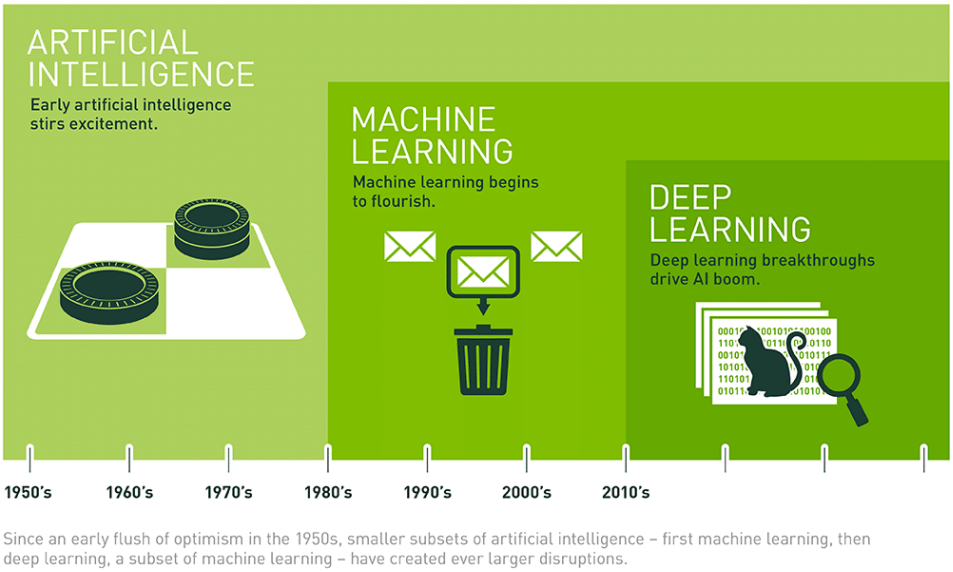


Hình 2.3 Sự khác nhau giữa ML và DL

Một trong các thuật toán được đưa ra trong thời kỳ đầu của AI, artificial neural networks (mạng lưới thần kinh nhân tạo), đã được xem xét và nghiên cứu trong nhiều thập kỷ. Nó lấy cảm hứng từ chính bộ não của chúng ta – các liên kết mật thiết giữa các neuron thần kinh. Nhưng không giống với não bộ với các neuron có thể liên kết đến bất cứ neuron khác trong một khoảng cách vật lý xác định, mạng lưới thần kinh nhân tạo có các lớp, kết nối và hướng truyền dữ liệu riêng biệt.

Và cùng với sức mạnh máy tính tăng lên đáng kể, nhất là sự ra đời của GPU, cùng với lượng dữ liệu khổng lồ được thu thập bởi các hãng công nghệ lớn, Deep Learning đã khiến cho việc tiến đến một hệ thống AI hoàn thiện – "General AI" đến gần hơn, với các ứng dụng như xe tự lái, hệ thống nhận diện gương mặt, dịch thuật trực tiếp,…

Cách dễ nhất để diễn tả mối quan hệ giữa AI, Machine Learning và Deep Learning là, AI – thứ chung nhất, tiếp đến, Machine Learning là một tập con của AI, giúp máy tính có khả năng tự học hỏi mà không cần lập trình cụ thể, và sau cùng, một bước tiến dài của Machine Learning, sinh ra một lĩnh vực mới là Deep Learning – thứ khiến cho AI bùng nổ trong những năm gần đây.



Hình 2.4 AI, ML và DL

2.3 Phân loại

Trong machine learning, dữ liệu đầu vào được chia làm 2 loại: labeled data (dữ liệu được gán nhãn) và unlabeled data (dữ liệu không được gán nhãn). Labeled data có cả thông số đầu vào và đầu ra (thường gọi là input/output) dưới dạng máy tính có thể dễ dàng hiểu được, nhưng lại mất rất nhiều thời gian và công sức của con người để dán nhãn các dữ liệu. Unlabeled data thì chỉ có một hoặc thậm chí không có thông số nào dưới dạng máy tính có thể đọc được, để thu thập dữ liệu là rất dễ dàng, không tốn nhiều công sức của con người, nhưng lại cần các thuật toán phức tạp.

Để phân loại machine learning, cách phổ biến nhất là chia theo phương thức học. Tuy có một số phương thức được tạo ra để sử dụng cho các bài toán đặc biệt cụ thể, chúng ta có thể chia machine learning thành 3 loại chính: *Supervised learning, Unsupervised learning và Reinforcement learning*.

2.3.1 Supervised Learning

Supervised learning (Học có giám sát) là một trong những loại cơ bản trong machine learning. Thuật toán học máy này được huấn luyện dựa trên labeled data. Nó dự đoán đầu ra của một dữ liệu mới dựa trên các dữ liệu đầu vào (input data), gọi là dữ liệu huấn luyện (training data) có các kết quả đầu ra (output data). Supervised learning cực kỳ mạnh nếu được sử dụng trong các hoàn cảnh phù hợp.

Trong supervised learning, người ta đưa vào một lượng dữ liệu đầu vào, là một phần nhỏ của dữ liệu cần được xử lý, và để phục vụ cho việc máy tính tìm ra được mối liên hệ giữa dữ liệu đầu vào và kết quả đầu ra. Từ mối quan hệ này, thuật toán xử lý các dữ liệu mới, gần giống với dữ liệu huấn luyện, và cho ra kết quả, hay là dự đoán đầu ra của tập dữ liệu mới này.

Các vấn đề của học máy có giám sát chia thành “bài toán dự đoán” (regression) và “bài toán phân loại” (classification).

Trong “bài toán dự đoán”, nó dùng để dự đoán các kết quả đầu ra, một giá trị cụ thể với các dữ liệu đầu vào mới. “Bài toán phân loại” dùng để phân loại các kết quả đầu ra vào các nhóm hữu hạn riêng biệt, nói cách khác là ánh xạ các biến đầu vào thành các danh mục riêng biệt.

2.3.2 Unsupervised Learning

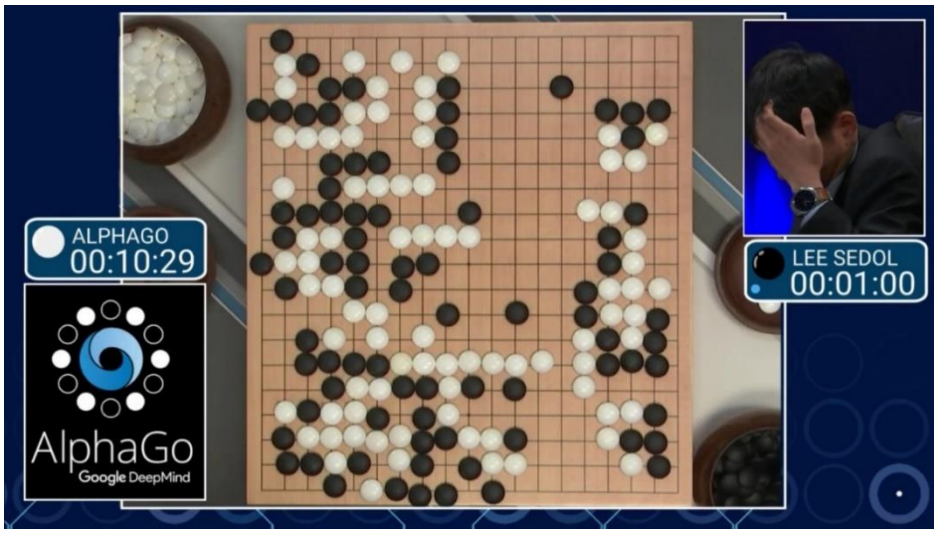
Unsupervised learning (Học không có giám sát) có điểm mạnh là huấn luyện trên dữ liệu không cần dán nhãn, và dữ liệu không dán nhãn thì internet là một nguồn vô cùng lớn, thu thập nhanh chóng và tốn ít kinh phí. Thuật toán sẽ dựa vào cấu trúc của dữ liệu đầu vào để thực hiện một công việc nào đó. Mối quan hệ giữa các dữ liệu được thuật toán cảm nhận theo cách trừu tượng, vì không biết câu trả lời chính xác cho dữ liệu đầu vào.

Các bài toán unsupervised learning được chia thành các loại: clustering (phân nhóm), dimensionality reduction (nén dữ liệu), và association (kết hợp).

2.3.3 Reinforcement Learning

Reinforcement learning (Học củng cố) được lấy cảm hứng trực tiếp từ cách con người học những thứ mới mẻ. Thuật toán tự phát triển chính bản thân và học hỏi từ các tình huống mới bằng phương pháp thử-và-lỗi, tức là học hỏi từ những lần mắc lỗi. Không hề có kết quả đầu ra, nhưng máy tính nhận được phản hồi với mỗi hành động, nếu hành động đúng đắn, chúng sẽ nhận được các phản hồi tích cực, nếu hành động sai, chúng sẽ nhận được phản hồi tiêu cực. Dựa vào đó, máy tính sẽ điều chỉnh hành động sao phù hợp với các phản hồi tích cực.

Hiện tại, Reinforcement learning chủ yếu được áp dụng vào Lý Thuyết Trò Chơi (Game Theory), các thuật toán cần xác định nước đi tiếp theo để đạt được điểm số cao nhất. Ví dụ như AlphaGo của Google trong bộ môn cờ vây hay IBM Deep Blue của IBM, đã đánh bại con người trong bộ môn cờ vua từ rất lâu về trước.



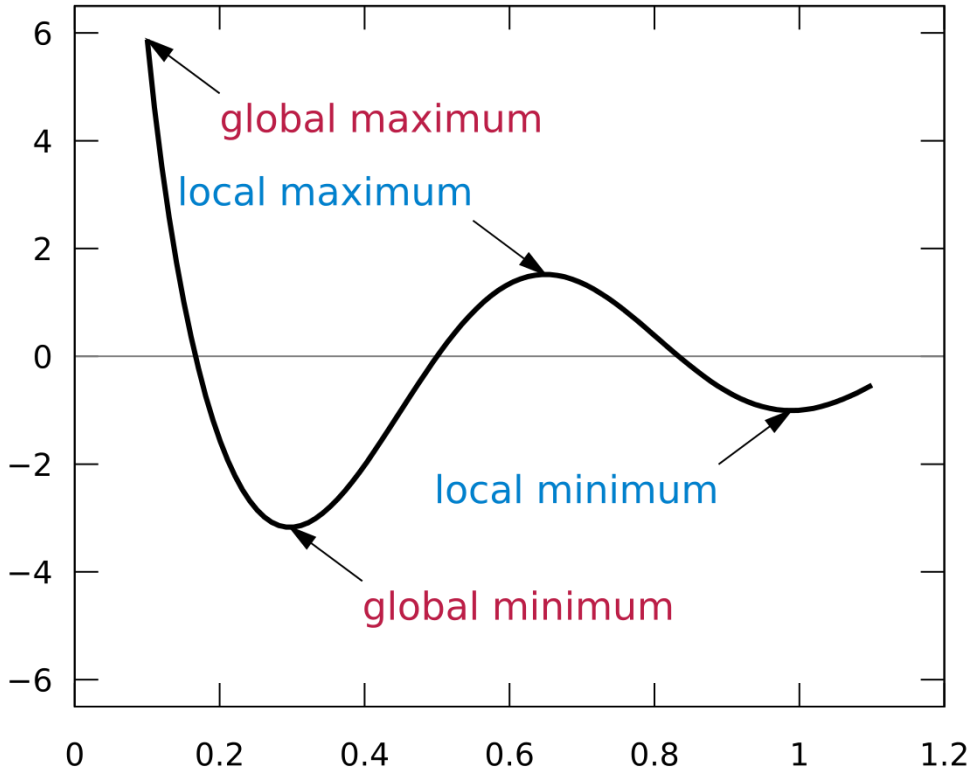
Hình 2.5 AlphaGo của Google đánh bại kỳ thủ

CHƯƠNG 3. GRADIENT DESCENT

3.1 Gradient Descent.

Gradient descent là thuật toán tối ưu lặp, phục vụ mục đích tìm điểm "local minimum" của một hàm số.

Điểm local minimum là điểm cực tiểu của hàm số, tại đó hàm số có đạo hàm bằng 0. Điểm global minimum là điểm cực tiểu mà tại đó hàm số đạt giá trị nhỏ nhất. Global minimum là một trường hợp đặc biệt của local minimum.



Hình 3.1 Local minimum và Global minimum

Công thức cập nhật cho GD:

$$\theta := \theta - \eta \nabla_\theta f(\theta)$$

trong đó θ là tập các biến cần cập nhật, η là tốc độ học (learning rate), $\nabla_\theta f(\theta)$ là đạo hàm của hàm mất mát f theo tập θ.

3.2 Tìm hiểu chi tiết

Để hiểu rõ về **Gradient Descent**, mình sẽ lấy ví dụ về cách nó được áp dụng trong thuật toán **Linear Regression**.

Bài toán đặt ra: chúng ta có dữ liệu về ngôi nhà bao gồm diện tích, số phòng ngủ và khoảng cách đến trung tâm thành phố.

Giá của căn nhà đó là bao nhiêu?

Giả sử chúng ta đã có dữ liệu thống kê từ 1404 căn nhà trong thành phố đó.

Ta gọi diện tích là $x_1\ m^2$, số phòng ngủ là x_2, khoảng cách đến trung tâm thành phố là $x_3\ km$. Giá nhà là y.

Suy ra, ta có hàm dự đoán sẽ là:

$$y \approx f(x) = w_1x_1 + w_2x_2 + w_3x_3 + w_0$$

trong đó, w_1, w_2, w_3 là các hằng số, w_0 còn được gọi là bias.

Đặt $\boldsymbol{w} = [w_0, w_1, w_2, w_3]^T$ là vector hệ số cần tối ưu và $\overline{\boldsymbol{x}} = [1, x_1, x_2, x_3]$ là vector đầu vào mở rộng. Số 1 ở đầu được thêm để tính toán đơn giản hơn. Lúc này, phương trình (1) có thể viết lại dưới dạng:

$$y \approx f(x) = \overline{\boldsymbol{x}}\boldsymbol{w}$$

3.2.1 Hàm mất mát

Gọi y' là kết quả dự đoán. Ta sẽ mong muốn y' gần nhất so với y. Gọi e là độ chênh lệch giữa y' và y.

Ta luôn muốn sự chênh lệch giữa kết quả dự đoán và giá trị thực là nhỏ nhất, hay nói cách khác ta muốn có:

$$e = Min|y - y'| \text{ hay là } e = Min(y - y')^2$$

Đây chính là ý tưởng chính của **hàm mất mát (Coss function)**.

Nếu ta có một tập dữ liệu gồm các cặp (*input, outcome)* $(x_i, y_i),\ i = 1, 2, \dots, N$ với N là số lượng cặp dữ liệu. Thì hàm mất mát của tập dữ liệu trên có dạng

$$\frac{1}{2N}\sum_{i=1}^{N}(y_i - y_i')^2$$

Hàm mất mát thể hiện hiệu suất của thuật toán machine learning. Nó càng nhỏ thì càng chứng tỏ sự sai khác giữa kết quả dự đoán và kết quả thực tế càng đúng. Vì thế, ta luôn muốn làm sao cho giá trị hàm mất mát là **nhỏ nhất**.

Áp dụng vào ví dụ trên, ta muốn tìm giá trị vector cột $\boldsymbol{w}$ sao cho hàm số sau là nhỏ nhất:

$$C(\boldsymbol{w}) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - \overline{\boldsymbol{x}_i}\boldsymbol{w})^2$$

Đặt $\boldsymbol{y} = [y_1;\ y_2; \dots; y_N]^T$ là vector cột chứa tất cả kết quả đầu ra, $\overline{\boldsymbol{X}} = [\overline{\boldsymbol{x}_1};\ \overline{\boldsymbol{x}_2}; \dots;\ \overline{\boldsymbol{x}_N}]$ là ma trận dữ liệu đầu vào mà mỗi hàng là một điểm dữ liệu.

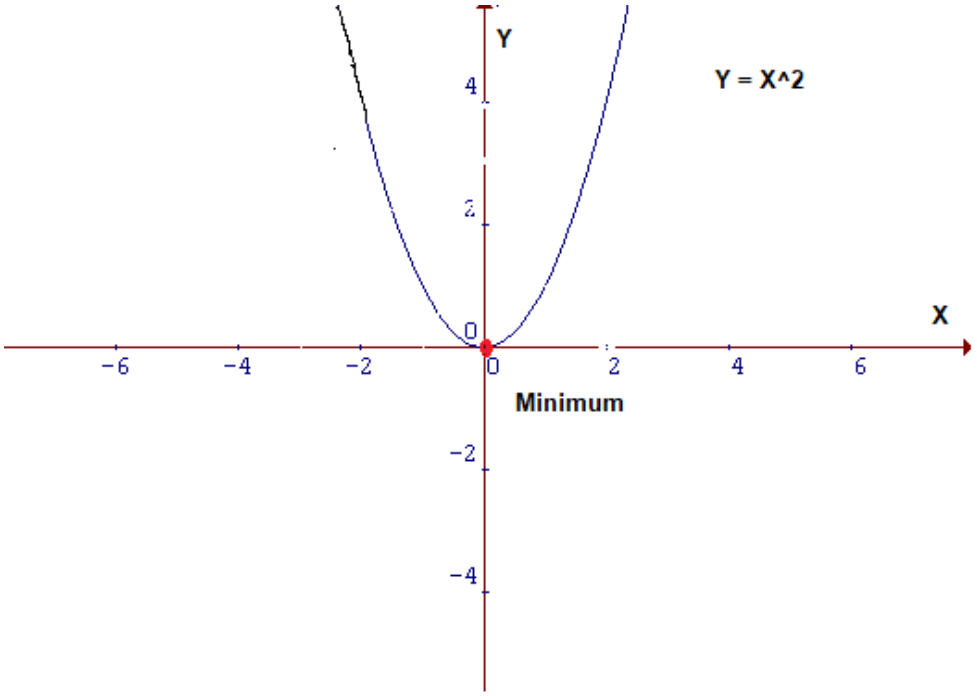
Khi này, hàm số mất mát được viết dưới dạng đơn giản hơn:

$$C(\boldsymbol{w}) = \frac{1}{2N}\|\boldsymbol{y} - \overline{\boldsymbol{X}}\boldsymbol{w}\|_2^2 \quad (\mathbf{1})$$

Với $\|\boldsymbol{z}\|_2$ là Euclidean norm (khoảng cách Euclid), $\|\boldsymbol{z}\|_2^2$ là tổng của bình phương mỗi phần tử của vector **z.**

3.2.2 Tối ưu hàm mất mát

Nếu như quan sát kỹ, hàm mất mát có dạng $y = x^2$. Vì vậy hàm sẽ có dạng parabol và có thể được biểu diễn dưới như sau:



Hình 3.2 Đồ thị hàm số mất mát

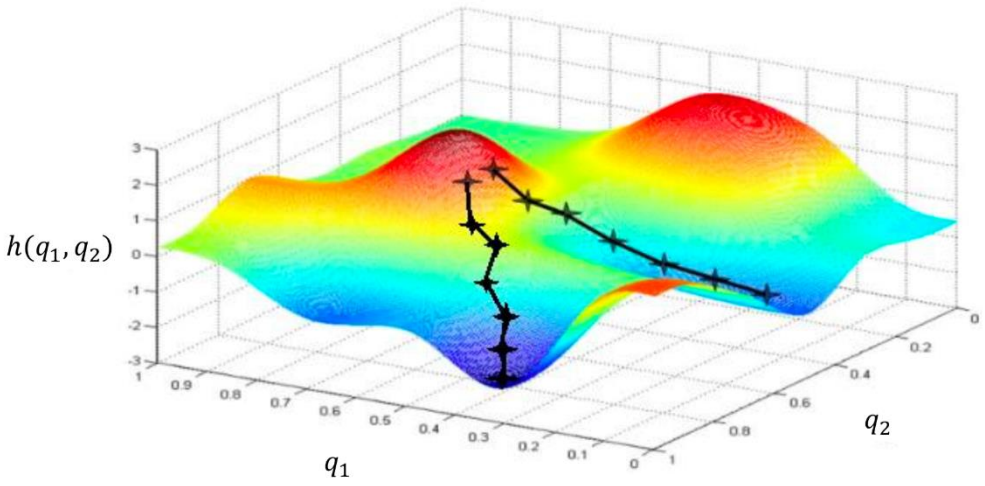
Suy ra, để tìm được giá trị nhỏ nhất của hàm mất mát, ta phải tìm giá trị của x để giá trị của y là nhỏ nhất, tức tìm điểm "global minimum".

Tuy nhiên, việc tìm "global minimum" của hàm mất mát trong machine learning là rất phức tạp, có thể là bất khả thi. Vì thế, người ta thường tìm các điểm "local minimum", và ở một mức độ nào đó, có thể xem đó là nghiệm cần tìm của bài toán. Để tìm các điểm "local minimum" thì ta tìm phương trình đạo hàm bằng 0, tuy vậy việc giải phương trình đạo hàm bằng 0 là bất khả thi trong hầu hết trường hợp, nguyên nhân có thể là vì đạo hàm phức tạp, các điểm dữ liệu có số chiều lớn hoặc có rất nhiều điểm dữ liệu.

Vì các nguyên nhân trên, chúng ta cần một thuật toán tốt để có thể tìm được điểm "local minimum" hoặc thậm chí là "global minimum", và thuật toán đó có tên là **Gradient Descent.**

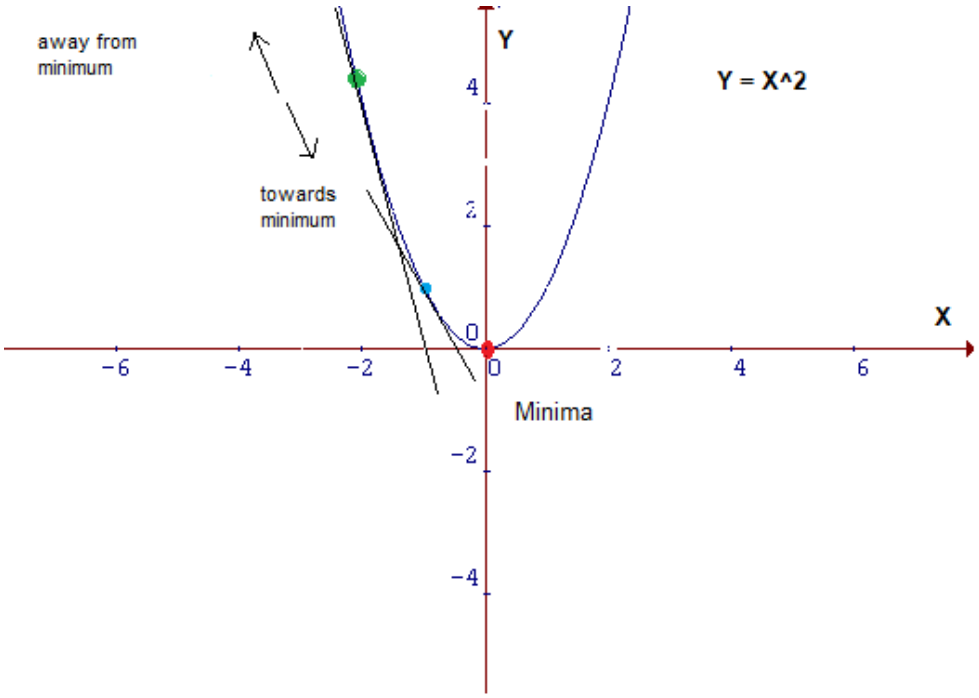
3.2.3 Gradient Descent

Giả sử, bạn đang đứng gần đỉnh một ngọn đồi, và bạn muốn đi xuống. Bạn chắc sẽ bước xuống điểm gần bạn mà bạn thấy dốc xuống, sau đó bước tiếp đến chỗ thấp hơn, dốc xuống như thế. Lặp đi lặp lại quá trình này đến khi bạn đến được nơi mà xung quanh bạn không còn thấy điểm thấp hơn nữa, không còn độ dốc nữa. Đây đúng là nơi thấp nhất đối với bạn, nhưng chưa chắc nơi bạn đang đứng là nơi thấp nhất, một điểm khác thấp hơn lại ở cách xa bạn một khoảng.



Hình 3.3 Ví dụ minh họa GD

Gradient descent cũng hoạt động với cơ chế tương tự vậy, thuật toán sẽ bắt đầu tại một điểm bất kỳ trên đồ thị, lấy đạo hàm (gradient) tại đó, rồi tìm một điểm có đạo hàm thấp hơn (descent) và hàm số có giá trị thấp hơn, cứ thế đến khi tìm được điểm mà tại đó đạo hàm bằng 0 hoặc gần bằng 0 (tức tìm được local minimum), mặc dù có thể không phải là "global minimum" nhưng ta cũng có thể chấp nhận được giá trị đó.



Hình 3.4 Đồ thi minh họa hướng di chuyển với GD

Ví dụ như hàm mất mát $y = x^2$ trên, chúng ta lấy điểm màu xanh lá là điểm bắt đầu, có độ dốc lớn, thuật toán sẽ bước xuống điểm màu xanh nước biển, điểm có độ dốc nhỏ hơn, và giá trị của hàm số nhỏ hơn, sau đó dần đến điểm màu đỏ,

điểm vừa là local minimum vừa là global minimum, tại đây đạo hàm bằng 0 và có giá trị nhỏ nhất.

Quan sát tiếp đồ thị trên, ta có một vài nhận xét:

- Thứ nhất: Đạo hàm của hàm số tại điểm màu xanh lục âm: $f'(x_l) < 0$, và điểm nằm bên trái so với điểm màu đỏ, để điểm tiếp theo: x_{l+1} gần với điểm màu đỏ hơn thì cần di chuyển điểm sang bên phải, tức về phía *dương*. Nói cách khác, ta cần di chuyển điểm **ngược dấu với đạo hàm**:

$$x_{l+1} = x_l + \Delta$$

Với Δ là đại lượng ngược dấu với đạo hàm $f'(x_l)$.

- Thứ hai: Điểm càng xa điểm màu đỏ về phía bên trái thì đạo hàm càng bé hơn 0 (và ngược lại). Vậy, lượng di chuyển Δ, một cách trực quan nhất, là tỉ lệ thuận với $-f'(x_l)$.

Từ 2 nhận xét chung, chúng ta có cách cập nhật điểm tiếp theo:

$$x_{l+1} = x_l - \eta f'(x_l).$$

Với η là một số dương, gọi là *learning rate (tốc độc học).* Dấu trừ thể hiện việc ta phải đi ngược với đạo hàm.

Tổng quát hơn, nếu ta cần tìm local minimim cho hàm $f(\theta)$, ta có quy tắc cập nhật:

$$\theta_{t+1} \coloneqq \theta_t - \eta \nabla_\theta f(\theta_t)$$

Hoặc đơn giản là:

$$\theta \coloneqq \theta - \eta \nabla_\theta f(\theta)$$

với:

θ là một vector, thường được ký hiệu của tập hợp các tham số của một mô hình cần tối ưu.

$\nabla_\theta f(\theta)$ là đạo hàm của hàm số đó tại một điểm θ bất kỳ.

Áp dụng vào bài toán Linear Regression phía trên, ta có:

Đạo hàm của hàm mất mát là:

$$\nabla_w C(\boldsymbol{w}) = \frac{1}{N} \bar{\boldsymbol{X}}^T (\bar{\boldsymbol{X}}\boldsymbol{w} - \boldsymbol{y})$$

Suy ra, công thức cập nhật của bài toán là:

$$\boldsymbol{w} \coloneqq \boldsymbol{w} - \eta \frac{1}{N} \bar{\boldsymbol{X}}^T (\bar{\boldsymbol{X}}\boldsymbol{w} - \boldsymbol{y})$$

3.2.4 Stochastic Gradient Descent

Thuật toán Gradient Descent chúng ta nhắc đến từ đầu đến giờ được gọi là Batch Gradient Descent. Batch có nghĩa là tất cả, tức cập nhật θ, chúng ta sử dụng tất cả các điểm dữ liệu $\boldsymbol{x}_i$. Cách làm này hạn chế với những dữ liệu lớn, chúng ta phải tính toán lại đạo hàm của tất cả các điểm sau mỗi lần cập nhật.

Với thuật toán Stochastic Gradient Descent, tại 1 thời điểm, ta chỉ tính đạo hàm của hàm mất mát dựa trên 1 điểm dữ liệu $\boldsymbol{x}_i$ rồi cập nhật θ dựa trên đạo hàm này. Thực hiện việc này trên từng điểm trên toàn bộ dữ liệu, sau đó lặp lại quá trình này.

Epoch là thuật ngữ được sử dụng trong machine learning và cho biết số lần thuật toán duyệt qua toàn bộ tập dữ liệu. Đối với GD thông thường, mỗi epoch là một lần cập nhật θ, với SGD thì mỗi epoch ứng với N lần cập nhật θ với N điểm dữ liệu.

Thông thường, SGD chỉ yêu cầu 1 lượng rất nhỏ epoch để đến gần điểm local minimum. Vì vậy SGD phù hợp với bài toán có lượng dữ liệu lớn (chủ yếu là Deep Learning) và các bài toán yêu cầu mô hình thay đổi liên tục, tức online learning.

Quy tắc cập nhật của SGD là:

$$\theta \coloneqq \theta - \eta \nabla_\theta f(\theta;\ x_i;\ y_i)$$

Trong đó $f(\theta;\ x_i;\ y_i)$ là hàm mất mát với chỉ 1 cặp điểm dữ liệu là $(x_i;\ y_i)$.

CHƯƠNG 4. LOGISTIC REGRESSION

Logistic Regression là thuật toán phổ biến và thường được sử dụng trong bài toán phân loại, đặc biệt là Binary Classification.

Binary classification là những bài toán có kết quả đầu ra chỉ phân vào 2 nhóm, ví dụ như spam/không spam, đỗ/trượt, có/không, … và nó sẽ được biểu diễn dưới dạng nhị phân, tương ứng với giá trị 0 và 1.

Đầu ra của Logistic regression thường được viết chung dưới dạng:

$$f(x) = \theta(\boldsymbol{w}^T \boldsymbol{x})$$

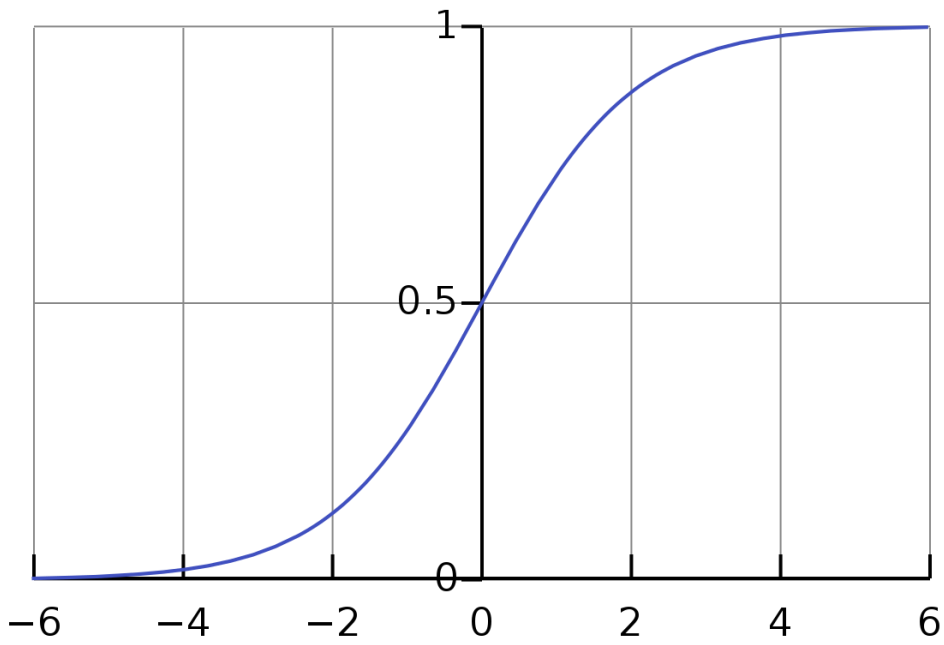
Với θ là hàm logistic, $\boldsymbol{w}^T$ là vector tham số và $\boldsymbol{x}$ là dữ liệu đầu vào.

4.1 Sigmoid function

Hàm sigmoid thường được sử dụng là hàm logistic, hàm có công thức:

$$f(x) = \frac{1}{1 + e^x} \triangleq \sigma(x)$$

Hàm được biểu diễn như sau:



Hình 4.1 Đồ thị hàm số Sigmoid

Ta có thể thấy:

- Hàm số liên tục, bị chặn trong khoảng (0, 1).
- Nếu chia hàm làm 2 phần, phân chia tại điểm có tung độ là 0.5, thì điểm càng xa điểm này về phía bên trái có giá trị càng gần 0. Ngược lại, điểm càng về bên phải thì có giá trị càng gần 1.
  Đây là lí do tại sao hàm được sử dụng chính cho bài toán phân loại, ta có thể xem như:
  - Giá trị đầu ra $y = 0$ nếu giá trị hàm số $\sigma(x) \geq 0.5$.
  - Giá trị đầu ra $y = 1$ nếu giá trị hàm số $\sigma(x) < 0.5$.

Thêm nữa:

$$\lim_{x\to-\infty}\sigma(x)=0;\ \lim_{x\to+\infty}\sigma(x)=1$$

Và:

$$\sigma'(x)=\frac{e^{-x}}{(1+e^{-x})^2}$$

$$\Leftrightarrow\ \sigma'(x)\ =\ \frac{1}{1+e^{-x}}\frac{e^{-x}}{1+e^{-x}}$$

$$\Leftrightarrow\ \sigma'(x)=\ \sigma(x)(1-\ \sigma(x))$$

Có thể thấy là công thức đạo hàm rất đơn giản, điều này giúp hàm số được sử dụng rộng rãi.

4.2 Hàm mất mát

Giả sử rằng xác suất để một điểm dữ liệu đầu vào $\boldsymbol{x}$ có giá trị đầu ra "$y\ =1$" là $f(\boldsymbol{w}^T\boldsymbol{x})$ và có giá trị đầu ra "$y\ =0$" là 1 - $f(\boldsymbol{w}^T\boldsymbol{x})$.

Vậy xác suất đầu ra tại một điểm $\boldsymbol{x_i}$ bất kỳ là:

$$P(y_i=1|\boldsymbol{x_i};\boldsymbol{w})\ =\ \ \ \ f(\boldsymbol{w}^T\boldsymbol{x_i})$$

$$P(y_i=0|\boldsymbol{x_i};\boldsymbol{w})\ =\ 1-\ f(\boldsymbol{w}^T\boldsymbol{x_i})$$

Vậy, mục đích là tìm các hệ số $\boldsymbol{w}$ sao cho $f(\boldsymbol{w}^T\boldsymbol{x_i})$ càng gần với 1 càng tốt với các điểm có giá trị đầu ra bằng 1, và càng gần với 0 càng tốt với các điểm có giá trị đầu ra là 0.

Kí hiệu $z_i=\ f(\boldsymbol{w}^T\boldsymbol{x_i})$ và viết gộp hai biểu thức trên, ta có:

$$P(y_i|\boldsymbol{x_i};\boldsymbol{w})\ =\ z_i^{y_i}\ (1-\ z_i)^{1-\ y_i}$$

Xét toàn bộ tập dữ liệu huấn luyện với $\boldsymbol{X}=[x_1;\ x_2;\ldots;x_N]$ và $\boldsymbol{y}=[y_1;\ y_2;\ldots;y_N]$, ta cần tìm $\boldsymbol{w}$ để biểu thức:

$P(\boldsymbol{Y}|\boldsymbol{X};\boldsymbol{w})$ đạt giá trị lớn nhất

Giả sử, các điểm dữ liệu sinh ra một cách ngẫu nhiên độc lập với nhau, ta có thể viết:

$$P(\boldsymbol{Y}|\boldsymbol{X};\boldsymbol{w})=\ \prod_{i=1}^{N}P(y_i|\boldsymbol{x_i};\boldsymbol{w})$$

$$\Leftrightarrow P(\boldsymbol{Y}|\boldsymbol{X};\boldsymbol{w})=\ \prod_{i=1}^{N}z_i^{y_i}\ (1-\ z_i)^{1-\ y_i}$$

Tối ưu hàm này theo $\boldsymbol{w}$ có vẻ rất phức tạp và khi N lớn, tích của N số nhỏ hơn 1 có thể sẽ dẫn tới sai số trong tính toán vì tích quá nhỏ.

Ta lấy logarit tự nhiên cơ số e để biến phép nhân thành phép cộng, sau đó lấy ngược dấu và coi nó là **hàm mất mát**. Lúc này, bài toán sẽ trở thành tìm giá trị nhỏ nhất của hàm mất mát:

$$C(\boldsymbol{w})=\ -\log P(\boldsymbol{Y}|\boldsymbol{X};\boldsymbol{w})$$

$$\Leftrightarrow\ C(\boldsymbol{w})=\ -\sum_{i=1}^{N}(\ y_i\log z_i+(1-\ y_i)\log\ (1-z_i))$$

4.3 Tối ưu hàm mất mát

Chúng ta lại sử dụng phương pháp Stochastic Gradient Descent (SGD) ở đây. Hàm mất mát với chỉ 1 điểm dữ liệu $(\boldsymbol{x}_i, y_i)$ là:

$$C(\boldsymbol{w}; \boldsymbol{x}_i, y_i) = -(y_i \log z_i + (1 - y_i)\log(1 - z_i))$$

Lấy đạo hàm:

$$\frac{\partial C(\boldsymbol{w}; \boldsymbol{x}_i, y_i)}{\partial \boldsymbol{w}} = -\left(\frac{y_i}{z_i} - \frac{1 - y_i}{1 - z_i}\right)\frac{\partial z_i}{\partial \boldsymbol{w}} = \frac{z_i - y_i}{z_i(1 - z_i)}\frac{\partial z_i}{\partial \boldsymbol{w}}$$

Mà $z_i = f(\boldsymbol{w}^T \boldsymbol{x_i})$ chính là hàm sigmoid, đặt $s = \boldsymbol{w}^T \boldsymbol{x_i}$, ta có:

$$z_i = \frac{1}{1 + e^s}$$

Và đạo hàm của z_i theo s là:

$$z_i'(s) = z_i(1 - z_i)$$

Đạo hàm của s theo $\boldsymbol{w}$ là: $s'(\boldsymbol{w}) = \boldsymbol{x_i}$

Suy ra, ta có:

$$\frac{\partial z_i}{\partial \boldsymbol{w}} = \frac{\partial z_i}{\partial s}\frac{\partial s}{\partial \boldsymbol{w}} = \frac{\partial z_i}{\partial s}\boldsymbol{x_i} = z_i(1 - z_i)\boldsymbol{x_i}$$

Kết quả, ta có hàm mất mát là:

$$\frac{\partial C(\boldsymbol{w}; \boldsymbol{x}_i, y_i)}{\partial \boldsymbol{w}} = (z_i - y_i)\boldsymbol{x_i}$$

Và công thức cập nhật (theo thuật toán SGD) cho logistic regression là:

$$\boldsymbol{w} := \boldsymbol{w} - \eta(z_i - y_i)\boldsymbol{x_i}$$

CHƯƠNG 5. GIẢI QUYẾT BÀI TOÁN

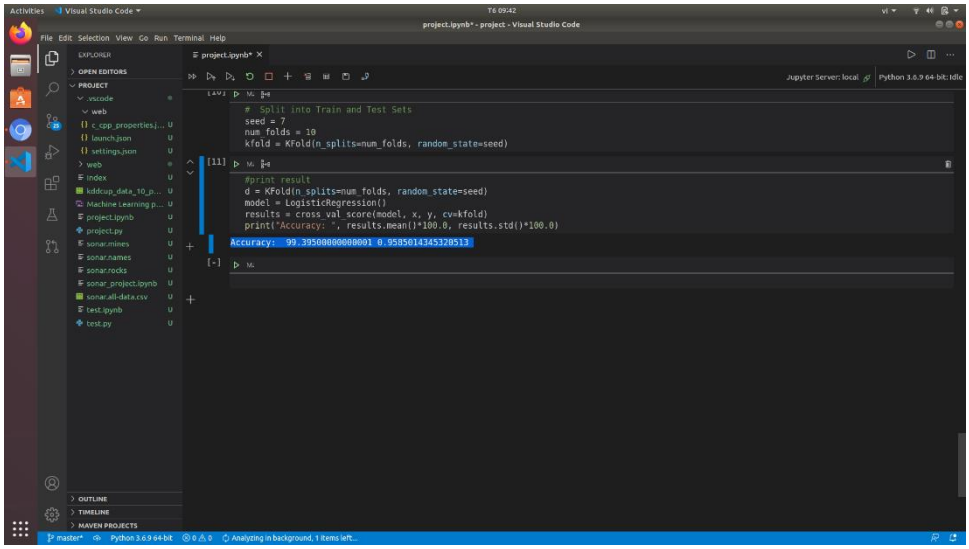
Code của chương trình:

```
from __future__ import division, print_function, unicode_literals
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from numpy import set_printoptions
from sklearn.preprocessing import MinMaxScaler
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.feature_selection import RFE
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import KFold

# import data
filename = '/home/suma/project/kddcup_data_10_percent_corrected.csv'
names = ['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot',
'num_failed_logins', 'logged_in',
'num_compromised', 'root_shell', 'su_attempted', 'num_root',
'num_file_creations', 'num_shells', 'num_access_files',
'num_outbound_cmds', 'is_hot_login', 'is_guest_login', 'count',
'srv_count', 'serror_rate', 'srv_serror_rate', 'rerror_rate',
'srv_rerror_rate', 'same_srv_rate', 'diff_srv_rate',
'srv_diff_host_rate', 'dst_host_count', 'dst_host_srv_count',
'dst_host_same_srv_rate', 'dst_host_diff_srv_rate',
'dst_host_same_src_port_rate', 'dst_host_srv_diff_host_rate',
'dst_host_serror_rate', 'dst_host_srv_serror_rate',
'dst_host_rerror_rate',  'dst_host_srv_rerror_rate', 'type']

# take a look on data
data = pd.read_csv(filename, names=names)
data_short=data.head(20000)
print(data.head(10))

# check information about data
print("data shape:", data.shape)

types = data.dtypes
print("data type:", types)

data_short['type'] = data_short['type'].replace(['back.',
'buffer_overflow.', 'ftp_write.', 'guess_passwd.', 'imap.',
'ipsweep.', 'land.', 'loadmodule.', 'multihop.', 'neptune.', 'nmap.',
'perl.', 'phf.', 'pod.', 'portsweep.', 'rootkit.', 'satan.', 'smurf.',
'spy.', 'teardrop.', 'warezclient.', 'warezmaster.'], 'attack')
print(data_short)

# encoding categorical data of 3 fields: protocol_type, service, and
flag
labelencoder_x_1 = LabelEncoder()
labelencoder_x_2 = LabelEncoder()
labelencoder_x_3 = LabelEncoder()
```

```
x[:, 1] = labelencoder_x_1.fit_transform(x[:, 1])
x[:, 2] = labelencoder_x_2.fit_transform(x[:, 2])
x[:, 3] = labelencoder_x_3.fit_transform(x[:, 3])

#encoding data field: type
labelencoder_y = LabelEncoder()
y = labelencoder_y.fit_transform(y)

# Rescale data (between 0 and 1)
scaler = MinMaxScaler(feature_range=(0, 1))
rescaledX = scaler.fit_transform(x)
# summarize transformed data
set_printoptions(precision=3)

# Feature selection
model = LogisticRegression()
rfe = RFE(model, 20)
fit = rfe.fit(x, y)
print("Num Features: ", fit.n_features_)
print("Selected Features: ", fit.support_)
print("Feature Ranking: ", fit.ranking_)
x_select = fit.transform(x)
data_select = fit.transform(x)

#  Split into Train and Test Sets
seed = 7
num_folds = 10
kfold = KFold(n_splits=num_folds, random_state=seed)

#print result
d = KFold(n_splits=num_folds, random_state=seed)
model = LogisticRegression()
results = cross_val_score(model, x, y, cv=kfold)
print("Accuracy: ", results.mean()*100.0, results.std()*100.0)
```

Kết quả chương trình:

Thuật toán có độ chính xác: 99.395



CHƯƠNG 6. KẾT LUẬN

6.1 Kết luận

Sau khi hoàn thành tìm hiểu và giải quyết bài toán về bộ dữ liệu KDD, em tự nhận thấy mình còn tìm hiểu và giải quyết bài toán theo cách khá cơ bản, chưa áp dụng vào thực tế được. Tuy vậy, nhờ nghiên cứu và làm việc với đồ án môn học, em đã học được nhiều kiến thức mới về Machine Learning, đã hiểu hơn về các kiến thức toán học về ma trận, vi phân, nguyên hàm, cũng tìm hiểu được một số thuật toán cơ bản của Machine Learning, nó sẽ giúp ích cho em rất nhiều về sau. Em cảm ơn thầy đã cho em một đề tài thú vị để nghiên cứu.

6.2 Hướng phát triển của đồ án trong tương lai

Em mong muốn sự tìm hiểu của em được áp dụng trong thực tế, vì vậy, hướng phát triển của em sẽ là nghiên cứu và áp dụng Machine Learning để phát hiện các hoạt động bất thường trong hệ thống thật.

TÀI LIỆU THAM KHẢO

[1] Jason Brownlee "Machine Learning Mastery With Python", 2016.

[2] Khóa học online về Machine Learning của Andrew Ng https://www.coursera.org/learn/machine-learning.

[3] Các bài về Machine Learning, Gradient Descent, Logistic Regression trên trang https://en.wikipedia.org/.

[4] Bài viết trên trang https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

PHỤ LỤC

A1. Danh mục các từ viết tắt

ML: Machine Learning

DL: Deep Learning

GD: Gradient Descent

SGD: Stochastic Gradient Descent