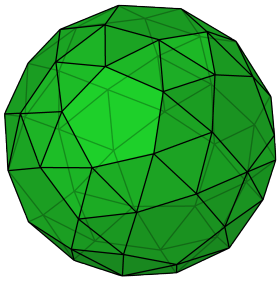
Projects in Mathematics and Applications

LINEAR PROGRAMMING FOR DATA SCIENCE

Ngày 29 tháng 6 năm 2025

Lê Tiến Hợp [*] [†]Trần Bảo Minh
Nguyễn Thị Bảo Tiên [‡] [§]Nguyễn Mai Anh Thư



[*]Trường THPT Chuyên Phan Bội Châu - Nghệ An
[†]Trường THPT Chuyên Hùng Vương - Bình Dương
[‡]Trường THPT Chuyên Lương Văn Chánh - Phú Yên
[§]Trường Phổ Thông Năng Khiếu - ĐHQG TP. Hồ Chí Minh

Lời cảm ơn

Chúng em xin chân thành cảm ơn Ban tổ chức Trại hè Toán học và Ứng dụng PiMA đã trao cho chúng em cơ hội để tham gia trại hè PiMA 2024 với chủ đề Quy hoạch Tuyến tính, đồng thời hỗ trợ tất cả chi phí di chuyển và ăn ở trong quá trình diễn ra trại hè. Thông qua những bài giảng của mentor và các buổi Workshop chia sẻ kiến thức từ khách mời, chúng em không chỉ học hỏi được những kiến thức mới mẻ, mà còn nhận ra Toán học có ứng dụng vô cùng quan trọng trong đời sống nói chung và các ngành về Khoa học nói riêng. Bên cạnh đó, chúng em cũng bày tỏ sự biết ơn sâu sắc đến Trường Đại Học Khoa Học Tự Nhiên - Đại Học Quốc Gia TP.HCM đã đồng ý trở thành địa điểm tổ chức sự kiện để các hoạt động chuyên môn được diễn ra thuận lợi. Cuối cùng, chúng em - Tiên, Thư, Hợp, và Minh - cảm thấy vô cùng may mắn khi được hướng dẫn bởi anh Nguyễn Ngọc Cảnh và anh Vòng Vĩnh Toàn. Sự tận tình trong chuyên môn và sự hài hước trong tính cách đã khiến cho hành trình 7 ngày của tụi em đáng nhớ hơn bao giờ hết.

Chúng em mong rằng trong tương lai phong trào Toán học ở Việt Nam nói chung sẽ ngày càng phát triển và Trại hè Toán học và Ứng dụng PiMA sẽ luôn được tổ chức rộng rãi.

Tóm tắt nội dung

Linear Regression là một kỹ thuật cơ bản nhưng hiệu quả trong phân tích dữ liệu và thống kê. Nó giúp ước lượng mối quan hệ tuyến tính giữa biến phụ thuộc và các biến độc lập trong bộ dữ liệu có sẵn, từ đó đưa ra dự đoán cho giá trị của những điểm dữ liệu mới trong tương lai. Trong LR, L^1-regression là một phương pháp quan trọng giúp giảm thiểu ảnh hưởng của các giá trị ngoại lai (outliers) trong dữ liệu. Giúp đưa ra dự đoán chính xác và hợp lý hơn khi dữ liệu bị nhiễu bởi những yếu tố bất thường.

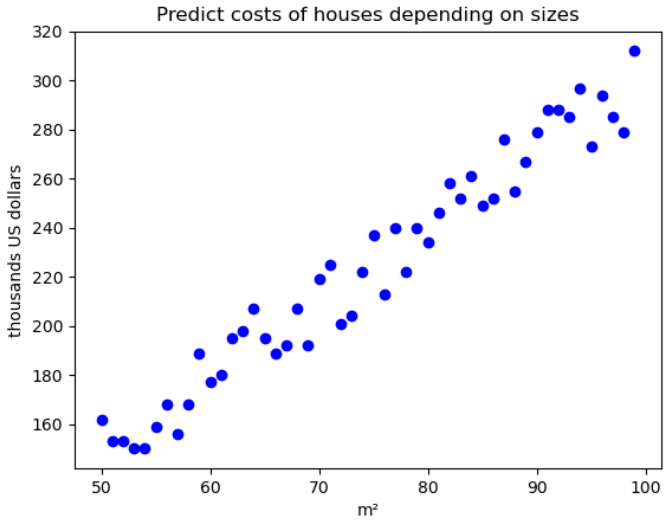
Support Vector Machine là một trong những thuật toán phân lớp hiệu quả và phổ biến nhất trong học máy. Dựa trên một tập các điểm dữ liệu thuộc một trong hai lớp cho trước, mô hình SVM sẽ được huấn luyện để dự đoán liệu một điểm dữ liệu mới nên thuộc về lớp nào. Khi phân loại trên dữ liệu nhiều chiều, một giải pháp hiệu quả trong SVM là sử dụng L^1-SVM. Tuy đây là một thuật toán phân loại nhưng nó cũng gián tiếp giảm chiều dữ liệu bằng cách triệt tiêu một số đặc trưng kém quan trọng trong dữ liệu, đồng thời đặt trọng số cao hơn vào những đặc trưng có ý nghĩa. Trong dự án này, chúng ta sẽ tìm hiểu và áp dụng L^1-SVM để giải quyết một bài toán thực tế thú vị: nhận dạng kí tự viết tay.

Mục lục

1 Linear Regression

1.1 Động lực

Trước tiên, ta xét bài toán ước lượng giá của một căn nhà có diện tích x m^2. Giả sử ta đã thu thập được số liệu từ 100 căn nhà trong một thành phố. Liệu rằng khi có một căn nhà mới xây có thông tin như trên thì ta có thể đưa ra mức giá phù hợp cho căn nhà đó không? Nếu có thì hàm dự đoán giá nhà $\hat{y} = f(x)$ sẽ được biểu diễn như thế nào?

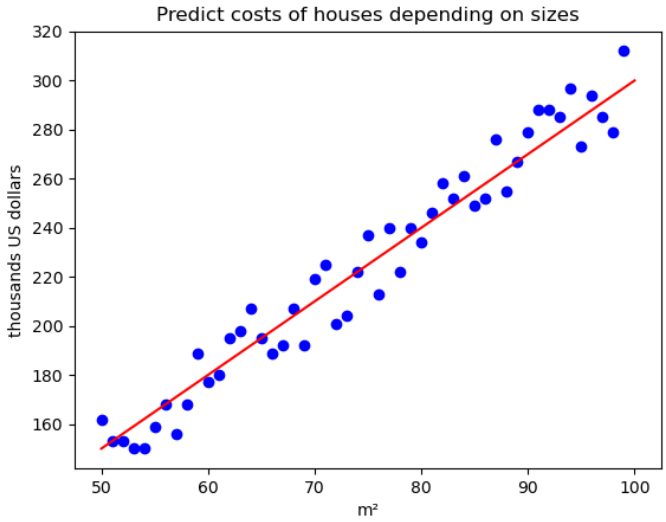


Đây là đồ thị biểu diễn số liệu của các căn nhà ở trên. Một cách trực quan, ta có thể thấy rằng nếu diện tích nhà càng lớn thì giá nhà càng cao. Mối quan hệ giữa diện tích và giá nhà có thể coi như một đường thẳng. Dựa trên quan sát này, ta có thể dự đoán đường thẳng biểu diễn mối quan hệ giữa giá nhà và diện tích bằng một hàm bậc nhất có dạng:

$$y \approx \hat{y} = f(x) = w_1 x + w_0$$

với $\hat{y}$ là hàm dự đoán được xây dựng dựa trên dữ liệu có sẵn.

Từ đó ta tìm cách tìm w_1 và w_0 để sai số so với các điểm dữ liệu là thấp nhất.



1.2 Các hướng tiếp cận Linear Regression là gì?

Trên thực tế, để dự đoán giá nhà ta cần rất nhiều yếu tố như số thiết bị, số phòng hay khoảng cách từ căn nhà đến trung tâm thành phố là bao nhiêu km.

Trong cuộc sống có rất nhiều bài toán cũng như vấn đề cần dự đoán dựa trên một vài số liệu cho trước nào đó. Và để giải quyết các bài toán hay vấn đề trong cuộc sống ta đưa các bài toán đó về bài toán có các giá trị đầu ra và giá trị đầu vào. Và trong dự án này, ta đề đến một bài toán có dạng có tên gọi là Linear Regression. Ở bài báo cáo này, ta tổng quát mô hình mối quan hệ giữa đầu ra và đầu vào bằng một hàm có dạng như sau:

$$y = w_0 + w_1 x_1 + \cdots + w_d x_d = x^T w,$$

trong đó: $w = [w_0, w_1, \cdots, w_d]^T$, $x = [1, x_1, \cdots, x_d]^T$, $d \in \mathbb{N}$ là số đặc tính của mỗi điểm dữ liệu.

Đến đây, ta xây dựng hàm mất mát để tối ưu các bài toán thuộc dạng trên.

1.2.1 L^1-regression

Xét các bài toán với các điểm (x_i, y_i), $i = 1, 2, \cdots, n$.

Ta xây dựng bài toán mất mát cho L^1-regression:

$$\mathcal{L}_1(w) = \frac{1}{n} \sum_{i=1}^{n} \left| y_i - x_i^T w \right|,$$

trong đó: n là số điểm dữ liệu ta thu thập được, y_i là giá trị đầu ra của mỗi điểm, $x_i = [1, x_{i1}, x_{i2}, \cdots, x_{id}]^T$ với x_{ij} là đặc tính thứ j của dữ liệu thứ i. Mặt khác, nhận thấy $\frac{1}{n}$ là một hằng số dương nên không ảnh hưởng đến giá trị tối ưu của $\mathcal{L}_1(w)$. Đến đây ta đưa bài về tìm giá trị tối ưu của: $\sum_{i=1}^{n} |y_i - x_i^T w|$.

Trước khi giải bài toán, ta nhắc lại bài toán Linear Programming như sau: Với $n, m_{ub}, m_{eq} \in \mathbb{N}$, $A_{ub} \in \mathbb{R}^{m_{ub} \times n}$, $b_{ub} \in \mathbb{R}^{m_{ub}}$, $A_{eq} \in \mathbb{R}^{m_{eq} \times n}$, $b_{eq} \in \mathbb{R}^{m_{eq}}$, $c, l, u \in \mathbb{R}^n$:

$$\begin{array}{rc} \text{minimize:} & c^T x \\ \text{subject to:} & A_{ub} x \leq b_{ub} \\ & A_{eq} x = b_{eq} \\ & l \leq x \leq u \end{array}$$

Để giải quyết bài toán Linear Programming, ta sử dụng đến hàm `scipy.optimize.linprog`: Ta chuyển đổi bài toán Linear Regression về bài toán Linear Programming.

Ta đặt $t_i = \left| y_i - \sum_j w_j x_{ij} \right|$

Ta chuyển bài toán về dạng:

$$\begin{array}{rc} \text{minimize:} & \sum_i t_i \\ \text{subject to:} & \left| y_i - \sum_j w_j x_{ij} \right| = t_i, i = 1, 2, \cdots, n. \end{array}$$

Trên thực tế khi để bài toán gốc thì ta rất khó xử lí do có rất nhiều trường hợp và số liệu. Vì vậy, để giải quyết bài toán gốc ta chuyển bài toán gốc về bài toán dưới đây để có thể sử dụng Linear Programming:

$$\begin{aligned} \text{minimize:} \quad & \sum_i t_i \\ \text{subject to:} \quad & \left|y_i - \sum_j w_j x_{ij}\right| \leq t_i, i = 1, 2, \cdots, n. \end{aligned}$$

Khi chuyển từ bài toán gốc qua bài toán Linear Programming thì nghiệm tối ưu của hai bài toán không đổi do:

- Nhận xét 1: Nghiệm tối ưu của bài toán Linear Programming là một nghiệm chấp nhận được của bài toán Linear Regression. Thật vậy, nếu $t_i > y_i - \sum_j w_j x_{ij}$ thì ta có thể kéo t_i về gần hơn thì ta có giá trị tối ưu hơn.
- Nhận xét 2: Miền nghiệm của bài toán Linear Programming là tập con của miền nghiệm của bài toán Linear Regression. Thật vậy, mỗi nghiệm của mãn bài toán Linear Programming đều thỏa mãn bài toán gốc. Do đó, giá trị tối ưu của hàm mục tiêu của bài toán LP sẽ nhỏ hơn hoặc bằng giá trị tối ưu của bài toán gốc.

Hay ta chuyển về bài toán sau:

$$\begin{aligned} \text{minimize:} \quad & \sum_i t_i \\ \text{subject to:} \quad & -t_i \leq y_i - \sum_j w_j x_{ij} \leq t_i, i = 1, 2, \cdots, n. \end{aligned}$$

Cụ thể, ta có thể biến đổi bài toán về dạng sau:

$$\begin{aligned} \text{minimize:} \quad & 0w_0 + 0w_1 + \cdots + 0w_m + t_1 + \cdots + t_n \\ \text{subject to:} \quad & w_0 + w_1 x_{11} + \cdots + w_d x_{1d} - t_1 \leq y_1 \\ & \vdots \\ & w_0 + w_1 x_{n1} + \cdots + w_d x_{nd} - t_n \leq y_n \\ & -w_0 - w_1 x_{11} - \cdots - w_d x_{1d} - t_1 \leq -y_1 \\ & \vdots \\ & -w_0 - w_1 x_{n1} - \cdots - w_d x_{nd} - t_n \leq -y_n \end{aligned}$$

Sau khi đã hoàn thiện việc chuyển đổi bài toán Linear Regression ban đầu về bài toán Linear Programming và mô hình hoá bài toán. Ta có thể bắt đầu cài đặt thuật toán, ở đây ta có thể sử dụng thư viện `scipy.optimize` để giải quyết bài toán Linear Programming.

1.2.2 scipy.optimize

`linprog(c, A_ub=None, b_ub=None, A_eq=None, b_eq=None, bounds=(0, None), methods='highs', callback=None, options=None, x_0 = None, integrality=None)` Tuy nhiên, ở đây ta chỉ quan tâm đến 4 tham số, lần lượt là: `c, A_ub, b _ub, bounds`. Từ việc phân tích bài toán trên, ta có thể cài đặt thuật toán một cách dễ dàng:

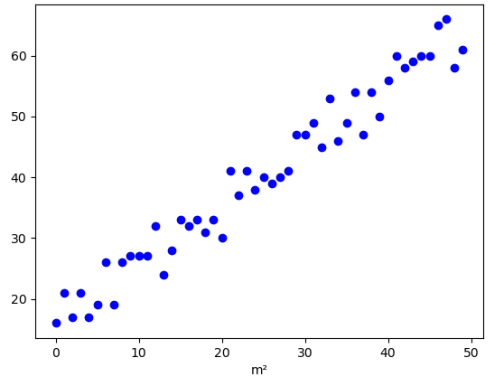
$$\texttt{c} = \left[(0)_{1\times d} \quad | \quad (1)_{1\times n}\right] = \left[0 \quad 0 \quad \cdots \quad 0 \quad 1 \quad 1 \quad \cdots \quad 1\right]$$

$$\texttt{b\_ub} = \left[(y)_{1\times n} \quad | \quad -(y)_{1\times n}\right] = \left[y_1 \cdots y_n - y_1 \cdots - y_n\right]$$

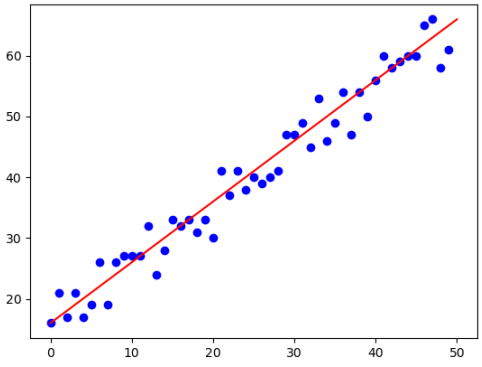
$$\texttt{A\_ub} = \left[\begin{array}{c|c|c} (1)_{n\times 1} & X_{n\times d} & I_{n\times n} \\ \hline -(1)_{n\times 1} & -X_{n\times d} & -I_{n\times n} \end{array}\right]$$
$$= \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} & 1 & 0 & \cdots & 0 \\ 1 & x_{21} & \cdots & x_{2d} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nd} & 0 & 0 & \cdots & 1 \\ -1 & -x_{11} & \cdots & -x_{1d} & -1 & 0 & \cdots & 0 \\ -1 & -x_{21} & \cdots & -x_{2d} & 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & -x_{n1} & \cdots & -x_{nd} & 0 & 0 & \cdots & -1 \end{bmatrix}$$

$$\texttt{bounds} = (\text{None}, \text{None})$$

Đến đây, ta chỉ việc gọi hàm linprog là đã hoàn thành bài toán.



(a) Dataset



(b) L^1 Regression on the dataset

1.2.3 L^2 - regression

Tương tự với L^1-regression, ta xây dựng hàm mất mát với L^2-regression như sau:

$$\mathcal{L}_2(w) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - x_i^T w\right)^2.$$

Trước khi giải bài toán trên, ta viết gọn hàm số dưới dạng ma trận, vector và norm:

$$\mathcal{L}_2(w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i^T w)^2 = \frac{1}{n}\left\|\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} - \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix} w\right\|^2 = \frac{1}{n}\|y - Xw\|^2.$$

với $y = [y_1, y_2, \cdots, y_n]^T, X = [x_1, x_2, \cdots, x_n]^T$.

Do $1/n$ là hằng số dương nên ta cần tìm giá trị tối ưu của Để tối ưu hàm mất mất trên, ta đưa về bài toán tìm giá trị tối ưu của $f(w) = \|y - Xw\|^2$.

Giải ta có:

$$
\begin{aligned}
f(w) &= \|y - Xw\|^2 \\
&= \langle Xw - y, Xw - y \rangle \\
&= (Xw - y)^T (Xw - y) \\
&= (X^T w^T - y^T)(Xw - y) \\
&= X^T w^T Xw - y^T Xw - w^T X^T y + y^T y \\
&= w^T (X^T X) xw - (2y^T X) w + y^T y.
\end{aligned}
$$

Do $y^T Xw = X^T w^T y$ nên $\nabla f(w) = 2(X^T X)w - 2X^T y$ và $\nabla^2 f(w) = 2(X^T X)$. Hơn nữa, $h^T \nabla^2 f(w) h = 2[h^T (X^T X) h] = 2[(h^T X^T)(h)] = 2||Xh||^2 \geq 0 \ \forall h \in \mathbb{R}^N$ hay ma trận $\nabla^2 f(w)$ xác định dương.

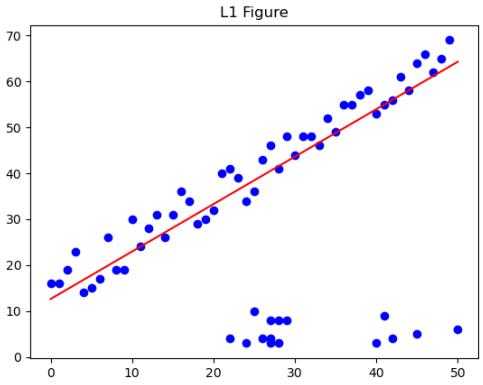
Hơn nữa, rank$(X) = n$ nên rank$(X^T X) = n$, hay $X^T X$ khả nghịch.

Vậy nghiệm $w^* = (X^T X)^{-1} X^T y$ của $\nabla f(x) = 0$ là cực tiểu toàn cục của hàm f.

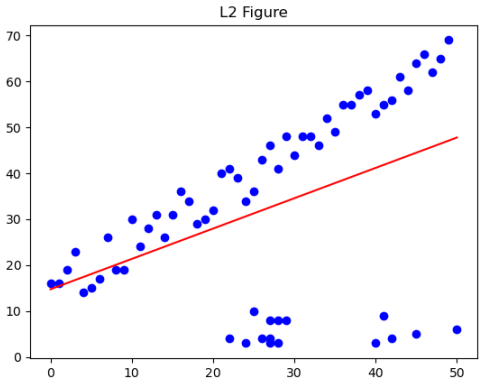
Đến đây ta tìm được giá trị tối ưu.

1.2.4 So sánh L^1-regression và L^2-regression

- Nhận xét: So với L^1-regression thì L^2-regression rất nhạy với các điểm nhiễu. Do vậy, khi xuất hiện các điểm nhiễu thì độ chính xác của L^1-regression thường cao hơn so với L^2-regreesion.



(a) L^1-regression



(b) L^2-regression

Ta có nhận xét về 2 công thức hàm mất mát của L^1 và L^2 regression:

$$\mathcal{L}_1(w) = \frac{1}{n} \sum_{i=1}^{n} \left| y_i - x_i^T w \right|$$

$$\mathcal{L}_2(w) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - x_i^T w \right)^2$$

Ta nhận thấy, mỗi sai số của L^2 sẽ được bình phương dẫn đến sai số nhiều hơn so với L^1.

1.3 Polynomial Regression

Ta có thể mở rộng và áp dụng Linear Programming để thực hiện Polynomial Regession.

Mô hình hồi quy đa thức bậc d (Polynomial Regression) có dạng:

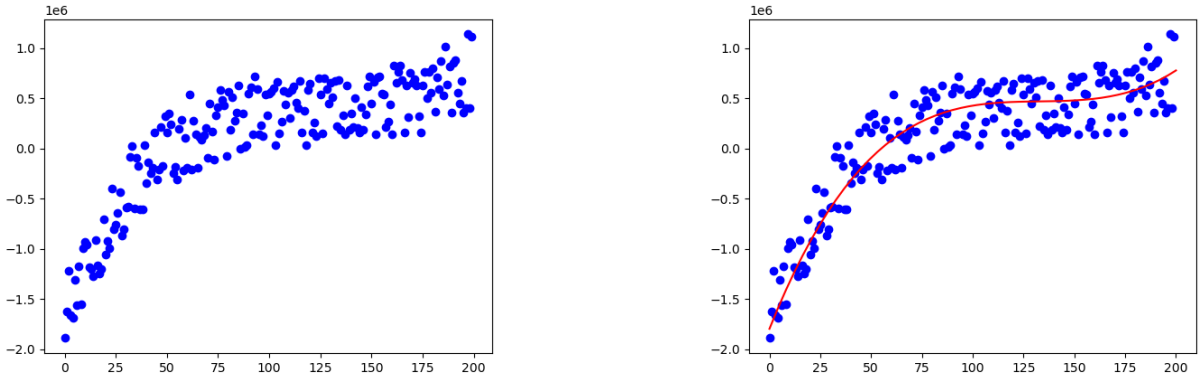
$$\begin{aligned} f : \mathbb{R} &\to \mathbb{R} \\ x &\mapsto \hat{y} = f(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_d x^d \end{aligned}$$

Ta gọi $\mathbf{x} = [x^0, x^1, \cdots, x^d]^T$ và $w = [w_0, w_1, \cdots, w_d]^T$.

Dễ nhận thấy, (x_i, y_i) là dữ liệu đầu vào. Do đó, ta chỉ đang tìm w bằng việc tối ưu $|y - w^T \mathbf{x}|$.

Xem xét kỹ về cách ta cài đặt hàm `linprog` để giải bài toán L^1-regression bằng Linear Programming. Nhận thấy, ở đây ta chỉ tối ưu bài toán bằng việc tìm những giá trị w_i và t_i. Từ đó, để thực hiện hồi quy đa thức (hay Polynomial Regession) ta chỉ cần thay đổi `A_ub` như sau:

$$\begin{aligned} \texttt{A\_ub} &= \left[\begin{array}{c|c|c} (1)_{n\times 1} & X_{n\times d} & I_{n\times n} \\ \hline -(1)_{n\times 1} & -X_{n\times d} & -I_{n\times n} \end{array}\right] \\ &= \begin{bmatrix} 1 & x_1 & (x_1)^2 & \cdots & (x_1)^d & 1 & 0 & \cdots & 0 \\ 1 & x_2 & (x_2)^2 & \cdots & (x_2)^d & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & (x_n)^2 & \cdots & (x_n)^d & 0 & 0 & \cdots & 1 \\ -1 & -x_1 & -(x_1)^2 & \cdots & -(x_1)^d & -1 & 0 & \cdots & 0 \\ -1 & -x_2 & -(x_2)^2 & \cdots & -(x_2)^d & 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & -x_n & -(x_n)^2 & \cdots & -(x_n)^d & 0 & 0 & \cdots & -1 \end{bmatrix} \end{aligned}$$



(a) Dataset

(b) Polynomial Regression on the dataset

1.4 Ứng dụng của Linear Regression

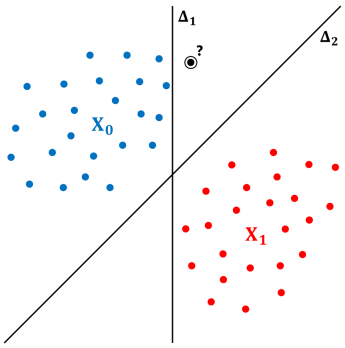
Ngày nay, mô hình Linear Regression được ứng dụng trong nhiều lĩnh vực như:

- Dự báo giá cả: dự đoán giá nhà, giá cổ phiếu, giá nhiên liệu dựa trên các yếu tố như vị trí, kích thước, chất lượng, lượng cung cầu,...

- Dự báo điểm số: dự đoán điểm số của học sinh dựa trên thời gian học, nỗ lực, kỹ năng, trình độ giáo viên,...
- Dự báo sản phẩm: dự đoán đầu ra sản xuất dựa trên thời gian, công suất, nguyên liệu, lao động,...
- Phân tích chuỗi thời gian: dự đoán xu hướng và chu kỳ của các chuỗi dữ liệu, như bất động sản, thời tiết, xu hướng sản xuất,...

2 Support Vector Machine

2.1 Tổng quan



Hình 4: Có nhiều đường phân chia 2 lớp X_0 và X_1

Support Vector Machine (SVM) là một trong những thuật toán phân lớp phổ biến và hiệu quả trong lĩnh vực Machine Learning. SVM cực kỳ hữu ích khi làm việc với dữ liệu có nhiều chiều. Một số ứng dụng phổ biến của nó là phân loại hình ảnh, phát hiện gian lận, ...

Trong bài toán học có giám sát (supervised learning), để huấn luyện mô hình và giúp mô hình nhận diện được các kí tự viết tay, ta sẽ có các dữ liệu (data) đầu vào của ảnh kí tự viết tay và các nhãn tương ứng.

Ví dụ như trong **Hình 4**, tập hợp các điểm X_0 (màu xanh) sẽ tượng trưng cho dữ liệu ảnh của số 0 và tập hợp các điểm X_1 (màu đỏ) sẽ tượng trưng cho dữ liệu ảnh của số 1. Mỗi tập hợp sẽ có các nhãn (label) tương ứng để mô hình nhận diện.

Để mô hình học cách nhận diện tốt nhất, ta cần tìm mặt phân chia 2 tập hợp dữ liệu tốt nhất, tức để 2 tập hợp dữ liệu cách xa nhau nhất. Khoảng cách từ các điểm gần nhất của một tập hợp tới mặt phân chia được gọi là margin. Vậy ta cần đi tìm mặt phân chia sao cho margin lớn nhất.

Bài toán tối ưu trong Support Vector Machine là bài toán đi tìm mặt phân chia tốt nhất (sao cho margin giữa hai lớp bằng nhau và lớn nhất). Ranh giới đơn giản nhất trong không gian một chiều là một đường thẳng, trong không gian ba chiều là một mặt phẳng, trong không gian nhiều chiều hơn là một siêu mặt phẳng hoặc siêu phẳng (hyperplane).

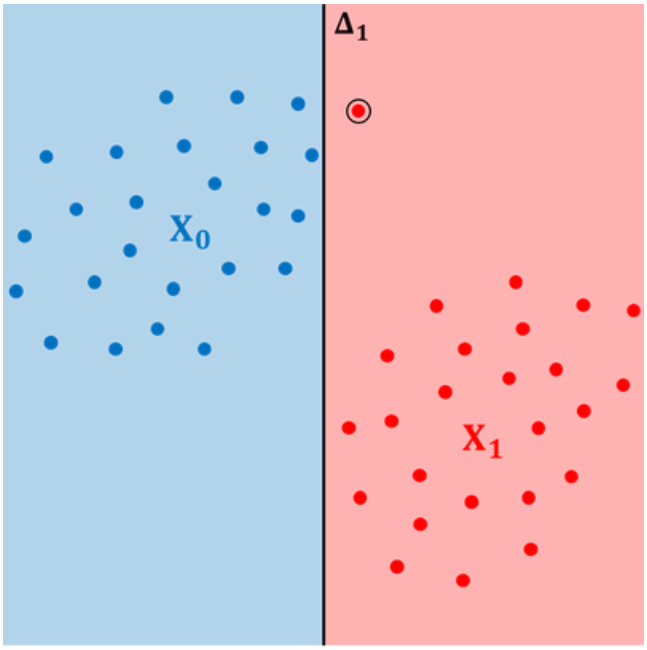
2.2 Xây dựng bài toán tối ưu cho thuật toán SVM

Cho các cặp dữ liệu $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$ với vectơ $x_i \in R^d$ thể hiện đầu vào của một điểm dữ liệu và y_i là nhãn của dữ liệu đó.

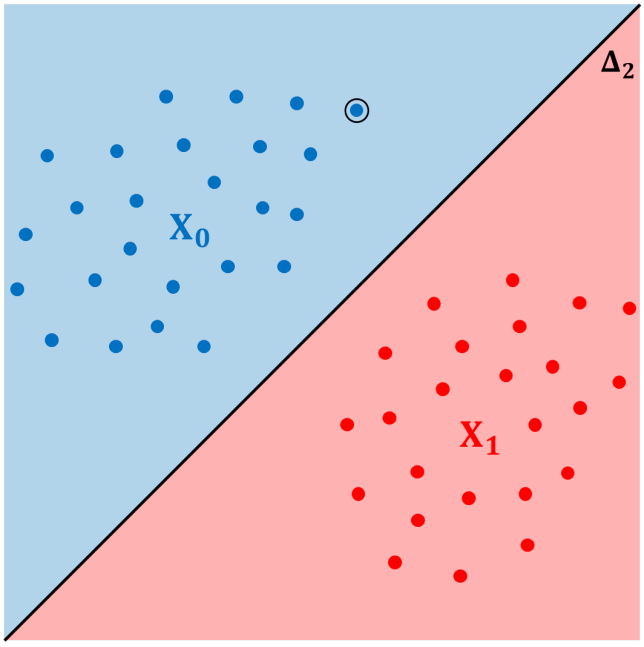
Giả sử rằng các điểm màu xanh có nhãn là 1, các điểm màu đỏ có nhãn là -1. Khi đó, y_n luôn cùng dấu với $w^T x_n + w_0$.

Với cặp dữ liệu (x_i, y_i) bất kỳ, khoảng cách từ điểm đó tới siêu phẳng $w_1 X_0 + w_2 X_2 + \cdots + w_d X_d + w_0 = 0$ là:

$$\frac{\left|w^T x_i + w_0\right|}{\|w\|} = \frac{y_i(w^T x_i + w_0)}{\|w\|}.$$



Hình 5: Đường phân chia thứ nhất (chưa phải đường tối ưu)



Hình 6: Đường phân chia thứ hai (tốt hơn đường phân chia thứ nhất)

Với mặt phân chia này, margin được tính là khoảng cách gần nhất từ một điểm (trong cả hai lớp, vì cuối cùng margin của cả hai lớp sẽ bằng nhau) tới mặt đó, tức là

$$\text{margin} = \min_i \frac{y_i(w^T x_i + w_0)}{\|w\|}.$$

Bài toán Support Vector Machine (SVM): tìm w và b sao cho margin đạt max

$$\begin{aligned}(w, w_0) &= \arg\max_{w,w_0} \left\{ \min \frac{y_i(w^T x_i + w_0)}{\|w\|} \right\} \\ &= \arg\max_{w,w_0} \left\{ \frac{1}{\|w\|} \min_i y_i(w^T x_i + w_0) \right\}.\end{aligned}$$

Nhận xét: Xét khoảng cách từ một điểm tới mặt phân chia là $y_i(w^T x_i + w_0)/\|w\|$.

Gọi (x_k, y_k) là điểm gần nhất của một lớp tới mặt phân chia. Giả sử $y_k(w^T x_k + w_0) = a$. Khi ta chia cả tử và mẫu của phân số cho a, giá trị của tử số bằng 1, còn mẫu số sẽ trở thành:

$$\left\|\frac{1}{a}w\right\|_p = \left(\sum_{i=1}^{d}\left|\frac{1}{a}w\right|^p\right)^{\frac{1}{p}} = \frac{1}{a}\left(\sum_{i=1}^{d}|w|^p\right)^{\frac{1}{p}}$$

Nếu đặt z = w/a , ta thấy nó lại tương tự với phương trình ban đầu trước khi chia a. Vậy nên, không mất tính tổng quát:

$$y_i(w^T x_i + w_0) = 1$$

với những điểm nằm gần mặt phân chia nhất. Như vậy, với mọi n ta luôn có:

$$y_i(w^T x_i + w_0) \geq 1$$

Do đó, bài toán tối ưu có thể viết lại dưới dạng:

$$\begin{aligned} \underset{w,w_0}{\text{maximize}} \quad & \frac{1}{\|w\|} \\ \text{subject to:} \quad & y_i(w^T x_i + w_0) \geq 1 \quad \forall i = 1, 2, \cdots, n \end{aligned}$$

Hay nói cách khác, bài toán có thể chuyển đổi về dạng:

$$\begin{aligned} \underset{w,w_0}{\text{minimize}} \quad & \|w\| \\ \text{subject to:} \quad & y_i(w^T x_i + w_0) \geq 1 \quad \forall i = 1, 2, \cdots, n \end{aligned}$$

2.3 Hướng tiếp cận Linear Programming

Để đưa bài toán về dạng Linear Programming, ta có thể sử dụng L^1-norm để đưa $\|w\|$ về dạng tổng của các phần tử $|w_1| + |w_2| + \cdots + |w_n|$.

Việc xuất hiện phép lấy giá trị tuyệt đối sẽ gây không ít khó khăn trong quá trình giải toán nên ta sẽ biến đổi nó thành dạng bất đẳng thức.

Bài toán viết lại dưới dạng: Đặt $t_j = \|w_j\|, \quad \forall j = 1, 2, \cdots, d$

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{d} t_j \\ \text{subject to:} \quad & y_n(w^T x_i + w_0) \geq 1 \quad \forall i = 1, 2, \cdots, n \\ & t_j = |w_j| \quad \forall j = 1, 2, \cdots, d \end{aligned}$$

Nghiệm tối ưu của bài toán trên cũng chính là nghiệm tối ưu của bài toán sau đây (Xem mục 1.2.1 - Nhận xét 1):

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{d} t_j \\ \text{subject to:} \quad & y_i(w^T x_i + w_0) \geq 1 \quad \forall i = 1, 2, \cdots, n \\ & -t_j \leq w_j \leq t_j \quad \forall j = 1, 2, \cdots, d. \end{aligned}$$

2.4 Khả năng giảm chiều dữ liệu của L^1 - SVM

Trong các bài toán thực tế, số lượng điểm dữ liệu và kích thước của các vector thường rất lớn. Dữ liệu quá lớn sẽ gây cản trở trong quá trình lưu trữ và tốc độ tính toán. Chính vì thế, chúng ta cần phải có phương pháp giảm chiều dữ liệu hoặc lựa chọn đặc trưng quan trọng. Trong đoạn **code của Support Vector Machine**, ta thấy giá trị 0 xuất hiện khá nhiều ở phần kết quả. Điều này là minh chứng cho việc L_1 - SVM có thể giảm số chiều dữ liệu. Công thức SVM sử dụng L_1 - norm được đề xuất trong [2] hầu hết sử dụng ít biến hơn và tốn ít thời gian hơn so với các công thức SVM sử dụng norm khác.

2.5 Ứng dụng của Support Vector Machine

- Nhận diện hình ảnh: chẩn đoán hình ảnh y khoa, nhận diện khuôn mặt, nhận diện chữ viết tay,...
- Tin sinh học
- Nhận diện ký tự: phân loại thư rác, phân tích quan điểm,...

3 Phụ lục

Code mẫu của phần Linear Regression: **Hands-on Demo Code of Linear Regression**

Code mẫu của phần Support Vector Machine: **Hands-on Demo Code of Support Vector Machine**

Tài liệu

[1] *Các bài giảng PiMA 2024*.

[2] Paul S. Bradley & O. L. Mangasarian. Feature selection via concave minimization and support vector machines. *In Proceedings of the Fifteenth International Conference (ICML)*, pages 82–90, 1998.

[3] Vũ Hữu Tiệp. *Machine Learning cơ bản, [online]*. 2018.

[4] Robert J. Vanderbei. *Linear Programming: Foundations and Extensions (4th edition)*. Springer, 2020.