Báo cáo thực tập ngày 20/06/2024

Nguyễn Tiến Đạt - Thực tập sinh AI

1 HỒI QUY SOFTMAX - SOFTMAX REGRESSION

Multinomial Logistic Regression (Softmax Regression) là một phương pháp phân loại đa lớp phổ biến trong học máy và trí tuệ nhân tạo. Nó mở rộng từ logistic regression, thường được sử dụng cho các bài toán phân loại nhị phân, để xử lý các bài toán phân loại có nhiều lớp.

Softmax Regression là tên gọi khác của Multinomial Logistic Regression vì nó sử dụng hàm softmax để chuyển đổi các đầu ra của mô hình thành xác suất của các lớp. Tên "softmax" xuất phát từ tính chất của hàm này, chuyển đổi một vector các giá trị thực (logits) thành một phân phối xác suất mềm (soft probability distribution).

Softmax regression thường được ứng dụng trong:Nhận dạng chữ viết tay, phân loại ảnh, xử lý ngôn ngữ tự nhiên (ví dụ: phân loại văn bản) cũng như các bài toán khác yêu cầu phân loại nhiều lớp.

1.1 One-Hot Encoding

1.1.1 Định nghĩa

Trong học máy và xử lý dữ liệu, one-hot encoding là một kỹ thuật biến đổi dữ liệu phân loại thành dạng số học, để mô hình có thể hiểu và xử lý dữ liệu này. Kỹ thuật này thường được sử dụng khi biến đổi biến phân loại thành các biến giả, sao cho mỗi giá trị phân loại được biểu diễn bằng một vectơ có kích thước bằng số lượng giá trị phân loại, trong đó chỉ có một phần tử bằng 1 (được gọi là "hot"), còn lại đều bằng 0.

1.1.2 Mục đích

Biểu Diễn Dữ Liệu Phân Loại: Mục tiêu chính của one-hot encoding là biến đổi dữ liệu phân loại thành dạng số học, giúp mô hình có thể hiểu và xử lý dữ liệu này. Với one-hot encoding, mỗi giá trị phân loại được biểu diễn bằng một vectơ, nơi mà chỉ có một phần tử có giá trị là 1 và các phần tử còn lại bằng 0. Điều này giúp mô hình hiểu được mối quan hệ giữa các giá trị phân loại mà không cần phải giả định mối quan hệ tuyến tính.

Áp Dụng Cho Phân Loại Đa Lớp: One-hot encoding thường được sử dụng trong các bài toán phân loại đa lớp, nơi mà mỗi mẫu dữ liệu có thể thuộc vào một trong nhiều lớp khác nhau. Bằng cách biến đổi các nhãn lớp thành dạng one-hot encoding, mô hình có thể dễ dàng hiểu và xử lý thông tin về các lớp mà mỗi mẫu dữ liệu có thể thuộc vào.

Sử Dụng trong Mạng Nơ-ron và Học Sâu: Trong mạng nơ-ron và các mô hình học sâu, one-hot encoding thường được sử dụng để biểu diễn các nhãn lớp đầu ra. Mỗi lớp đầu ra của mạng nơ-ron có thể được biểu diễn bằng dạng one-hot encoding, giúp mô hình dễ dàng học được mối quan hệ giữa các lớp và đưa ra dự đoán chính xác.

1.1.3 Ví dụ

Giả sử bạn có một bộ dữ liệu về chữ số viết tay, bao gồm 3 ảnh như sau:

1) Ảnh thứ nhất: Một bức ảnh của chữ số 2.
2) Ảnh thứ hai: Một bức ảnh của chữ số 4.
3) Ảnh thứ ba: Một bức ảnh của chữ số 9.

Sau khi áp dụng one-hot encoding, giá trị của mỗi ảnh sẽ được biểu diễn như sau:

- Ảnh thứ nhất: [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
- Ảnh thứ hai: [0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
- Ảnh thứ ba: [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]

Điều này giúp mô hình máy học hiểu và xử lý dữ liệu phân loại một cách hiệu quả, và dễ dàng diễn giải kết quả dự đoán của mô hình.

1.2 Hàm Softmax

Hàm softmax chuyển đổi một vector các giá trị thực thành một vector xác suất, trong đó tổng các xác suất bằng 1. Hàm softmax được định nghĩa như sau:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}$$

trong đó:

- z_i là giá trị đầu vào của lớp i.
- C là số lượng lớp.
- e là cơ số của logarit tự nhiên.

Tuy nhiên, Khi z_i quá lớn, việc tính toán $\exp(z_i)$ có thể gây ra hiện tượng tràn số .Ta khắc phục bằng cách sử dụng phiên bản ổn định hơn của hàm SoftMax là Softmax-stable:

$$\frac{\exp(z_i)}{\sum_{j=1}^{C}\exp(z_j)} = \frac{\exp(-Cnt)\exp(z_i)}{\exp(-Cnt)\sum_{j=1}^{C}\exp(z_j)} = \frac{\exp(z_i - Cnt)}{\sum_{j=1}^{C}\exp(z_j - Cnt)}$$

với Cnt là một hằng số bất kỳ. Trong thực nghiệm, giá trị dự lớn này thường được chọn là $Cnt = \max_i z_i$.

1.3 Loss Function - Cross Entropy

1.3.1 Định nghĩa

Cross entropy là một khái niệm quan trọng trong lý thuyết thông tin và học máy. Nó đo lường sự khác biệt giữa phân phối thực tế P và phân phối dự đoán Q. Công thức tính cross entropy giữa hai phân phối rời rạc P và Q trên một tập hợp các sự kiện C được biểu diễn như sau:

$$H(P, Q) = -\sum_{x \in C} P(x) \log Q(x)$$

Trong trường hợp này ta sẽ thay các giá trị P(X) bằng y_x và thay Q(X) bằng giá trị của hàm Softmax hoặc Softmax-stable để được công thức loss funtion cho quá trình training như sau:

$$\ell_n(\theta) = -\sum_{x \in C} y_x \log\left(\frac{e^{x^T\theta}}{\sum_j e^{x_j^T\theta}}\right)$$

Trong đó:

- C là tập hợp các lớp (classes).
- y_x là nhãn thực tế (ground truth), với $y_x = 1$ nếu x là lớp đúng, và $y_x = 0$ với các lớp khác.
- x là vector đặc trưng của mẫu dữ liệu.
- θ là vector trọng số của mô hình.
- $x^T\theta$ là tích vô hướng giữa vector đặc trưng và vector trọng số, đại diện cho logit (giá trị đầu ra của lớp tương ứng) từ mô hình.
- $\sum_j e^{x_j^T\theta}$ là tổng của các exponential của logits để normalize thành phân phối xác suất.

1.4 Stochastic Gradient Descent - SGD

1.4.1 Định nghĩa

Stochastic Gradient Descent (SGD) là một phương pháp tối ưu hóa được sử dụng rộng rãi trong việc huấn luyện các mô hình học máy, đặc biệt là các mạng nơ-ron. SGD là một biến thể của phương pháp gradient descent truyền thống, nhưng thay vì tính toán gradient trên toàn bộ tập dữ liệu, nó chỉ sử dụng một phần nhỏ của dữ liệu (một mini-batch) để tính toán gradient và cập nhật trọng số mô hình.

1.4.2 Thuật toán

Thuật toán SGD hoạt động như sau:

1) Khởi tạo trọng số của mô hình với giá trị ngẫu nhiên.
2) Lặp lại các bước sau cho đến khi đạt được điều kiện dừng:
   a) Chọn ngẫu nhiên một mini-batch từ tập dữ liệu huấn luyện.
   b) Tính toán gradient của hàm mất mát (loss function) trên mini-batch đó.
   c) Cập nhật trọng số của mô hình theo hướng ngược lại với gradient, với một tỷ lệ học tập (learning rate) nhất định.

1.4.3 Công thức cập nhật trọng số

Trước tiên, ta sẽ tính gradient của hàm mất mát (loss function) của từng mẫu dữ liệu đối với tham số mô hình:

$$\frac{\partial J_i(\theta)}{\partial \theta} = -\frac{\partial \ell_i(\theta)}{\partial \theta} = -\mathbf{x}_i \mathbf{e}_i^T \tag{1}$$

Trong đó:

- $J_i(\theta)$: Hàm loss đối với mẫu dữ liệu thứ i.
- $\ell_i(\theta)$: Hàm log-likelihood âm (negative log-likelihood loss) cho mẫu thứ i.
- θ: Vector các tham số của mô hình.
- $\mathbf{x}_i$: Vector đặc trưng của mẫu dữ liệu thứ i.
- $\mathbf{e}_i$: Vector sai số (error) của mẫu thứ i, thường được định nghĩa là sự khác biệt giữa giá trị dự đoán và giá trị thực tế.

Cập nhật trọng số W bằng Stochastic Gradient Descent - SGD như sau:

$$\mathbf{W} := \mathbf{W} - \eta \left(\frac{\partial J_i(\theta)}{\partial \theta} \right) = \mathbf{W} - \eta \left(-\mathbf{x}_i \mathbf{e}_i^T \right) \tag{2}$$

- $\mathbf{W}$: Trọng số của mô hình.
- $J_i(\theta)$: Hàm mất mát cho mẫu dữ liệu thứ i với tham số θ.
- $\frac{\partial J_i(\theta)}{\partial \theta}$: Gradient của hàm mất mát $J_n(\theta)$ theo tham số θ.
- η: Tốc độ học (learning rate).
- $\mathbf{x}_i$: Đầu vào của mẫu dữ liệu thứ n.
- $\mathbf{e}_i$: Sai số dự đoán của mẫu dữ liệu thứ i.
- $\mathbf{e}_i^T$: Chuyển vị của sai số dự đoán $\mathbf{e}_i$.

1.5 Thực nghiệm kết quả

1.5.1 Input

Tập dữ liệu đầu vào bao gồm 48 000 ảnh chữ viết tay có kích thước 28x28 pixel, tương đương 784 điểm ảnh (pixels). Tập dữ liệu này được chia thành hai phần: tập huấn luyện và tập kiểm tra, với tỷ lệ tương ứng là 0.8 và 0.2.

1.5.2 Tập huấn luyện

Tập huấn luyện bao gồm $N_1 = 38.400$ mẫu dữ liệu, mỗi mẫu là một ảnh chữ viết tay. Dữ liệu huấn luyện được biểu diễn dưới dạng ma trận có kích thước $N_1 \times d$, trong đó $d = 784$ là số điểm ảnh của mỗi ảnh.

$$X_{\text{train}} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,784} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,784} \\ \vdots & \vdots & \ddots & \vdots \\ x_{38400,1} & x_{38400,2} & \cdots & x_{38400,784} \end{pmatrix}_{38400 \times 784}$$

1.5.3 Tập kiểm tra

Tập kiểm tra bao gồm $N_2 = 9.600$ mẫu dữ liệu, cũng là các ảnh chữ viết tay. Dữ liệu kiểm tra được biểu diễn dưới dạng ma trận có kích thước $N_2 \times d$, với $d = 784$ là số điểm ảnh của mỗi ảnh.

1.5.4 Nhãn đầu vào

Mỗi ảnh chữ viết tay trong tập dữ liệu được gán một nhãn từ 0 đến 9, tương ứng với chữ số viết tay trong ảnh đó. Để phù hợp với quá trình huấn luyện, các nhãn đầu vào cần được chuyển đổi sang dạng one-hot encoding, trong đó mỗi nhãn được biểu diễn bằng một vector có chiều dài 10, với phần tử tại vị trí tương ứng với nhãn có giá trị 1, các phần tử khác có giá trị 0.

1.6 Quá trình thực hiện

Bước 1: Khởi tạo ma trận trọng số

Trong bài toán phân loại nhiều lớp này, chúng ta sẽ khởi tạo ma trận trọng số ban đầu là một ma trận toàn số 0 với kích thước $d \times C$, trong đó $d = 784$ (tương ứng với số điểm ảnh của mỗi ảnh đầu vào) và $C = 10$ (tương ứng với số lớp cần dự đoán, từ 0 đến 9).

$$W = \begin{pmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,9} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,9} \\ \vdots & \vdots & \ddots & \vdots \\ w_{783,0} & w_{783,1} & \cdots & w_{783,9} \end{pmatrix}$$

Việc khởi tạo trọng số ban đầu bằng 0 là một cách khởi tạo phổ biến trong các mô hình học máy. Sau đó, trong quá trình huấn luyện, các giá trị trọng số sẽ được cập nhật dần dần sử dụng các thuật toán tối ưu như Stochastic Gradient Descent (SGD) hoặc các biến thể của nó, nhằm tối thiểu hóa hàm mất mát và cải thiện hiệu năng dự đoán của mô hình.

Bước 2: Sử dụng hàm Softmax để tính xác suất của từng ảnh với mỗi lớp phân loại.

Để dự đoán xác suất của một ảnh đầu vào thuộc vào mỗi lớp từ 0 đến 9, chúng ta sẽ áp dụng phép nhân ma trận X_{train} với ma trận trọng số W, và sau đó áp dụng hàm softmax để tính xác suất của mỗi ảnh thuộc vào từng lớp (từ 0 đến 9). Ví dụ với dòng thứ i của ma trận X_{train}, tương ứng với ảnh thứ i, ta có thể tính toán như sau

$$\begin{aligned} x_i &= \begin{pmatrix} x_{i,1} & x_{i,2} & \cdots & x_{i,784} \end{pmatrix} \\ z_i &= \begin{pmatrix} x_{i,1} & x_{i,2} & \cdots & x_{i,784} \end{pmatrix} \begin{pmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,9} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,9} \\ \vdots & \vdots & \ddots & \vdots \\ w_{783,0} & w_{783,1} & \cdots & w_{783,9} \end{pmatrix} \\ &= \begin{pmatrix} z_{i,0} & z_{i,1} & \cdots & z_{i,9} \end{pmatrix} \\ y_i &= \text{softmax}(z_i) \\ &= \frac{1}{\sum_{j=0}^{9} e^{z_{i,j}}} \begin{pmatrix} e^{z_{i,0}} & e^{z_{i,1}} & \cdots & e^{z_{i,9}} \end{pmatrix} \end{aligned}$$

Trong đó:

x_i là dòng thứ i của ma trận đầu vào X_{train}, biểu diễn ảnh thứ i, có kích thước 1×784.

z_i là vector được tính bằng cách nhân x_i với ma trận trọng số W, có kích thước 1×10.

y_i là vector xác suất dự đoán cho ảnh thứ i, cũng có kích thước 1×10.

Bước 3: Cập nhật trọng số ma trận $\mathbf{W}$ trong một mô hình học máy dựa trên quá trình giảm dần sai số (Stochastic gradient descent)

$$\mathbf{W} := \mathbf{W} - \eta \left(-\mathbf{x}_i \mathbf{e}_i^T \right) \tag{3}$$

Với $\eta = 1 \times 10^{-5}$, x_i đại diện cho từng dòng của ma trận X_{train}, tức là dữ liệu điểm ảnh của mỗi ảnh trong tổng số N ảnh.

$$\begin{aligned} e_i &= y_i - \hat{y}_i \\ &= \begin{pmatrix} y_i^{(1)} \\ y_i^{(2)} \\ \vdots \\ y_i^{(C)} \end{pmatrix} - \begin{pmatrix} \hat{y}_i^{(1)} \\ \hat{y}_i^{(2)} \\ \vdots \\ \hat{y}_i^{(C)} \end{pmatrix} \\ &= \begin{pmatrix} y_i^{(1)} - \hat{y}_i^{(1)} \\ y_i^{(2)} - \hat{y}_i^{(2)} \\ \vdots \\ y_i^{(C)} - \hat{y}_i^{(C)} \end{pmatrix} \end{aligned}$$

Trong đó:

- y_i là vector nhãn thực tế (one-hot encoding) của mẫu thứ i, với C phần tử tương ứng với C lớp.
- $\hat{y}_i$ là vector xác suất dự đoán của mô hình cho mẫu thứ i, cũng với C phần tử.
- e_i là vector sai số, với mỗi phần tử là hiệu của giá trị thực tế và giá trị dự đoán tương ứng.

Ví dụ, với $C = 3$ lớp và mẫu thứ n có nhãn thực tế $y_n = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$, nếu mô hình dự đoán $\hat{y}_i = \begin{pmatrix} 0.8 \\ 0.1 \\ 0.1 \end{pmatrix}$, thì vector sai số e_i sẽ là:

$$e_i = \begin{pmatrix} 1 - 0.8 \\ 0 - 0.1 \\ 0 - 0.1 \end{pmatrix} = \begin{pmatrix} 0.2 \\ -0.1 \\ -0.1 \end{pmatrix}$$

Cập nhật trọng số đến khi nào đủ số vòng lặp thì chúng ta sẽ cập nhật loss function cuối cùng.

Công thức cập nhật trọng số $\mathbf{W}$ theo từng mẫu dữ liệu $\mathbf{x}_n$ một cách tuần tự cho phép mô hình học liên tục từ dữ liệu mới mà không cần tính toán lại gradient trên toàn bộ tập huấn luyện cũ. Phương pháp này hiệu quả trong thực tế khi dữ liệu luôn được cập nhật liên tục, đặc biệt với học máy trực tuyến hoặc phân tán, giúp tiết kiệm tài nguyên tính toán và cập nhật mô hình kịp thời.

Bước 4: Dữ đoán với tập dữ liệu X_{test}

Với ma trận trọng số W đã tính được sau các bước cập nhật từ các bước trước chúng ta sẽ nhân ma trận trọng số $\mathbf{W}$ có kích thước $(d \times C)$ với dữ liệu kiểm tra X_{test} có kích thước $(d \times N_2)$.

Chúng ta sẽ có kết quả là một ma trận $\mathbf{Y}$ có kích thước $(C \times N_2)$, trong đó mỗi phần tử y_{ij} đại diện cho xác suất mà mẫu thứ j thuộc lớp thứ i.

$$\mathbf{Y} = \mathbf{W} \times X_{test} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1N_2} \\ y_{21} & y_{22} & \cdots & y_{2N_2} \\ \vdots & \vdots & \ddots & \vdots \\ y_{C1} & y_{C2} & \cdots & y_{CN_2} \end{bmatrix}$$

Để dự đoán lớp cho mỗi mẫu, chúng ta chọn lớp có xác suất lớn nhất trên mỗi cột (tương ứng với mỗi mẫu). Ví dụ, với mẫu thứ j, chúng ta sẽ chọn lớp k nếu y_{kj} là phần tử lớn nhất trong cột thứ j:

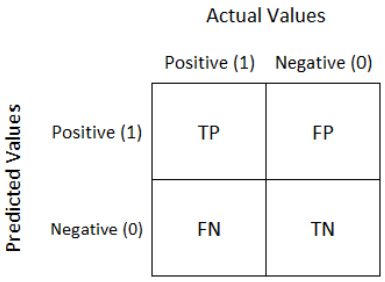
$$\text{Lớp dự đoán cho mẫu } j = \underset{i}{\operatorname{argmax}}\, y_{ij}$$

1.7 Độ đo đánh giá kết quả

1.7.1 Ma trận mâu thuẫn (Confused Matrix)

Ma trận mẫu thuẫn (Confusion Matrix) Là một phương pháp đánh giá kết quả của những bài toán phân loại với việc xem xét cả những chỉ số về độ chính xác và độ bao quát của các dự đoán cho từng lớp. Một confusion matrix gồm 4 chỉ số sau đối với mỗi lớp phân loại:

- TP (True Positive): Số lượng dự đoán dương tính thực sự.
- TN (True Negative): Số lương dự đoán âm tính thực sự.
- FP (False Positive - Type 1 Error): Số lượng các dự đoán dương tính giả.
- FN (False Negative - Type 2 Error): Số lượng các dự đoán âm tính giả.



1.7.2 Accuracy

Độ chính xác Accuracy đùng để tính toán tỉ lệ dự đoán đúng trên tổng số mẫu dự đoán là bao nhiêu. Chỉ số này càng cao tức là mô hình dự đoán càng đúng và càng tốt (đối với dữ liệu không quá mất cân bằng).

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

1.7.3 Precision

Precision cũng là một thang đo độ chính xác. Cụ thể trong tất cả các dự đoán Positive được đưa ra, bao nhiêu dự đoán là chính xác? Chỉ số này được tính theo công thức:

$$Precision = \frac{TP}{TP + FP}$$

1.7.4 Recall

Độ đo Recall (còn được gọi là sensitivity hoặc true positive rate) là một độ đo đánh giá khả năng của một mô hình hoặc hệ thống trong việc bắt capture tất cả các trường hợp thực sự thuộc một lớp cụ thể. Độ đo này quan tâm đến khả năng của mô hình trong việc không bỏ sót các trường hợp dương thực sự (true positive). Công thức tính sẽ là:

$$Recall = \frac{TP}{TP + FN}$$

1.8 Kết quả và thực nghiệm

Accuracy trên tập dữ liệu kiểm tra: **0.8904166666666666**
Precision trên tập dữ liệu kiểm tra: **0.8941137863463393**
Recall trên tập dữ liệu kiểm tra: **0.8900599977043162**

Ma trận nhầm lẫn trên tập kiểm tra:

	0	1	2	3	4	5	6	7	8	9
0	1089	0	43	1	3	21	9	0	6	3
1	0	1273	12	4	2	13	1	2	15	0
2	1	14	1093	4	11	12	9	9	18	3
3	2	9	65	973	1	89	3	9	51	17
4	5	2	13	0	1075	8	12	2	6	53
5	10	4	24	17	8	994	13	0	24	10
6	6	1	61	1	11	40	1055	0	2	0
7	6	11	27	11	26	6	0	1123	2	87
8	8	25	41	13	5	93	18	1	933	23
9	2	5	10	12	38	19	0	19	12	1077

Nhận xét:

- Độ chính xác (accuracy) trên tập kiểm tra đạt 89.04%, điều này cho thấy mô hình có khả năng nhận dạng chữ số viết tay tương đối tốt.
- Tuy nhiên, từ ma trận nhầm lẫn ta thấy mô hình gặp khó khăn trong việc phân biệt một số cặp số như 5 và 3, 9 và 4, 9 và 7, 8 và 3, ...