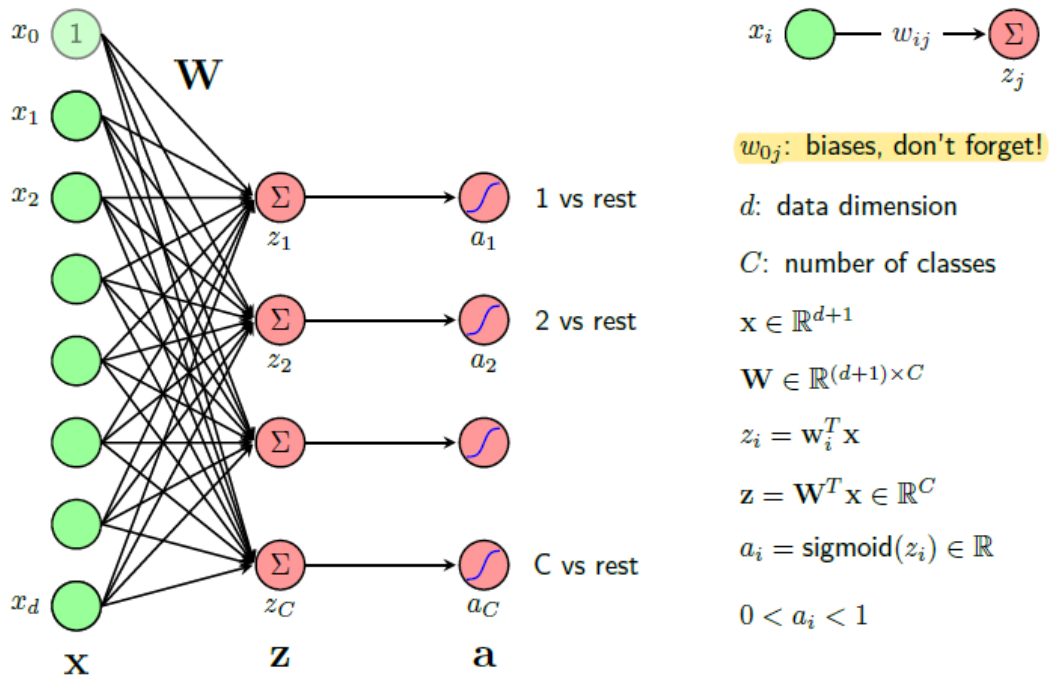


SOFTMAX REGRESSION

1. Giới thiệu



Hình 15.1: Phân lớp đa lớp với logistic regression và one-vs-rest.

Với bài toán phân lớp nhị phân sử dụng logistic regression, đầu ra của neural network là một số thực trong khoảng $(0, 1)$, đóng vai trò như là xác suất để đầu vào thuộc một trong hai lớp. Ý tưởng này cũng có thể mở rộng cho bài toán phân lớp đa lớp, ở đó có C node ở output layer và giá trị mỗi node đóng vai trò như xác suất để đầu vào rơi vào lớp tương ứng. Như vậy, các đầu ra này liên kết với nhau qua việc chúng đều là các số dương và có tổng bằng một. [Mô hình softmax regression thảo luận trong chương này đảm bảo tính chất đó.]

Notes : Bài toán Softmax regression có đầu ra là nhiều node, giá trị mỗi node như xác suất để đầu vào rơi vào lớp tương ứng.

Hỏi : ví dụ Phân biệt Chó-Mèo-Gà dựa vào đặc điểm Cổ-Mồm-đuôi thì mình hiểu thế nào về các đặc điểm trên trong bài toán ?

Nhắc lại kỹ thuật *one-vs-rest* được trình bày trong chương trước được biểu diễn dưới dạng neuron network như trong Hình 15.1. Output layer màu đỏ có thể phân tách thành hai *sublayer* và mỗi thành phần của sublayer thứ hai a_i chỉ phụ thuộc vào thành phần tương ứng ở sublayer thứ nhất z_i thông qua hàm sigmoid $a_i = \sigma(z_i)$. Các giá trị đầu ra a_i đều là các số dương nhưng vì không có ràng buộc giữa chúng, tổng của chúng có thể là một số dương bất kỳ.

Notes:

Chú ý rằng các mô hình linear regression, PLA, và logistic regression chỉ có một node ở output layer. Trong các trường hợp đó, tham số mô hình chỉ là một vector w . Trong trường hợp output layer có nhiều hơn một node, tham số mô hình sẽ là tập hợp tham số w_i ứng với từng node. Lúc này, ta có một *ma trận trọng số* (*weight matrix*) $W = [w_1, w_2, \dots, w_C]$, mỗi cột ứng với một node ở output layer.

Notes: 1 vecto trọng số w tương ứng cho một (input, output).

$Y \rightarrow Z = w_0 + w_1.x_1$: mô hình Linear Regression

Nhiều output : nhiều vecto w xếp thành cột tương ứng thành một ma trận trọng số.

Hỏi : Lập bảng data như vậy có đúng với bài toán ?

	x1	x2	x3	xN
$y_1 \rightarrow z_1$:	w 11	w 21	w 31	w N1
$y_2 \rightarrow z_2$:	w 12	w 22	w 32	w N2
$y_3 \rightarrow z_3$:	w 13	w 23	w 33	w N3
$y_C \rightarrow z_C$:	w 1C	w 2C	w 3C	w NC
Sum(a) =1	a 1	a 2	a 3	a C

Từ bảng ta có : $Y = w.X$ tương ứng khai triển ra theo hàng.

Giải thích : x : giá trị input

y : giá trị output

w : trọng số

$I = 1 \rightarrow N$: số chiều dữ liệu (tập training data có N hàng)

$J = 1 \rightarrow C$: số lớp dữ liệu (tập training data có C cột)

Mỗi vecto x nhân với lại w tạo ra z (hoặc y_{hat}). Từng vecto z nhân với nhau tạo ra a là giá trị xác suất của output.

2. Softmax function

15.2.1 Công thức của Softmax function

Chúng ta cần một mô hình xác suất sao cho với mỗi input \mathbf{x} , a_i thể hiện xác suất để input đó rơi vào lớp thứ i . Vậy điều kiện cần là các a_i phải dương và tổng của chúng bằng một. Ngoài ra, ta sẽ thêm một điều kiện cũng rất tự nhiên nữa, đó là giá trị $z_i = \mathbf{w}_i^T \mathbf{x}$ càng lớn thì xác suất dữ liệu rơi vào lớp thứ i càng cao. Điều kiện cuối này chỉ ra rằng ta cần một quan hệ đồng biến ở đây.

Chú ý rằng z_i có thể nhận giá trị cả âm và dương vì nó là một tổ hợp tuyến tính của các thành phần của vector đặc trưng \mathbf{x} . Một hàm số khả vi đơn giản có thể chắc chắn biến z_i thành một giá trị dương, và hơn nữa, đồng biến, là hàm $\exp(z_i) = e^{z_i}$. Điều kiện khả vi để thuận lợi cho việc sử dụng đạo hàm cho việc tối ưu. Điều kiện cuối cùng, tổng các a_i bằng một có thể được đảm bảo nếu

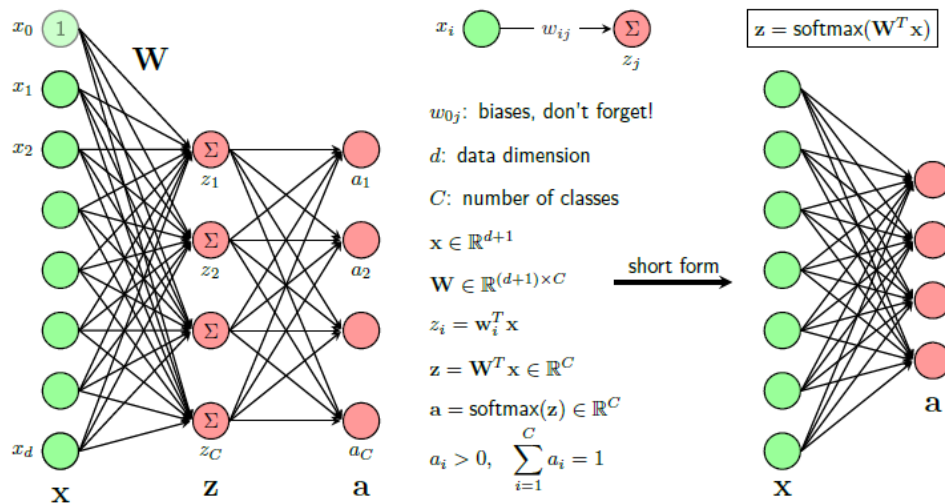
$$a_i = \frac{\exp(z_i)}{\sum_{j=1}^C \exp(z_j)}, \quad \forall i = 1, 2, \dots, C \quad (15.1)$$

Mối quan hệ này, với mỗi a_i phụ thuộc vào tất cả các z_i , thỏa mãn tất cả các điều kiện đã xét: dương, tổng bằng một, giữ được thứ tự của z_i . Hàm số này được gọi là *softmax function*. Lúc này, ta có thể coi rằng

$$p(y_k = i | \mathbf{x}_k; \mathbf{W}) = a_i \quad (15.2)$$

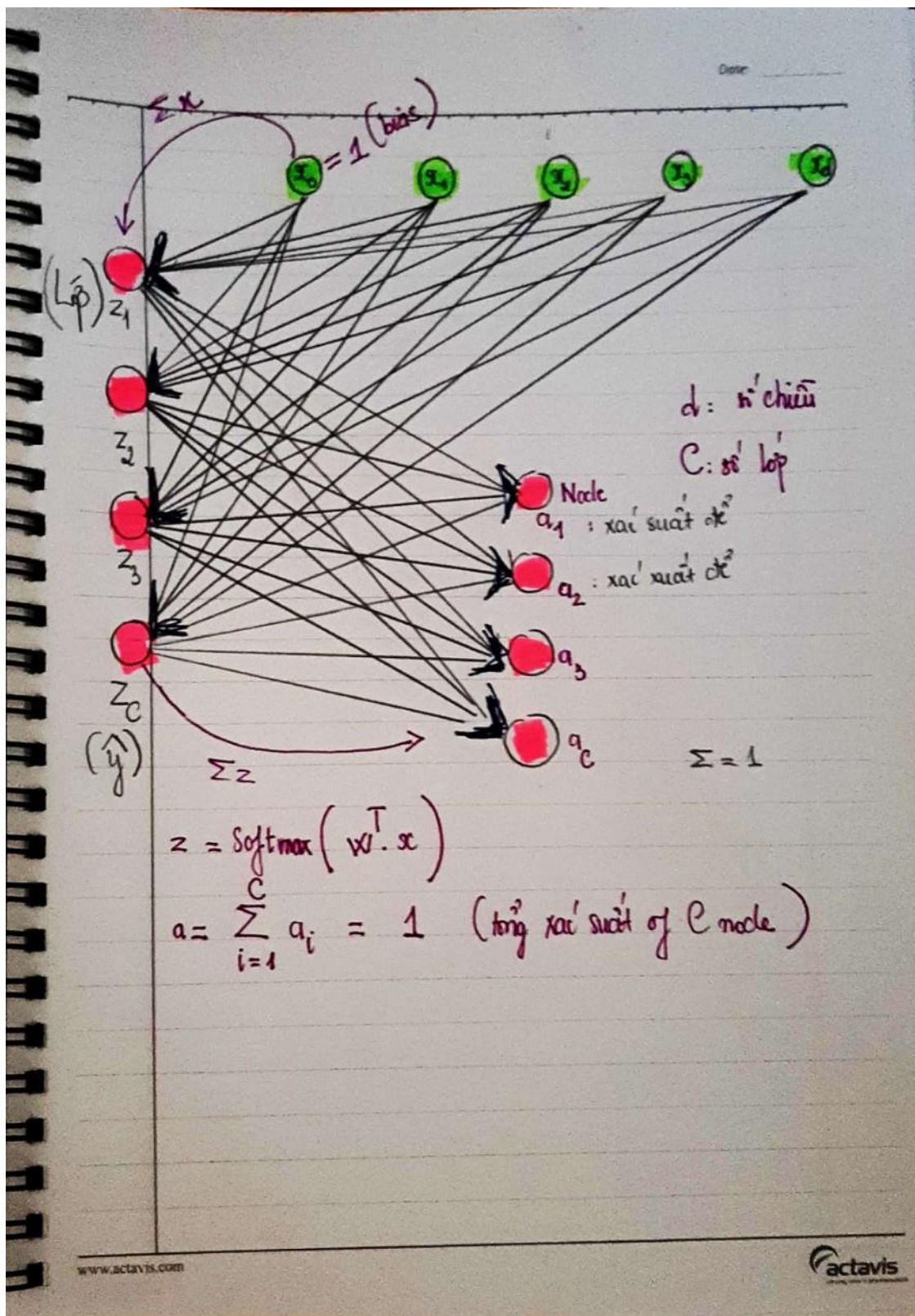
Trong đó, $p(y = i | \mathbf{x}; \mathbf{W})$ được hiểu là xác suất để một điểm dữ liệu \mathbf{x} rơi vào lớp thứ i nếu biết tham số mô hình là ma trận trọng số \mathbf{W} . Hình 15.2 thể hiện mô hình softmax regression

dưới dạng neural network. Sự khác nhau giữa mô hình này và mô hình one-vs-rest nằm ở chỗ nó có các liên kết giữa mọi node của hai sublayer màu đỏ.

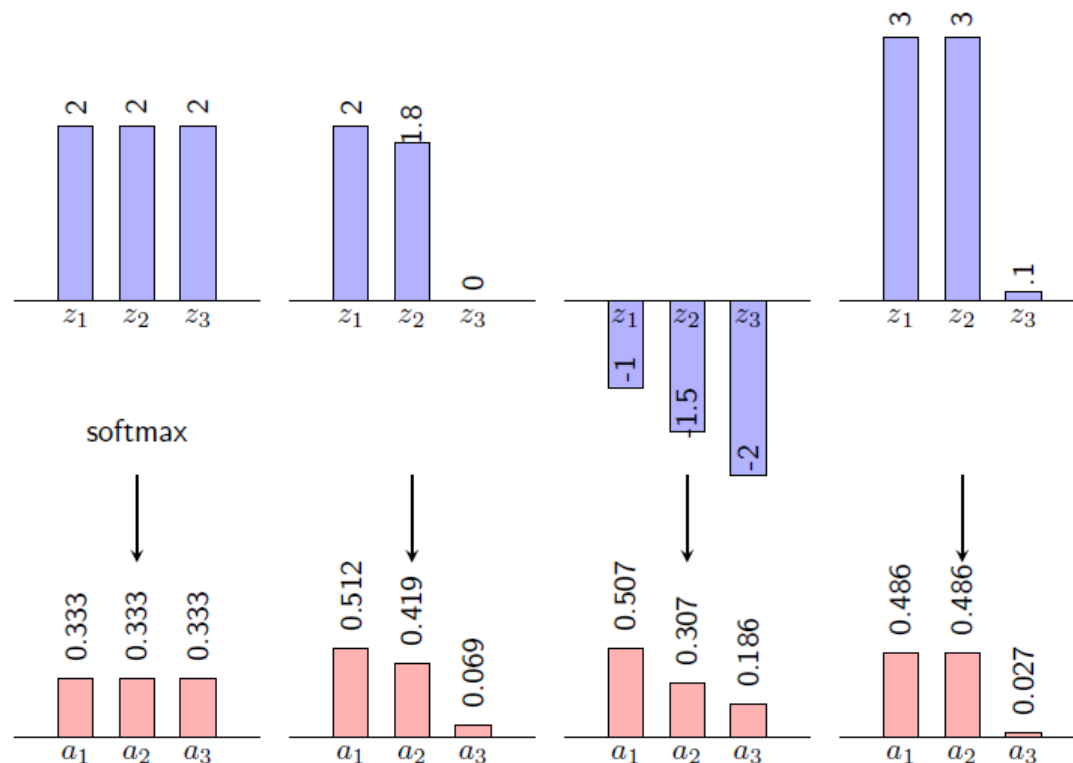


Hình 15.2: Mô hình softmax regression dưới dạng neural network.

Notes: Vẽ lại mạng neuron theo cách hiểu



Một vài ví dụ :



Hình 15.3: Một số ví dụ về đầu vào và đầu ra của hàm softmax.

Có một vài quan sát như sau:

- Cột 1: Nếu các z_i bằng nhau (bằng 2 hoặc một số bất kỳ), thì các a_i cũng bằng nhau và bằng $1/3$.
- Cột 2: Nếu giá trị lớn nhất trong các z_i là z_1 vẫn bằng 2, thì mặc dù xác suất tương ứng a_1 vẫn là lớn nhất, nó đã thay đổi lên hơn 0.5. Sự chênh lệch ở đầu ra là đáng kể, nhưng thứ tự tương ứng không thay đổi.
- Cột 3: Khi các giá trị z_i là âm thì các giá trị a_i vẫn là dương và thứ tự vẫn được đảm bảo.
- Cột 4: Nếu $z_1 = z_2$, thì $a_1 = a_2$.

Phiên bản ổn định hơn của softmax function :

Khi một trong các z_i quá lớn, việc tính toán $\exp(z_i)$ có thể gây ra hiện tượng *tràn số* (*overflow*), ảnh hưởng lớn tới kết quả của hàm softmax. Có một cách khắc phục hiện tượng này bằng cách dựa trên quan sát

$$\frac{\exp(z_i)}{\sum_{j=1}^C \exp(z_j)} = \frac{\exp(-c) \exp(z_i)}{\exp(-c) \sum_{j=1}^C \exp(z_j)} = \frac{\exp(z_i - c)}{\sum_{j=1}^C \exp(z_j - c)} \quad (15.3)$$

với c là một hằng số bất kỳ. Vậy một phương pháp đơn giản giúp khắc phục hiện tượng overflow là trừ tất cả các z_i đi một giá trị đủ lớn. Trong thực nghiệm, giá trị đủ lớn này thường được chọn là $c = \max_i z_i$. Vậy chúng ta có thể sửa đoạn code cho hàm softmax phía

Notes:

3. Hàm mất mát và phương pháp tối ưu

Xây dựng hàm mất mát

Trong trường hợp có C lớp dữ liệu, *mất mát* giữa đầu ra dự đoán và đầu ra thực sự của một điểm dữ liệu x_i với label (one-hot) y_i được tính bởi

$$J_i(\mathbf{W}) \triangleq J(\mathbf{W}; x_i, y_i) = - \sum_{j=1}^C y_{ji} \log(a_{ji}) \quad (15.5)$$

với y_{ji} và a_{ji} lần lượt là phần tử thứ j của vector xác suất y_i và a_i . Nhắc lại rằng đầu ra a_i phụ thuộc vào đầu vào x_i và ma trận trọng số \mathbf{W} . Tới đây, nếu để ý rằng chỉ có đúng một j sao cho $y_{ji} = 1, \forall i$, biểu thức (15.5) chỉ còn lại một số hạng tương ứng với giá trị j đó. Để tránh việc sử dụng quá nhiều ký hiệu, chúng ta giả sử rằng y_i là nhãn của điểm dữ liệu x_i (các nhãn là các số tự nhiên từ 1 tới C), khi đó j chính bằng y_i . Sau khi có ký hiệu này, ta có thể viết lại

$$J_i(\mathbf{W}) = -\log(a_{y_i, i}) \quad (15.6)$$

với $a_{y_i, i}$ là phần tử thứ y_i của vector a_i .

Kết hợp tất cả các cặp dữ liệu $x_i, y_i, i = 1, 2, \dots, N$, hàm mất mát cho softmax regression được xác định bởi

$$J(\mathbf{W}; \mathbf{X}, \mathbf{Y}) = -\frac{1}{N} \sum_{i=1}^N \log(a_{y_i, i}) \quad (15.7)$$

Ở đây, ma trận trọng số \mathbf{W} là biến cần tối ưu. Mặc dù hàm mất mát này trông phức tạp, đạo hàm của nó rất gọn. Ta cũng có thể thêm weight decay để tránh overfitting bằng cách cộng thêm một đại lượng tỉ lệ với $\|\mathbf{W}\|_F^2$.

$$\bar{J}(\mathbf{W}; \mathbf{X}, \mathbf{Y}) = -\frac{1}{N} \left(\sum_{i=1}^N \log(a_{y_i, i}) + \frac{\lambda}{2} \|\mathbf{W}\|_F^2 \right) \quad (15.8)$$

Tiếp theo ta sử dụng công thức

$$\nabla_{\mathbf{W}} J_i(\mathbf{W}) = [\nabla_{\mathbf{w}_1} J_i(\mathbf{W}), \nabla_{\mathbf{w}_2} J_i(\mathbf{W}), \dots, \nabla_{\mathbf{w}_C} J_i(\mathbf{W})] \quad (15.10)$$

Trong đó, gradient theo từng cột của \mathbf{w}_j có thể tính được dựa theo (15.9) và quy tắc chuỗi tính gradient

$$\begin{aligned} \nabla_{\mathbf{w}_j} J_i(\mathbf{W}) &= -y_{ji} \mathbf{x}_i + \frac{\exp(\mathbf{w}_j^T \mathbf{x}_i)}{\sum_{k=1}^C \exp(\mathbf{w}_k^T \mathbf{x}_i)} \mathbf{x}_i \\ &= -y_{ji} \mathbf{x}_i + a_{ji} \mathbf{x}_i = \mathbf{x}_i (a_{ji} - y_{ji}) \\ &= e_{ji} \mathbf{x}_i \quad (\text{với } e_{ji} = a_{ji} - y_{ji}) \end{aligned} \quad (15.11)$$

Giá trị $e_{ji} = a_{ji} - y_{ji}$ chính là sự sai khác giữa đầu ra dự đoán và đầu ra thực sự tại thành phần thứ j . Kết hợp (15.10) và (15.11) với $\mathbf{e}_i = \mathbf{a}_i - \mathbf{y}_i$, ta có

$$\nabla_{\mathbf{W}} J_i(\mathbf{W}) = \mathbf{x}_i [e_{1i}, e_{2i}, \dots, e_{Ci}] = \mathbf{x}_i \mathbf{e}_i^T \quad (15.12)$$

$$\Rightarrow \nabla_{\mathbf{W}} J(\mathbf{W}) = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i \mathbf{e}_i^T = \frac{1}{N} \mathbf{X} \mathbf{E}^T \quad (15.13)$$

với $\mathbf{E} = \mathbf{A} - \mathbf{Y}$. Công thức tính đạo hàm đơn giản thế này giúp cho cả batch gradient descent, và mini-batch gradient descent đều có thể dễ dàng được áp dụng. Trong trường hợp mini-batch gradient, giả sử kích thước batch là k , ký hiệu $\mathbf{X}_b \in \mathbb{R}^{d \times k}$, $\mathbf{Y}_b \in \{0, 1\}^{C \times k}$, $\mathbf{A}_b \in \mathbb{R}^{C \times k}$ là dữ liệu ứng với một batch, công thức cập nhật cho một batch sẽ là

$$\mathbf{W} \leftarrow \mathbf{W} - \frac{\eta}{N_b} \mathbf{X}_b \mathbf{E}_b^T \quad (15.14)$$

Chứng minh, khai triển các công thức :

Ta có: $J_i(\mathbf{x}_i, \mathbf{y}_i)$

$$J_i(\mathbf{w}) = - \sum_{j=1}^C y_{ji} \log(a_{ji}) \quad \left(\begin{array}{l} \text{Khai triển ra} \\ C=3 \end{array} \right)$$

$$= - y_{1i} \log(a_{1i}) - y_{2i} \log(a_{2i}) - y_{3i} \log(a_{3i}) \quad (i=1-N)$$

$$(*) \quad \log a_{ji} = \log \left(\frac{\exp(\mathbf{w}_j^T \mathbf{x}_i)}{\sum_{k=1}^C \exp(\mathbf{w}_k^T \mathbf{x}_i)} \right) = \mathbf{w}_j^T \mathbf{x}_i - \log \sum_{k=1}^C \exp(\mathbf{w}_k^T \mathbf{x}_i)$$

$\left[\log \frac{a}{b} = \log a - \log b \right]$

www.actavis.com

actavis
creating value in pharma

www.actavis.com

Date: / /

Suy ra:

$$J_i(w) = - \sum_{j=1}^C \left(y_{ji} w_j^T x_i - y_{ji} \log \left(\sum_{k=1}^C \exp(w_k^T x_i) \right) \right)$$

$$= - \sum_{j=1}^C y_{ji} w_j^T x_i + \log \left(\sum_{k=1}^C \exp(w_k^T x_i) \right)$$

⊛ y_{ji} biến mất vì $\sum_{j=1}^C y_{ji} = 1$ (tổng xác suất)

$$\nabla_w J_i(w) = \left[\nabla_{w_1} J_i(w) \quad \nabla_{w_2} J_i(w) \quad \nabla_{w_3} J_i(w) \right]$$

↓
Gradient cho w cột j → xếp vào ⇒ Ma trận


⊛ $\nabla_{w_j} J_i(w)$

$$= -y_{ji} x_i + \frac{\exp(w_j^T x_i)}{\sum_{k=1}^C \exp(w_k^T x_i)} x_i$$

$$= -y_{ji} x_i + a_{ji} x_i = (a_{ji} - y_{ji}) x_i = e_{ji} x_i$$

$e_{ji} = a_{ji} - y_{ji}$: sai số (y & \hat{y})

www.actavis.com

 creating value in pharmaceuticals

$$\left[\log(e^{w_1^T x_i} + e^{w_2^T x_i} + e^{w_3^T x_i}) \right]'$$

$$= \frac{(e^{w_1^T x_i} + e^{w_2^T x_i} + e^{w_3^T x_i})'}{\sum_{k=1}^3 e^{w_k^T x_i}} = x_i \cdot \frac{e^{w_j^T x_i}}{\sum_{k=1}^3 e^{w_k^T x_i}}$$

$$\nabla_{w_j} J_i(w) = e_j x_i \quad \text{bung j ra} \quad = \boxed{A \cdot B = B \cdot A^T}$$

$$\Rightarrow \nabla_w J_i(w) = x_i [e_{1i} \ e_{2i} \ e_{3i} \ e_{4i}]$$

$$= x_i \cdot e_i^T \quad \text{bung i ra}$$

$$\Rightarrow \nabla_w J(w) = \frac{1}{N} \sum_{i=1}^N x_i e_i^T = \frac{1}{N} X E^T \quad (N=4)$$

$$= \frac{1}{4} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} [A - Y] = (4 \times 1) \otimes (1 \times 1):$$

$$= \boxed{4 \times 1} \quad \boxed{\text{Nhos chuc}}$$

$$= \frac{1}{4} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \begin{bmatrix} a_{11} - y_{11} & a_{12} - y_{12} & a_{13} - y_{13} & a_{14} - y_{14} \end{bmatrix}$$

4. Thảo luận

15.5.1 Logistic regression là một trường hợp đặc biệt của softmax regression

Khi $C = 2$, softmax regression và logistic regression là giống nhau. Thật vậy, với $C = 2$, đầu ra của hàm softmax cho một đầu vào \mathbf{x} là

$$a_1 = \frac{\exp(\mathbf{w}_1^T \mathbf{x})}{\exp(\mathbf{w}_1^T \mathbf{x}) + \exp(\mathbf{w}_2^T \mathbf{x})} = \frac{1}{1 + \exp((\mathbf{w}_2 - \mathbf{w}_1)^T \mathbf{x})}; \quad a_2 = 1 - a_1 \quad (15.15)$$

Từ đây ta thấy rằng, a_1 có dạng là một hàm sigmoid với vector hệ số $\mathbf{w} = -(\mathbf{w}_2 - \mathbf{w}_1)$. Khi $C = 2$, bạn đọc cũng có thể thấy rằng hàm mất mát của logistic regression và softmax regression là như nhau. Hơn nữa, mặc dù có hai outputs, softmax regression có thể biểu diễn bởi một output vì tổng của hai outputs luôn luôn bằng 1.

Softmax regression còn có các tên gọi khác là multinomial logistic regression, hay maximum entropy classifier. Giống như logistic regression, softmax regression được sử dụng trong các bài toán classification. Các tên gọi này được giữ lại vì vấn đề lịch sử.

Notes :

15.5.2 Ranh giới tạo bởi softmax regression là một mặt tuyến tính

Thật vậy, dựa vào hàm softmax thì một điểm dữ liệu \mathbf{x} được dự đoán là rơi vào class j nếu $a_j \geq a_k, \forall k \neq j$. Bạn đọc có thể chứng minh được rằng

$$a_j \geq a_k \Leftrightarrow z_j \geq z_k \Leftrightarrow \mathbf{w}_j^T \mathbf{x} \geq \mathbf{w}_k^T \mathbf{x} \Leftrightarrow (\mathbf{w}_j - \mathbf{w}_k)^T \mathbf{x} \geq 0 \quad (15.16)$$

Như vậy, một điểm thuộc lớp thứ j nếu và chỉ nếu $(\mathbf{w}_j - \mathbf{w}_k)^T \mathbf{x} \geq 0, \forall k \neq j$. Như vậy, lãnh thổ của mỗi lớp dữ liệu là giao của các nửa không gian. Nói cách khác, đường ranh giới giữa các lớp là các mặt tuyến tính.

15.5.3 Softmax Regression là một trong hai classifiers phổ biến nhất

Softmax regression cùng với multi-class support vector machine (Chương 29) là hai bộ phân lớp phổ biến nhất được dùng hiện nay. Softmax regression đặc biệt được sử dụng nhiều trong các deep neural network với rất nhiều hidden layer. Những layer phía trước có thể được coi như một bộ tạo vector đặc trưng, layer cuối cùng thường là một softmax regression.

Notes :