MACHINE LEARNING

Cao Văn Chung
cvanchung@hus.edu.vn
Informatics Dept., MIM, HUS, VNU Hanoi

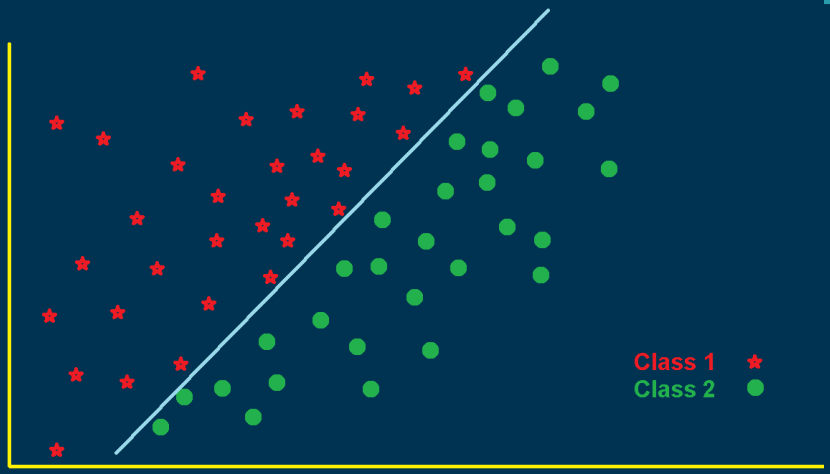
Multinomial Logistic Regression (Entropy cực đại)

Binary vs. Multinomial Classifiers
Hồi quy Logistic nhiều lớp
Xây dựng hàm xác suất phân lớp
Softmax
Xây dựng hàm tổn thất
Đạo hàm hàm tổn thất
GD Method vs. Softmax Regression
Examples

Binary Classifiers (B.C)

- Hồi quy logistic dùng trong phân loại nhị phân (Binary Classifiers), tức đầu ra chỉ có 02 lớp: $y \in \{0, 1\}$.
- Một số mô hình khác cũng thuộc nhóm phân loại nhị phân: Thuật toán Perceptron, Support Vector Machine...
- Mô hình hồi quy logistic cũng như các mô hình phân loại nhị phân nói trên, yêu cầu dữ liệu hầu như tách được tuyến tính - *nearly linearly separable* (đặc điểm chung của các phương pháp phân loại tuyến tính).

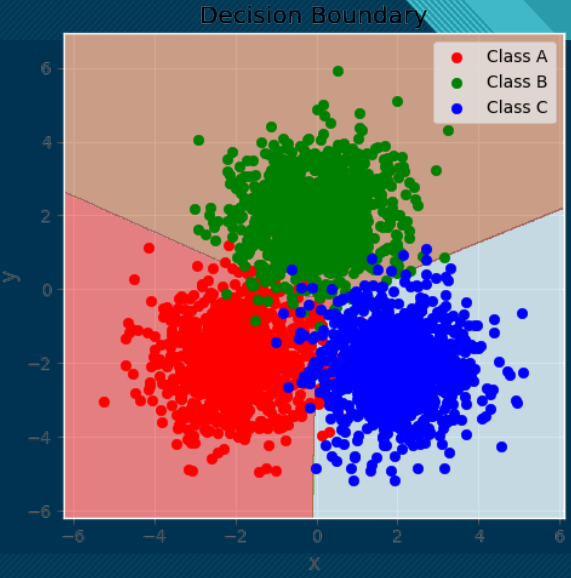
Binary Classifiers (B.C)



Binary Classifiers to Multi-class classification

- Thực tế nhiều trường hợp đầu ra thuộc $C > 2$ lớp $C = \{1, 2, \dots C\}$ (*Multi-Class* hay *Multinomial classification* - MC).
- Ví dụ: Xác định chữ số viết tay, đầu ra sẽ có 10 lớp: $\{0, 1, 2, \dots 9\}$.
- Ta muốn phát triển các thuật toán phân loại nhị phân để áp dụng cho bài toán này.
- Ta vẫn giả thiết tập dữ liệu hầu như tách được tuyến tính (nearly linearly separable).

Binary Classifiers to Multi-class classification



Binary Classifiers to Multi-class classification

- Một cách tự nhiên, sẽ có các hướng tiếp cận:
  - **One-vs-One**: Áp dụng BC theo từng cặp, ví dụ với đầu vào x, dùng BC để kiểm tra và phân loại lần lượt với 2 lớp $\{0, 1\}$, $\{0, 2\}$, ... Với C lớp đầu ra, ta cần $\frac{C(C-1)}{2}$ cặp! Hơn nữa dữ liệu x có thể bị đưa đi kiểm tra ở 2 phân lớp hoàn toàn sai. Ví dụ ảnh của số 2 nhưng lại thử ở mô hình 2 lớp $\{5, 6\}$.
  - **Hierarchical** (phân tầng): Trước hết chia dữ liệu thành 2 lớp tương đối giống nhau, phân loại dữ liệu vào 1 trong 2 lớp đó. Sau đó với mỗi lớp lại chia tiếp thành 2 lớp con... Ví dụ với phân loại chữ số, đầu tiên ta cố gắng xác định ảnh x thuộc nhóm nào trong các nhóm: $\{1, 4, 7\}$ và $\{0, 2, 3, 5, 6, 8, 9\}$. Sau đó ví dụ x rơi vào nhóm 2, ta lại áp dụng với 2 lớp con: $\{0, 8, 9\}$ và $\{2, 3, 5, 6\}$... Dễ thấy, nếu phân loại lớp gốc sai thì kết quả chắc chắn sai!.

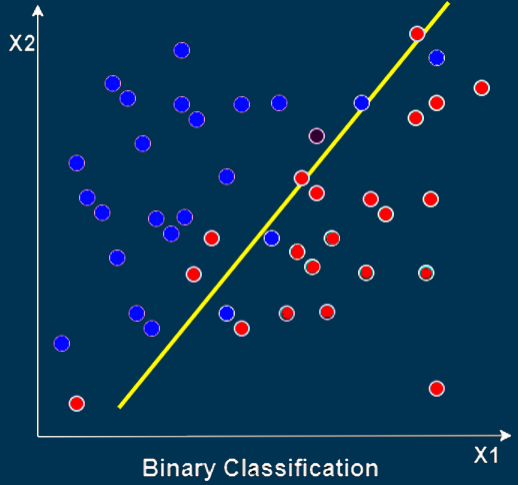
Binary Classifiers to Multi-class classification

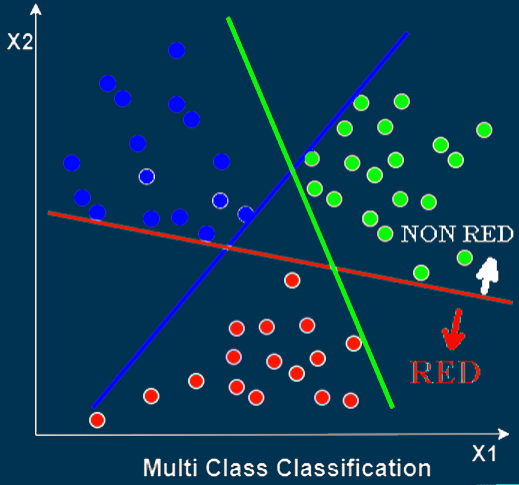
- Hướng tiếp cận:
  - **Binary coding**: giảm số lớp bằng mã hóa output của mỗi class bằng một số nhị phân. Ví dụ, nếu có 4 classes thì sẽ được mã hóa là 00, 01, 10 và 11. Số cặp lớp phải thực hiện chỉ là $m = [\log_2(C) + 0.5]$. Hạn chế:

    (1) Nếu một bít bị phân loại sai $\Rightarrow$ Toàn dữ liệu bị phân loại sai!

    (2) Nếu số class C không phải lũy thừa của 2 $\Rightarrow$ Tồn tại những mã nhị phân không ứng với class nào.
  - **One-vs-Rest**: Với C lớp, ta xây dựng C classifiers, mỗi classifier ứng với một class: Classifier thứ nhất phân biệt class 1 với not class 1 (theo xác suất như LR); Classifier thứ hai phân biệt class 2 với not class 2, và tiếp tục... Kết quả cuối cùng dựa vào xác định class mà dữ liệu rơi vào với xác suất cao nhất...

One-vs-Rest

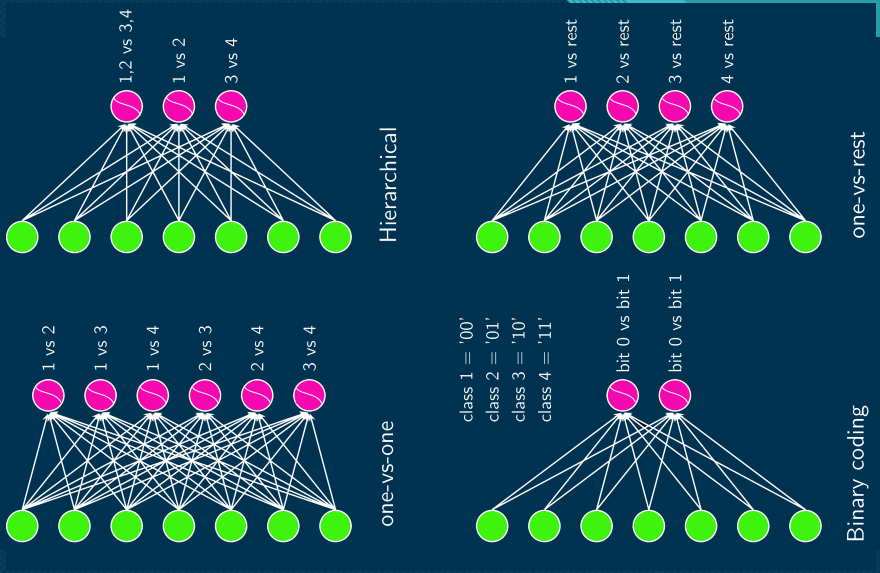

X2
X1
Binary Classification


X2
NON RED
RED
X1
Multi Class Classification

One-vs-Rest

- One-vs-Rest còn được gọi là one-hot coding, ove-vs-all, one-against-rest...
  - Nếu mã hóa các lớp đầu ra bằng C bit, lớp thứ k sẽ ứng với bit thứ k bật thành 1.
  - Ví dụ: Có 4 lớp đầu ra $\{1, 2, 3, 4\}$, ta sẽ mã hóa chúng bằng 1000, 0100, 0010 và 0001. Vì lý do này phương pháp còn được gọi là one-hot coding, tức mã hóa 01 bit vị trí chủ thành 1.
  - Chú ý: Giả sử xác suất để một điểm dữ liệu được dự đoán thuộc vào class $1, 2, \dots C$ lần lượt là $p_1, p_2, \dots p_C$, thì tổng $\sum_{i=1}^{C} p_i$ có thể khác 1! (tự giải thích).
- Chú ý: So sánh vector one-hot coding với vector xác suất biến quan sát rơi vào class $1 \leq c \leq C$ cũng sẽ dẫn đến Loss Function. Lưu ý ở đây ta so sánh hai phân phối xác suất, ví dụ với $C = 4$, biến x có dạng one-hot coding là $\{0, 1, 0, 0\}$ và xác suất x rơi vào các lớp tương ứng là $\{0.1, 0.6, 0.2, 0.1\}$. Vì vậy cần dùng Coss-Entropy để so sánh.

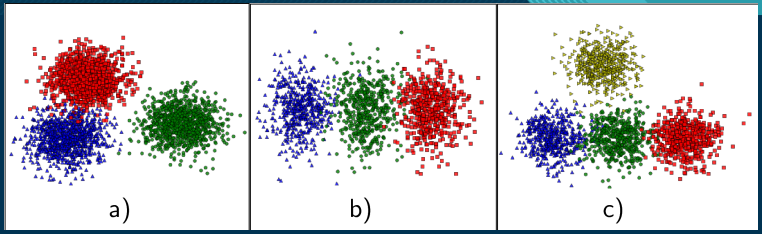
So sánh các hướng tiếp cận



Kết hợp các hướng tiếp cận

- Các phương pháp binary classifiers nhắc đến ở trên yêu cầu dữ liệu là linearly separable hoặc nearly linearly separable.
- Ta cũng có thể mở rộng định nghĩa này cho các bài toán multi-class: Nếu hai class bất kỳ là linearly separable thì ta coi dữ liệu đó là linearly separable, tương tự với nearly linearly separable.
- Ta sẽ áp dụng các phương pháp trong phần trên để có thể áp dụng phân loại nhị phân cho bài toán phân loại nhiều lớp.
- Tuy nhiên, có những loại dữ liệu linearly separable không phù hợp với một số phương pháp trên đây, hoặc yêu cầu phải kết hợp nhiều phương pháp mới thực hiện được.

Kết hợp các hướng tiếp cận



- Hình a): cả 4 phương pháp đều có thể áp dụng được.
- Hình b): one-vs-rest không phù hợp vì class green và phần còn lại (hợp của blue và red) không linearly separable.
- Hình c): Tương tự b), chỉ có thể dùng one-vs-one hoặc hierarchical. Ngoài ra có thể dùng one-vs-rest với tập red (chia thành red và non-red), tương tự blue, yellow. Phần còn lại dùng one-vs-one hoặc hierarchical.

Hồi quy Logistic cho phân loại nhiều lớp

- Khi bài toán phân loại có nhiều lớp, ta có thể mở rộng mô hình hồi quy logistic nhị phân ở trên cho trường hợp đa lớp.
- Mô hình hồi quy logistic đa lớp còn được gọi là mô hình entropy cực đại (maximum entropy–maxent), một dạng của mô hình log–tuyến tính (tức dạng phân phối xác suất sau khi lấy logarith tự nhiên thì được hàm tuyến tính).
- Mô hình Entropy cực đại được phát hiện trong những lĩnh vực khác nhau, dưới nhiều tên gọi khác nhau:
  - Trong lí thuyết xác suất: mô hình entropy cực đại; mô hình log–tuyến tính; trường ngẫu nhiên Markov (Markov random fields) và họ hàm mũ.
  - Trong thống kê toán học: Hồi quy logistic.
  - Trong cơ học thống kê và vật lí: phân phối Gibbs, phân phối Boltzmann.
  - Trong các mạng neural: Máy Boltzmann; hàm kích hoạt softmax.

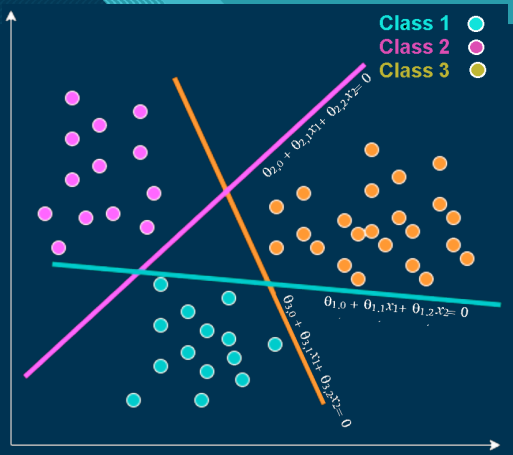
Hồi quy Logistic cho phân loại nhiều lớp

- Giả sử số lớp là C - các lớp là $\{1, 2, \dots C\}$.
- Các biến quan sát x thuộc không gian d chiều: $x \in X \subset \mathbb{R}^d$.
- Nhắc lại, trường hợp chỉ có 2 lớp, vector tham số θ có $d + 1$ chiều
  - Mô hình hồi quy logistic phân lớp dựa vào siêu phẳng tách $\theta^T x + \theta_0$.
  - Thành phần tham số $\theta_1, \dots \theta_d$ ứng với x;
  - Thành phần θ_0 - tham số tự do: Nếu không có θ_0 siêu phẳng $\theta^T x$ bắt buộc phải đi qua gốc tọa độ.
  - Thành phần θ_0 còn được gọi là bias.

Hồi quy Logistic cho phân loại nhiều lớp

- Giả sử ta nâng cấp Hồi quy Logistic để áp dụng cho phân loại nhiều lớp theo cách tiếp cận one-vs-rest:
  - Cần xây dựng C bộ phân loại, do đó
  - Cần C vector tham số $\theta_1, \ldots, \theta_C$. Vậy tham số tạo thành ma trận có kích thước $(d+1) \times C$

$$\begin{pmatrix} \theta_{0,1} & \theta_{0,2} & \ldots & \theta_{0,C} \\ \theta_{1,1} & \theta_{1,2} & \ldots & \theta_{1,C} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{d,1} & \theta_{d,2} & \ldots & \theta_{d,C} \\ \theta_{(d+1),1} & \theta_{(d+1),2} & \ldots & \theta_{(d+1),C} \end{pmatrix}$$



Hình: Lý do cần C vector tham số θ_c cho C lớp

Xây dựng hàm xác suất phân lớp

- Trước khi xây dựng hàm hợp lý cực đại - tức phân phối xác suất để một điểm dữ liệu x rơi vào lớp y - chúng ta sẽ xét lại cách xây dựng hàm này cho hồi quy Logistic:
  - Chú ý ta cần siêu phẳng $\theta^T x + \theta_0 = 0$ tách tập dữ liệu thành 2 phần chứ không phải để xấp xỉ tập dữ liệu như hồi quy tuyến tính.
  - Căn cứ giá trị của tổ hợp tuyến tính $z = \theta^T x + \theta_0$, đánh giá xác suất để x thuộc lớp $y = 1$ hay lớp $y = 0 \Rightarrow$ cần ánh xạ biến z thành xác suất $P(y|\theta; x)$, tức là cần tìm: $g : z = \theta^T x \mapsto P(y|\theta; x)$
  - Do $P(y|\theta; x) \in (0, 1]$ và $P(y = 1|\theta; x) + P(y = 0|\theta; x) = 1$, vậy ta cần $g(z) \in (0, 1]$, và đạo hàm đủ trơn (để giải bài toán tối ưu).
  - Trong hồi quy Logistic, ta chọn $g(z) = [1 + \exp(-z)]^{-1}$. Do chỉ có 2 lớp $\{0, 1\}$ nên điều kiện tổng xác suất bằng 1 luôn đảm bảo.

Xây dựng hàm xác suất phân lớp

- Nếu áp dụng trực tiếp sigmoid(z) theo kiểu one-vs-rest cho trường hợp C lớp, ta không có mối liên hệ giữa các lớp để thể hiện xác suất x không thuộc lớp m với điều kiện không thuộc lớp n: $P(y \neq m | y \neq n; x; \theta)$.
- Do đó $\sum_{k=1}^{C} P(y = k|x; \theta_k)$ có thể khác 1. Để tránh điều này, ta cần tìm hàm $g(z) : \mathbb{R} \rightarrow (0, 1]$ như sau
  - Giá trị của $g(z)$ phụ thuộc vào một tập C phần tử khác nhau từng đôi một $\{z_1, \ldots, z_C\}$. Xét trong tập $g(z)$ đơn trị $z_i = z_j \Leftrightarrow g(z_i) = g(z_j)$ và

$$\sum_{c=1}^{C} g(z_c) = 1.$$

  - $g(z) : \mathbb{R} \rightarrow (0, 1)$, $g(z_1) > g(z_2)$ nếu $z_1 > z_2$.
  - $g(z)$ đủ trơn và đạo hàm tính được dễ dàng.

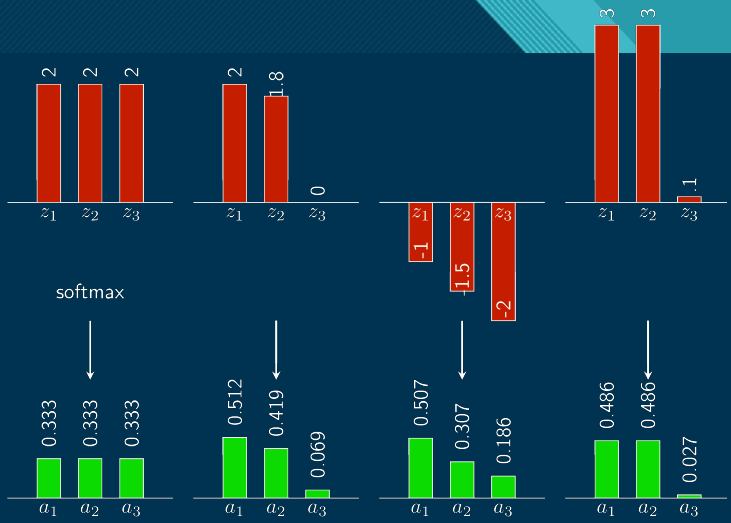
Hàm softmax

- Người ta sử dụng hàm **Softmax** $g(z) : \mathbb{R} \to (0,1)$, xây dựng trên tập $\{z_1, \ldots, z_C\}$, xác định như sau

$$g(z_k) = \frac{\exp(z_k)}{\sum_{c=1}^{C} \exp(z_c).}$$

- Dễ thấy $g(z) \in (0,1)$ với mọi $z \in \mathbb{R}$ và $\sum_{c=1}^{C} g(z_c) = 1$.
- Xét trong $\{z_1, \ldots, z_C\}$, $g(z)$ tăng chặt: $z_i > z_j \Leftrightarrow g(z_i) > g(z_j)$.
- Ở đây, cũng như hồi quy Logistic, chúng ta dùng $(e)^z$ vì hàm này đủ trơn và đạo hàm tính được dễ dàng.
- Hình ở trang sau mình họa rõ hơn một số tính chất của hàm softmax.

Hàm softmax



Hình: Minh họa tính chất hàm SoftMax $a_i = g(z_i)$

Hàm softmax

- Hình trên cho thấy $g : \{z_i\}_{i=1}^{C} \subset \mathbb{R} \to (0, 1)$ có giá trị phụ thuộc theo tập $\{z_i\}_{i=1}^{C}$, tăng chặt trong mỗi tập $\{z_c\}$, $\sum_{c=1}^{C} g(z_c) = 1$ và
  - Từ cột 1: với tập $\{z_1, \ldots, z_C\}$ mà $z_1 = z_2 = \ldots = z_C$ thì

$$g(z_1) = g(z_2) = \ldots = g(z_C) = \frac{1}{C}.$$

  - Từ cột 1 và 2: Nếu $z^* = \max\{z_c\} = \max\{\bar{z}_c\}$ mà $z_1 = z_2 = \ldots = z_C$ và $z^* \neq \min\{\bar{z}_c\}$ thì $g(z^*)_{\{\bar{z}_c\}} > g(z^*)_{\{z_c\}}$.
  - Từ cột 2 và 3: Khi các giá trị $z_i \leq 0$ thì các giá trị $a_i = g(z_i)$ vẫn dương và thứ tự vẫn được đảm bảo.
  - Từ cột 4: $z_i = z_j \Leftrightarrow g(z_i) = g(z_j)$.

Softmax-stable

- 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

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

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

Xây dựng hàm xác suất phân lớp

- Do $\theta_k = (\theta_{k,j})^T$ $(j = 0, \ldots d))$, từ công thức của $g(z)$ ta có thể viết chi tiết xác suất để x thuộc vào lớp y:

$$P(y|x;\theta) = \frac{\exp(\sum_{j=1}^{d} \theta_{y,j} x_j)}{\sum_{c=1}^{C} \exp\left(\sum_{j=1}^{d} \theta_{c,j} x_j\right)} = \frac{\exp(\theta_y^T x)}{\sum_{c=1}^{C} \exp\left(\theta_c^T x\right)} = g(\theta_y^T x) \quad (1)$$

- với g là hàm softmax xây dựng trên tập $\{(\theta_c^T x)\}_{c=1}^{C}$.
- Do $\sum_{k=1}^{C} P(y = k|x;\theta_k) = 1$, thực sự chỉ cần tính $C - 1$ vector θ_k.
- Có hai cách để đi đến hàm mục tiêu:
  - Sử dụng ước lượng hợp lý cực đại - hàm log-likelihood hoặc
  - Sử dụng so sánh phân phối xác suất với vector one-hot-coding qua Entropy chéo (Cross Entropy).
- Ta sẽ sử dụng cách thứ nhất và giới thiệu về cách thứ hai.

Hàm tổn thất

- Từ giả thiết xác suất xẩy ra của dữ liệu là độc lập, trung bình log-Likelihood của tập dữ liệu huấn luyện là:

$$\begin{aligned}\ell(\theta) &= \frac{1}{N}\sum_{n=1}^{N}\log P(y_n|\mathbf{x}_n;\theta) = \frac{1}{N}\sum_{n=1}^{N}\log\left(\frac{\exp(\theta_{y_n}^T\mathbf{x}_n)}{\sum_{c=1}^{C}\exp\left(\theta_c^T x_n\right)}\right) \\ &= \frac{1}{N}\sum_{n=1}^{N}\left\{(\theta_{y_n}^T\mathbf{x}_n) - \log\left[\sum_{c=1}^{C}\exp\left(\theta_c^T\mathbf{x}_n\right)\right]\right\}\end{aligned} \tag{2}$$

- Theo phương pháp MLE, chúng ta sẽ tìm các tham số tối ưu $\theta_c \in \mathbb{R}^d$ $(c = 1, \ldots, C)$ thông qua cực đại hóa phiếm hàm log-Likelihood $\ell(\theta)$.

Hàm tổn thất

- Tương tự cho trường hợp 2 class, thay cho cực đại hóa log-Likelihood $\ell(\theta)$, ta tìm $\theta_c \in \mathbb{R}^d$ $(c = 1, \ldots, C)$ để cực tiểu hóa

$$-\ell(\theta) = -\frac{1}{N}\sum_{n=1}^{N}\left\{(\theta_{y_n}^T \mathbf{x}_n) - \log\left[\sum_{c=1}^{C}\exp\left(\theta_c^T \mathbf{x}_n\right)\right]\right\}.$$

  Đây chính là một dạng hàm tổn thất chúng ta xây dựng dựa trên MLE.

- Trường hợp sử dụng hiệu chỉnh L_2, chúng ta có hàm tổn thất

$$J(\theta) = -\ell(\theta) + \frac{\lambda}{2}\sum_{c=1}^{C}\|\theta_c\|_2^2 = -\ell(\theta) + \frac{\lambda}{2}\sum_{c=1}^{C}\sum_{j=1}^{d}\theta_{cj}^2. \qquad (3)$$

Hàm tổn thất

- Xét cho 01 cặp dữ liệu duy nhất $(\mathbf{x}_n, y_n)$, ta có hàm log-Likelihood

$$\ell_n(\theta) = \ell(\theta; \mathbf{x}_n, y_n) = (\theta_{y_n}^T \mathbf{x}_n) - \log \left[\sum_{c=1}^{C} \exp\left(\theta_c^T \mathbf{x}_n\right) \right].$$

- Để giải bài toán cực tiểu, chúng ta sẽ dùng đến các phương pháp dạng Gradient Descent. Với $k = 1, \ldots, C$ ứng với các phân lớp; $j = 1, \ldots, d$ ứng với các chiều dữ liệu, ta tính các đạo hàm

$$\begin{aligned} \frac{\partial \ell_n(\theta)}{\partial \theta_{k,j}} &= \frac{\partial}{\partial \theta_{k,j}} (\theta_{y_n}^T \mathbf{x}_n) - \frac{\partial}{\partial \theta_{k,j}} \log \left[\sum_{c=1}^{C} \exp\left(\theta_c^T \mathbf{x}_n\right) \right] \\ &= \delta(y_n = k)(\mathbf{x}_n)_j - \frac{1}{\sum_{c=1}^{C} \exp\left(\theta_c^T \mathbf{x}_n\right)} \cdot \frac{\partial}{\partial \theta_{k,j}} \left[\sum_{c=1}^{C} \exp\left(\theta_c^T \mathbf{x}_n\right) \right] \end{aligned} \tag{4}$$

Hàm tổn thất

- Trong (4), hàm $\delta(i=j) = \begin{cases} 1 & \text{nếu} \quad i=j \\ 0 & \text{nếu} \quad i \neq j \end{cases}$. Bên cạnh đó ta có

$$\frac{\partial}{\partial \theta_{k,j}} \left[\sum_{c=1}^{C} \exp\left(\theta_c^T \mathbf{x}_n\right) \right] = \sum_{c=1}^{C} \left[\frac{\partial}{\partial \theta_{k,j}} \exp\left(\theta_c^T \mathbf{x}_n\right) \right]$$
$$= \frac{\partial}{\partial \theta_{k,j}} \exp\left(\theta_k^T \mathbf{x}_n\right) = \exp\left(\theta_k^T \mathbf{x}_n\right)(\mathbf{x}_n)_j$$

vì trong tổng trên, các biểu thức $\exp(\theta_c^T \mathbf{x}_n)$ với $c \neq k$ không chứa biến $\theta_{k,j}$. Với mọi $k = 1, 2, \ldots, C$ và $j = 1, 2, \ldots, d$.

Hàm tổn thất

- Thay vào (4) suy ra

$$\begin{aligned}\frac{\partial \ell_n(\theta)}{\partial \theta_{k,j}} &= \delta(y_n = k)(\mathbf{x}_n)_j - \frac{\exp\left(\theta_k^T \mathbf{x}_n\right)(\mathbf{x}_n)_j}{\sum_{c=1}^{C} \exp\left(\theta_c^T \mathbf{x}_n\right)} \\ &= \left[\delta(y_n = k) - P(y_n = k|x;\theta)\right](\mathbf{x}_n)_j.\end{aligned}$$

- Lấy tổng theo tất cả $(\mathbf{x}_n, y_n)$, $n = 1, \ldots, N$, ta được các đạo hàm

$$\begin{aligned}\frac{\partial}{\partial \theta_{k,j}} \ell(\theta) &= \frac{1}{N} \sum_{n=1}^{N} \delta(y_n = k)(\mathbf{x}_n)_j - \frac{1}{N} \sum_{n=1}^{N} P(y_n = k|x;\theta)(\mathbf{x}_n)_j \\ &= \frac{1}{N} \sum_{n=1}^{N} \left\{\delta(y_n = k) - g(\theta_k^T \mathbf{x}_n)\right\}(\mathbf{x}_n)_j.\end{aligned} \tag{5}$$

Hàm tổn thất

- Từ (5) ta có thể tính được tất cả đạo hàm $\frac{\partial \ell(\theta)}{\partial \theta_{k,j}}$, với mọi $k = 1, \ldots, C$ và $j = 1, \ldots, d$. Chú ý, trong vế thứ 2 của phương trình (5)
  - Đại lượng $\frac{1}{N} \sum_{n=1}^{N} \delta(y_n = k)(\mathbf{x}_n)_j$ là kì vọng mẫu của đặc trưng (chiều - tọa độ) thứ j trên dữ liệu huấn luyện ứng với phân lớp k;
  - Đại lượng $\frac{1}{N} \sum_{n=1}^{N} P(y_n = k|x; \theta)(\mathbf{x}_n)_j$ là kì vọng của đặc trưng thứ j ứng với mô hình $P(y = k|\mathbf{x}; \theta)$

  Nghĩa là, $\frac{\partial \ell(\theta)}{\partial \theta_{k,j}}$ là hiệu của kỳ vọng mẫu và kỳ vọng theo mô hình, ở phân lớp k, chiều (đặc trưng) thứ j.

Hàm tổn thất

Từ (5) và các nhận xét trên, ta có thể đi đến một dạng khác của $\dfrac{\partial \ell(\theta)}{\partial \theta_{k,j}}$.

- Với mỗi nhãn $y_n = c \in \{1, \ldots, C\}$, ta đặt $\bar{y}^n$ là vector one-hot-coding của nó, tức $\bar{y}^n \in \mathbb{R}^C$, nếu $y_n = c$ thì thành phần thứ c của $\bar{y}^n$ là $\bar{y}^n_c = 1$ và $\bar{y}^n_m = 0$ với $m \neq c$. Đặt $a_{kn} := P(y_n = k|\mathbf{x}_n; \theta) = g(\theta_k^T \mathbf{x}_n)$, (5) dẫn tới

$$\begin{aligned} \frac{\partial}{\partial \theta_{k,j}} \ell(\theta) &= \frac{1}{N} \sum_{n=1}^{N} \left\{ \delta(y_n = k) - g(\theta_k^T \mathbf{x}_n) \right\} (\mathbf{x}_n)_j \\ &= \frac{1}{N} \sum_{n=1}^{N} \left(\bar{y}^n_k - a_{kn} \right) (\mathbf{x}_n)_j = \frac{1}{N} \sum_{n=1}^{N} e_{k,n} (\mathbf{x}_n)_j, \end{aligned} \tag{6}$$

ở đây $e_{k,n} := \bar{y}^n_k - a_{kn}$, là sai số giữa phân lớp theo dữ liệu quan sát và xác suất phân lớp theo mô hình.

Hàm tổn thất

- Có thể viết công thức các đạo hàm dưới dạng ma trận. Ta có ma trận tham số $\theta = [\theta_1, \ldots, \theta_C]$, với $\theta_k = (\theta_{k1}, \ldots, \theta_{kd})^T \in \mathbb{R}^d$ - là vector cột. Lúc đó đạo hàm của $\ell(\theta)$ theo từng vector cột có thể viết dạng

$$\frac{\partial \ell_n(\theta)}{\partial \theta} = \left[\frac{\partial \ell_n(\theta)}{\partial \theta_1}, \frac{\partial \ell_n(\theta)}{\partial \theta_2}, \ldots, \frac{\partial \ell_n(\theta)}{\partial \theta_C}\right] = [e_{1,n}\mathbf{x}_n, e_{2,n}\mathbf{x}_n \ldots, e_{C,n}\mathbf{x}_n]$$
$$= \mathbf{x}_n [e_{1,n}, e_{2,n}, \ldots, e_{C,n}] =: \mathbf{x}_n \mathbf{e}_n^T.$$

- Vậy dạng ma trận của đạo hàm của $\ell(\theta)$ sẽ là

$$\frac{\partial \ell(\theta)}{\partial \theta} = \sum_{n=1}^{N} \mathbf{x}_n \mathbf{e}_n^T = \mathbf{X}\mathbf{E}^T,$$

với $\mathbf{E} = \bar{\mathbf{Y}} - \mathbf{A}$, các ma trận đã được định nghĩa $\bar{\mathbf{Y}} = (\bar{y}_k^n)$; $\mathbf{A} = (a_{kn})$.

Hàm tổn thất

- Ta có dạng ma trận của đạo hàm hàm tổn thất cho một cặp dữ liệu

$$\frac{\partial J_n^0(\theta)}{\partial \theta} = -\frac{\partial \ell_n(\theta)}{\partial \theta} = -\mathbf{x}_n \mathbf{e}_n^T.$$

- Vậy dạng ma trận của đạo hàm của hàm tổn thất cho toàn bộ dữ liệu

$$\frac{\partial J^0(\theta)}{\partial \theta} = -\sum_{n=1}^{N} \frac{\partial \ell_n(\theta)}{\partial \theta} = \mathbf{X}\left(\mathbf{A} - \bar{\mathbf{Y}}\right)^T.$$

- Trường hợp sử dụng hiệu chỉnh L_2, hàm tổn thất xác định theo (3). Đạo hàm hàm tổn thất cho 01 dữ liệu và cho toàn bộ N dữ liệu tương ứng là

$$\frac{\partial J_n(\theta)}{\partial \theta} = -\mathbf{x}_n \mathbf{e}_n^T + \lambda\theta \quad \text{và} \quad \frac{\partial J(\theta)}{\partial \theta} = \mathbf{X}\left(\mathbf{A} - \bar{\mathbf{Y}}\right)^T + \lambda\theta. \qquad (7)$$

GD Method vs. Softmax Regression

Ta áp dụng (7) vào các thuật toán Gradient Descent để giải bài toán tối ưu $J(\theta) \rightarrow \min_\theta$

- **Stochastic Gradient Descent - SGD**
  1. Đầu vào: tham số học $\eta > 0$; $(\mathbf{x}_n, y_n)$. Khởi tạo: Chọn $\theta \in \mathbb{R}^{d \times C}$ bất kỳ ở bước $st = 0$;
  2. Tráo đổi ngẫu nhiên thứ tự các cặp dữ liệu $(\mathbf{x}_n, y_n)$. Lần lượt với $n = 1, 2, \ldots, N$, thực hiện:
  3. Tính $\dfrac{\partial J_n(\theta)}{\partial \theta}$ theo (7);
  4. Cập nhật tham số:
  $$\theta = \theta - \eta \cdot \frac{\partial J_n(\theta)}{\partial \theta}.$$
  5. Nếu đạt điều kiện dừng, kết thúc. Ngược lại quay lại bước 2.

GD Method vs. Softmax Regression

- **Batch Gradient Descent - BGD**
  1. Đầu vào: tham số học $\eta > 0$; $(\mathbf{x}_n, y_n)$. Khởi tạo: Chọn $\theta \in \mathbb{R}^{d \times C}$ bất kỳ ở bước $st = 0$;
  2. Tính $\dfrac{\partial J(\theta)}{\partial \theta}$ theo (7) với toàn bộ dữ liệu;
  3. Cập nhật tham số:
  $$\theta = \theta - \eta \cdot \frac{\partial J(\theta)}{\partial \theta}.$$
  4. Nếu đạt điều kiện dừng, kết thúc. Ngược lại quay lại bước 2.

Từ công thức ta có thể thấy Logistic Regression là trường hợp riêng của SoftmaxRegression.

1. A toy example

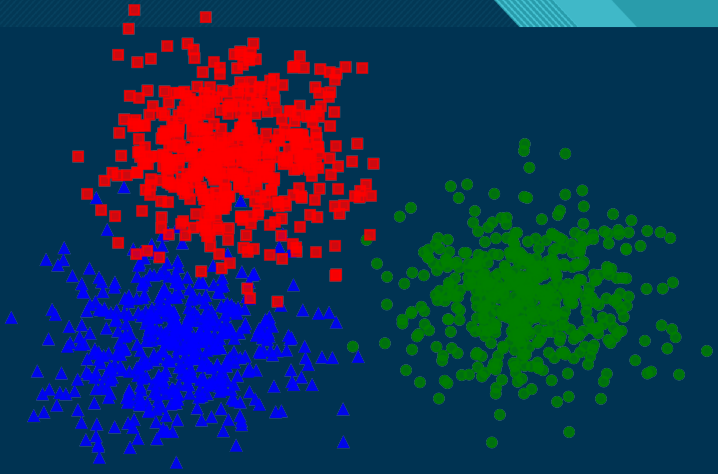
Ta bắt đầu với một ví dụ với dữ liệu tự tạo để minh họa thuật toán.

- Ta sử dụng $C = 3$ phân lớp; Không gian dữ liệu có $d = 2$ chiều và tập huấn luyện có $N = 500$ cặp dữ liệu.
- Dữ liệu $\mathbf{x}_n \in \mathbb{R}^2$ được tạo ngẫu nhiên theo phân bố chuẩn với kỳ vọng (tọa độ tâm của mỗi phân lớp) và hiệp phương sai cho trước.
- Mã lệnh tạo dữ liệu như sau

1. A toy example

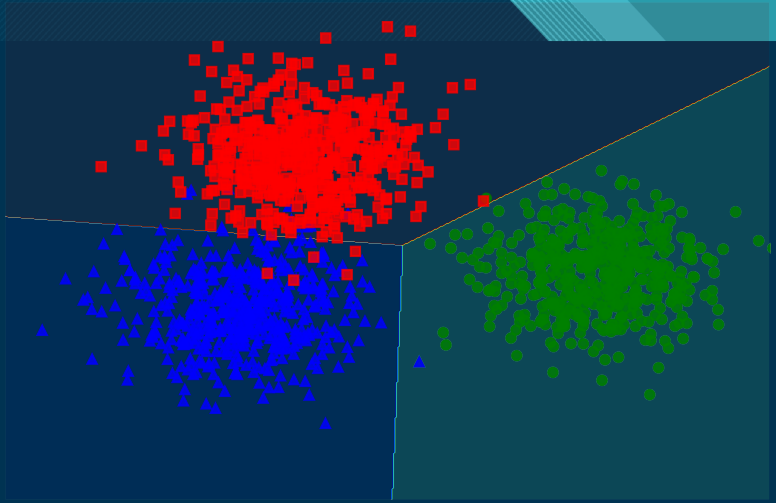
```
means = [[2, 2], [8, 3], [3, 6]]
cov = [[1, 0], [0, 1]]
N = 500
X0 = np.random.multivariate_normal(means[0], cov, N)
X1 = np.random.multivariate_normal(means[1], cov, N)
X2 = np.random.multivariate_normal(means[2], cov, N)

# each column is a datapoint
X = np.concatenate((X0, X1, X2), axis = 0).T
# extended data
X = np.concatenate((np.ones((1, 3*N)), X), axis = 0)
C = 3

original_label = np.asarray([0]*N + [1]*N + [2]*N).T
```

1. A toy example



Hình: Dữ liệu huấn luyện thuộc $C = 3$ phân lớp.

1. A toy example

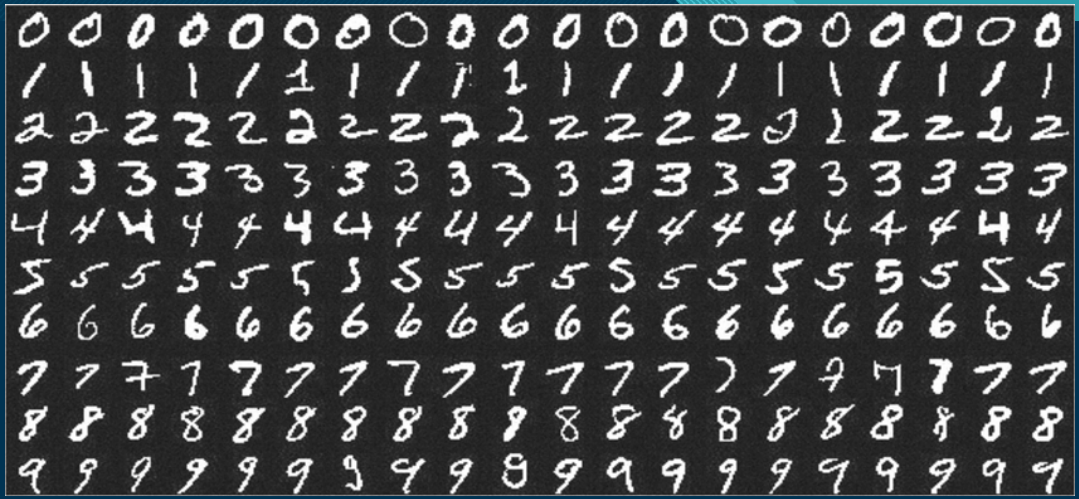


Hình: Kết quả phân lớp bằng Softmax Regression.

2. Handwritten Digit Recognition

- Ứng dụng Softmax Regression trên tập dữ liệu các chữ số viết tay. Bộ dữ liệu chứa 5000 mẫu dữ liệu huấn luyện.
- Dữ liệu là một phần của dữ liệu tại http://yann.lecun.com/exdb/mnist/.
- Mỗi dữ liệu huấn luyện là một ảnh đa cấp xám kích thước 28x28 pixel của các chữ số viết tay.
- Mỗi pixel lưu một số thực cho biết cường độ độ xám tại vị trí đó.

2. Handwritten Digit Recognition



Hình: Một phần của dữ liệu chữ số viết tay.