About Neural Network

Viet Hoang Duc

April 2019

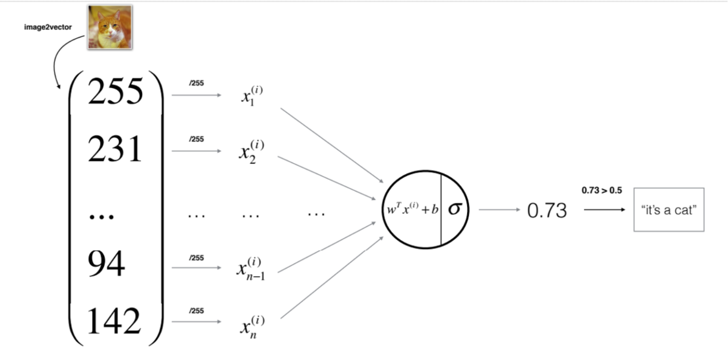
Content

- Logistic Regression.
- Softmax Regression.
- Shallow Neural Network.
- Deep Neural Network.

Logistic Regression

- Đầu ra của logistic regression thể hiện dưới dạng xác suất.
- Cho tập điều kiện x, mục tiêu tìm $\hat{y} = P(y = 1|x)$
- Ví dụ:
  - Xác suất 1 email là spam dựa trên người gửi, số lỗi chính tả hoặc số lần lặp lại của từ "offer", "free gift"...
  - Xác suất 1 khối u là ác tính hay không dựa trên đặc điểm ví dụ như kích thước khối u, khối u nằm ở đâu.
  - Xác định là ảnh con mèo dựa trên thông tin hình ảnh.

Logistic Regression



Logistic Regression

- Đầu ra dự đoán của logistic regression thường được viết dưới dạng:

$$\hat{y} = f(x) = \theta(\mathbf{w}^T x) \quad (*)$$

- Trong đó:
  - $\hat{y}$ là giá trị dự đoán. $0 \leq \hat{y} \leq 1$
  - θ là logistic funtion.
  - $\mathbf{w}^T$ là tham số.
- Trong phần này ta đề cập đến hàm sigmoid:

$$\theta(s) = \frac{1}{1 + e^{-s}} \triangleq \sigma(s)$$

- Tập dữ liệu (data set) nhận được: $\left\{\left(x^1, y^1\right), ..., (x^m, y^m)\right\}$. Tìm $\hat{y}^{(i)} \approx y^{(i)}$.
- $\hat{y}^{(i)} = \sigma(z^{(i)})$ với $\sigma(z^{(i)}) = \frac{1}{1+e^{-z^{(i)}}}$ và $z^{(i)} = \mathbf{w}^T x^{(i)}$.
- Loss Function:

$$\mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1-y)\log(1-\hat{y}))$$

- Cost function:

$$J(\mathbf{w}) = -\frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}(y \log \hat{y} + (1-y)\log(1-\hat{y}))$$

- Gradient descent, cần tìm đạo hàm J đối với $\mathbf{w}$:

$$\mathbf{w} = \mathbf{w} - \eta \frac{\partial J(\mathbf{w}, b)}{\partial \mathbf{w}}$$

- Cost function:

$$J(\mathbf{w}) = -\frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

- Loss function:

$$\mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$$

- Sigmoid function:

$$\hat{y} = \sigma(z) \quad với \quad z = \mathbf{w}^T x$$

- Đạo hàm J đối với $\mathbf{w}$:

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial J(\mathbf{w})}{\partial \mathcal{L}(\hat{y}, y)} \times \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial \mathbf{w}} \tag{1}$$

- Đạo hàm của J đối với $\mathcal{L}$:

$$\frac{\partial J(\mathbf{w})}{\partial \mathcal{L}(\hat{y}, y)} = -\frac{1}{m} \sum_{i=1}^{m} \frac{\partial \mathcal{L}(\hat{y}^{(i)}, y^{(i)})}{\partial \hat{y}^{(i)}} \tag{2}$$

- Đạo hàm của $\mathcal{L}$ đối với $\hat{y}$:

$$\frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} = -(\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}}) = \frac{\hat{y}-y}{\hat{y}(1-\hat{y})} \quad (3) \tag{3}$$

- Đạo hàm của $\hat{y}$ đối với biến z

$$\begin{aligned}\frac{\partial \hat{y}}{\partial z} &= \sigma'(z) = \frac{e^{-z}}{(1+e^{-z})^2} \\ &= \frac{1}{1+e^{-z}} \frac{e^{-z}}{1+e^{-z}} = \sigma(z)(1-\sigma(z)) \\ &= \hat{y}(1-\hat{y})\end{aligned} \tag{4}$$

- Đạo hàm của z đối với $\mathbf{w}$:

$$\frac{\partial z}{\partial \mathbf{w}} = x \tag{5}$$

- Thay (2) (3) (4) (5) vào (1):

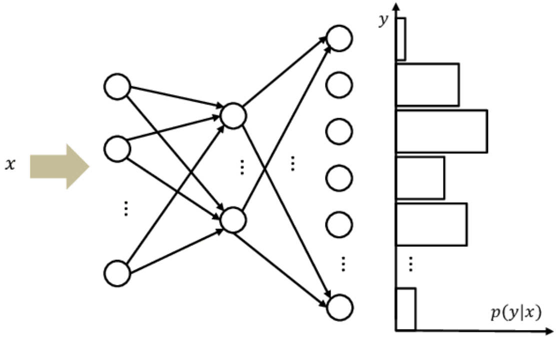
$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = -\frac{1}{m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})x^{(i)}$$

- Công thức cập nhật theo Gradient Descent:

$$\mathbf{w} = \mathbf{w} - \eta \frac{\partial J(\mathbf{w}, b)}{\partial \mathbf{w}}$$

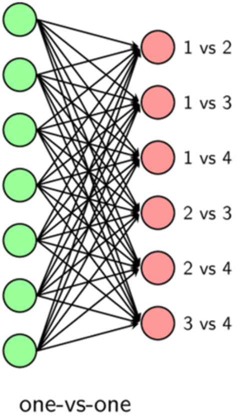
Với η là learning rate.

Softmax Regression



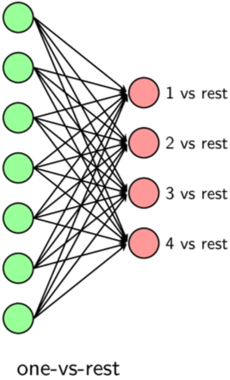
Softmax Regression biểu diễn dưới dạng Neural Network

Figure: One-vs-one biểu diễn dưới dạng Neural Network



- Xây dựng nhiều bộ binary classifiers cho từng cặp classes. Bộ thứ nhât phân biệt class 1 và class 2, bộ thứ 2 phần biệt class 1 và class 3, ... Tuy nhiên, khi đó cần rất nhiều bộ binary classifiers.
- Số lượng bộ cần dùng là $\frac{n(n-1)}{2}$

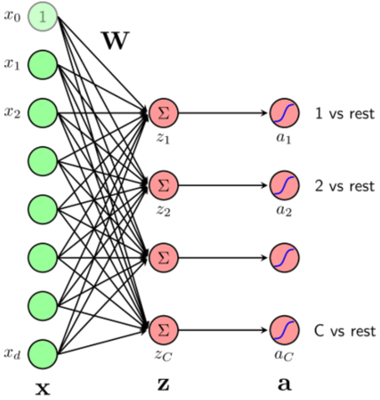
Figure: One-vs-one biểu diễn dưới dạng Neural Network

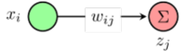


- Phương pháp này còn được gọi là one-hot coding. Ta sẽ xây dựng những bộ classifiers để phân loại C classes. Classifiers thứ nhất phân biệt class 1 vs not class 1, tương tự như thế với các classes còn lại
- Phương pháp one-vs-rest theo góc nhìn xác suất, một điểm dữ liệu có thể được dự đoán thuộc vào class $1, 2, \ldots, C$ với xác suất lần lượt là $p_1, p_2, \ldots, p_C$. Tuy nhiên, tổng các xác suất này có thể không bằng 1.

Softmax Regression

Figure: Multi-class classification với Logistic Regression và one-vs-rest.





w_{0j}: biases, don't forget!

d: data dimension

C: number of classes

$\mathbf{x} \in \mathbb{R}^{d+1}$

$\mathbf{W} \in \mathbb{R}^{(d+1)\times C}$

$z_i = \mathbf{w}_i^T \mathbf{x}$

$\mathbf{z} = \mathbf{W}^T \mathbf{x} \in \mathbb{R}^C$

$a_i = \text{sigmoid}(z_i) \in \mathbb{R}$

$0 < a_i < 1$

$$a^{(i)} = \mathrm{sigmoid}(z^{(i)}) = \mathrm{sigmoid}(\mathbf{W}^{(i)T}\mathbf{x})$$

Tập dữ liệu (data set) nhận được: $\left\{\left(x^1, y^1\right), ..., \left(x^m, y^m\right)\right\}$. Và $y^{(i)} \in \{1, 2, \ldots, C\}$. Với C là số lượng classes. Tìm $\hat{y}^{(i)} \approx y^{(i)}$. Ví dụ trong bộ dữ liệu MNIST thì C $= 10$. Cho một điểm dữ liệu x, mục tiêu tìm $\hat{y} = P(y = k|x), \quad \forall k = 1, \ldots, C$ Do ta cần $\sum\limits_{i=1}^{C} \hat{y}^{(i)} = 1$ nên nếu áp dụng hàm sigmoid như Logistic Regression thì tổng các xác suất không bằng một. Phương án là sử dụng một hàm số khác thoả mãn:

- Hàm đồng biến do $z^{(i)}$ càng lớn thì xác suất rơi vào class i càng tăng cao.
- Do $z^{(i)}$ có thể nhận giá trị âm nên hàm số cần chắc chắn cho ra giá trị của $\hat{y}^{(i)}$ chạy trong đoạn 0 đến 1.

- Hàm số ta cần tìm:

$$\hat{y}^{(i)} = \begin{bmatrix} P(y=1|x;\mathbf{W}) \\ P(y=2|x;\mathbf{W}) \\ \vdots \\ P(y=C|x;\mathbf{W}) \end{bmatrix} = \theta(z^{(i)}) = \frac{e^{z^{(i)}}}{\sum_{j=1}^{C} e^{z_j^{(i)}}} \tag{6}$$

- Trong đó:

$$z^{(i)} = \mathbf{W}^T x^{(i)}$$

Ví dụ trong tập dữ liệu MNIST, input là một ảnh grayscale kích thước 28x28 px. x là một vector (784,1) có output là y là một vector (10,1). Suy ra:

$$z = \mathbf{W}^T x$$

Với:

- z có kích thước (10,1).
- $\mathbf{W}$ có kích thước (784,10).

Cần chú ý rằng trong Logisitc Regression chỉ có 1 node ở output layer nên trong trường hợp đó, tham số mô hình chỉ là 1 vector $\mathbf{w}$ có kích thước (784,1). Tuy nhiên trong bài toán này, có 10 node ở output layer nên ta có ma trận trọng số $\mathbf{W} = [\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_9]$.

Ví dụ với C = 4: $z = \begin{bmatrix} 5 \\ 9 \\ -4 \\ 3 \end{bmatrix}$ $\quad \hat{y} = \theta(z) = \begin{bmatrix} 0.018 \\ 0.980 \\ 0.000 \\ 0.002 \end{bmatrix}$ và $\quad y = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$

Hàm mất mát - Loss Function:

$$\begin{aligned} \mathcal{L}(\hat{y}, y) &= -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \\ &= -\log 0.980 \end{aligned}$$

Hàm chi phí - cost function:

$$
\begin{aligned}
J(\mathbf{W}) &= \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) \\
&= -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{C} y_j^{(i)} \log(\hat{y}_j^{(i)})
\end{aligned}
$$

- Có m điểm dữ liệu: $\left\{\left(x^1, y^1\right), ..., \left(x^m, y^m\right)\right\}$ $\quad x^{(i)} \in \mathbb{R}^{n_x}$

$$X = \begin{bmatrix} | & | & | & | \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ | & | & | & | \end{bmatrix}$$

$$X.shape = (n_x, m)$$

- Tương tự:

$$Y = \left[y^{(1)}, y^{(2)}, \dots, y^{(m)}\right] \quad y^{(i)} \in \mathbb{R}^C$$

$$Y.shape = (C, m)$$

- Và:

$$Z = \left[z^{(1)}, z^{(2)}, \dots, z^{(m)}\right] \quad z^{(i)} \in \mathbb{R}^C$$

$$Z.shape = (C, m)$$

Giả sử rằng chúng ta sử dụng SGD, công thức cập nhật cho ma trận trọng số **W** sẽ là:

$$\mathbf{W} = \mathbf{W} + \eta x^{(i)}(y^{(i)} - \hat{y}^{(i)})$$