Multi-class Classification

ACM AI | Intro to Machine Learning: Beginner Track #4

Slides: tinyurl.com/f20btrack4
Attendance code: **cars**
Discord: bit.ly/ACMdiscord

acm.ai

Logistic Regression (Binary Classification) Review

acm.ai

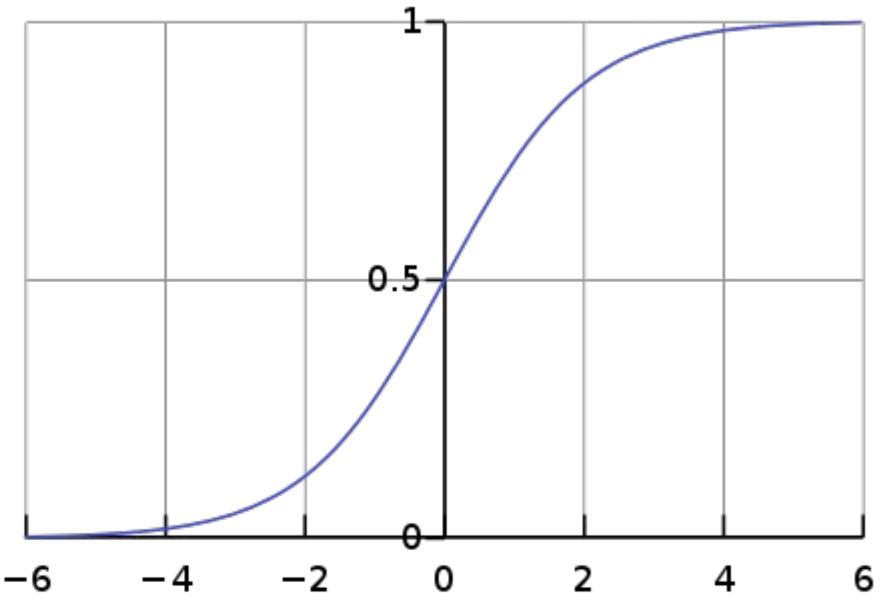
Sigmoid Function

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

If x is negative,
sigma(x) < 0.5

If x is positive,
sigma(x) > 0.5

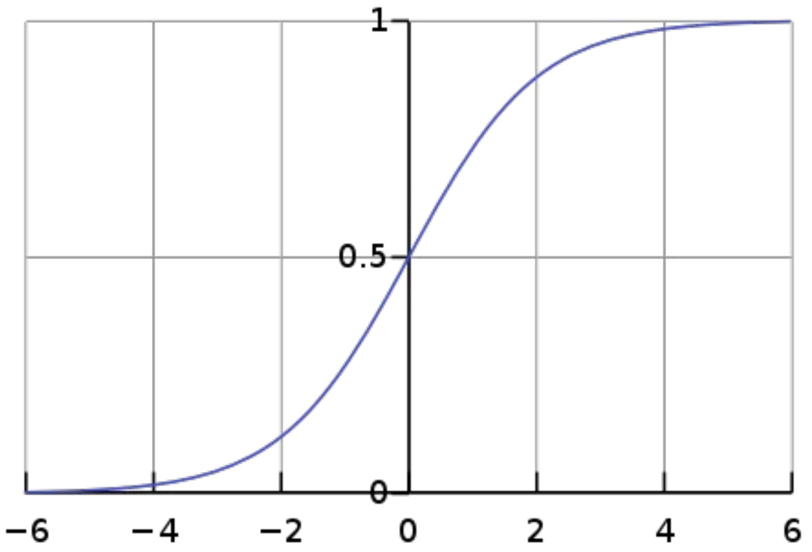
Also,
0 <= sigma(x) <= 1



Logistic Regression

$$\hat{y}(x) = \frac{1}{1 + e^{-(W^T X + b)}}$$

- An input needs to be classified as **0** or **1**
- We need to find the **decision boundary** or the **W** and **b** for our model
- This is known as **binary** classification



Cost Function: Binary Cross-Entropy Loss

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

- $\hat{y}$: prediction.
- y : label
- What happens when **y** is **1**? $L(\hat{y}, y) = -\log(\hat{y})$
- What happens when **y** is **0**? $L(\hat{y}, y) = -\log(1 - \hat{y})$

acm.ai

Quick Poll: Logistic Regression Review

Which task is logistic regression well suited for?

a. Predicting the price of a house
b. Predicting whether to approve a loan or deny a loan
c. Generating pictures of dogs and cats
d. Facial recognition software

Multi Class Classification

Labels

Imagine that we have a bunch of photos of cats, dogs, chickens, and fish that we want to classify.

0

1

2

0

2

3

??

To help our model distinguish them we can assign 0 to cats, 1 to dogs, 2 to chickens, and 3 to fish.

acm.ai

One Hot Encoding

For a single image we can assign a **0 or 1** to each category depending on whether or not the image is under that category.

$$\begin{array}{l} \text{Cat:} \\ \text{Dog:} \\ \text{Chicken:} \\ \text{Fish:} \end{array} \quad \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Then we put these labels into a vector indexed by each class.

This process is called **one hot encoding.**

One Hot Encoding

Now that our samples have one hot encoded labels, our model needs to have a similarly shaped output so that we can compare our predictions and labels.

$$\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$$

$$\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$$

$$\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$$

$$\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$$

$$\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$$

$$\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

acm.ai

Multi Class Model

For binary classification and linear regression, a single training example **X** was a **n-dimensional vector** for the n features in the example.

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$

The weight **W** was also an n-dimensional vector.

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix}$$

The bias **b** was a real number.

$$b$$

Multi Class Model

For multi-class classification, we have the same input **X.**

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

But now our weight **W** is an *(n x c)* matrix where **c** is the number of classes, **n** is the number of features

$$\begin{bmatrix} w_1^1 & w_1^2 & \cdots & w_1^c \\ w_2^1 & w_2^2 & \cdots & w_2^c \\ \vdots & \vdots & \ddots & \vdots \\ w_n^1 & w_n^2 & \cdots & w_n^c \end{bmatrix}$$

Our bias **b** becomes a c-dimensional vector.

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_c \end{bmatrix}$$

acm.ai

Multi Class Model

For our animal example, to generate the output we

1) take the pixel values from our image and put them in a vector for our **x**
2) multiply it by our weight matrix and add our bias vector
3) output our prediction **z**

$$\underset{\substack{(px,\,1)\\ x}}{\begin{bmatrix} 3 \\ 254 \\ 11 \\ \vdots \\ 19 \end{bmatrix}} \longrightarrow \underset{\substack{(4,\,px)\\ w^T}}{\begin{bmatrix} 8 & 32 & 99 & 10 \\ 13 & 17 & 12 & 38 \\ 14 & 2 & 58 & 31 \\ \vdots & \vdots & \vdots & \vdots \\ 19 & 39 & 13 & 29 \end{bmatrix}^T} \times \underset{\substack{(px,\,1)\\ x}}{\begin{bmatrix} 3 \\ 254 \\ 11 \\ \vdots \\ 19 \end{bmatrix}} + \underset{\substack{(4,\,1)\\ b}}{\begin{bmatrix} 2 \\ 3 \\ 19 \\ 8 \end{bmatrix}} \longrightarrow \underset{z}{\begin{bmatrix} 23 \\ 9 \\ 1 \\ 34 \end{bmatrix}}$$

Quick Poll: Challenge Question

In multi-class classification , with ‘f’ features, ‘c’ classes, ‘m’ training samples for X, the matrix W (weights) will h(f, ve dimensions:

a. (f, 1)
b. (f, c)
c. (m, f)
d. (c, m)

acm.ai

Softmax

Softmax

$$\mathbf{z} \qquad\qquad\qquad\qquad\qquad\qquad \hat{y}$$

$$\begin{bmatrix} 3 \\ 4 \\ 1 \\ 2 \end{bmatrix} \longrightarrow \frac{e^{z}}{\sum_{i=1}^{c} e^{z_i}} \longrightarrow \begin{bmatrix} 0.24 \\ 0.64 \\ 0.03 \\ 0.09 \end{bmatrix}$$

- takes in the output vector **z** from our model
- outputs vector $\hat{y}$ of probabilities for each class that sums to 1
- Why not use a simple ratio? (Think about negatives!)

Softmax

To convert our outputs **z** to probabilities $\hat{y}$ we,

1) raise *e* by component of our output vector **z**
2) divide by the sum of the previous vector to get a vector of probabilities $\hat{y}$

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_c \end{bmatrix} \longrightarrow \begin{bmatrix} e^{z_1} \\ e^{z_2} \\ e^{z_3} \\ \vdots \\ e^{z_c} \end{bmatrix} \longrightarrow \frac{1}{\sum_{i=1}^{c} e^{z_i}} \begin{bmatrix} e^{z_1} \\ e^{z_2} \\ e^{z_3} \\ \vdots \\ e^{z_c} \end{bmatrix} \longrightarrow \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_c \end{bmatrix}$$

acm.ai

Multi-class Cost Function

acm.ai

Cross Entropy

- need a general loss function that can apply to **c** number of classes
- needs to have a higher cost if our model makes a bad prediction
    - I.e. the probability for the correct class is far away from 1
- this function is called **cross entropy** or categorical cross entropy

Cross Entropy

$$L(\hat{y}, y) = \sum_{i=1}^{c} -y_i \log(\hat{y}_i)$$

$$\underset{y}{\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}} \leftarrow \quad \underset{\hat{y}}{\begin{bmatrix} .18 \\ .27 \\ .54 \\ .01 \end{bmatrix}} \leftarrow$$

- the only class that will contribute to the loss is the class that has a 1 in the label
- to minimize the cost, the model needs to make the corresponding class in $\hat{y}$ as close to 1 as possible

Cross Entropy

So total cost across all training sample becomes:

$$J(w,b) = \frac{1}{m}\sum_{j=1}^{m} L(\hat{y}_j, y_j)$$

$$J(w,b) = \frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{c} -y_{ji} log(\hat{y}_{ji})$$

Gradient Descent in Multi-Class Classification

acm.ai

Gradient Descent

The derivatives **dJ /dw** and **dJ / db** are the same as those in binary classification

$$\frac{\delta J}{\delta w} = \frac{1}{m}(\hat{Y} - Y)X^T \qquad w = w - \alpha\frac{\delta J}{\delta w}$$

$$\frac{\delta J}{\delta b} = \frac{1}{m}\sum(\hat{Y} - Y) \qquad b = b - \alpha\frac{\delta J}{\delta b}$$

Why is this true?
Because softmax is a **generalization** of the sigmoid function
and cross-entropy loss is a generalization of log loss

acm.ai

Putting it all together

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Pixels

$$\begin{bmatrix} 3 \\ 254 \\ 11 \\ \vdots \\ 19 \end{bmatrix}$$

(n, 1)

Linear model →

Outputs

$$\begin{bmatrix} 3 \\ 4 \\ 1 \\ 2 \end{bmatrix}$$

(c, 1)

Softmax →

Probabilities

$$\begin{bmatrix} 0.24 \\ 0.64 \\ 0.03 \\ 0.09 \end{bmatrix}$$

(c, 1)

Cross Entropy →

2.4 = loss

Quick Poll

Your model outputs the following probabilities for multi-class classification with 5 classes

[0.3, 0.2, 0.3, 0.1, x]

x = ?

a. 0.3
b. 0.2
c. 0.5
d. 0.1

Thank you! We'll see you next week!

Feedback form: tinyurl.com/btrackfeedback4

Next week: K-Nearest Neighbours + Python

Learn the most popular programming language in the world

Today's event code: **cars**

FB group: facebook.com/groups/uclaacmai

Github: github.com/uclaacmai/beginner-track-fall-2020

acm.ai