Logistic Regression and Multiclass Classification

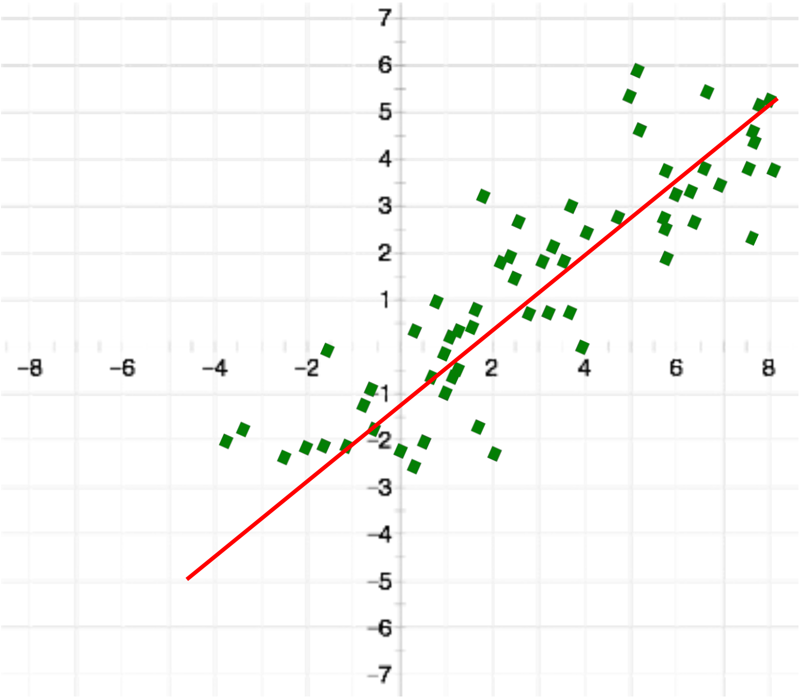
Intro to Machine Learning: Beginner Track #4

Feedback form: https://tinyurl.com/s21-btrack4-feedback
Attendance code: **Nigeria**
Discord: **bit.ly/ACMdiscord**

acm.ai

Linear regression recap

What is linear regression?



- Goal: to find the equation of a line that best fits our data
  - We want to be able to use this line to predict outputs from given inputs
- Classification or regression?

acm.ai

Linear Regression

$$\hat{y}(x) = b + w_1 x_1 + w_2 x_2 + \ldots\ldots w_n x_n$$

An input **X** is an **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** is also an n-dimensional vector.

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

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

$$b$$

acm.ai

Loss function

$$L(\hat{y_1}, \hat{y_2}, \ldots \hat{y_m}) = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y_i})^2$$

yhat$_i$ is the output of your model (**prediction**),

y$_i$ is the actual value (**target**),

all for training example number **i**

Gradient Descent

Use gradient descent on loss function to determine how to change each of the weights!

$$\frac{\delta L}{\delta w_j} = \frac{2}{m}\sum_{i=1}^{m}(\hat{y_i} - y_i)x_{ij} \qquad w_j = w_j - \alpha\frac{\delta L}{\delta w_j}$$

$$\frac{\delta L}{\delta b} = \frac{2}{m}\sum_{i=1}^{m}(\hat{y_i} - y_i) \qquad b = b - \alpha\frac{\delta L}{\delta b}$$

i refers to the ith training sample, **j** refers to the jth feature

Quick Poll: Linear Regression Review

What is linear regression well suited for?

a. Determining whether an image is a dog or a cat
b. Predicting whether a tumor is benign or harmful
c. Creating Siri
d. Predicting MCAT scores based on college GPA

Motivation

Motivation

- Linear Regression: predict continuous data (eg: house price)

- Logistic Regression: Classify data (this or that)

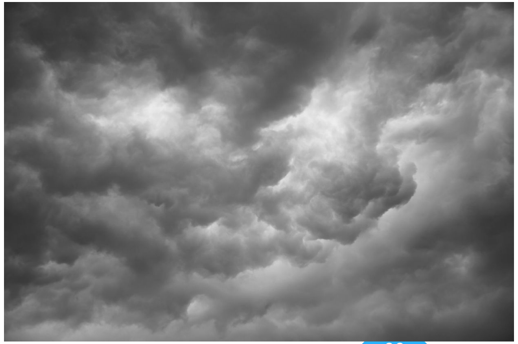
Output: $250,000

Output: Cat

acm.ai

An Example Problem

- Suppose we want to predict if it is going to rain today or not.
- What kind of features would we look at?
- Given these *features* how would we determine whether it is going to rain?
- **Logistic Regression!**



Example Problem

Suppose we are given the following data:

- Location: Los Angeles
- Temperature: 105F
- Cloud cover : Low
- Humidity: 50%
- Wind Speed: 10mph

Is it more likely to rain or not to rain?

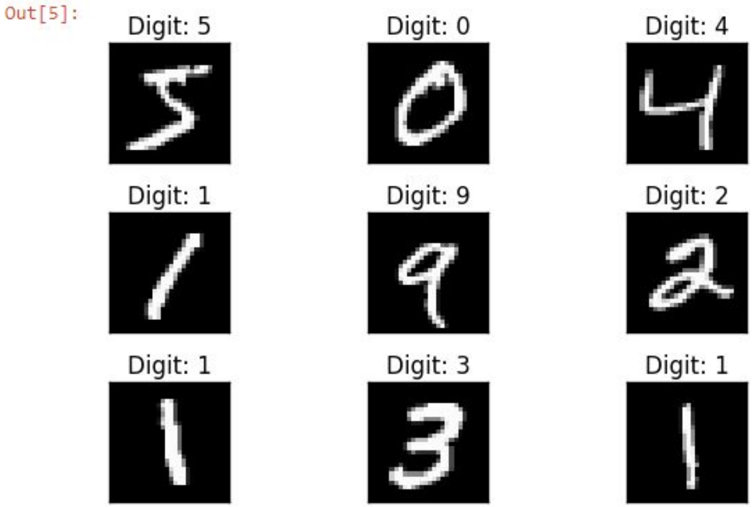
It's probably not going to rain!

I.e. the probability of it raining is less than **50%**

acm.ai

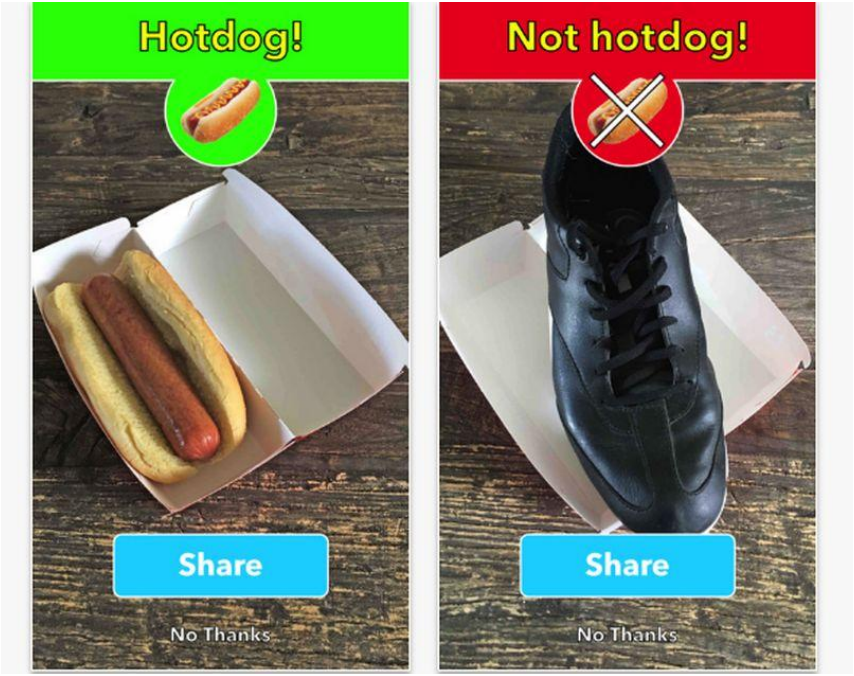
Examples of classification tasks

- Tumors - benign or malignant
- Numerical digits - one or two or three or …



Input and Output for binary classification

- Input for image classification task:
    - Array of pixels - the image
    - Each pixel is a feature
- Output for image classification task:
    - Binary Classification: 2 classes
    - Class label: **0** or **1**
    - Examples
        - 0 - Cat, 1 - dog
        - 0 - hotdog, 1 - not hotdog

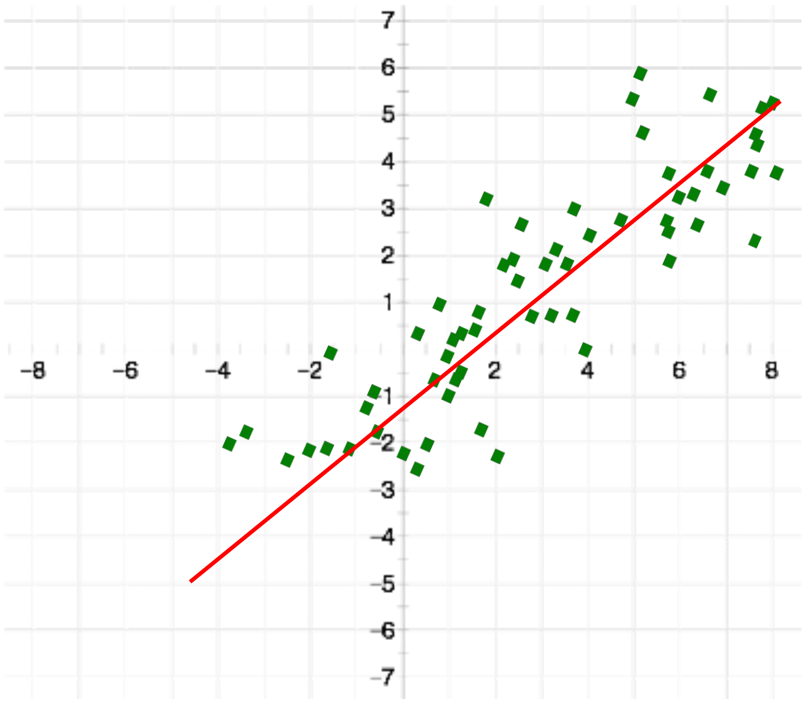


Logistic regression

acm.ai

Logistic Regression

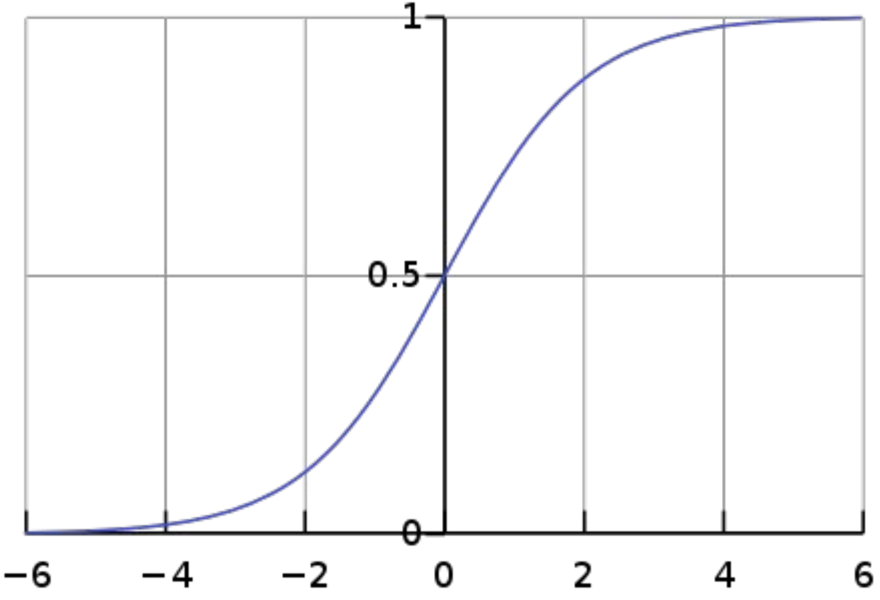
- We want our hypothesis to be a function to output a probability
- Our linear function maps to any real number
- How do we fix this?



acm.ai

Solution: The 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

Discussion Questions: Sigmoid Function

- Is this output continuous?
- How can we use this for classification?

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

acm.ai

Building the model

acm.ai

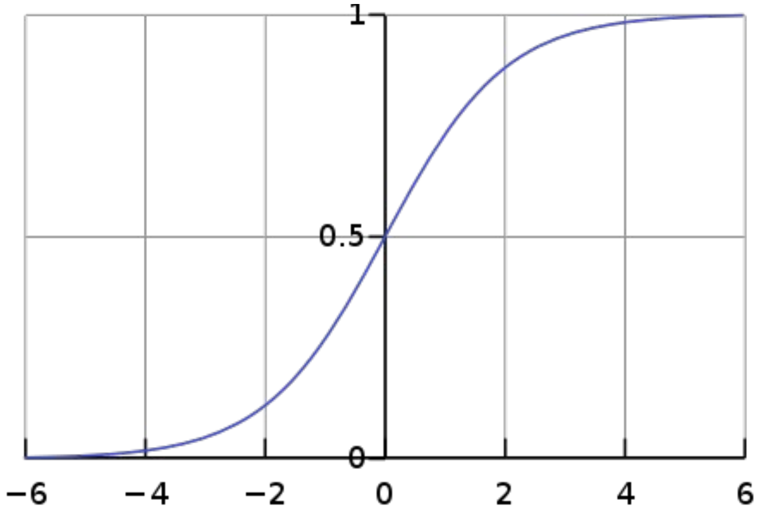
The Hypothesis

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

Instead of x, let's use our old linear function as the input

$$h(x) = W^T X + b$$

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



So depending on the values of **W** and **b**, an input **X** will result in a prediction ŷ that is either greater than 0.5 or lesser than 0.5

If ŷ<0.5 we can classify it as **0**
If ŷ>0.5 we can classify it as **1**

acm.ai

Probability of being a particular class

- Think of the output of the model as the **probability** of the input being class 1 given the features X.

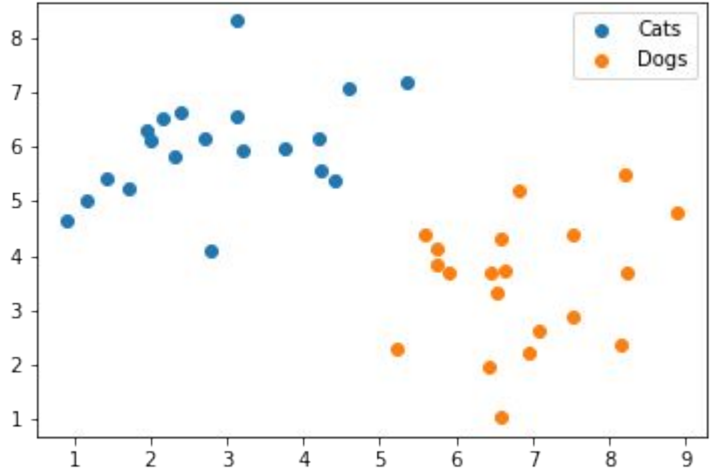
$$\hat{y}(x) = P(Y = 1|X)$$

- This is read as “Probability that the label **Y** is **1** given the features **X** we have”

So what do we need to find?

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

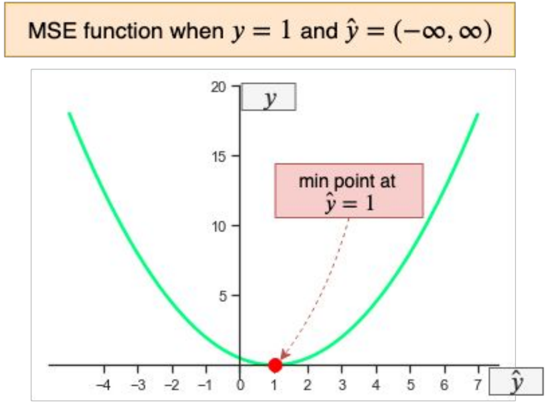
- We need to find the **decision boundary**
- That is, we must learn **W** and **b** such that an input **X** when transformed, is correctly classified as **0** or **1**
- How do we do this?
- Gradient descent on cost function!

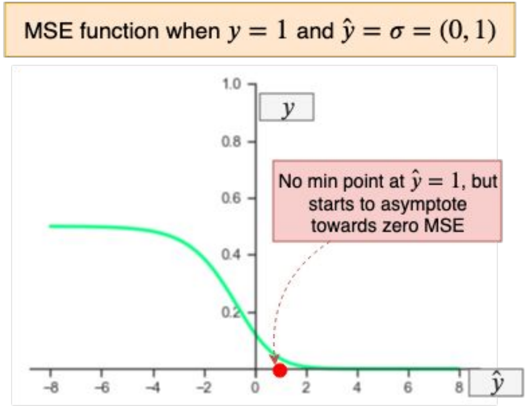


acm.ai

Cost Function: Why not mean squared error?

$$L(\hat{y_1}, \hat{y_2}, \dots \hat{y_m}) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y_i})^2$$





https://towardsdatascience.com/why-using-mean-squared-error-mse-cost-function-for-binary-classification-is-a-bad-idea-933089e90df7

Cost Function: Binary Cross-Entropy Loss

- Instead, we use **Binary Cross-Entropy Loss** or **Log Loss**

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

Cost Function: Binary Cross-Entropy Loss (aka Log Loss)

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

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

acm.ai

Cost Function: Binary Cross-Entropy Loss

So the total cost across all the samples becomes :

$$L(w,b) = \frac{1}{m}\sum_{i=1}^{m} L_{single}(\hat{y_i}, y_i)$$

$$L(w,b) = \frac{1}{m}\sum_{i=1}^{m} -y_i \log(\hat{y_i}) - (1-y_i)\log(1-\hat{y_i})$$

Quick poll: Correct Loss Function

Which of the following is the correct loss function for binary classification?

(a) $(1-y)\log(\hat{y}) + (y)\log(1-\hat{y})$

(b) $-(y)\log(\hat{y}) - (1-y)\log(1-\hat{y})$

(c) $(y)\log(\hat{y}) + (1-y)\log(1-\hat{y})$

(d) $-(1-y)\log(\hat{y}) - (y)\log(1-\hat{y})$

Gradient Descent

The derivatives **dL /dw** and **dL / db** are similar to those in linear regression.

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

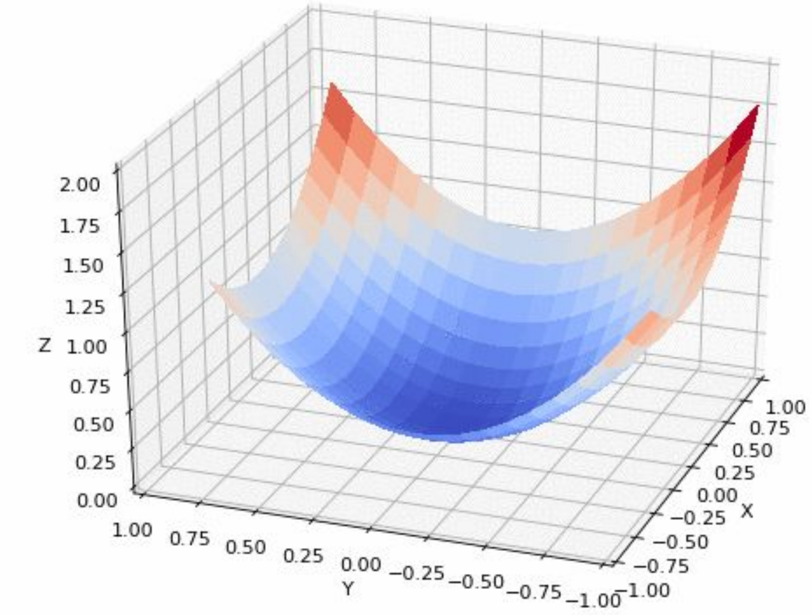
$$\frac{\partial L}{\partial b} = \frac{1}{m} \sum_{i=1}^{m} (\hat{Y}_i - Y_i) \qquad b = b - \alpha \frac{\partial L}{\partial b}$$

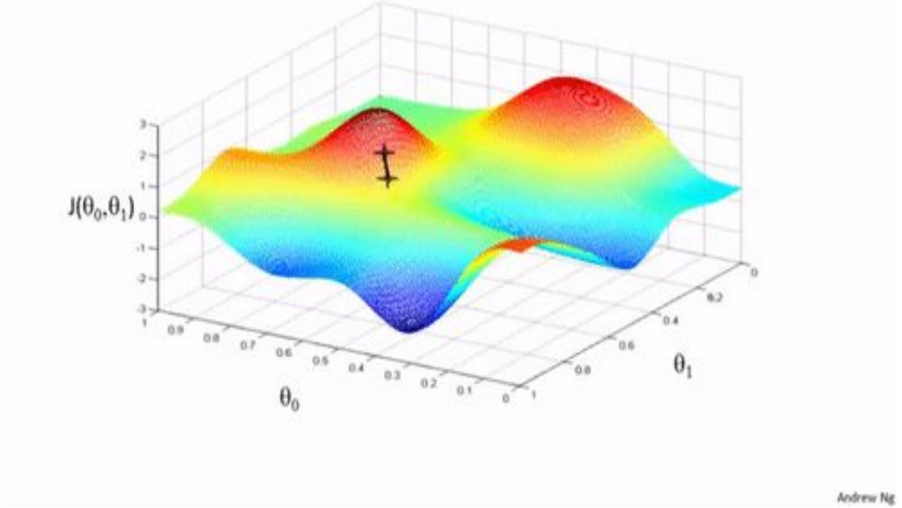
Yhat is a column vector (m x 1) containing all the predictions, ***Y*** is a column vector (m x 1) with the labels/target, and **X** is the data (m x n)

Check out the full [derivation]() if you are interested in the Math
Credit to **towardsdatascience.com**

acm.ai

Binary Cross-Entropy is Convex too!





Binary Classification Demo!

- http://playground.tensorflow.org

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

acm.ai

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.

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.

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

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

Cat:
Dog:
Chicken:
Fish:

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

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 (scalar).

$$b$$

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**

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

Quick Poll: Challenge Question

In multi-class classification , with ***f*** features, ***c*** classes, and ***m*** training samples for X, the matrix W (weights) will have dimensions:

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

Softmax

Softmax

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

- 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 $\hat{y}$

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

Multi-class Cost Function

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

acm.ai

Cross Entropy

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

$$\begin{array}{cc} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} .18 \\ .27 \\ .54 \\ .01 \end{bmatrix} \\ \text{y} & \hat{y} \end{array}$$

- 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

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

$$\frac{\partial L}{\partial b} = \frac{1}{m} \sum_{i=1}^{m} (\hat{Y}_i - Y_i) \qquad b = b - \alpha \frac{\partial L}{\partial b}$$

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

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

$$\underset{(n,\ 1)}{\overset{\text{Pixels}}{\begin{bmatrix}3\\254\\11\\\vdots\\19\end{bmatrix}}} \xrightarrow{\text{Linear model}} \underset{(c,\ 1)}{\overset{\text{Outputs}}{\begin{bmatrix}3\\4\\1\\2\end{bmatrix}}} \xrightarrow{\text{Softmax}} \underset{(c,\ 1)}{\overset{\text{Probabilities}}{\begin{bmatrix}0.24\\0.64\\0.03\\0.09\end{bmatrix}}} \xrightarrow{\text{Cross Entropy}} 2.4 = \text{loss}$$

Quick Poll

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

[0.3, 0.2, 0.3, 0.1, x]

What is x?

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

Coding Exercise

- https://tinyurl.com/s21-btrack4-colab

Thank you! We'll see you next week!

Please fill out our feedback form:
https://tinyurl.com/s21-btrack4-feedback

Next week: Numpy and Pandas

Grumpy Pandas more like Numpy and Pandas am I right?

Today's event code: **Nigeria**

FB group: facebook.com/groups/uclaacmai

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

acm.ai