Google Developer Student Clubs

ML Study Jams

Session #3 - Classification

Vijay Jaisankar
@Vijay73893478

Tech Lead, GDSC-IIITB

Classification

Essence

Categorise a set of data into a **set of classes**.

Learn the classification rules from **labelled** input data

Where do you think classification is used?

Hint: There are many 🌚

Go to slido.com and use the code
3597601

slido

Use cases of classification

ⓘ Start presenting to display the poll results on this slide.

You will “discover” a classifier from scratch!

With the help of a few of these 🏓

… and a few volunteers.

K - Nearest Neighbours

One of the simplest classifiers

- Find the k nearest neighbours of the target point given the input data
- Assign the target point based on the majority class of the nearest neighbours
- Tie-breakers based on nearest distance

I know what you’re thinking - can we **extend Linear Regression to classification**? We learned about it in previous study jams and it seems natural!

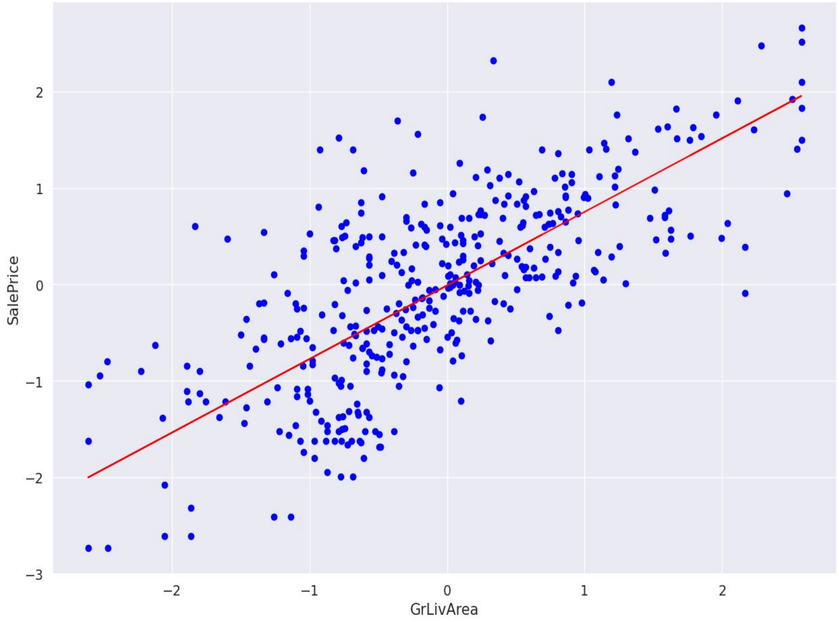
`Okay, you weren’t thinking about it - but you are now!`

Recap : Linear Regression

a.k.a “Fitting a line to data”

Linear Regression - the big picture

Credits: ML Study Jams - sessions 1 and 2



Linear Regression for everyone!

a.k.a Gentle Introduction

- We estimate a target variable by representing it as a *linear sum* of input features

- The parameters of the linear sum (weights) are learned through the rinse-repeat cycle of changing them based on a *Loss function*

- At the end of training, we have a **function** that takes features' data and gives a **real number** that bears some resemblance to what the target variable would be for the same inputs.

Linear Regression - equations

Credits: ML study Jams - sessions 1 and 2

Loss function - we just saw this!

$$J(\theta) = \sum_{i=1}^{N} (y_i - x_i^T \theta)^2 \quad \Rightarrow \quad J(\theta) = ||Y - X\theta||^2$$

$$\frac{\partial J}{\partial \theta} = 2(X^T X\theta - X^T Y)$$

Remember Gradient descent?

Updating the parameters (theta) by progressing against the gradient

Okay, now that we have a function that gives a real value for inputs, how do we build a classifier out of it?

It's a good question!

```python
"""
    Thresholding
        Assume binary classification
"""
def condition(y):
    pass # What will you fill here?

"""
    Find class
        Based on the "linear sum" output, decide which class
"""
def find_class_0_1(X, theta):
    y = theta.T * X # For example
    if condition(y):
        return 0
    else:
        return 1
```

```python
"""
    Thresholding
        Assume binary classification
"""
def condition(y):
    pass # What will you fill here?

"""
    Find class
        Based on the "linear sum" output, decide which class
"""
def find_class_0_1(X, theta):
    y = theta.T * X # For example
    if condition(y):
        return 0
    else:
        return 1
```

y <= n

So, we can threshold the values of y. Okay, but...

Is that it? Is this a complete classifier?

Google Developer Student Clubs

What is your intuition?

Vote now on your phones!

Stepping stones from LR to Classification

There are no right or wrong answers here, we're all learning :)

Go to slido.com and use the code
3597601

slido

Our current loss function is Mean Squared Error. Do you think that should change for Binary Classification?

ⓘ Start presenting to display the poll results on this slide.

slido

Do you think it's easier to work with (threshold) any value in R or a smaller range say [0,1]?

ⓘ Start presenting to display the poll results on this slide.

Logistic Regression to the rescue!

Wait, what? Logistic “Regression” is a classifier? 👀

First, let's look at the questions again

`Remember Slido?` 🌚

- Keeping the thresholding range as **R** might come with a major unintended consequence - outliers would play a much bigger scale.

- Arithmetic errors might come into the picture, too.

Logistic Regression

The essence of the approach

- Much like Linear Regression, we learn a line.
- We feed this line into some function to get a value between 0 and 1.
- We then threshold this value to a particular class (0 or 1).

About the Loss Function

This is where things get interesting.

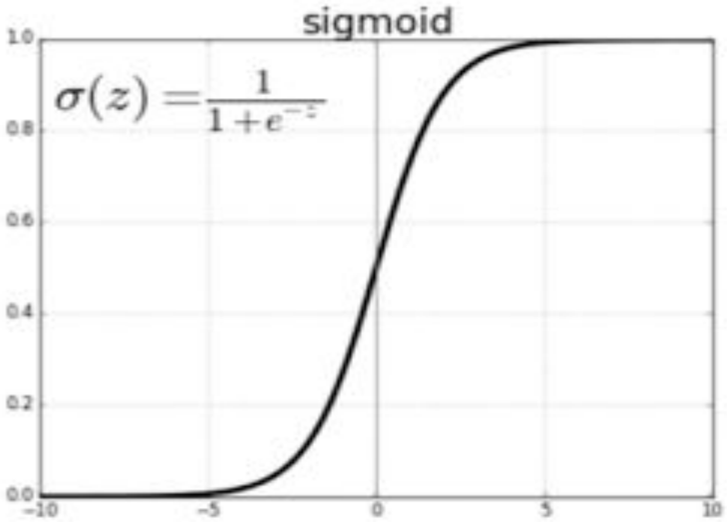
While RMSE is definitely valid, there's a far more effective Loss function for Binary Classification: **The Log-Loss function**

$$J = -\sum_{i=1}^{N} y_i \log(h_\theta(x_i)) + (1 - y_i) \log(1 - h_\theta(x_i))$$

h is the thresholded value
based on theta

How do we generate h(θ) though?

How does one convert a real number into a range from [0,1]?



Wow

That’s a lot of maths.

Logistic Regression - Summary

`a.k.a "What just happened for the last 15 minutes?"`

- We learn a line and threshold it using the **Sigmoid function**

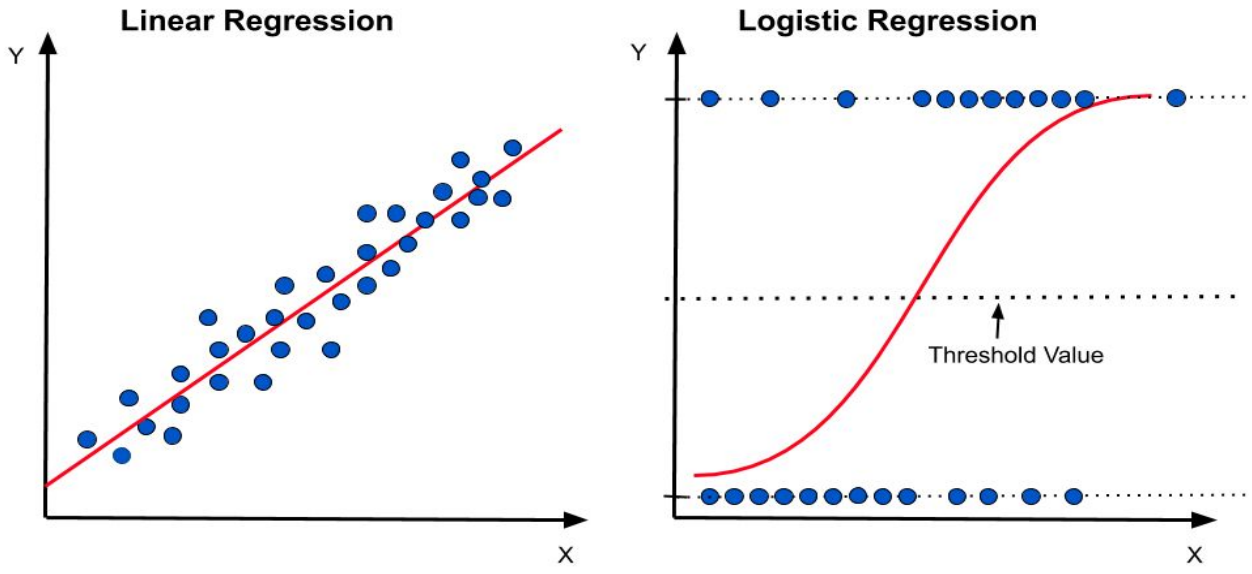
- After every iteration, the loss corresponding to the current parameters are calculated through **Log loss**.

- The parameters of the line are learnt through a variation of Gradient Descent similar to Linear Regression, but the gradients will change owing to the new loss function.

- At the end of training, the output of the function will be 0/1 - a **discrete class label**!

Logistic Regression - Summary

Comparison with Linear Regression



Let’s build a simple LogReg Classifier!

Please make a copy of the ipynb file and let’s get started!

https://colab.research.google.com/drive/1FpeoFaWKUcq7OeacjyIq5nJdVuLBo6vm?usp=sharing

Cool! We learnt about a few classifiers - what next?

Let’s look at some practical things to take care of.

PSA

From now on, consider your classifier to be a “black box”.

Dealing with multiple classes

`More than just 0/1`

- One vs All method
  - Make k classifiers - one for each class
  - Most confident class

- Some classifiers have support for multiple classes
  - More complex loss functions and prediction functions
  - Look into Multinomial Log Loss

Dealing with Class Imbalance

`Relative frequency of data points`

- Oversampling minority class
  - Example: SMOTE
- Undersampling majority class
- Passing class_weights as a parameter to the model

Dealing with Class Imbalance

How about testing the predictions?

This is a loaded question.

Scenario

Fraud Detection Problem - ~ 200 features, ~120000 rows.

Binary Classification - 0 for no fraud and 1 for fraud

A “model” trained in under 2 minutes and had a 95% accuracy!

Can you guess what it was?

```python
"""
    Crazy Model
        Trained under 2 minutes and gave 95% accuracy
"""
def get_predictions(X_test):
    # What do you think goes here?
```

```python
"""
    Crazy Model
        Trained under 2 minutes and gave 95% accuracy
"""
def get_predictions(X_test):
    return [0] * len(X_test)
```

It just returned "Not fraud" for all samples!

Dealing with Class Imbalance

`Sometimes, accuracy is not enough`

We need to account for class frequencies too. A good tester is the **F1 score** metric that weights false positives and false negatives too!

Dealing with underfitting and overfitting

a.k.a “nah m8 not happening” and “suffering from success”

- Underfitting
    - Increasing training time
    - Making the hypothesis more complex
    - Making the data more streamlined

- Overfitting
    - Validation sets and validation testing
        - Cross-validation

There are many classifiers.

Logistic regression
Naive Bayes
KNN
SVM

The list goes on and on…

So, how would you choose the best classifier for your problem?

Try them out one-by-one and choose the best classifier based on a testing metric.

Can we do better?

`The PTSD associated with this statement…`

Should we only stick to **one** model?

What if we can combine models? Won't that be good?

Ensemble Methods

Combining multiple models

- Bagging
  - Take m models
  - Find out the predictions of each model
  - Take the most frequent prediction
  - Does this sound familiar? 🌚

- There's also boosting and stacking, which is left as an exercise to the reader 🌚

Yeet

That’s a lot of things to take care of.

Let’s take a breather and do something fun

Please sit in pairs.

Teachable Machine

Note: You need to enable your webcams for this.

https://teachablemachine.withgoogle.com/train/image

Teachable Machine

How do they do it?

- Very complex neural networks!

- Transfer learning

Does that sound interesting to you?

`Please say yes :)`

Find out more about these topics in our next session!

Session #4: **Big-brain time with neural networks**

Thank you!

If you’d like to post about the event,
please tag @gdsc_iiitb

Any questions?