# Lecture 3:

## Machine Learning Essentials

Book chapters 3.5.1, 4.1 to 4.5, 4.7 to 4.10 (Excluding 4.9.1)

### Classification:

- The task of assigning input data to a specific category or label.

- input data (aka object) has features or properties
- Classifier makes prediction based on these feature and properties
- Good classifiers consider many properties jointly

### Formal Definitions:

A classifier, $h \in O \rightarrow$ is a function that maps an input object $o \in O$, to an output category, $y \in Y$

Example: BRACU Graduate Classifier where

$O$ = All students of BRACU

$Y$ = {will graduate, won't graduate}

Here, the features of each students such as GPA, Program, Complete credit etc are denoted as $f_i$ aka feature function.

eg: $f_{CGPA}$, $f_{program}$ ....

All of these feature functions forms a vector called feature vector.

↳ $x = [f_1(o), f_2(o), \ldots, f_m(o)]$

- Two main type of classifier –

① Generative classifier: focuses on how input data is generated by learning the joint probability $P(X, Y)$

ex: Naive Bayes

② Discriminative Classifiers:

focuses on finding the boundary between classes by learning $P(Y|X)$ directly.

ex: Logistic Regression

## Probability Review:

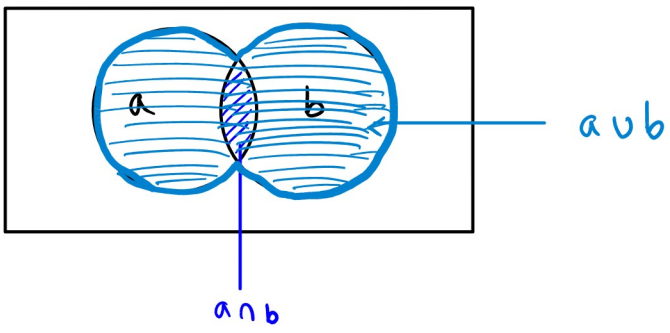
$P(a)$ is our degree of belief in a proposition $a$

- $P(\text{Dhaka} = \text{rainy}) = 0.75$

means there is 75% chance of rain in Dhaka.

Rules: $0 \leq P(a) \leq 1$ $P(\text{true}) = 1$ $P(\text{false}) = 0$

$$P(a \cup b) = P(a) + P(b) - P(a \cap b)$$



## Prior Probability:

* Initial probability of an event before seeing any new evidence.
* Unconditional.

eg: $P(\text{Dhaka} = \text{Rain}) = 0.75$

Here, we are not considering any other conditions like humidity, temp etc.

## Joint probability:

* The probability of two or more events happening together,

eg: $P(A, \text{spade}) = 1/52$

## Conditional Probability:

- It is the probability of one event given that another event has already occured. $P(A|B)$

$$P(\text{Card} = A \text{ spade} \mid \text{Cardsuit} = \text{spade}) = \frac{1}{13}$$

## Relation between joint & conditional:

Product Rule:

$$P(A \cap B) = P(A|B)\, P(B)$$

$$\Rightarrow P(A|B) = \frac{P(A \cap B)}{P(B)}$$

# Bayes Rule:

**Purpose:** Swap the conditioned & conditioning variables

**Derivation:**

From product rule,

$$P(A \cap B) = P(A|B)\,P(B) \quad \text{——— (1)}$$

$$\Rightarrow P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Now, swapping A and B,

$$\Rightarrow P(B|A) = \frac{P(B \cap A)}{P(A)}$$

from (1),

$$\Rightarrow \boxed{P(B|A) = \frac{P(A|B)\,P(B)}{P(A)}} \leftarrow \text{Bayes' Rule}$$

It tells us how likely an email is spam after seeing certain words.

In Bayes theorem, all features/words are dependent with each other.

But in Naive Bayes, All feature/words are independent to each other given the class label.

↳ Naive Bayes Assumption

Before moving on to Naive Bayes Classifier, we will see a technique to convert text documents into feature vector.

## Bag of Word (BOW)

- An unorder set of word with their position ignored, keeping only their frequency in the documents.

| | Documents | Features | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $f_{and}$ | $f_{boxing}$ | $f_{film}$ | $f_{great}$ | $f_{plot}$ | $f_{satire}$ | $f_{scenes}$ | $f_{twists}$ | $f_{worst}$ |
| document 1 → | *worst boxing scenes* | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| document 2 → | *satire and great plot twists* | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| document 3 → | *great scenes great film* | 0 | 0 | 1 | 2 | 0 | 0 | 1 | 0 | 0 |

→ We create a set of all possible unique data, W. [We call this Vocabulary]

→ For each document we create a vector v of size |W| [length of vocabulary] with every element initialized to 0.

→ If a word i appear in the sentence, replace 0 with the frequency of that word in that sentence/ documents.

## Issues of BOW:

- Only one feature function per word.
- Completely ignores word order
- large & sparse
- Useful for simple classification tasks.

We will learn more about word representation techniques later.

# Naive Bayes Classifier:

[Ch 4.1, 4.2, 4.3]

→ A probabilistic classifier that assumes that the feature are independent given the class.

$$h(o) = \underset{y \in Y}{\operatorname{argmax}}\ P(y) \prod_{i=1}^{m} P(f_i(o) \mid y)$$

# Basically, if a email contains money lottery win, and classes are spam, not spam we will find,

$$\underset{y \in \{\text{spam}, \text{not spam}\}}{\operatorname{argmax}}\ P(y) \prod_{i=1}^{m} P(f_i(o) \mid y)$$

Prior probability; Conditional probabilities

or, $P(\text{spam} \mid \text{money}, \text{lottery}, \text{win}) \propto P(\text{spam}) \cdot P(\text{money} \mid \text{spam}) \cdot P(\text{lottery} \mid \text{spam}) \cdot P(\text{win} \mid \text{spam}) = A$

$P(\text{not spam} \mid \text{money}, \text{lottery}, \text{win}) \propto P(\text{not spam}) \cdot P(\text{money} \mid \text{not spam}) \cdot P(\text{lottery} \mid \text{not spam}) \cdot P(\text{win} \mid \text{not spam})$

$= B$

The classifier will predict spam or not spam based on the maximum value between A, B.

Now, how do we calculate $P(\text{word} \mid \text{class})$ ?

We can find this by,

$$P(\text{word} \mid \text{class}) = \frac{\text{count of 'word' in that class}}{\text{count of all word in that class}}$$

eg, $P(\text{money} \mid \text{spam}) = \dfrac{\text{count of 'money' in the spam class}}{\text{count of all word in the spam class}}$

But there is a problem, what if the training data doesn't have 'money' word in spam class?

Then, $P(\text{money} \mid \text{spam}) = 0$

This will cause

$$P(\text{spam} \mid \text{money}, \text{lottery}, \text{win}) \propto P(\text{spam}) \cdot \underbrace{P(\text{money} \mid \text{spam})}_{0} \cdot P(\text{lottery} \mid \text{spam}) P(\text{win} \mid \text{spam}) = 0$$

To solve this we use smoothing.

Simplest one is add-one smoothing. (aka Laplace smoothing)

[Book Ch. 3.5.1]

$$P(\text{word} \mid \text{class}) = \frac{\text{count of 'word' in that class} + 1}{\text{count of all word in that class} + |V|}$$

$|V|$ : len(vocabulary)

So, The train phase of the naive bayes classifier consists of finding prior probability $P(\text{class})$ and conditional probabilities $P(\text{word} \mid \text{class})$ from the training data.

The test phase / inference means multiplying the probabilities of a test data.

eg:

$$P(\text{spam} \mid \text{money}, \text{lottery}, \text{win}) \propto P(\text{spam}) \cdot P(\text{money} \mid \text{spam}) \cdot P(\text{lottery} \mid \text{spam}) P(\text{win} \mid \text{spam})$$

$$P(\text{not spam} \mid \text{money}, \text{lottery}, \text{win}) \propto P(\text{not spam}) \cdot P(\text{money} \mid \text{not spam}) \cdot P(\text{lottery} \mid \text{not spam}) \cdot P(\text{win} \mid \text{not spam})$$

# Worked Example:

[Book Ch 4.3]

|V| = 12

| Training data: | $f_{-}$ | $f_{!!}$ | $f_{!}$ | $f_{Sorry}$ | $f_{U}$ | $f_{call}$ | $f_{can}$ | $f_{have}$ | $f_{later}$ | $f_{me}$ | $f_{now}$ | $f_{won}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| + *Sorry I'll call later* | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| + *U can call me now* | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| – *U have won call now!!* | 2 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| Testing data: | | | | | | | | | | | | |
| *Sorry! U can not unsubscribe* | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

5+5 = 10 words in positive class

$P(+|o) \propto P(+)P(f_!|+)P(f_{Sorry}|+)P(f_U|+)P(f_{can}|+)$

$P(-|o) \propto P(-)P(f_!|-)P(f_{Sorry}|-)P(f_U|-)P(f_{can}|-)$

$$P(word|class) = \frac{\text{count of 'word' in that class} + 1}{\text{count of all word in that class} + |V|}$$

$$P(+) = \frac{2}{3} \qquad P(!|+) = \frac{0+1}{10+12} = \frac{1}{22}$$

$$P(sorry|+) = \frac{1+1}{10+12} = \frac{2}{22}$$

$$P(U|+) = \frac{1+1}{10+12} = \frac{2}{22}$$

$$P(can|+) = \frac{1+1}{10+12} = \frac{2}{22}$$

$$\therefore P(+|o) \propto \frac{2}{3} * \frac{1}{22} * \frac{2}{22} * \frac{2}{22} * \frac{2}{22} = 0.000023$$

Similarly,

| Training data: | $f_{-}$ | $f_{!!}$ | $f_{!}$ | $f_{Sorry}$ | $f_{U}$ | $f_{call}$ | $f_{can}$ | $f_{have}$ | $f_{later}$ | $f_{me}$ | $f_{now}$ | $f_{won}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| + *Sorry I'll call later* | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| + *U can call me now* | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| – *U have won call now!!* | 2 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| Testing data: | | | | | | | | | | | | |
| *Sorry! U can not unsubscribe* | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

7 words in negative class

$P(+|o) \propto P(+)P(f_!|+)P(f_{Sorry}|+)P(f_U|+)P(f_{can}|+)$

$P(-|o) \propto P(-)P(f_!|-)P(f_{Sorry}|-)P(f_U|-)P(f_{can}|-)$

$$P(-) = \frac{1}{3} \qquad P(!|-) = \frac{2+1}{7+12} = \frac{3}{19}$$

$$P(sorry|-) = \frac{0+1}{7+12} = \frac{1}{19}$$

$$P(U|-) = \frac{1+1}{7+12} = \frac{1}{19}$$

$$P(can|-) = \frac{0+1}{7+12} = \frac{1}{19}$$

$$\therefore P(-|o) \propto \frac{1}{3} * \frac{3}{19} * \frac{1}{19} * \frac{1}{19} = 0.000015$$

$\therefore P(+|o)$ is higher. So, predicted class = +

# More examples are given in the book and drive.

Think. what if we get new words that was not present in the vocabulary ?

eg: 'not', 'unsubscribe' in the previous example ?

Bag of word model doesn't recognize the new word. So, we ignore them.

## Refining feature :

Optimizing for sentiment analysis [Ch 4.4]

① Binarization :

→ Instead of using frequency, we only focus if the word is present or not.

→ Equivalent of removing duplicate words in each document

| $f_i$ | $f_{!!}$ | $f_!$ | $f_{Sorry}$ |
|---|---|---|---|
| 0 | 1 | 1 | 3 |
| 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |

⇒

| $f_i$ | $f_{!!}$ | $f_!$ | $f_{Sorry}$ |
|---|---|---|---|
| 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |

② Lexicon features :

→ In some situation, we maynot have enough training data

→ In such case we can use lexicons.

→ Lexicon is a pre-annotated list of word for specific task.

Example: LIWC

» Lexicons are typically used to produce count feature.

$f_{positive\ emotion}$ (great scenes beautiful film) = 2

when great and beautiful are in positive emotion lexicon.

③ Rule based features:

Examples:

$f_{capital}$ = how many capital letter in the text

$f_{puncfreq}$ = how many punctuation in the text

⋮

There is no limit

* ④ N-gram : [Details Ch 3.1]

- An n-gram is a sequence of n words.
- Instead of one word at a time, we consider n word.
- Unigram = 1-gram = 1 word [the BoW we saw previously is unigram]
- Bigram = 2-gram = 2 words
- Trigram = 3-gram = 3 words

Example: This movie is very good.

1-gram : (This) (movie) (is) (very) (good) (.)

2-gram : (This, movie) (movie, is) (is, very) (very, good) (good, .)

3-gram : (This, movie, is) (movie, is, very) (is, very, good) (very, good, .)

- A bag of word is basically bag of 1-grams. (unigrams)

Ex: $f_{great}$ (great scene great movie) = 2

- A bag of n-gram feature representation has a vocabulary that includes phrases (up to n-tokens long)

$f_{great,\ movie}$ (great scene great movie) = 1

  - Even though N-grams can capture context a bit better, it has consequences.
    - Vocabulary size increases rapidly for n increase
    - small counts as most feature n-grams has a frequency of 0 for large n.

  - Usually we use,
    $n \leq 3$ for word n-grams
    $n \leq 10$ for character n-grams

# Probabilistic Learners are simple. Now we will discuss about a simple Discriminative Classifier, Logistic Regression.

# Logistic Regression [ch 5]

→ A classification Algorithm that predicts the probability of a sample belonging to a class (usually 0 or 1)

Equation:

$$z = w_1 x_1 + w_2 x_2 + \dots + b$$

$$\hat{y} = \sigma(z) = \frac{1}{1+e^{-z}}$$

Here, $x = [x_1, x_2 \dots]$ is input vector [example BoW representation]

$w$ is the weights [learned from traing phase and data]

$b$ = bias [" " " " " "]

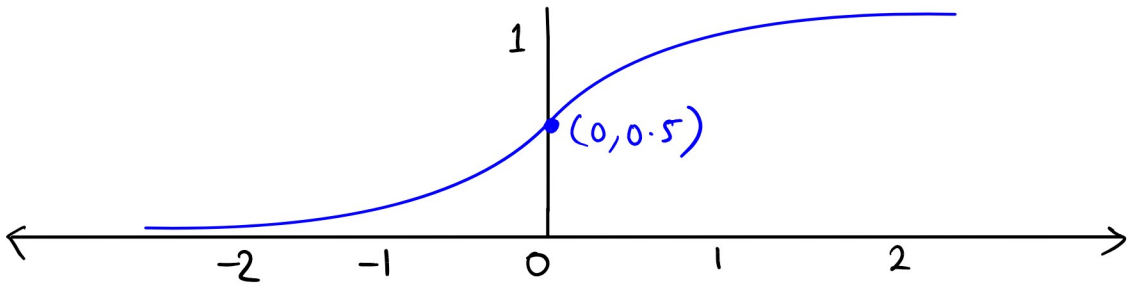
$\hat{y}$ = predicted probability

$\sigma(z)$ = Sigmoid function [maps $z$ into range $(0,1)$]

Decision Rule, if $\hat{y} \geq 0.5$ predicted class = 1

else predicted class = 0

Sigmoid function plot



## Worked Example:

* Use logistic Regression to classify the following reviews using BOW.

Vocabulary = { very ($x_0$), good ($x_1$), movie ($x_2$), not ($x_3$), bad ($x_4$) }

Weights $[w_0, w_1, w_2, w_3, w_4] = [0.3, 1.5, 1, -1.5, -2]$ $\quad b = -0.5$

Reviews: ① Not that bad
② Very good

class = 1 – positive
0 – Negative

Soln: ① Not that bad vector = $[\overset{x_0}{0}, \overset{x_1}{0}, \overset{x_2}{0}, \overset{x_3}{1}, \overset{x_4}{1}]$ [ignoring unknown words in BOW]

We know, $z = w_0 x_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + b$

$= 0.3 * 0 + 1.5 * 0 + 1 * 0 + (-1.5) * 1 + (-2) * 1 - 0.5$

$= -4$

$\therefore \hat{y} = \sigma(z) = \frac{1}{1 - e^{-(-4)}} = 0.018 < 0.5$

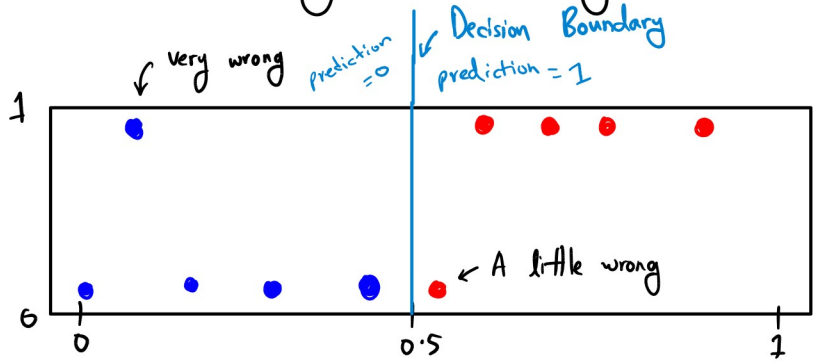
$\therefore$ Predicted class = 0 = Negative

② Very Good vector = $[1, 1, 0, 0, 0]$

Try it yourself.

From the above example, we can see 'not bad' is classified as negative. In everyday life, the learner we use make mistakes & Learn from mistake.
How do we calculate how much mistake a model is making?

We use Loss. It says how wrong is a classifier?




We can find easily how many error the classifier made easily.
Here,

$$\text{classification error} = \frac{2}{10}$$

← Wrong prediction
← Total datapoint

But all errors are not equal.

- Error near decision boundary are 'less bad.'

## Cross Entropy Loss

[Details ch 5.5]

A good model,

Prediction $\hat{y}$, Actual label $y$

gives high $\hat{y}$ when $y = 1$
gives low $\hat{y}$ when $y = 0$

Cross entropy loss takes negative log to penalize wrong classification

Equation:

$$\text{Loss}_{CE} = -[\, y \log(\hat{y}) + (1-y) \log(1-\hat{y}) \,]$$

This function measures how bad the parameters $(w, b)$ are on an single example $(x, y)$

- A model runs through each training examples and update its w and b accordingly. [using Gradient Descent or other techniques]
- Going through all training samples once is called an epoch.