Naïve Bayes



Coneixement, Raonament i Incertesa

Enginyeria Informàtica 2021-2022

Martí Caixal Joaniquet - 1563587

Índex

[Introducció 3](#_Toc90738951)

[Dades 3](#_Toc90738952)

[Disseny i implementació 3](#_Toc90738953)

[Funció “Fit” 4](#_Toc90738954)

[Funció “Predict” 4](#_Toc90738955)

[Execució 5](#_Toc90738956)

[Mètrica i anàlisi 6](#_Toc90738957)

[Diferents mides de Train i diccionaris 7](#_Toc90738958)

[Laplace Smoothing 8](#_Toc90738959)

# Introducció

La pràctica tracta de dissenyar, programar i testejar un aprenentatge bayesià. Més concreatament, s’ha d’utilitzar l’algorisme “Naïve Bayes” per poder classificar un tweet entre positiu o negatius. Està dividida en 3 aparts:

1. Disseny i implementació de l’algorisme.
2. Modificar mides de diccionari i conjunt d’aprenentatge.
3. Efecte del Laplace Smoothing

# Dades

Les dades amb les que es treballa són un conjunt de tweets emmagatzemats a un arxiu csv. Es tracta de més d’un milió de mostres, cada una d’elles en el següent format:

*16; I fell in love again; 02/12/2015; 1*

El 16 és un identificador, la segona columna és el tweet en qüestió, la tercera la data de publicació i per últim l’etiqueta classificadora (0 si és negatiu, 1 si és positiu).

Estem davant d’un conjunt de dades balancejat, significant que es té el mateix nombre de mostres d’una classe que de l’altra.

# Disseny i implementació

Per poder implementar l’algorisme es fa ús d’un seguit de variables:

* wc (diccionari): Per cada classe, emmagatzema el número de vegades que una paraula apareix. Es tracta, doncs, de diccionaris anidats. A continuació es motra un exemple del format

wc **=** **{**'positive'**:** **{** 'hello' **:** 4**,**

'car' **:** 7**,**

'pencil' **:** 5

**},**

'negative'**:** **{**'window'**:** 2**,**

'paper' **:** 10

**}**

**}**

* dictionary (set): emmagatzema cada paraula que es troba durant l’entrenament. Al tractar-se d’un set, ens assegurem que totes les paraules són úniques i no es repeteixen.
* log\_prior\_probability (diccionari): Guarda la probabilitat a priori que un tweet sigui positiu o negatiu.
* tweet\_num (diccionari): Guarda el nombre de tweets que hi ha positius i negatius.
* laplace\_smoothing (float): Conté el valor del Laplace Smoothing que s’utilitza durant la fase de predicció.

Tot i que moltes variables podrien ser simples llistes, el fet d’utilitzar diccionaris va bé per poder accedir directament al valor que es necessita.

L’algorisme està implementat dins d’una classe anomenada “NaiveBayes”. Les principals funcions que conté són “Fit” i “Predict”. La classe hereda de “sklearn.base.BaseEstimator” per tal de poder utilitzar el “cross\_validate” de “Sklearn”.

### Funció “Fit”

És la funció utilitzada durant l’entrenament del model. Primer omple les variables “tweet\_num” i “log\_prior\_probability”.

Seguidament fa els següents passos per cada tweet:

1. Separar tweet per paraules.
2. Mirar la classe a la que pertany el tweet.
3. Per cada paraula:
   1. Afegir-la al set “dictionary”.
   2. Incrementar el nombre d’aparicions per classe al diccionari “wc”.

### Funció “Predict”

És la funció utilitzada durant les prediccions de les mostres. Per cada mostra fa els següents passos:

1. Separar tweet per paraules.
2. Posar variables positive\_count i negative\_count a 0.
3. Per cada paraula:
4. Si no apareix al diccionari, continuar a la següent.
5. Calcular p(wi | ‘positive’) i p(wi | ‘negative’).
6. Sumar aquests valors a positive\_count i negative\_count
7. Afegir la probabilitat apriori de cada classe a positive\_count i negative\_count
8. Guardar com a resultat el la classe amb el valor més gran entre positive\_count i negative\_count

# Execució

# Mètrica i anàlisi

# Diferents mides de Train i diccionaris

# Laplace Smoothing