2nd

Naive Bayes :→ classification algorithm
↳ probability - based

KNN → neighborhood based classification.

Conditional Probability = $(P(A/B)) = Pr(A = a / B = b)$

$A = a$ ↓ value that A takes

$B = b$ ↓ value that B takes.

Always read equations in english.

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

* Independent Events & mutually Exclusive events :→

A, B are said to be independent.

$P(A/B) = P(A)$

$P(B/A) = P(B)$

, for ex :→
A: getting value of 6 in die 1 throw $(D_1 = 6)$
B: getting a value of 3 in die 2's throw $(D_2 = 3)$

A, B are said to be mutually exclusive if.

$P(A/B) = P(B/A) = 0$

$\frac{P(A \cap B)}{P(B)}$ $\frac{P(B \cap A)}{P(A)}$, so, $P(A \cap B)$ should be 0

as $A \cap B = B \cap B$.

For ex :→ probability of getting 3 in die 1 is 0 if die 2 is getting 6 in it.

Scanned by TapScanner

imp

(1700's)

Bayes Theorem :→

$$P(A/B) = \frac{\overbrace{P(B/A)}^{\text{likelihood}} \cdot \overbrace{P(A)}^{\text{prior}}}{\underbrace{P(B)}_{\text{evidence}}} \quad \text{if } P(B) \neq 0.$$

posterior probability

Proof of Bayes theorem :→

$$P(A/B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A,B)}{P(B)}$$

also means $\cap$

In set theory $A \cap B = B \cap A$

$$P(A/B) = \frac{P(B \cap A)}{P(B)} = \frac{P(B,A)}{P(B)} \quad - (1)$$

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

$$P(B \cap A) = P(B/A) \cdot P(A) \quad (2)$$

Put ~~(2)~~ $P(B \cap A)$ in (1) eq

$$P(A/B) = \frac{P(B/A) \cdot P(A)}{P(B)}$$

* Naive Bayes Algorithm

↓

unsophistic

↓

simplistic

$$P(c/\underbrace{f_1, f_2, f_3, f_4}_{x_q}) = P(f_1/c) * P(f_2/c) * P(f_3/c) * P(f_4/c) * P(c)$$

Naive bayes is much better in terms of space complexity at runtime

$O(d*c) \rightarrow$ naive bayes

$O(n*d) \rightarrow$ KNN

memory efficient at run-time

* Naive Bayes on text data 6→

↓

first technique used in case of spam filters → email → spam / not spam

review → +ve / → -ve

Text classification → Naive Bayes is a simple used method.

← text1 →	0
← text2 →	1
	0
	1
	⋮

Task

① ← $P(y_q=1 / text_q)$ → compute these

② ← $P(y_q = 0 / text_q)$

So if ① > ②

then we can say that $y_q = 1$ for given $text_q$.

Preprocessing :- text ⟶ stopwords
Stemming
n-grams
$\hookrightarrow \{w, w_2, w_3, \dots w_d\}$
$\downarrow$
Binary BoW

text $\xrightarrow{\text{pre-pro}} \underbrace{\{w_1, w_2, \dots w_d\}}_{\text{set of words.}}$

$$P(y_1 = 1 / \text{text}) = P(y_1 = 1 / w_1, \underbrace{w_2, w_3, \dots, w_d}_{\text{features}})$$

class prior

likelihoods

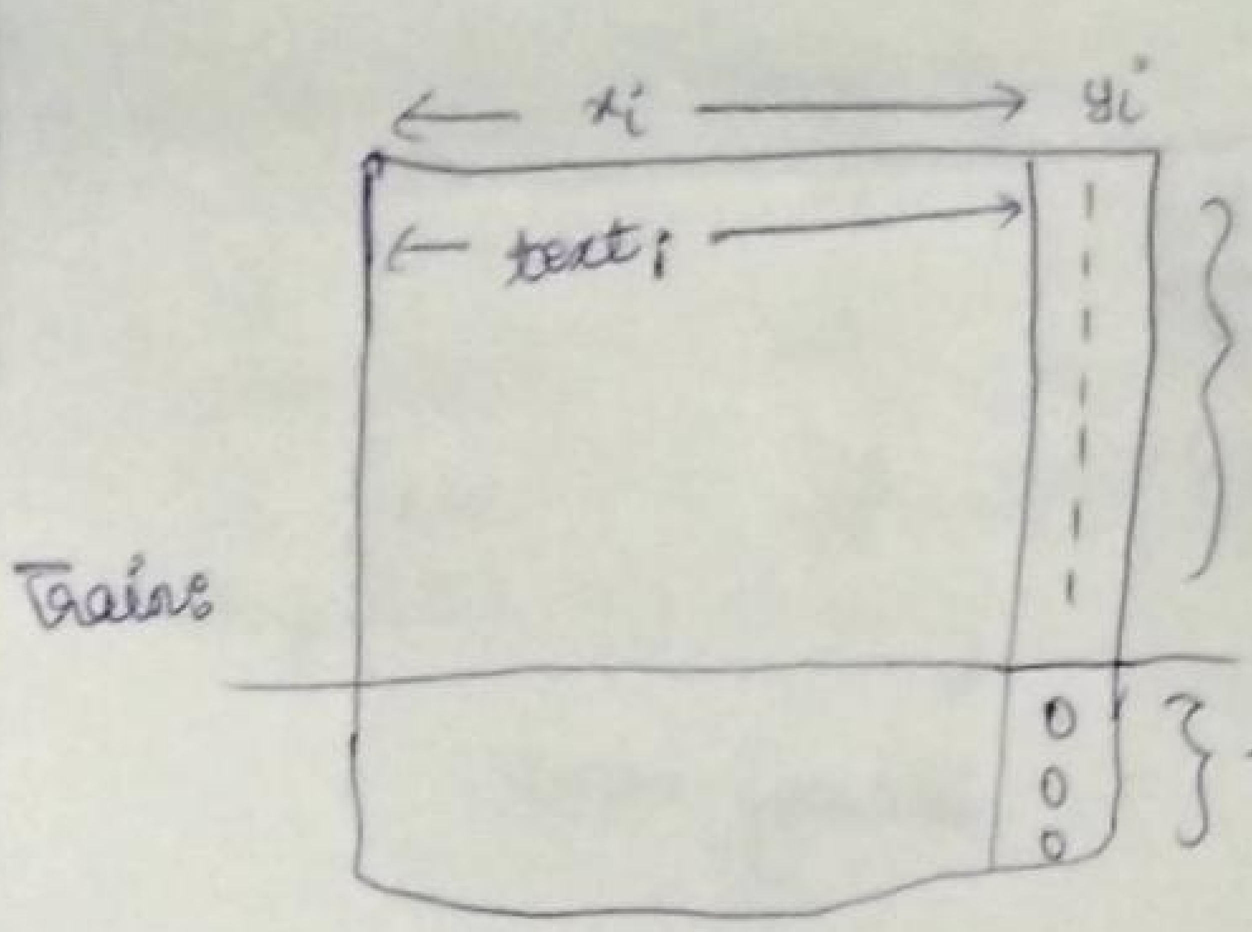
$$= P(y=1) * P(w_1/y=1) * P(w_2/y=1) \dots P(w_d/y=1)$$

$$P(y_1 = 1/\text{text}) \propto P(y=1) * \prod_{i=1}^{d} P(w_i/y=1)$$

$$P(y = 0/\text{text}) \propto P(y=0) * \prod_{i=1}^{d} P(w_i/y=0)$$

$$P(y=1) = \frac{\text{no. of train points with } y=1}{\text{Total no. of train points.}}$$

$$P(y=0) = \frac{\text{no. of train points with } y=0}{\text{Total no. of train points.}}$$



} Train data with $y=1$

} Train data with $y=0$

$$P(w_i | y=1) = \frac{\text{no. of Train data points which contain } w_i \text{ \&\& } y=1}{\text{no. of Train data points with } y=1}$$

Text - classification problems.

{ spam - detection
  Polarity of a review } Naive Bayes is a very good Baseline $\rightarrow$ benchmark

not laplacian smoothing

* Laplace - Additive Smoothing :-

After training $\rightarrow$ all this data is already computed.

$$\left\{\begin{array}{ll} P(y=1) ; P(y=0) & \leftarrow \text{class priors} \\ P(w_1 | y=1) & P(w_1 | y=0) \\ P(w_2 | y=1) & P(w_2 | y=0) \\ \vdots & \\ P(w_m | y=1) & P(w_m | y=0) \end{array}\right\} \rightarrow \text{likelihoods}$$

Test:→ $text_q = (w_1, w_2, w_3, w')$

training data.

very often ← * w' is not present in $\{w_1, w_2, w_3, \ldots w_m\}$

$$P(1/text_q) = P(y=1/w_1, w_2, w_3, w')$$

$$= P(y=1) * P(w_1/y=1) * P(w_2/y=1) * P(w_3/y=1) * P(w'/y=1)$$

how do you get this probability as w' is not present in training data.

ignoring or dropping it will mean $P(w'/y=1) = 1$ which is not correct.

We have to get values of $P(w'/y=1)$ and $P(w'/y=0)$

$$P(w'/y=1) = \frac{P(w', y=1)}{P(y=1)}$$

$$= \frac{\text{no. of train points such that } w' \text{ occurs in } y=1}{\text{no. of train points where } y=1}$$

$$= \frac{0}{n_1} = 0$$ → This is also dangerous as it will make whole probability to be 0.

Laplace smoothing or additive smoothing :-

$$P(f_1 = a / y = 1) = \frac{0 + \alpha}{n_1 + \alpha k}$$

$\alpha = 1$ → typically (not always)

k = no. of distinct values which f_1 can take

f_1 ⇒ feature

$$P(w' / y = 1) = \frac{0 + \alpha}{100 + 2\alpha}$$

$K = 2$ → because w' is 0 or 1.
present or not.

Let $n_1 = 100$

Case 1 :- $\alpha = 1$ = $\frac{1}{102} \neq 0$

α small, we are getting rid of multiplying all the probabilities with 0.

$P(w' / y = 1) \neq 0$

⇓ which implies

$P(y = 1 | text_q) \neq 0$.

So, its not 0 anymore

Case 2 :- $\alpha = 10000$ → when α is large

$$P(w' / y = 1) = \frac{0 + 10k}{100 + 2 \times 10k} = \frac{10k}{20100} \approx \frac{1}{2}$$

$$\boxed{P(w' / y = 1) = P(w' / y = 0) = \frac{1}{2}}$$

means equal probability of w' to be 0 or 1 because w' have only two possibilities (0 or 1)

Laplace smoothing :→

find this for all words

$$P(w_i/y=1) = \frac{(\text{no. of data points with } w_i \text{ \& } y=1) + \alpha}{(\text{no. of data points with } y=1) + \alpha k}$$

present in my training data

adding something to numerator & denominator, that's why it is called additive Smoothing

In this formula, as $\alpha \uparrow$,

$P(w_i/y=1) \rightarrow$ likelihood probability is moving to a uniform distribution.

if n_1 is small

num or den is small → less confidence in ratio

often times, $\alpha = 1 \rightsquigarrow$ add one smoothing

It is called smoothing because you are moving likelihood probability towards uniform distribution

* log-probabilities for numerical stability :→

$$\log\left(P(y=1/w_1, w_2, \ldots w_d)\right) = \log\left\{P(y=1) * \prod_{i=1}^{d} P(w_i/y=1)\right\}$$

$$\log\left(P(y=0/w_1, w_2, \ldots w_d)\right) = \log\left\{P(y=0) * \prod_{i=1}^{d} P(w_i/y=0)\right\}$$

$x \uparrow$; $\log x \uparrow$ → monotonic fxn.

Scanned by TapScanner

* Bias and Variance Tradeoff :→

Case 1 :→ $\alpha = 0$ → small change in D_{train} results in large change in the model. ⇒ high variance = overfitting.

Case-2 :→ α = very large = $\alpha = 10000$

$$P(w_i/y=1) = \frac{2+10000}{1000 + \underbrace{20000}_{K=2}} \approx \frac{1}{2}$$

↓
0 or 1

high bias ⇐ (Underfitting) like in case of KNN (K = n) case.
↓
based on $P(y=1) / P(y=0)$

* How to find the right α ?
⇒ using simple cross ~~fold~~ validation or 10-fold cross validation

* Feature Importance and Interpretability :→

words w_i's which have high values of $P(w_i/y=0)$ and $P(w_i/y=1)$ → important/meaningful words (features)

So, unlike KNN, we can directly find important features by using likelihoods in Naive Bayes.

Interpretability :→ We can give reason by saying :→

$x_q \longrightarrow y_q = 1$

as x_q contains words w_3, w_6, w_9 ... etc for which $P(w_3/y=1)$, ... are high.

* Imbalanced data :→

Impact on Priors

In case of Imbalanced data, because of class priors ($P(y=1)$ and $P(y=0)$) majority/ dominating class has an advantage.

Sol^n (1) upsampling or downsampling.

motive ⇒ $n_1 \simeq n_2$ and $P(y=1) = P(y=0) = \frac{1}{2}$

(2) drop $P(y=1)$ & $P(y=0)$ by putting
$P(y=1) = P(y=0) = 1$

(3) modified naive Bayes for class-imbalance
not often used

Impact on likelihood ratios

minimum (-ve)	maximum (+ve)
$P(w_i/y=1000)$ $\frac{2}{100} = 2\%$	$\frac{18}{900} = 2\%$
let's assume, $\alpha = 10$	$\frac{18+10}{900+20} = \frac{28}{920} = 3.04\%$
$10\% = \frac{12}{120} = \frac{2+10}{100+20}$	

minority class label is benifited which is not good.

Sol^n (1) unsampling or downsampling

(2) Hacks forex ?
α_{+ve}, α_{-ve}
~~two~~ αs

* Outliers :→ Both in training & test data

hack/solution :→

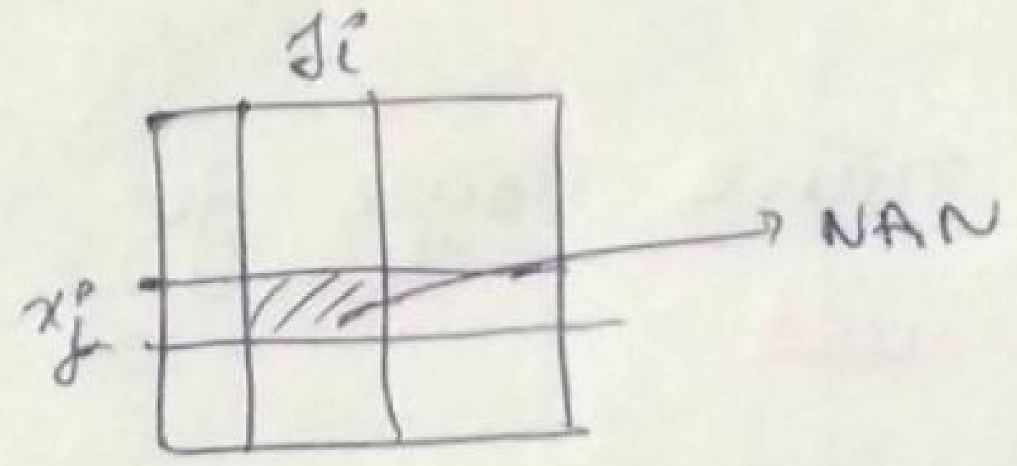
① If a word (w_j) occurs fewer than $\tau = 10$ times, then just ignore that word.

② Apply laplace smoothing to that word with a reasonable value of α.

* Missing values :→

① Text-data :→ no case of missing data, as a word either occurs or not.

② Categorical features :→



Hack :→ Consider NAN as a category too.

So, $f_i \in \{a_1, a_2, a_3, \ldots, NAN\}$

③ Numerical features :→ use of standard feature imputation methods like mean, median etc. based

Gausian Naive Bayes.

Naive Bayes with Numerical or realvalued features :→

assume f_j in $D' \longrightarrow$ Gaussian dist.

$\downarrow$

D^{+ve}_{train} $\qquad N(\mu_j^1, \sigma_j^1)$

assume f_j in $D'' \longrightarrow$ Gaussian dist

$\hookrightarrow D^{-ve}_{train}$

$N(\mu_j^0, \sigma_j^0)$

Scanned by TapScanner

This is also called Gaussian Naive Bayes. → likelihood probabilities

we take $P(w_i/y=1)$ & $P(w_i/y=0)$ values from Gaussian curve, thats why its name is this'

$P(w_1/y=1)$

0 1 → Bernoulli Naive Bayes.

another is → Multinomial Naive Bayes.

Naive Bayes is making an assumption of Conditional independence.

Multiclass Classification:→

Same as $P(y_i=0/w_1, w_2, \dots w_d)$

we can compute as many classes probability

like $P(y_i=2/w_1, w_2, \dots w_d)$

and compare them. → Naive Bayes is by default made for multiclass classification.

Similarity matrix or Distance matrix:→

Naive Bayes cannot use similarity or distance matrix.

as to compute probabilities (likelihood) we need actual features values for that.

* Large Dimensionality :-

Naive Bayes is easily applied on large dimensionality as in case of text classification.

Just use log-probabilities in place of normal probabilities to maintain numerical stability.

* Best and worst cases for Naive Bayes :-

① Naive Bayes perform very well with conditional independence assumption, but conditional independence ↓ NB degrades.

But when some features are dependent → NB works reasonably well.

+ve → good
→ great
→ Phenomenal
} → conditional dependent.

② Text Classification → email spam
→ review polarity { high-dim }.

↓

NB is baseline for these problems.

③ Categorical features :- NB is used ⩽ much.

④ Real-valued features :- NB is not often used.

⑤ Interpretable, feature Importance :-
→ medical domain

run-time complexity → low
Train time complexity - low
run-time space → low.

Scanned by TapScanner

NB → Simple to implement
↳ simple to understand.

(5) easily overfit if you don't do laplace smoothing.