Naive Bayes

- It is a classification Algorithm.
- It is a probability base Technique.
- k-nn: neighborhood based Technique.

Conditional probabity

$P(A/B) = P_r(A=a / B=b)$

- Variable, variable (A, B); value, value (a, b)
- A : random variable
- B : random variable

Two dies $D_1 + D_2$

D2 across the top, D1 down the side:

+	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

→ what is the probability that $D1 + D_2 \leq 5$?

$P(D_1 + D_2 \leq 5) = 10/36$

→ what is the probability that $D1 = 2$, given $d_1 + d_2 \leq 5$?

$P(D_1 = 2 \mid D_1 + D_2 \leq 5) = 3/10$

(given: $D_1 + D_2 \leq 5$; Condition on: $D_1 = 2$)

Let us assume

$D_1 = 2$ (is an Event A)

$D_1 + D_2 \leq 5$ (is an Event B)

$P(A|B) = \frac{P(A \cap B)}{P(B)}$ (definition of Conditional probability)

$P(A \cap B)$ = Probability of Event A has happened and also Event B also has happend.

This is Valid $P(B) \neq 0$

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{3/36}{10/36} = 3/10$$

Reference

https://en.wikipedia.org/wiki/Conditional_probability

23.2 Independent Vs mutually exclusive Events

Independent Events

A, B are said to be independent if $P(A|B) = P(A)$

$P(B|A) = P(B)$

A : getting Value of 6 in die 1's throw $(D_1 = 6)$

B : getting Value of 3 in die2's throw $(D_2 = 3)$

$P(D_1 = 6 | D_2 = 3) = P(D_1 = 6)$

Here throwing D_1 and D_2 are two independant Event. There is no point that when $D_2 = 3$, D_1 cannot not get 6

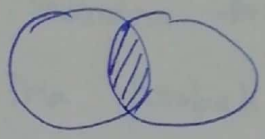
$P(D_2 = 3 | D_1 = 6) = P(D = 3)$

Mutually Exclusive Events

If $P(A/B) = P(B/A) = 0$ Then A & B are said to be mutually Exclusive Events

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

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



Event A : $D_1 = 6$

Event B : $D_1 = 3$

$$P(A/B) = P(D_1 = 6 / D_1 = 3)$$

As we already got 3 probability of getting 6 is zero

Bayes Theorem

→ Simple, Elegant, beautiful, useful theorem

→ Early 1700's

Thm:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)} \quad \text{if } P(B) \neq 0$$

Joint likelihood

prior

Posterior prob

Evidence & class.

Proof

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A, B)}{P(B)} \text{ (or) } \frac{P(A \cup B)}{P(B)}$$

$A \cap B = B \cap A$ → Set Theory

$$P(A|B) = \frac{P(B \cap A)}{P(B)} = \frac{P(B, A)}{P(B)} \rightarrow \text{Eqn } (1)$$

$$P(B|A) = \frac{P(B \cap A)}{P(A)} \Leftarrow \text{By deprod.}$$

$$P(A)\,P(B|A) = P(B \cap A) \rightarrow \text{Eqn } (2)$$

Replacing Equation (2) with Eq (1)

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

https://en.wikipedia.org/wiki/Bayes%27_theorem

Example

We have three machines and produce an output $m_1 = 0.2$, $m_2 = 0.3$, $m_3 = 0.5$ and the fraction of defective items produced is this

$d_1 = 0.05$, $d_2 = 0.03$, $d_3 = 0.01$

if an item is chosen at random from the total o/p and is found to be defective what is the probability that is produced by m_3

B = item is defective.

$P(B/A_1)$: Probability of an item is defective given it is made by m_1

$P(B/A_2)$: " made by m_2

$P(B/A_3)$: by m_3

$P(B/A_1) = 0.05$ $\quad P(A_1) = 0.2$

$P(B/A_2) = 0.03$ $\quad P(A_2) = 0.3$

$P(B/A_3) = 0.01$ $\quad P(A_3) = 0.5$

$$P(B) = P(B/A_1)P(A_1) + P(B/A_2)P(A_2) + P(B/A_3)P(A_3)$$

$$= (0.05)(0.2) + (0.03)(0.3) + (0.01)(0.5)$$

$$P(B) = 0.024$$

$P(B) = 2.4\%$ of all the products are defective.

$$P(A_3/B) = \frac{P(B/A_3)\,P(A_3)}{P(B)} = 5/24$$

↓ by m_3

what is the probility, given an item is defective what is probability that it is made

1. A total of 46% of the voters in a certain city classify themselves as independents, whereas 30% classify themselves as liberals & 24% say they are Conservative.

In recent local election 35% of the independents, 62% of the liberals and 58% of Conservative voted.

If a voter is choosen at random. Given that this person is in the local election, that probability that he or she is an independent is _____

P(I) : Probability of person being ~~an~~ independent voter

P(I) = 46%

P(L) = 30%

P(C) = 24%

⇒ And also given that 35% of independent are voted

$P(V/I) = 0.35$, $P(V/L) = 0.62$, $P(V/C) = 0.58$

$$P(I/V) = \frac{P(V/I)\,P(I)}{P(V)}$$

$$= \frac{0.35 \times 0.46}{P(V/I)P(I) + P(V/L)P(L) + P(V/C)P(C)}$$

$$\boxed{P(I/V) = 0.331}$$

Naive Bayes Algorithm

https://en.wikipedia.org/wiki/Naive_Bayes_classifier.

→ Abstractly, Naive Bayes is a Conditional probability

Naive means simplistic or unsophisticated

I have a datapoint x with n features

$x = (x_1, x_2, \ldots x_n)$

If have a k classes Each datapoint represent a k

$x \rightarrow 1, 2, 3, \ldots k$

$k \rightarrow C_k$

$P(C_k/x)$ = probability of class label given a point x, random variable has n features and hence n random variable.

$$P(C_k/x) = \frac{P(x/C_k)\ P(C_k)}{P(x)}$$

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{Evidence}}$$

Numerator

$$P(C_k)\ P(x/C_k) = P(x \cap C_k)$$

I have calculated

$P(C_1/x)$, $P(C_2/x)$ $P(C_3/x)$ $\ldots$ $P(C_k/x)$

↓

Let assign this to largest

$P(x)$

→ The denominator is same for all Condition, hence we can ignore them for a while.

→ Lets focus only on numerator.

$$P(C_k|x) = P(C_k)\ P(x/C_k) = P(C_k \cap x) = P(C_k, x)$$

$$P(C_k/x) \underset{\text{proportional to}}{\propto} P(C_k, x)$$

The numerator is called Joint probability model, C_k & x are jointly happening.

$$P(C_k, x_1, \cdots x_n)$$

→ There is something called as chain rule for Conditional probability

As $A \cap B = B \cap A$

$P(C_k, x_1 \cdots x_n) = P(x_1 \cdots x_n, C_k)$ ← Conditional probability not bayes theorem

$$= P(x_1 | x_2 \cdots x_n, C_k)\ P(x_2 \cdots x_n, C_k)$$

$$= P(x_1 | x_2 \cdots x_n, C_k)\ P(x_2 | x_3 \cdots x_n, C_k)\ P(x_3 \cdots x_n, C_k)$$

$$= \cdots\cdots$$

$$= P(x_1 | x_2 - x_n, C_k)\ P(x_2 | x_3 \cdots x_n, C_k) \cdots P(x_{n-1} | x_n, C_k)$$

$$P(x_n | C_k)\ P(C_k)$$

We want to Compute $P(C_k/x)$ for Each class and this is Equal to $P(C_k, x)$

class label
obese
fit
lean

Let take the first Condition.

$P(x_1 | x_2, x_3 \cdots x_n, C_k)$

$P(w=180 | \text{haircolor}=b, \text{hairlength}=5cm, \text{eyecolor}=br, C_k=\text{obies})$

This is very hard find there in the real data.

what naive bayes is lets make naive assumption of Conditional Independence.

Simple independance

$P(A|B) = P(A)$ A & B are two independent Events

$P(A|\underbrace{B, C}_{B \cap C}) = P(A|C)$ i.e., A is Conditionally independent of B given C

$P(x_i | x_{i+1} \cdots, x_n, C_k) = p(x_i | C_k)$

x_i is Conditionally independent of $x_{i+1}, x_{i+2} \cdots x_n$ given C_k

Naive : Conditionally independence of Each feature pairs

$X = (x_1, x_2 \cdots x_n)$ are features

$p(x_i | x_{i+1} \cdots, x_n, C_k) = p(x_i | C_k)$

$p(C_k | x_1 \cdots x_n) \propto p(C_k, x_1 \cdots x_n)$

$\propto p(x_1|C_k)\, p(x_2|C_k)\, p(x_3|C_k) \cdots p(x_n|C_k)\, p(C_k)$

$\propto p(C_k) \prod_{i=1}^{n} p(x_i|C_k)$

This means that under the above independence assumptions, the Conditional distribution over the class variable C is:

$p(C_k | x_1 \cdots x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^{n} p(x_i|C_k)$

where the Evidence $Z = p(x) = \sum_k p(C_k)\, p(x|C_k)$ is a scaling depending only on $x_1, \ldots, x_n$, that is a constant if the values of the feature variables are known.

Constructing a classifier from the probability model

The discussion so far has derived the independent feature model, that is, the naive Bayes probability model. The naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable. This is known as the maximum a posteriori or MAP decision rule.

The corresponding classifier, a Bayes classifier is the function that assigns a class label $\hat{y} = C_k$ for some k as follows:

$$\hat{y} = \underset{k \in \{1 \ldots K\}}{\operatorname{argmax}} \; p(C_k) \prod_{i=1}^{n} p(x_i|C_k)$$

Toy Example

shatterline.com/blog/2013/09/12/not-so-naive-classification-coin-the-naive-bayes-classifier/

	Outlook	Temperature	Humidity	Wind	class (play=yes, play=no)
Day1	Sunny	Hot	high	weak	no
Day2	Sunny	Hot	high	strong	no
Day3	overcast	Hot	high	weak	yes
Day4	Rain	mild	high	weak	yes
Day5	Rain	Cool	normal	Weak ~~strong~~	yes
Day6	Rain	Cool	normal	strong	no
Day7	overcast	Cool	normal	Strong ~~weak~~	yes
Day8	Sunny	mild	high	weak	no
Day9	Sunny	Cool	normal	weak	yes
Day10	Rain	mild	normal	Weak ~~Low~~	yes
Day11	Sunny	mild	normal	strong	yes
Day12	overcast	mild	high	strong	yes
Day13	overcast	Hot	normal	weak	yes
Day14	Rain	mild	high	Strong	no

$$P(c \mid \overbrace{f_1, f_2, f_3, f_4}^{x_q}) \propto \underbrace{P(f_1|c) * P(f_2|c) * P(f_3|c) * P(f_4|c)}_{\text{likelihoods}} * P(c)$$

$P(C=yes) = 9/14$

$P(C=no) = 5/14$

In the training phase

→ we done frequency counting

f_1 { $P(\text{outlook} = o \mid \text{class play} = yes/no)$

Outlook =	Frequency Play=yes	Play=no	Probability in class Play=yes	Play=no
Sunny	2	3	2/9	3/5
overcast	4	0	4/9	0/5
Rain	3	2	3/9	2/5
	9	5		

$P(\text{outlook} = \text{sunny} \mid C = yes) = \frac{2/14}{9/14} = 2/9$

$P(\text{outlook} = \text{sunny} \mid C = \text{no}) = \frac{3/14}{5/14} = 3/5$

Temperature	Frequency		Probability in class	
Hot	2	2	2/9	2/5
mild	4	2	4/9	2/5
cool	3	1	3/9	1/5
	9	5		

Humidity =	Play=yes	Play=NO	play: yes	play=NO
High	3	4	3/9	4/5
normal	6	1	6/9	1/5

wind				
strong ~~weak~~	3	3	3/9	3/5
weak	6	2	6/9	2/5

we also calculate $P(\text{class}_{play=yes})$ & $P(\text{class}_{play=NO})$

$P(\text{class}_{play=yes})$ & $P(\text{class}_{play=NO})$

Play	Frequency		Probability in class	
	Play=yes	Play=NO	Play=yes	play=NO
	9	5	9/14	5/14

Classification Phase

Let's say, we get a new instance of the weather condition,

x' = (outlook = Sunny, Temperature : Cool, Humidity = High, Wind = Strong) that will have to be classified

(i.e., are we going to play tennis under the condition specified by x')

with MAP rule, we compute the posterior probabilities. This is easily done by looking up the tables we built in the learning phase

$P(\text{Class}_{\text{play=yes}} \mid x') \propto [P(\text{Sunny} \mid \text{play=yes}) * P(\text{Cool} \mid \text{play=yes}) *$

$P(\text{High} \mid \text{play=yes}) * P(\text{Strong} \mid \text{class}_{\text{play=yes}})] *$

$P(\text{class}_{\text{play=yes}})$

$= 2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14$

$= 0.0053$

$P(\text{Class}_{\text{play=no}} \mid x') \propto [P(\text{Sunny} \mid \text{no}) * P(\text{Cool} \mid \text{no}) *$

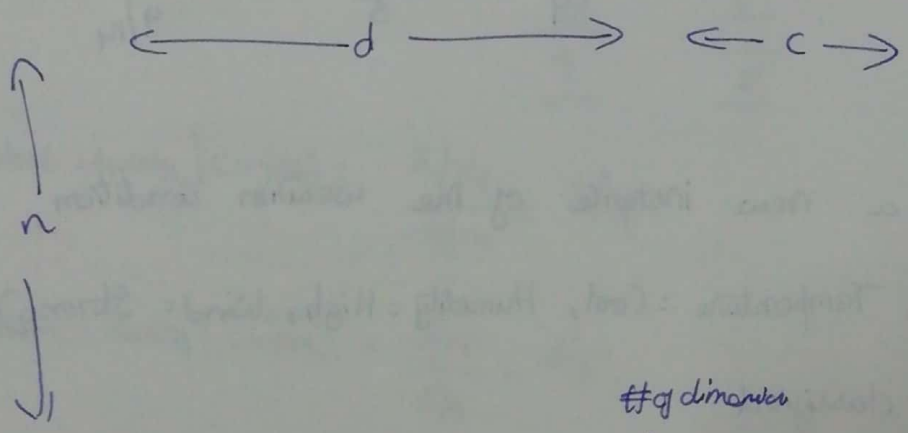
$P(\text{high} \mid \text{no}) * P(\text{strong} \mid \text{no})] * P(\text{no})$

$= 3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14$

$= 0.0205$

→ In training phase we will calculate all the likelihood probabilities and the class probabilities

→ P(class)



Time taken to calculate is $O(n d c)$

n → # of points, d → # of dimensions, c → # of classes.

→ $P(f_i \mid c)$ (d) (c)

Storing

The space complexity is $O(d * c)$

For Training

→ Time Complexity for Naive Bayes is $O(n)$

Space " " " $O(d * c)$ } d & c should be much smaller than n

↳ dictionary

$P(Yes|x') \propto 0.0053$

$P(NO|x') \propto 0.0205$

Since No is large $x' : y' = NO$

For Test :-

If i have d dimensions, we have to do d-lookups.

→ Time Complexity $O(d * c)$

If d is small & c is small, the training time is less, space is also less.

→ Compared to KNN, NB is much better in terms of space Complexity @ runtime.

$O(d * c) \rightarrow NB$

$O(n * d) \rightarrow KNN$

→ NB is much more memory efficient at RUNTIME

NB Bernauli → Binary classification

NB Gaussian → 1–5 classes

NB multinomial → More than 5 classes.

Naive Bayes on Text Data

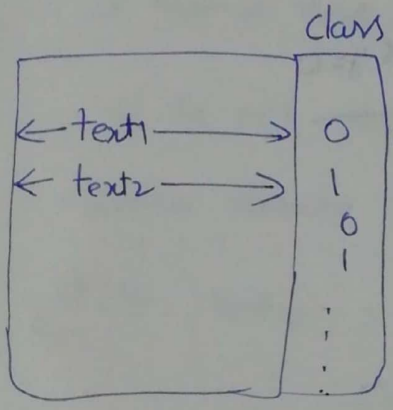
→ Navies bayes is first method to create a Spam filter

Email → SPAM
↳ not Spam.

Review → +ve
↳ -ve. } Polarity

→ Textclassification → NB a very simple and heavyly applied method.

→ NB can be applied on many languages.



Task:

$P(y_q=1 \mid text_q)$

$P(y_q=0 \mid text_q)$

text → Stopwords
stemming
n-grams
} Pre processing steps most ppl apply

↳ $w_1, w_2, w_3 \cdots w_d$ } Binary Bow

↳ whether a word is present or not
we are not counting number of occurances.

text → $\{w_1, w_2, \cdots w_d\}$

$P(y=1 \mid text) \propto P(y=1 \mid w_1, w_2, w_3 \cdots w_d)$

$\propto \underbrace{P(y=1)}_{\text{Class prior}} * \underbrace{P(w_1 \mid y=1) * P(w_2 \mid y=1) \cdots P(w_d \mid y=1)}_{\text{likelihood}}$

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

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

$$P(y=1) = \frac{\#\text{ of Train points with } y=1}{\text{Total \# of Train points}}$$

$$P(y=0) = \frac{\#\text{ of Training points with } y=0}{\text{Total \# of train points}}$$

$$P(w_i \mid y=1) = \frac{\#\text{ of data pts which contain } w_i \text{ \& } y=1}{\#\text{ data points with } y=1}$$

$$P(w_i \mid y=0) = \frac{\#\text{ of data pts which Contains } w_i \text{ \& } y=0}{\#\text{ data points with } y=0.}$$

→ NB is a very good baseline
↗
benchmark

LAPLACE / ADDITIVE Smoothing

Training : $P(y=1)$; $P(y=0)$ ← class priors.

$P(w_1|y=1)$ $P(w_1|y=0)$
$P(w_2|y=1)$ $P(w_2|y=0)$
⋮
$P(w_m|y=1)$ $P(w_m|y=0)$

} likelihoods.

Test :

I have a text$_q$ = (w_1, w_2, w_3, w')

w' is not present in the set of words $\{w_1, w_2, w_3 \cdots w_n\}$ (training corpus)

→ Let us assume w' is a crazy words which is not often used

$P(y=1 \mid \text{text}_q) = P(y=1 \mid w_1, w_2, w_3, w')$

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

now the challenge is how do we get

$P(w'|y=1)$; $P(w'|y=0)$

Ignoring w' or dropping w' is equal to $P(w'|y=1) = 1$ & $P(w'|y=1) = 0$. This is incorrect.

$$P(w'|y=1) = \frac{P(w', y=1)}{P(y=1)} = \frac{\#\text{pts } w' \text{ occurs } \& \ y=1}{\#\text{pts where } y=1}$$

$$= \frac{0}{n_1} = 0$$

If I put, it will be dangerous. The whole probability will be Zero

work around for this problem is Laplace Smoothing / Additive Smoothing.

Laplace Smoothing is different from Laplacian Smoothing

↓ this could be used in Image processing.

$$P(w^i|y=1) = \frac{0+\alpha}{n_1+\alpha k}$$

k: # of distinct values that w^i can take

w^i can be present & not present. So it can take 0 & 1 In text processing case.

In non text processing

$$P(\text{feature}=a|y=1) = \frac{0+\alpha}{n_1+\alpha}$$

Here k = # distinct values w^i can take.

α = Any numeric value

Typically we will take $\alpha=1$

Let $n_1=100$ $\quad P(w^i|y=1) = \frac{0+\alpha}{100+2\alpha}$

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

$= P(w^i|y=1) \neq 0$

Why we are doing this Entire process why can we replace with a simple Epsilon which is $\varepsilon = 0.001$

Case 2: $\alpha=10,000 = \frac{10,000}{100+20,000} = \frac{10000}{20,100} \approx \frac{1}{2}$

when your alpha is large, we are saying Two probabilities are Equal $w^i=0$

$w^i=0$

$P(w^i|y=1) = P(w^i|y=0) = 1/2$

The probability of w_i occurs for positive class and the probability of w_i occurs for negative class and those both probabilities are Equal to half

→ In Laplace Smoothing

$$P(w_i | y=1) = \frac{\#\text{ of datapoints with } w_i \text{ \& } y=1 \; + \alpha}{\#\text{ of datapts with } y=1 \; + \alpha K}$$

$P(w_i | y=1)$: Present in my Training data

This is called additive smoothing as we are adding α & αK

→ $P(w_i | y=1) = 2/50 \quad \frac{2+\alpha}{50+\alpha K} = \frac{2}{5}$ when $\alpha = 0$

$\alpha = 1$, $K = 2$ $\quad \frac{2+2}{50+4} = 3/54$

$\alpha = 1000 = \frac{2+1000}{50+2000} = \frac{1002}{2050}$

$\alpha = 10 \quad \frac{2+10}{50+20} = \frac{12}{70}$

$\alpha = 100 \quad \frac{2+100}{50+200} = \frac{102}{250}$

As alpha is increasing o/p1 < o/p2 < o/p3 < o/p4

$\alpha \uparrow$ we are moving likelihood probabilities to uniform distribution

If numerator & denominator is small we have less confidence in ratio

Often times people apply $\alpha = 1$ which is add-one-smoothing

we are smoothing it towards uniform smoothing $\left(\frac{1002}{2050}\right) \approx 1/2$

α is directly proportional to bias & variance.

Log Probabilities & Numerical Stability

$$P(y=1 \mid w_1, w_2 \cdots w_d) = \underbrace{P(y=1)}_{0 \text{ to } 1} * \underbrace{P(w_1 \mid y=1)}_{0 \text{ to } 1} * \underbrace{P(w_2 \mid y=1)}_{0 \text{ to } 1} * \cdots \underbrace{P(w_d \mid y=1)}_{0 \text{ to } 1}$$

Suppose d is large, Let d=100

we are multiplying 100 numbers which lie b/w 0 & 1

$(0.2 * 0.1 * 0.2 * 0.1) = 0.0004$ (It is a very small decimal)

If I multiply 100 numbers all of which lies b/w 0 to 1 we can run into numerical stability

In python if we take double precision floating point #. it has 16 significal values.

when we multiplying lot of small number we will end up at Numeric underflow

Then python starts doing rounding and result in Errors

$\log(0.0004) \simeq -2.23979$

Log probabilities :

we initially have.

$$P(y=1 \mid w_1 w_2 \cdots w_d) = P(y=1) * \prod_{i=1}^{d} P(w_i \mid y=1)$$

$$P(y=0 \mid w_1, w_2 \cdots w_d) = P(y=0) * \prod_{i=1}^{d} P(w_i \mid y=0)$$

log doesn't change the order $x \uparrow$; $\log(x) \uparrow$

↳ monotonic function.

~~$\log P(y=1 \mid w)$~~

$$\log(a*b) = \log(a) + \log(b)$$

$$\log\left(P(y=1 \mid \omega_1, \omega_2 \cdots \omega_d)\right) = \log\left(P(y=1) * \prod_{i=1}^{d} (\omega_i \mid y=1)\right)$$

$$= \log\left(P(y=1)\right) + \sum_{i=1}^{d} \log\left(P(\omega_i \mid y=1)\right)$$

log converts mult into addition $\log(a*b) = \log(a) + \log(b)$

log convert exponential into multiple $\log(a^b) = b * \log a$.

Bias and Variance Trade off

Naive Bayes : high Bias → underfitting
high variance → overfitting.

α in laplace smoothing defines underfitting & overfitting
α changes either high bias & high variance.

Case 1 $\alpha = 0$

$$P(w_i | y=1) = \frac{\#\text{ of train datat pts } w_i \text{ occurs } \& y=1}{\#\text{ Trainpts with } y=1}$$

Let's assume a words occurs two times : $\frac{2}{1000}$

n = 2000pts — 1000 +ve
1000 -ve

Even for words that are rare we are giving the probability $P(w_i | y=1)$ result in overfitting. (high variance)

→ Small Training data changes result in large model change } deg of variance
dramatically

Since w_i occurs only in (2) out of 2000 cases
1000 +ve, 1000 -ve.

Small change in my D_{train}
↳ remove the 2 texts (x_i, y_i) that contain w_i

$$P(w_i | y=1) = \frac{2}{1000} \text{ to } \frac{0}{1000}$$

As probability changed the model changes.

when $\alpha = 0$ small change in D_{train} results in large change in the model → high variance → overfitting

Case 2: α is very large : $\alpha = 10000$

$$P(w_i | y=1) = \frac{2}{1000} = \frac{2+10000}{1000+2(10000)} \approx \frac{1}{2}$$

w_i ↓ 0 or 1

when α become very large

for all w_i

$$P(w_i | y=1) \approx P(w_i | y=0) \approx \frac{1}{2}$$

Model cannot distinguish whether w_i is 0 or 1

α = v-large.

$$P(y=1 | w_1, w_2 \cdots w_d) = P(y=1) * \prod_{i=1}^{d} P(w_i | y=1) \approx 1/2$$

$$P(y=0 | w_1, w_2 \cdots w_d) = P(y=0) * \prod_{i=1}^{d} P(w_i | y=0) \approx 1/2$$

→ How to find the right alpha

→ know: right K → using simple CV or 10 fold CV

→ Same ~~is done~~ with α also.

α in NB, K in KNN → hyper Parameters

→ Right hyper parameter in most model are decided using Cross Validat

Feature importance and interpretability

How to get Feature Importance :

Likelihood

$\forall w_i$, $P(w_i|y=1)$

$\forall w_i$, $P(w_i|y=0)$

w_i	$P(w_i\|y=1)$	$P(w_i\|y=2)$
w_1		
w_2		
w_3		
w_4		
:		
w_{in}		

(1) Sort w_i s based on $P(w_i|y=1)$ in desc.

Top words occurs very often

w_i s which has high value of probabilities & important words & features in determining that pairs of the class.

+ve class : find words/feature (w_i) with highest value of $P(w_i|y=1)$

-ve class : " of $P(w_i|y=0)$

In NB feature importance determined/obtained directly from the model.

which is not possible in knn

Interpretability

Suppose a query text x_q is given $\rightarrow y_q = 1$

$\downarrow$

$\{w_1, w_2, w_3 \ldots w_d\}$

I am concluding $y_q = 1$ because x_q contains words w_3, w_6, w_{10} which have a high value of $P(w_3|y=1)$ $P(w_{10}|y=1)$ $P(w_6|y=1)$

Similarly

I am not saying $y_q \neq 0$; w_8, w_{15}, w_{20}

$P(w_8|y=0)$ is very small

$P(w_{15}|y=0)$ is very small.

IMBALANCED DATA

What happens to NB with imbalance data.

$n \rightarrow n_1$ +ve
$\quad\quad\; n_1$ -ve

$n_1 >> n_2$
or
$n_2 >> n_1$

Basic formula for NB

$$P(y=1 \mid w_1, w_2, w_3 \ldots w_d) \propto P(y=1) \prod_{i=1}^{d} P(w_i \mid y=1)$$

$$P(y=0 \mid w_1, w_2, w_3 \ldots w_d) \propto P(y=0) \prod_{i=1}^{d} P(w_i \mid y=0)$$

Compare

Let's assume 90% of pts are +ve pts
10% of pts are -ve pts

$P(y=1) = n_1/n = 0.9$
$P(y=0) = n_2/n = 0.1$

When
$\prod_{i=1}^{d} P(w_i \mid y=1)$ exactly equal to $\prod_{i=1}^{d} (w_i \mid y=0)$

The first one has advantage to be a positive class as it 0.9%.

When we have an imbalanced dataset, because of class priors $\Rightarrow$ majority/dominating class has an advantage.

Sol1 (1) up sampling
down sampling. $n_1 \approx n_2$ $P(y=1) = P(y=0) = 1/2$

(2) Simply drop probability terms. $P(y=1)$ & $P(y=0)$

$P(y=1)$ & $P(y=0) = 1$

(3) modified NB to account for class imbalanced
↳ not often used

Imbalance data

+ve $n_1 = 900$

-ve $n_2 = 100$

$P(w_i | y=1) = \frac{\text{freq}}{900}$ → majority class → 0 to 900

$P(w_i | y=0) = \frac{\text{freq}}{100}$ → 0 to 100 → minority class.

The minority class tend to have a small numerator when compared to majority class.

Laplace Smoothing

We use same alpha for positive & negative class

α impact more for minority class & less for majority class

Ex:- minority → majority

$P(w_i | y=0)$ $\frac{2}{100}$, $\frac{18}{900}$ ← $P(w_i | y=1)$

2% 2%.

Let's assume $\alpha = 100$ $\frac{2+10}{100+20}$, $\frac{18+10}{900+20}$:

$\frac{12}{120}$ → 0.1 or 10% $= 28/920$ 3.04%.

2% got change to 10%.

Now a Hack

We can create a ~~the~~ α for +ve class & α for -ve class.

→ NB is impacted by imbalance data

Outliers

How are outliers are handled in NB?

Test classification : outlier at test time.
$q = w_1, w_2, w_3, w'$

$w' \notin \{w_1, w_2, w_3 \cdots w_m\}$ ← Set of words that I have seen in Dtrain.
↑ not present

Laplace smoothing

$$P(w_i|y=1) = \frac{0}{n_1} + \frac{\alpha}{2\alpha}$$

when we get a outlier at test time it is handled by α

Outlier in training data :

$\{w_1, w_2, w_3 \cdots w_m\}$ ← set of words in Dtrain.

Let us conside w_8 occurs very few time in +ve & -ve class.

→ To handle outlier in Training

1. if a word (w_j) occurs fewer than 10 time, then just ignore that word.

$w_j \notin \{w_1, w_2, w_3 \cdots w_m\}$
↑ occurs very few time in Dtrain.

2. Laplace smoothing (α)

Missing Values

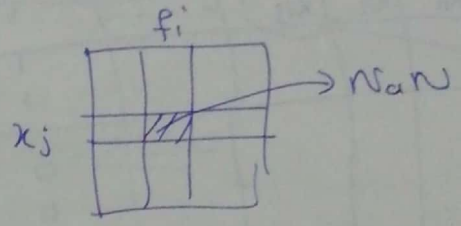
Case ①

If we have a text data : Text = {$w_1, w_2 \cdots w_d$)

↑ no care of missing data

Case ② Categorical features

$f_i \in \{a_1, a_2, a_3\}$



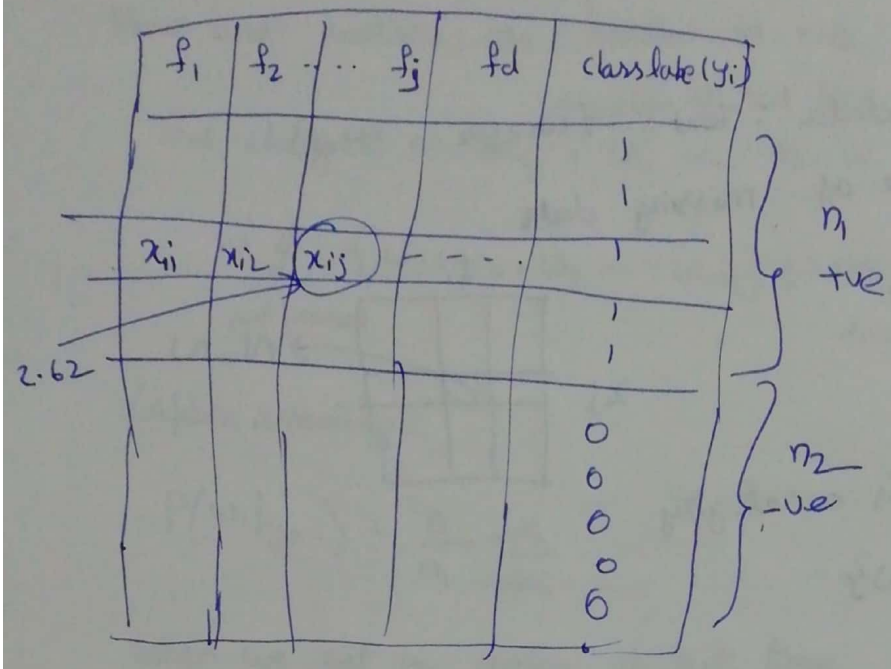
we can consider NaN is a category

$f_i \in \{a_1, a_2, a_3, nan\}$

Case 3 Numerical features

↳ Standard feature Imputation (mean, median, KNN)

NB with Numerical features

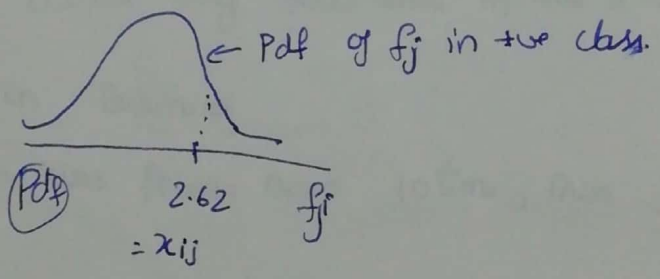


$$P(y_i=1 \mid x_{i1}, x_{i2} \cdots x_{id}) \propto P(y_i=1) \prod_{j=1}^{d} P(x_{ij} \mid y_i=1)$$

$\Rightarrow$ class prior $\frac{n_1}{n_1+n_2}$

$P(x_{ij} \mid y=1)$

$\Rightarrow$ we can obtain this probability from the Pdf of f_j in +ve



Assumption

→ Ppl assume f_j in +ve will come from Gaussian distribution $N(\mu_j, \sigma_j^2)$

Such a model is called Gaussian Navie Bayes

$P(w_i \mid y=1)$	$P(w_i \mid y=1)$	Gaussian NB ← likelihood prob are Gaussian.
0 1	0 1 2	
Bernoulli NB	Multinomial NB	

NB: Conditional Independences

It is assuming that each of your features are independent of other given the class label.

MULTI CLASS CLASSIFICATION

NB fundamentally can do multiclass classification.

$$P(y_i=1 \mid \omega_1, \omega_2, \omega_3 \cdots \omega_d)$$
$$P(y_i=0 \mid \omega_1, \omega_2 \cdots \omega_d)$$
$$P(y_i=2 \mid \omega_1, \omega_2 \cdots \omega_d)$$
$$\vdots$$
$$P(y=c-1 \mid \omega_1, \omega_2 \cdots \omega_d)$$

} c-class classification.

Similarity & Distance Matrix

Can naive bayes classification instead of giving points explicit, we are given a distance/similarity metrics.

→ KNN can do.

→ NB cannot use distance/similarity matrix.

Large Dimensionality

→ Can NB handle large dimensionality

↳ we have to use log probabilities.

Best & Worst Cases

① Conditional Independence of features

↳ True : NB perform very well

↳ False : NB deteriorates / degrades

When some features are dependent NB works reasonable well.

tve → good
great
phenominal
} these words are independent

② Text classification → Email spam
↳ review polarity (high dimension)

③ NB is Extensively used when categorical features / binary feature
Typically when we have realvalued features → NB is not used much.

④ NB is super ~~intre~~ interpredible

↳ we can get feature importance

↳ runtime Complexity is low

↳ Train Time Complexity is low

↳ run time Space is low (Prior) (Likelihood)

⑤ Easily overfit if you do not laplace smoothing and we have to find with cross validation

EDE (Example)

Points

→ alpha is typically taken to be valued b/w 0 & 1. In some cases where want to regularize heavily as we may be overfitting a lot, we could have alpha > 1 also

How to find feature importance?

```
from sklearn.naive_bayes import BernoulliNB.
bnb = BernoulliNB()
bnb.fit(X_train, y_train)

neg_class_prob_sorted = bnb.feature_log_prob_[0, :].argsort()
pos_class_prob_sorted = bnb.feature_log_prob_[1, :].argsort()

print(np.take(count_vect.get_feature_names(), neg_class_prob_sorted[:10]))
```

Gaussian NB

multinomial NB
Bernoulli NB
Out of core NB

```
from sklearn import dataset
iris = datasets.load_iris()
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
y_pred = gnb.fit(iris.data, iris.target).
                                predict(iris.data)
print("Number of mislabeled points out of a
total %d points : %d " %. (iris.data.shape[0],
(iris.target != y_pred).sum()))
```

Attributes

class_prior : probability of each class.
class_count_ : number of training samples in each class
theta_ : mean of each feature per class
sigma_ : Variance of each feature per class.

Methods

fit(X, y[, sample_weight])
get_params
partial_fit
predict
predict_log_proba
predict_proba
Score
set_params.

BernoulliNB

BernoulliNB(alpha=1.0, binarize=0.0, fit_prior=True, class_prior=None)

→ Naive Bayes classifier for multivariate Bernoulli models

→ Like MultinomialNB, this classifier is suitable for discrete data. The difference is that while MultinomialNB works with occurrence counts,

→ BernoulliNB is designed for binary/boolean features

Parameters

alpha : float, optional (default=1.0)

Additive (Laplace/Lidstone) smoothing parameter (0 for no smoothing)

binarize : float, None, optional, default=0.0

Threshold for binarizing (mapping to booleans) of sample features. If none input is presumed to already consist of binary vectors

fit_prior : boolean, optional (default=True)

Whether to learn class prior probabilities or not. If false, a uniform prior will be used

class_prior : array-like, size=[n_classes], optional (default=None)

Prior probabilities of the classes. If specified the priors are not adjusted according to the data.

Attributes

class_log_prior_ : array, shape = [n_classes]

Log probabilities of each class.

feature_log_prob:

Empirical log probability of features given a class

class_count_:

feature_count:

Number of samples encountered for each (class, feature) during fitting.

This value is weighted by the sample weight when provided.