Naive Bayes

→ It's a probability based classification algorithm.

Conditional probability

$$P(A|B) = \underbrace{P(A=a \mid B=b)}_{\text{Probability that } A=a \text{ given } B=b}$$

A, B → Random variables

Ex: let, D1: the outcome of die 1

D2: the outcome of die 2

Question: What is the probability that D1 = 2 given that $D_1 + D_2 \leq 5$

Soln: $P(D1 = 2 \mid D_1 + D_2 \leq 5)$

① possible values for $D_1 + D_2 \leq 5$

$= \{(1,1), (1,2), (1,3), (1,4), (2,1)\ (2,2), (2,3)$
$(3,1), (3,2), (4,1)\}$

outcomes when $D_1 = 2$

$= 10$

$$\Rightarrow P(D_1 = 2 \mid D_1 + D_2 \leq 5) = \frac{3}{10} = \frac{3/36}{10/36} = \frac{3}{10}$$

$$\boxed{P(A|B) = \frac{P(A \cap B)}{P(B)}}$$ } Valid only if $P(B) \neq 0$

Note: $P(A \cap B)$ → probability of both events occurring

$P(A|B)$ → Individual probability of A occurring given that B has occurred

Ex 2: Suppose an individual applying to college determines that he has 80% change of being accepted & he knows that dormitory housing will only be provided to 60% of all the accepted students. Then

$P(\text{Accepted and Dormitory housing})$

$= P(\text{Dormitory housing} \mid \text{Accepted}) * P(\text{Accepted})$

$= \cancel{0.6}\ 0.6 * 0.8 = 0.48$

Independent v/s Mutually Exclusive events

→ A & B are independent } $P(A|B) = P(A)$, $P(B|A) = P(B)$

Example 1:

A: Getting value of 6 in die 1 throw $(D_1 = 6)$

B: " " " 3 in die 2 throw $(D_2 = 3)$

⇒ $P(D_1 = 6 \mid D_2 = 3) = P(D_1 = 6) = 1/6$

→ $P(A|B) = P(B|A) = 0$ } then A & B are mutually exclusive events

$\frac{P(A \cap B)}{P(B)} = \frac{P(B \cap A)}{P(A)} = 0$

$A \cap B = B \cap A = 0$ for mutually exclusive events

Bayes Theorem:

theorem: $\underbrace{P(A|B)}_{\text{posterior probability}} = \frac{\overbrace{P(B|A)}^{\text{likelihood}}\ \overbrace{P(A)}^{\text{prior}}}{\underbrace{P(B)}_{\text{Evidence}}}$ if $P(B) \neq 0$

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

$A \cap B = B \cap A$ → from Set theory

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

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

⇒ $P(B \cap A) = \cancel{P(A)}\ P(B|A) * P(A)$

$\therefore \boxed{P(A|B) = \frac{P(B|A) * P(A)}{P(B)}}$

Naive Bayes Algorithm

Naive → Unsophisticated or Simplistic

Bayes → Bayes' theorem

→ $P(\underbrace{C_k}_{\text{class label}} \mid \underbrace{x_1, x_2 ---- x_n}_{\text{vector of features (independent) of a data point}})$

$P(C_k | x) = \frac{P(C_k) \cdot P(x | C_k)}{P(x)}$

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

$\text{Posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$

→ $p(C_k|x) = \frac{p(C_k)\,p(x|C_k)}{p(x)}$

In practice, there is interest only in the numerator of the above fraction, because the ~~dep~~ denominator $p(x)$ is constant and independent of C_k

$\therefore$ we consider only numerator

$\Rightarrow p(C_k|x) = p(C_k).\,p(x|C_k) = p(x \cap C_k)$

$= p(C_k, x) \rightarrow$ Joint probability model

→ ~~If for C_i~~ $\therefore$ If $p(C_i|x)$ is high for class "i" then we consider "x" to belong to class "i"

$\boxed{p(C_k, x) = \text{Joint probability model}}$

→ $p(C_k, x_1, x_2 \dots x_n) = p(x_1, x_2 \dots, x_n, C_k)$

$= p(x_1 | x_2 \dots x_n, C_k).\ p(x_2 \dots x_n, C_k)$

$= p(x_1|x_2, \dots x_n, C_k) \cdot p(x_2 | x_3, x_4 \dots x_n, C_k).$
$p(x_3, \dots x_n, C_k)$

$\vdots$

$= p(x_1|x_2 \dots x_n, C_k) \cdot p(x_2 | x_3, x_4 \dots x_n, C_k)$
$\dots\dots\ p(x_{n-1}|x_n C_k) \cdot p(x_n|C_k) \cdot p(C_k)$

→ In naive Bayes, we assume that each feature x_i is conditionally independent of every other feature.

Simple independence : $p(A|B) = p(A)$

Conditional independence : $p(A|B, C) = p(A|C)$

$\Rightarrow p(x_i | x_{i+1} \dots, x_n, C_k) = p(x_i | C_k)$ $\left.\right\}$ x_i is independent of $x_{i+1} \dots x_n$ given C_k

NOTE : "Naive" assumption : Features are conditionally independent of each other.

→ Thus the joint model can be expressed as

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

$\propto p(C_k)\ p(x_1|C_k).\ p(x_2|C_k) \dots$

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

$\therefore$

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

where the evidence, $Z = p(x)$

$= \sum_K p(C_k)\,p(x|C_k)$

is a scaling factor dependent only on $x_1, \dots x_n$

$\hat{y} = \underset{k \in \{1,2 \dots K\}}{\text{argmax}}\ p(C_k) \prod_{i=1}^{n} p(x_i | C_k)$

Toy example : Train and Test Stages

Source : Shatterline.com/———/naive-bayes

Data

	Predictors				Response
	outlook f_1	Temperature f_2	Humidity f_3	wind f_4	play = yes/no
Day 1	Sunny	Hot	High	weak	Yes
Day 2	overcast	mild	Normal	Strong	No
Day 3	Rain	cool	Normal	weak	Yes
⋮	⋮	⋮	⋮	⋮	⋮

The learning (or training) phase

In the learning phase, we compute the table of likelihoods (probabilities) from the training data

→ In Naive Bayes,

$$p(c \mid f_1, f_2, f_3, f_4) \propto p(f_1 \mid c) \cdot p(f_2 \mid c) \cdot p(f_3 \mid c) \cdot p(f_4 \mid c) \cdot p(c)$$

likelihoods

$p(\text{outlook} = \text{sunny} \mid c = \text{yes})$

outlook (f_1)	Frequency: play = Yes	Frequency: play = No	probabilities: $p(f_1 \mid c=\text{Yes})$	probabilities: $p(f_1 \mid c=\text{No})$
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rain	3	2	3/9	2/5
	9	5		

→ likelihood table for (f_1) feature "outlook"

Similarly calculate the likelihood table for other features temp, humidity and wind.

→ Also calculate $p(c)$: $p(c = \text{Yes}) = 9/14$, $p(c = \text{No}) = 5/14$

Time complexity $O(ndc)$, with optimization it is $O(n)$
storage complexity $O(dc)$

The classification phase

let's say, we get a new instance of the weather condition, ~~x' = (o~~

$x' = (\text{outlook} = \text{Sunny}, \text{Temp} = \text{Cool}, \text{Humidity} = \text{high}, \text{wind} = \text{strong})$

Question: Are we going to play tennis under the conditions specified by x'?

Soln:

~~p(clas~~

$p(\text{play}=\text{yes} \mid x') \propto p(\text{sunny} \mid \text{play}=\text{yes}) \times p(\text{cool} \mid \text{play}=\text{yes}) \times p(\text{high} \mid \text{play}=\text{yes}) \times p(\text{wind}=\text{strong} \mid \text{play}=\text{yes}) \times p(\text{yes})$

$\propto \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14}$

$\propto \frac{1}{189} \propto 0.0053$

$p(\text{play}=\text{no} \mid x') \propto p(\text{outlook}=\text{sunny} \mid \text{play}=\text{no}) \times p(\text{temp}=\text{cold} \mid \text{play}=\text{no}) \times p(\text{hum}=\text{high} \mid \text{play}=\text{no}) \times p(\text{wind}=\text{strong} \mid \text{play}=\text{no}) \times p(\text{play}=\text{no})$

$\propto \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} \propto 0.0205$

Since $p(\text{play}=\text{no} \mid x') > p(\text{play}=\text{yes} \mid x')$, we classify the new instance x' to be the conditions where game cannot be played. i.e for x', we give $y' = 0$

NOTE: The above implementation is ~~for~~ for categorical features

Naive Bayes on Text data

→ Naive Bayes is very popular for Text classification

→ Example:

	class
←— text 1 —→	0
←— text 2 —→	1
←— text 3 —→	0
⋮	⋮

Task

$p(y_q = 1 \mid x_q)$?

(or)

$p(y_q = 0 \mid x_q)$?

text → preprocessing → $w_1 w_2 \ldots w_d$ → Binary BoW

$p(y=1 \mid \text{text}) = p(y=1 \mid \underbrace{w_1, w_2 \ldots w_d}_{\text{features}})$

$\propto p(y=1) \times p(w_1 \mid y=1) \times p(w_2 \mid y=1) \ldots p(w_d \mid y=1)$

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

Similarly

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

→ $p(y=1) = \dfrac{\#\text{ training pts with } y=1}{\text{Total }\#\text{ training pts}}$

$p(y=0) = \dfrac{\#\text{training pts with } y=0}{\text{total }\#\text{ training pts}}$

→ To compute the likelihood probabilities, divide the training set into two parts ($y=1$ & $y=0$).

→ $p(w_i \mid y=1) = \dfrac{\#\text{ data pts with } w_i \;\&\; y=1}{\#\text{ data pts with } y=1}$

NOTE: For text-classification problems, Naive Bayes is a very good baseline.

Laplace / Additive Smoothing

Training

$p(y=1)$; $p(y=0)$ ← Class priors

$p(w_1 \mid y=1)$; $p(w_1 \mid y=0)$
$p(w_2 \mid y=1)$; $p(w_2 \mid y=0)$
⋮
$p(w_m \mid y=1)$; $p(w_m \mid y=0)$

likelihoods

Test: ~~text~~ → $\text{text}_q = (w_1, w_2, w_3, w')$

Say, w' is not present in list of words $\{w_1, w_2 \ldots w_m\}$ then what to do?

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

$= p(y=1) \times p(w_1 \mid y=1) \times p(w_2 \mid y=1) \times p(w_3 \mid y=1) \times p(w' \mid y=1)$?

How do we get this ?

→ $p(w' \mid y=1) = \dfrac{p(w', y=1)}{p(y=1)}$

$= \dfrac{\#\text{ pts such that } w' \text{ occurs in } y=1}{\#\text{ pts where } y=1}$

$= \dfrac{0}{n_1} = 0$

Note ~~If~~ we substitute above $p(w'|y=1)=0$, then we get $p(y=1|text_q)$ also zero. That's a problem!

There should be some way of handling it

The workaround for this problem is Laplace/Additive Smoothing

→ $p(w'|y=1) = \dfrac{0+\alpha}{n_1+\alpha K}$

K = # distinct values w' can take

$\alpha=1$ typically | In this case $K=2$

→ let $n_1 = 0$

$\Rightarrow p(w'|y=1) = \dfrac{0+\alpha}{100+2\alpha}$

Case 1:

$\alpha = 1 = \dfrac{1}{102} \neq 0 \Rightarrow p(w'|y=1) \neq 0$

Why $\dfrac{0+\alpha}{n_1+\alpha K}$?

Case 2: $\alpha = 10000$

$\Rightarrow p(w'|y=1) = \dfrac{0+10000}{100+20000} = \dfrac{10000}{20100} \simeq \dfrac{1}{2}$

when α is very large (like in case 2)

we get

$$\boxed{p(w'|y=1) = p(w'|y=0) = \frac{1}{2}}$$

→ In Laplace Smoothing, we add α & αK for all the words.

$\therefore p(w_i|y=1) = \dfrac{\#\text{data pts with } w_i \text{ \& } y=1 + \alpha}{\#\text{data pts } y=1 + \alpha K}$

Note: $\alpha \uparrow \Rightarrow$ moving likelihood probabilities to uniform distribution

The prob will move close to $\frac{1}{2}$

Since we are adding terms in numerator & denom, it is called additive Smoothing

→ often times, $\alpha=1$, this is called add-one Smoothing

Log-probabilities for numerical stability

We have

$p(y=1|w_1,w_2,w_3 \ldots w_d)$

$= p(y=1) \times p(w_1|y=1) \times p(w_2|y=1) \ldots \times p(w_d|y=1)$

Here all probabilities lie b/w 0 & 1.

~~If we~~ let, $d=100$, then multiplying 100 small values $(0<p<1)$, will give very small value. This causes numerical stability issues. (ex: numerical underflow)

→ One soln for this is to use "log" probabilities

Log probabilities

$$p(y=1|w_1,w_2,\ldots,w_d) = p(y=1) \times \prod_{i=1}^{d} p(w_i|y=1)$$

$$p(y=0|w_1,w_2,\ldots,w_d) = p(y=0) \times \prod_{i=1}^{d} p(w_i|y=0)$$

→ "log" is monotonic func. ⇒ $x\uparrow$; $\log\uparrow$

$$\log(p(y=1|w_1,w_2,\ldots,w_d))$$
$$= \log(p(y=1)) + \sum_{i=1}^{d} \log(p(w_i|y=1))$$

Bias and Variance tradeoff

High bias → Underfitting
High variance → overfitting

→ For Naive Bayes, α in laplace smoothing determines underfit (or) overfit
→ high bias (or) high variance

→ Case 1: $\alpha=0$

$$p(w_i|y=1) = \frac{\text{\# train data points where } w_i \text{ occurs } \& \ y=1}{\text{\# train pts with } y=1}$$

Even for words which are very rare, we're giving probabilities
This results in overfit

when $\alpha=0$ ⇒ Small change in train results in large change in the model ⇒ high variance ⇓ overfitting

→ Case 2: when α is large,

$$p(w_i|y=1) \simeq p(w_i|y=0) \simeq 1/2$$

$$\therefore p(y=1|w_1,w_2,\ldots,w_d) = p(y=1) \times \prod_{i=1}^{d} p(w_i|y=1)$$

$$p(y=0|w_1,w_2,\ldots,w_d) = p(y=0) \times \prod_{i=1}^{d} p(w_i|y=0)$$

~~ex.~~ $n \to n_1$ (+ve pts), $n \to n_2$ (-ve pts) say ($n_1 > n_2$)

~~Since~~ since $n_1 > n_2$ ⇒ $p(y=1) > p(y=0)$.

$\therefore p(y=1|w_1,w_2,\ldots,w_d) > p(y=0|w_1,w_2,\ldots,w_d)$

For most of the points, it will classify as +ve
This is Underfitting

Question: How to find right "α"?

Ans: Using simple cross validation or 10-fold cross validation

α in NB is hyper parameter

Feature importance & Interpretability

We have,

w_i	$p(w_i|y=1)$	$p(w_i|y=0)$
w_1		
w_2		
w_3		
⋮		
w_n		

① Sorts w_i's based on $p(w_i|y=1)$ say in descending order

② Words which have high $p(w_i|y=1)$ are important features in determining the point $\in$ +ve class

→ Feature importance in NB :

For +ve class $\Big\}$ find w_i with highest value of $p(w_i | y=1)$

For -ve class $\Big\}$ - - - $p(w_i | y=0)$

Note: In NB, important features are obtained directly from the model.

Interpretability

$x_q \longrightarrow y_q = 1$

$\{w_1, w_2 \dots w_d\}$

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

Imbalanced data

$$P(y=1 | w_1, w_2 \dots w_d) = p(y=1) \cdot \prod_{i=1}^{d} p(w_i | y=1)$$

$$p(y=0 | w_1, w_2 \dots w_d) = p(y=0) \prod_{i=1}^{d} p(w_i | y=0)$$

Since imbalance data $\Big\}$ ⇒ $\underset{\substack{(+ve\ pts) \\ 90\%}}{n_1} >> \underset{\substack{(-ve\ pts) \\ 10\%}}{n_2}$

Then $p(y=1) = \frac{n_1}{n} = 0.9$, $p(y=0) = 0.1$

→ when we have imbalanced data then

① Due to class priori → majority/dominant class has an advantage.

Soln: ① upsampling (or) Down Sampling

② Drop $p(y=1)$ & $p(y=0)$

↳ make $p(y=1) = p(y=0) = 1$ or $1/2$

→ For an imbalanced dataset, while applying laplace smoothing, the effect of α is more on minority class probability.

Outliers:

① $x_q = w_1\ w_2\ w_3\ w'$

$w' \notin \{w_1, w_2 \dots w_m\}$ ← set of words in D_{train}

$p(w' | y=1) = \frac{0}{n_1}$ ⇒ w' is an outlier.

∴ Use laplace Smoothing.

Note Outlier occurs very few times in +ve & -ve classes

→ Drop outliers or apply laplace Smoothing

Missing Values :

① Text-data → No case of missing data

② Categorical features → ~~Cond~~ Consider NaN as category

③ Numerical features → Imputation

Scanned by CamScanner

Handling Numerical features (Gaussian NB)

Multi class classification

Similarity or distance matrix

→ NB cannot use Similarity or distance matrix

Large dimensionality

→ NB can be used with large dimensional data
When dimensionality is large, we log probabilities

Best & ~~best~~ worst cases for NB

1. Conditional independence of features (assumption)
2. Used for text classification
3. Extensively used when we have categorical features
4. Interpretability
5. Run-time Complexity, Train time complexity, Runtime space Complexity is low.