Naive Bayes Classifier

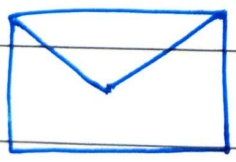
Bayes Theorem:

A, B events with prob. P(A) & P(B)

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

- $P(A/B)$ → Posterior Probability
- $P(B/A)$ → Likelihood : Conditional prob of B given A
- $P(A)$ → Prior Prob of A
- $P(B)$ → (Prob of B) (Normalisation factor)

Baye's Theorem for Classification:



→ Spam (0)

→ Not Spam (1)

x = mail

y = 1 (not spam)

$$P(Y=1/x) = \frac{P(X \cap Y=1)}{P(X)}$$

$$= \frac{P(Y=1) * P(X/Y=1)}{P(X)}$$

// Posterior Prob that your message belongs to class 1.

$$P(Y=0/x) = \frac{P(X/Y=0)\ P(Y=0)}{P(X)}$$

Final prediction = argmax ($P(y_i/x)$)
i.e. class y for which posterior prob is maximised.

Say you have 100 emails (60 spam, 40 not spam).

$P(Y=0) = 0.6$ & $P(Y=1) = 0.4$

Note: $P(X)$ is same for both & we are taking max, hence we can discard it.

Meaning: $P(Y=1/X) \propto P(X/Y=1)\, P(Y=1)$

OR

$P(Y=c/X) \propto$ likelihood * Prior

Say you have:

x_1 = ["get unlimited discount, 95% off"]

x_2 = ["I am in meeting"]

So, each word acts as feature.

$P(X/Y=0)$ = In how many spam mails these words occur.

= 0.8 (say) (means 80% of the time we see these words in spam)

Hence, $P(Y=0/X) = P(Y=0)\, P(X/Y=0)$

$= 0.6 * 0.8$

$= 0.48$

$P(Y=1/X) = 0.2 \times 0.4 = 0.08$

Hence, message is spam.

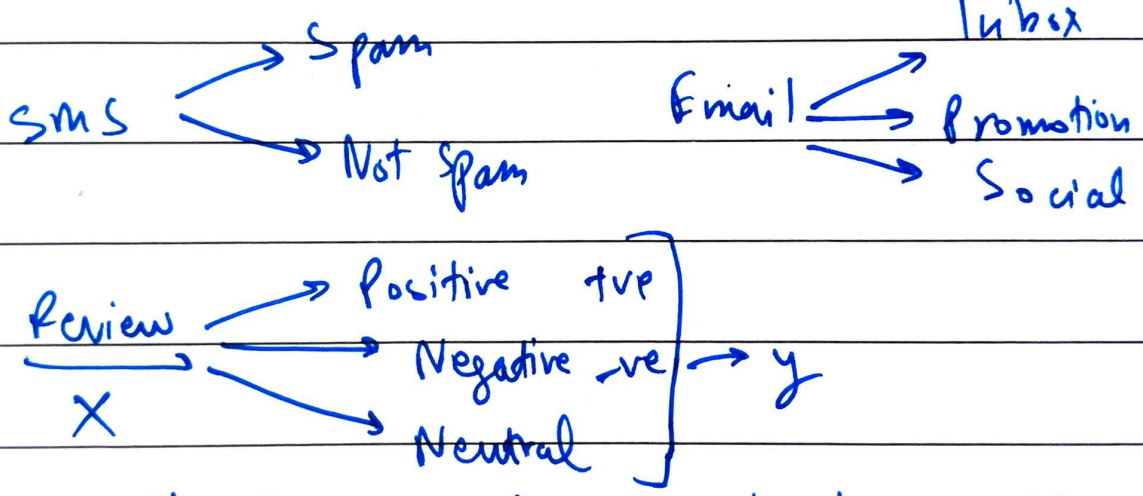
$$P(\text{spam} / \text{"lottery"}) = \frac{P(\text{"lottery"} / \text{spam}) \cdot P(\text{spam})}{P(\text{"lottery"})}$$

(likelihood, prior)

$$P(\text{"lottery"}) = P(\text{"lottery"} / \text{spam}) + P(\text{"lottery"} / \text{Not spam})$$

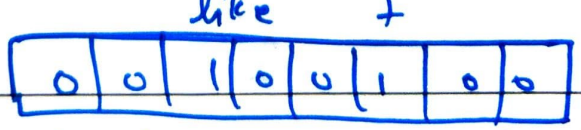
Naive Bayes Classifier :

Example: Text Classification.



You will have Discrete features. You can convert text into features using bag of words model.

Vector:



$X = x_1 x_2 \ldots \ldots \ldots x_{|V|}$

~~$P(\text{+ve} / X)$~~ $\quad P(Y/X) = \frac{P(Y) * P(X/Y)}{P(X)}$

$P(x/y)$ → Compute from training data.

$P(y)$ → Prior → what prob of total texts are positive, etc.

Posterior

$$P(Y=1/X) = \frac{P(Y=1)\ P(X/Y=1)}{P(X)}$$

$$P(Y=0/X) = \frac{P(Y=0)\ P(X/Y=0)}{P(X)}$$

$\text{argmax}\ P(Y_i/X)$ = hypothesis.

i.e. that value of i for which posterior probability is maximized.

$$P(Y=1) = \frac{\text{count all tve Reviews}}{\text{Total Reviews}}$$

$$= \frac{\sum_{i=1}^{m\ (\text{examples})} \mathbf{1}[y^{(i)}=1]}{m}$$ // Add 1 everytime $y=1$

$X = \langle x_1\ x_2\ x_3\ \ldots\ x_{|V|} \rangle$

To compute:

$P(x_1, x_2 \ldots x_{|v|} / y = 1) =$

$P(x_1 / y=1) \cdot P(x_2 / y=1, x_1) \cdot P(x_3 / y=1, x_1, x_2)$

$\ldots P(x_{|v|} / y=1, x_1, x_2 \ldots x_{|v|-1})$

x_1 .. good ...

x_2 .. awesome ...

x_3 .. like ...

$P(\text{awesome} / \text{+ve}, \text{good})$ → already seen good

This is becoming complex.

$P(x_i / y, x_1 \ldots x_{i-1})$

Prob. that word x_i is coming given you know the class, x_i is only conditioned on y & does not depend on if you have seen other words before.

$= P(x_i / y)$

* Used because it works pretty well in practice.
* This is Naive Baye's Assumption.

$$P(x/y=1) = P(x_1/y=1)\, P(x_2/y=1) \ldots$$
$$\ldots P(x_m/y=1)$$
$$= \prod P(x_i/y=1)$$

$$P(y=1/x) = \frac{\prod P(x_i/y=1)\, P(y=1)}{P(x)}$$

$$P(x) = P(y=0)\, P(x/y=0) + P(y=1)\, P(x/y=1)$$

Mushroom Classification:

Given mushrooms with say 3 features:

$$X = \begin{bmatrix} x_1 & x_2 & x_3 \\ - & - & - \\ - & - & - \\ - & - & - \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Naive Bayes: Compute Posterior prob. of mushroom of each type

$P(y=1/x)$, $P(y=2/x)$ & $P(y=3/x)$

↳ Given some features x of single! example.

Test Point (x, y)

Now, find class having max posterior prob.

$$P(y=1/x) = \frac{P(x/y=1)\ P(y=1)}{P(x)}$$

$x = \langle x_1\ x_2\ x_3 \rangle$

Naive Bayes Assumption :

$$P(x/y=1) = P(x_1/y=1) * P(x_2/y=1) * P(x_3/y=1)$$

$$P(y=1/x) \propto \prod_{i=1}^{n} P(x_i/y=1)$$

General :

$$P(y=c/x) \propto \prod_{i=1}^{n} P(x_i/y=c)\ P(y=c)$$

(n = no. of features)

This is a simple example training / testing

likelihood

Prior

$$= \frac{\prod_{i=1}^{n} P(x_i/y=c)\ P(y=c)}{\left(k = \text{no. of classes}\right) \sum_{c=1}^{k} \left(\prod_{i=1}^{n} P(x_i/y=c)\ P(y=c)\right)}$$

Remains same so ignore

Naive Bayes for text:

$$P(x_j / y_i = c) = \frac{\text{count}(x_j, y_i = c)}{\sum \text{words count}(w, y_i = c)}$$

(j^{th} feature) (i^{th} class)

No. of times x_j appeared in class c.

$$= \frac{\text{Count of words } x_j \text{ that occurred in class } c}{\text{Sum of all words in vocab in class } c}$$

* OR: For every word in vocab, no. of times it appears in $y = c$, whole sum.

$$P(y = c/x) \propto \pi P(x_i / y = c)\, P(y = c)$$

[multiply of all features of given x]

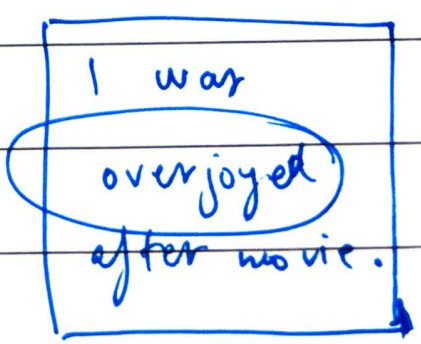
But there is a problem with this:

(Review)

$$P(y = +ve / x) = \pi P(x_i / y = +ve)\, P(y = +ve)$$

Now suppose training data had words like:

Training



Testing data

We know we will treat each word as a feature. Now,

$P(\text{overjoyed} / y = +ve) = 0$

because "overjoyed" is not in training set.

* If you have not seen something in your finite training set, it does not mean that it's probability is 0.

Solution:

Laplace smoothing: Say that the particular word appeared atleast 1 time by adding 1.

$$P(x_j / y_i = c) = \frac{[\text{count}(x_j, y = c) + 1]}{[\Sigma_{w \in V} (\text{count}(w, y = c) + 1)]}$$

$$= \frac{[\text{count}(x_j, y = c) + 1]}{\Sigma_{w \in V} (\text{count}(w, y = c)) + |V|}$$

($|V|$: vocab size)

Multivariate Bernoulli Naive Bayes

* We assume each feature will take value 0 or 1.
* We will not define freq like in vocab, we will define if feature was present or not.

D_1 = "I like I like to swim" → [1 1 0 1 1] (I, like, cooking, swim, to)

D_2 = "I like cooking" → [1 1 1 0 0]

Vocab = [I, like, cooking, swim, to] (same size)
0 1 2 3 4

Say, for $D_1 \Rightarrow y$ = "Sports"

$D_2 \Rightarrow y$ = "Code"

$$P(x/y=c) = \prod P(x_i/y=c)^{b} (1-P(x_i/y=c))^{(1-b)}$$

$$= (0.2)(1) * (0.3)(1) * (1)(0.9)$$

(0.2: P of having "like" in y=c; 0.9: Prob of not taking the word coding in y=c. = (1 − 0.1))

Here, b values are D_i i.e. [1 1 0 1 1]

So, $P(x/y=c)$ is probability of this sentence x being in class $y=c$.

$$P(y/x) = \pi P(x_i/y=c)\ P(y=c)$$

$P(x_i/y=c)$ = $df_{x_i, y=c}$ (frequency)

= count the no. of documents that contain the feature x_i and they belong to class c.

= $\frac{\text{count of documents having class c \& containing the feature } x_i.}{\text{Total no. of documents in class c.}}$

= $\frac{\text{"} + 1}{\text{"} + 2}$ // Laplace smoothing

every word has 2 choices, 0 OR 1

because it is a bernoulli model.

We added 2 ex. one has the word & other does not. In numerator, we add 1 example of having that word.

Now, if N == D, then we get $\frac{1}{2}$ likelihood. Exact 1 prob is not allowed as we are taking $P * (1-P)$. i.e. if does not exist, then its $(1-P)$ is also taken.

Multinomial Event model Naive Bayes

* we do not take binary data. (for text data)
* we may take frequency of words / terms.
  Frequency $tf(t,d)$ ⇒ No. of times a term appeared in a particular document.
* Normalised term freq = $\frac{tf(t,d)}{n_d}$ → (total no. of documents)

$$P(Y/X) = \pi P(x_i / y)\, P(y)$$

can also use $(tf \cdot idf)$ for x_i

$$P(x_i / y=c) = \frac{\left[\sum tf(x_i, d \in c)\right] + 1}{\left[\sum N_{d \in c}\right] + |V|}$$

= sum of term freq of word x_i where document d belongs to class c / Total no. of words in documents that belong to class c.

no. of times word x_i is appearing in all documents belonging to class c

$+1$ $+\alpha|V|$

$\alpha = 1 \Rightarrow$ laplace smoothing

$\alpha \Rightarrow$ hyper factor which changes amount of smoothing.

Difference between the two:

✱ Multivariate Bernoulli Event model:

d → "offer for you"

vector = [0 0 0 | 0 0 | 0 0 0 0 | 1]

size = |V| → vocab

$P(x_i|y=0)$

$1 - P(x_i|y=1)$

offer | for | you

(1 means is included in document)

$P(x=1 / y=spam) = P(x_i / y=spam)$

$P(x=0 / y=spam) = 1 - P(x_i / y=spam)$

$L(\theta) = \prod_{i=1}^{|V|} P(x_i/y=spam)^{b} (1 - P(x_i/y=spam))^{1-b} P(spam)$

✱ Multinomial Event model naive Bayes:

d = "offer for you" = $\langle x_1, x_2, x_3 \rangle$

n for d = (3)

$\prod_{i=1}^{n} P(x_i/y) P(y)$

(comes from multinomial distribution)

✱ Here we do not take entire vector of length |V|

✱ n can differ for every document.

✱ Performs better generally when vocab size is large.

✱ Also depends on features, you need to experiment.

Gaussian Naive Bayes :

* Multivariate & multinomial are used for discrete features (as boolean, or count)
* Gaussian NB : continuous value features

In ~~dat~~ this, data has gaussian distribution.

Prob. distribution formula:

$$P(x_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

continuous as in: x can be 1.1, 2.345, 8.995, etc

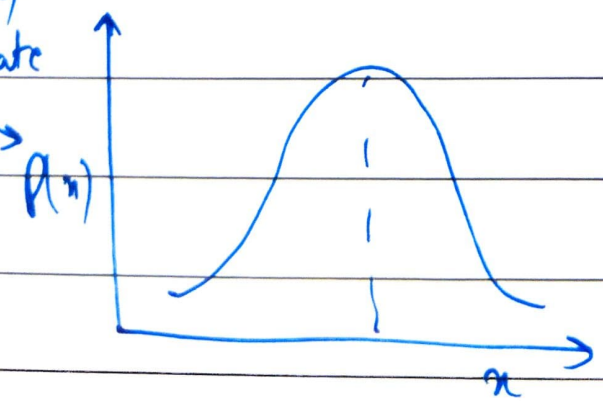
* As you go away from mean, the prob density decreases.
* σ : How quickly density will fall, or, the width of bell curve.

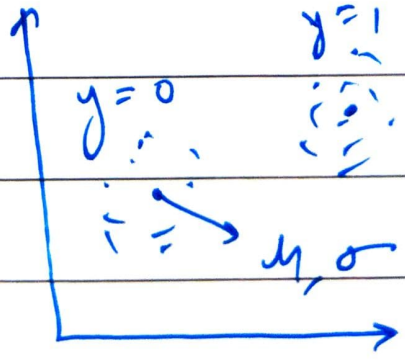
Conditional prob:

$P(x_i / y=1) = \{$ Normal Distribution $(\mu, \sigma)\}$

→ some N D

For Univariate/ multivariate





Confusion Matrix:

		Predicted N	Predicted P	Total (n = 165)
Actual	N	TN = 50	FP = 10	N → -ve
	P	FN = 5	TP = 100	P → +ve

* Accuracy = $\frac{TP + TN}{n \text{ (total examples)}}$

* Precision = $\frac{TP}{TP + FP}$

= % of examples that ~~~~ are really +ve from our prediction of +ve examples.

* Recall = $\frac{TP}{TP + FN}$

= % of examples that we predicted as +ve from all examples that were actually +ve.

* Goal is to achieve high value of precision & recall but in reality if precision ↑ then recall ↓ & vice-versa.
* To overcome this, we use:

F-measure, value between (0, 1)

Tells us the performance of model.

= Harmonic mean of prescision & Recall

$$\text{F-measure} = \frac{2 * TP}{2 * TP + FP + FN}$$

NB works better when you have:

* Small training data set
* Text classification
* Small features (dimensions)
* NB model size is low, wont overfit as it cannot represent complex behaviour.
* Use when data changes quickly.
* Conditional Independance of features.