Naive Bayes Classifier

Introduction

A **probability** is a number that reflects the chance or likelihood that a particular event will occur.

A probability of 0 indicates that there is no chance that a particular event will occur, whereas a probability of 1 indicates that an event is certain to occur. A probability of 0.45 (45%) indicates that there are 45 chances out of 100 of the event occurring.

Introduction

Let A be a random event

P(A) denotes the probability that event A happens

Example:

A is the event that I roll a die and obtain a 5

P(A) = 1/6

A is the event that I roll a die and obtain at least 5

P(A) = 2/6 = 1/3

Introduction

$P(A \wedge B)$ denotes the probability that both A and B happen

$P(A \vee B)$ denotes the probability that at least one of A and B happens

$P(A|B)$ (the probability of A given B)
denotes the probability that A happens GIVEN that B happened

$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$

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

A and B are independent if the result of A does not affect B (and vice versa)

Introduction

A and B are independent if the result of A does not affect B (and vice versa)

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

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

thus

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

Probability

Basic Formulas for Probabilities

- *Product Rule*: probability $P(A \wedge B)$ of a conjunction of two events A and B:

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

- *Sum Rule*: probability of a disjunction of two events A and B:

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$

- *Theorem of total probability*: if events $A_1, \ldots, A_n$ are mutually exclusive with $\Sigma_{i=1}^{n} P(A_i) = 1$, then

$$P(B) = \sum_{i=1}^{n} P(B|A_i)P(A_i)$$

Bayes Theorem

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- $P(h)$ = prior probability of hypothesis h
- $P(D)$ = prior probability of training data D
- $P(h|D)$ = probability of h given D
- $P(D|h)$ = probability of D given h

Probability

Bayes Theorem

Does patient have cancer or not?

A patient takes a lab test and the result comes back positive. The test returns a correct positive result in only 98% of the cases in which the disease is actually present, and a correct negative result in only 97% of the cases in which the disease is not present. Furthermore, .008 of the entire population have this cancer.

$$P(cancer) = \qquad P(\neg cancer) =$$
$$P(+|cancer) = \qquad P(-|cancer) =$$
$$P(+|\neg cancer) = \qquad P(-|\neg cancer) =$$

Probability

Bayes Theorem

Does patient have cancer or not?

A patient takes a lab test and the result comes back positive. The test returns a correct positive result in only 98% of the cases in which the disease is actually present, and a correct negative result in only 97% of the cases in which the disease is not present. Furthermore, .008 of the entire population have this cancer.

$P(cancer) =$ 0.008 $\quad P(\neg cancer) =$ 0.992

$P(+|cancer) =$ 0.98 $\quad P(-|cancer) =$ 0.02

$P(+|\neg cancer) =$ 0.03 $\quad P(-|\neg cancer) =$ 0.97

By Bayes Theorem:

P(cancer|+) = P(+|cancer)P(cancer)/P(+)

By theorem of total probability:

P(+) = P(+|cancer) P(cancer) + P(+|-cancer)P(-cancer)

P(cancer|+) = P(+|cancer) P(cancer) /

(P(+|cancer) P(cancer) + P(+|-cancer)P(-cancer))

= (0.98*0.008)/(0.98*0.008+0.03*0.992) = 0.2085

Probability

Bayes Theorem

Does patient have cancer or not?

A patient takes a lab test and the result comes back positive. The test returns a correct positive result in only 98% of the cases in which the disease is actually present, and a correct negative result in only 97% of the cases in which the disease is not present. Furthermore, .008 of the entire population have this cancer.

$P(cancer) =$ 0.008 $\quad P(\neg cancer) =$ 0.992

$P(+|cancer) =$ 0.98 $\quad P(-|cancer) =$ 0.02

$P(+|\neg cancer) =$ 0.03 $\quad P(-|\neg cancer) =$ 0.97

- By Bayes theorem:
  P(cancer|+) = P(+|cancer) P(cancer) / P(+)
- By theorem of total probability:
  P(+) = P(+|cancer) P(cancer) + P(+|-cancer)P(-cancer)
- Thus:

P(cancer|+) = P(+|cancer) P(cancer) /
(P(+|cancer) P(cancer) + P(+|-cancer)P(-cancer))

P(cancer|+) = (0.98*0.008)/(0.98*0.008+0.03*0.992)
= 0.2085

Probability

Three prisoners, A, B, and C, are in separate cells and sentenced to death. The governor has selected one of them at random to be pardoned. The warden knows which one is pardoned, but is not allowed to tell. Prisoner A begs the warden to let him know the identity of one of the two who are going to be executed. "If B is to be pardoned, give me C's name. If C is to be pardoned, give me B's name. And if I'm to be pardoned, secretly flip a coin to decide whether to name B or C."

The warden tells A that B is to be executed. Prisoner A is pleased because he believes that his probability of surviving has gone up from 1/3 to 1/2, as it is now between him and C. Prisoner A secretly tells C the news, who reasons that A's chance of being pardoned is unchanged at 1/3, but he is pleased because his own chance has gone up to 2/3. Which prisoner is correct?

Probability

We have the following events:

A – A is pardoned

B – B is pardoned

C – C is pardoned

W_B – Warden names B

W_C – Warden names C

$P(A) = P(B) = P(C) = 1/3$

$P(W_B) = P(W_C) = 1/2$

We need to find $P(A|W_B)$ and $P(C|W_B)$

Probability

By Bayes theorem:

$P(A|W_B) = P(W_B|A)\ P(A) / P(W_B)$

$P(C|W_B) = P(W_B|C)\ P(C) / P(W_B)$

- What is the probability that the warden will name B given that A will be pardoned?

- What is the probability that the warden will name B given that C will be pardoned?

Probability

By Bayes theorem:

$P(A|W_B) = P(W_B|A)\ P(A) / P(W_B)$

$P(C|W_B) = P(W_B|C)\ P(C) / P(W_B)$

- What is the probability that the warden will name B given that A will be pardoned?

  $P(W_B|A) = 1/2$

- What is the probability that the warden will name B given that C will be pardoned?

  $P(W_B|C) = 1$

Probability

By Bayes theorem:

$P(A|W_B) = P(W_B|A)\,P(A) / P(W_B) = (1/2) * (1/3) / (1/2) = 1/3$

$P(C|W_B) = P(W_B|C)\,P(C) / P(W_B) = (1) * (1/3) / (1/2) = 2/3$

The Naive Bayes Classifier

Naive Bayes Classifier

Assume target function $f : X \to V$, where each instance x described by attributes $\langle a_1, a_2 \ldots a_n \rangle$. Most probable value of $f(x)$ is:

$$
\begin{aligned}
v_{MAP} &= \underset{v_j \in V}{\operatorname{argmax}} P(v_j | a_1, a_2 \ldots a_n) \\
v_{MAP} &= \underset{v_j \in V}{\operatorname{argmax}} \frac{P(a_1, a_2 \ldots a_n | v_j) P(v_j)}{P(a_1, a_2 \ldots a_n)} \\
&= \underset{v_j \in V}{\operatorname{argmax}} P(a_1, a_2 \ldots a_n | v_j) P(v_j)
\end{aligned}
$$

Naive Bayes assumption:

$$P(a_1, a_2 \ldots a_n | v_j) = \prod_i P(a_i | v_j)$$

which gives

Naive Bayes classifier: $v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i | v_j)$

The Naive Bayes Classifier

Given a test example with attribute values $(a_1,...,a_n)$ assign x to the class c_i that maximizes $p(x|c_i)$

$$\text{pred}(x) = \text{argmax } p(c_i) * p(a_1|c_i) * p(a_2|c_i) * ... * p(a_n|c_i)$$

The Naive Bayes Classifier

Given a test example x with attribute values $(a_1,...,a_n)$ assign x to the class c_i that maximizes $p(x|c_i)$

$$pred(x) = \text{argmax } p(c_i) * p(a_1|c_i) * p(a_2|c_i) * ... * p(a_n|c_i)$$

In order to do this, we need to estimate, for every class c_i,

$p(c_i)$ and $p(a_j|c_i)$ for j in 1,...,n for every possible value of a_j

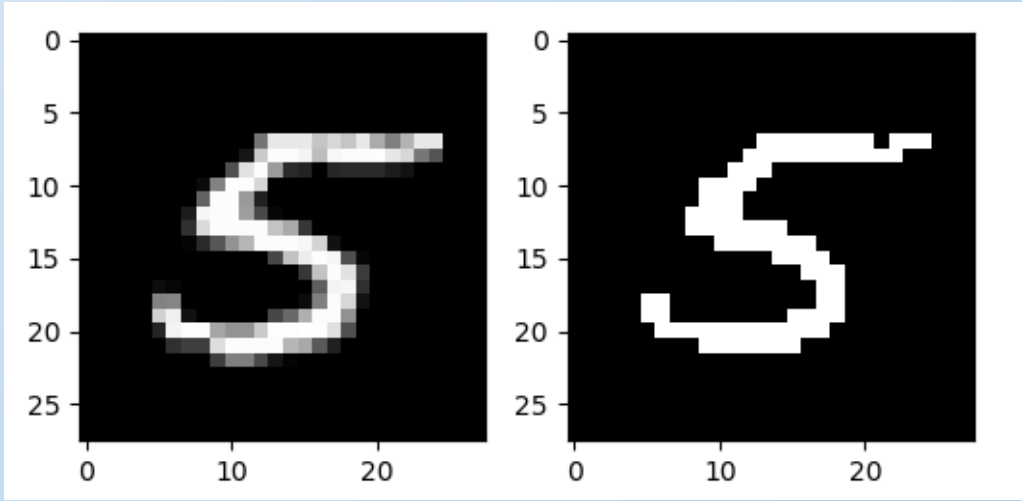
The Naive Bayes Classifier

Example MNIST with binary images

convert each image to a binary image

if pixel(i,j) >= 128, binary_pixel(i,j) = 1

else binary_pixel = 0

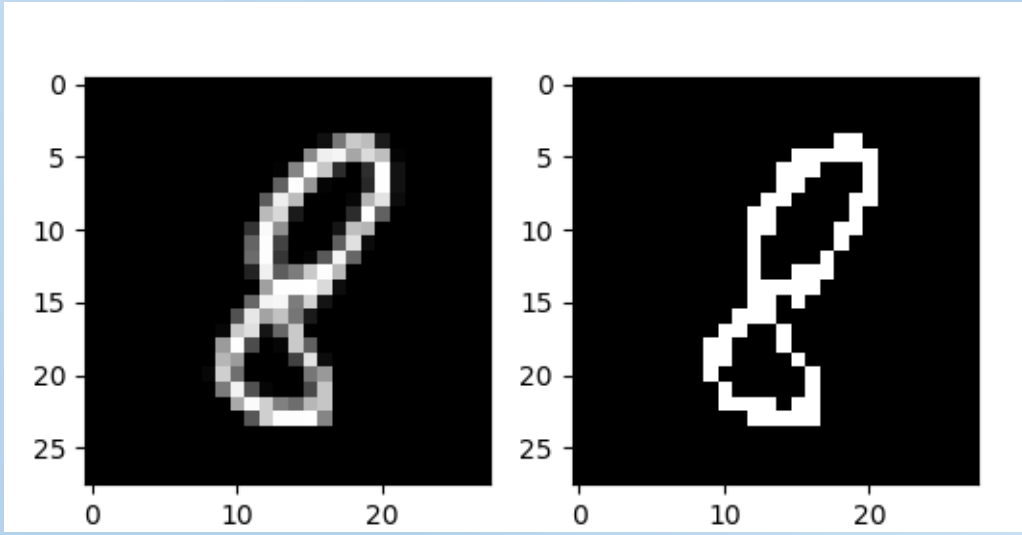


Thus every attribute has only 2 possible values

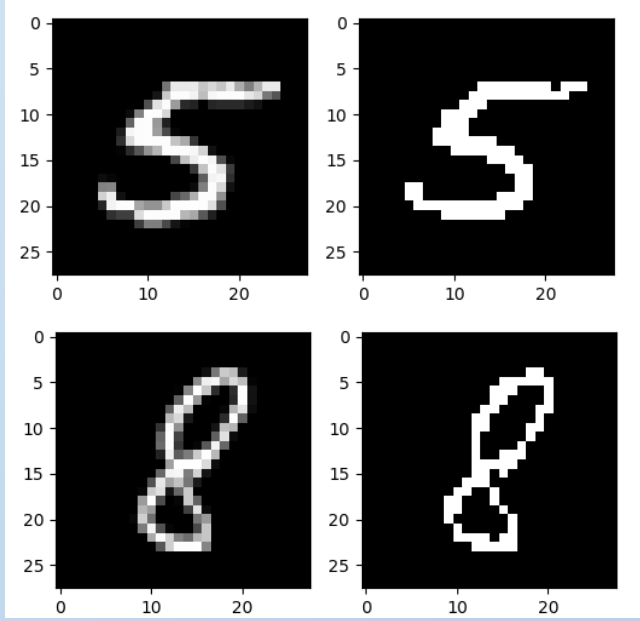
And we need to estimate

10 class probabilities

784 pixel probabilities for every class



Estimating probabilities in Python



Let X be the (binary) MNIST training dataset

Let y be the MNIST training class

We can store class probabilities in a 1D array of size 10

We can store attribute probabilities in a 2D array of size 10-by-784

p_class[i] is the probability that a randomly-chosen example belongs to class i

p_att[i,j] is the probability that a randomly-chosen example from class i has a value of 1 in attribute j

1-p_att[i,j] is the probability that a randomly-chosen example from class i has a value of 0 in attribute j

Estimating probabilities in Python – Boolean indices

Examples of Boolean indexing:

```
p = np.array([2,5,4,1,0,3])
print(p==4)          [False False  True False False False]
print(p%2 ==0)       [ True False  True False  True False]
print(p<2)           [False False False  True  True False]
```

Like all indices, Boolean indices can be used to access array elements

```
print(p[p==4])       [4]
print(p[p%2 ==0])    [2 4 0]
print(p[p<2])        [1 0]
```

Estimating probabilities in Python – Boolean indices

We can apply Boolean operators to Boolean vectors

* means 'and' and + means 'or'

```
p = np.array([2,5,4,1,0,3])
print((p<2)*(p%2 ==0))
print((p<2)+(p%2 ==0))
```

[False False False False True False]

[True False True True True False]

To count we can simply sum the values in the Boolean index vector, where True is cast to 1 and False to 0

```
p = np.array([2,5,4,1,0,3])
print(np.sum(p==4))
print(np.sum(p%2==0))
```

1

3

Estimating probabilities in Python – Boolean indices

To estimate probabilities, we divide the number of elements that satisfy the condition by the size of the universe

Example: Let y be the class vector. Compute the probability of class 4 (i.e. the probability that a randomly-chosen example belongs to class 4)

```
p_class[4] = np.sum(y==4)/len(y)
print(len(p[p==4])/len(p))
```

0.16666

Example: Let X be the training images from MNIST and let y be the class vector. Compute the probability that an example of class 4 has a value of 1 in pixel 400.

$P(a_{400}|c_4) = P(a_{400} \wedge c_4) / P(c_4)$

```
p_att_given_class[4,400] =
 = np.sum((y==4)*(X[:,400]==1))/len(y))/ (np.sum(y==4)/len(y))
 = np.sum((y==4)*(X[:,400]==1))/np.sum(y==4)
```

Dealing with underflow

What happens if we multiply a large number of small numbers?
Due to the finite precision in the computer, we eventually end up with a value of 0

To solve this, we observe:
argmax(x) = argmax(log(x))
$\log(p_1 * p_2 * \ldots * p_n) = \log(p_1) + \log(p_2) + \ldots + \log(p_n)$

Thus we replace:

$\text{pred}(x) = \text{argmax}\ p(c_i) * p(a_1|c_i) * p(a_2|c_i) * \ldots * p(a_n|c_i)$
where $x = [a_1, a_2, \ldots, a_n]$

by

$\text{pred}(x) = \text{argmax}\ \log p(c_i) + \log(p(a_1|c_i)) + \log(p(a_2|c_i)) + \ldots + \log(p(a_n|c_i))$

Dealing with real-valued attributes

Instead of computing the probability of an attribute having a value of 1 given the class, we compute the mean and standard deviation of every attribute given the class.
Given a test example, for each of its observed attribute values w, we compute the likelihood f(w) as shown below:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\sigma = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2$$

$$f(w) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(w-\mu)^2}{\sigma^2}}$$

Note:
f(w) is not really a probability, since adding the sum of f(w) over all possible values of w is not equal to 1.
If we compute the actual probabilities, the results don't change.