Supervised Learning (Classification: Bayesian Concept Learning)

Bayesian Classifier Concept

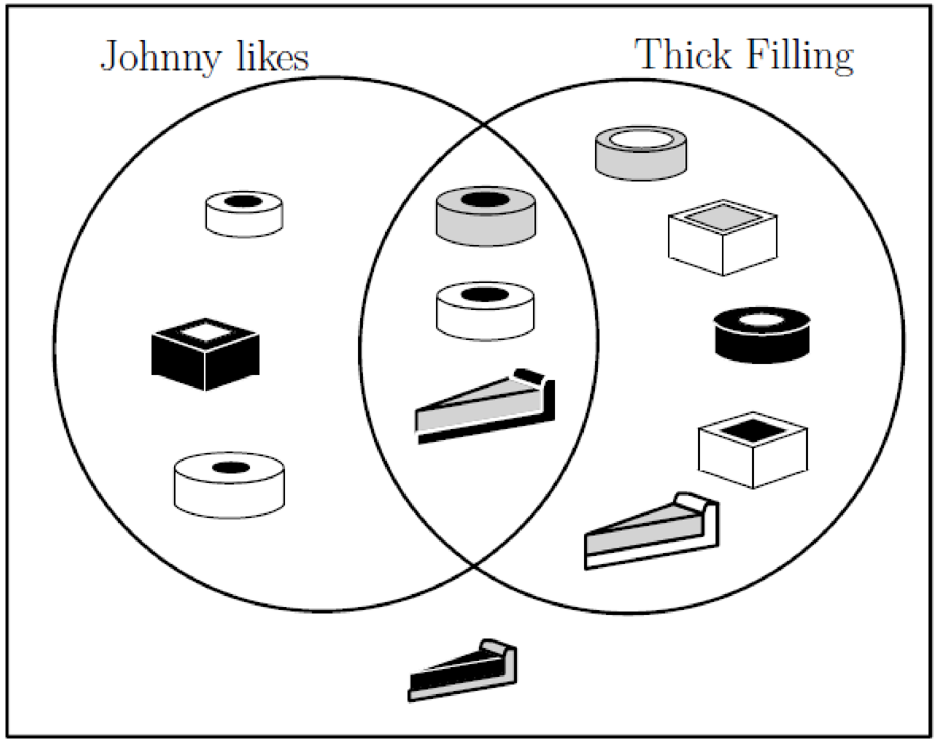
- **A Bayesian classifier** is a type of **probabilistic classifier** that applies **Bayesian probability theory** to make predictions or classifications. It models the probability distribution of different classes or categories given the observed data or features.
- There are several types of Bayesian classifiers, each with its own characteristics and applications such as : **Naive Bayes Classifier.**
- The foundation of Bayesian classifiers, including Naive Bayes and other Bayesian models, is **Bayesian probability theory**.

Example of Bayes Theorem

- Given:
  - A doctor knows that meningitis causes stiff neck 50% of the time
  - Prior probability of any patient having meningitis is 1/50,000
  - Prior probability of any patient having stiff neck is 1/20
- If a patient has stiff neck, what's the probability he/she has meningitis?

$$P(M \mid S) = \frac{P(S \mid M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

Training set in the pies domain



Basic probabilities (revision)

- Relative frequency:

$$P(pos) = \frac{N_{pos}}{N_{all}} = \frac{6}{12} = 0.5$$

- Conditional probability:

$$P(pos \mid thick) = \frac{N_{pos|thick}}{N_{thick}} = \frac{3}{8} = 0.375$$

- Joint probability:

$$P(pos, thick) = P(pos \mid thick) \cdot P(thick) = \frac{3}{8} \cdot \frac{8}{12} = \frac{3}{12}$$

Bayesian formula: derivation

- Joint probability is commutative. Therefore:

$$P(pos, thick) = P(pos \mid thick) \cdot P(thick) = P(thick \mid pos) \cdot P(pos)$$

- From here, the Bayesian formula is obtained:

$$P(pos \mid thick) = \frac{P(thick \mid pos) \cdot P(pos)}{P(thick)}$$

Bayesian formula used for classification

- Probability that a pie with *thick-filling* is positive:

$$P(pos \mid thick) = \frac{P(thick \mid pos) \cdot P(pos)}{P(thick)}$$

- Probability that a pie with *thick-filling* is negative:

$$P(neg \mid thick) = \frac{P(thick \mid neg) \cdot P(neg)}{P(thick)}$$

- Choose the label with the higher probability.

The "naïve Bayes" approach

- Probability of class c_j: $P(c_j \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid c_j) \cdot P(c_j)}{P(\mathbf{x})}$
- Naïve assumption: mutually independent attributes, x_i

$$P(\mathbf{x} \mid c_j) = \prod_{i=1}^{n} P(x_i \mid c_j)$$

- The whole Bayes formula

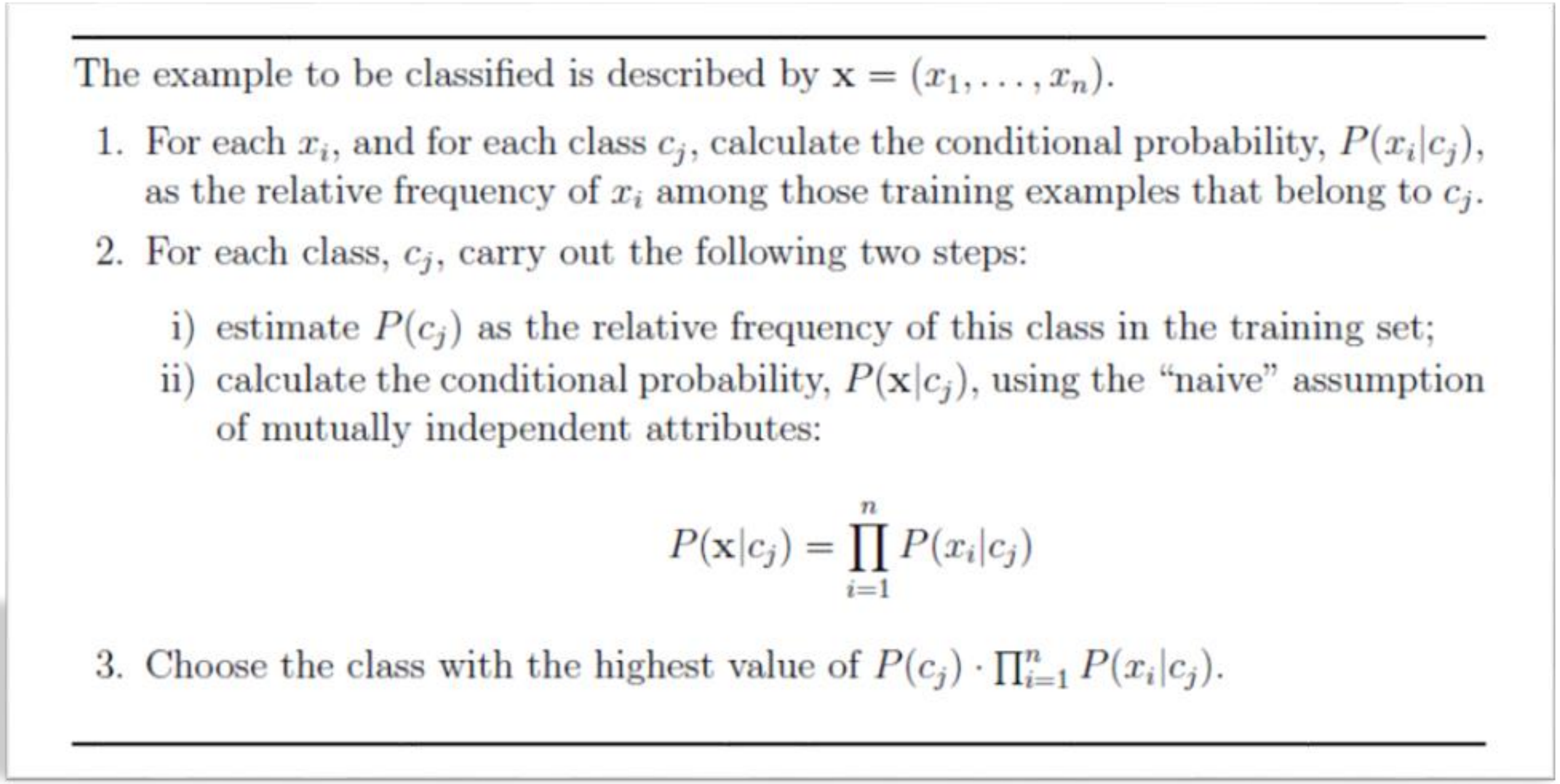
$$P(c_j \mid \mathbf{x}) = \frac{P(c_j) \cdot \prod_{i=1}^{n} P(x_i \mid c_j)}{P(\mathbf{x})}$$

"Naïve Bayes": one practical comment

- Probability of class c_j: $$P(c_j \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid c_j) \cdot P(c_j)}{P(\mathbf{x})}$$

- $P(\boldsymbol{x})$ is the same for each class, c_j.
- Therefore: choose c_j with the greatest numerator, $P(\boldsymbol{x}|c_j)P(c_j)$.

“Naïve Bayes” (algorithm)



The example to be classified is described by $\mathbf{x} = (x_1, \ldots, x_n)$.

1. For each x_i, and for each class c_j, calculate the conditional probability, $P(x_i|c_j)$, as the relative frequency of x_i among those training examples that belong to c_j.
2. For each class, c_j, carry out the following two steps:

   i) estimate $P(c_j)$ as the relative frequency of this class in the training set;

   ii) calculate the conditional probability, $P(\mathbf{x}|c_j)$, using the “naive” assumption of mutually independent attributes:

$$P(\mathbf{x}|c_j) = \prod_{i=1}^{n} P(x_i|c_j)$$

3. Choose the class with the highest value of $P(c_j) \cdot \prod_{i=1}^{n} P(x_i|c_j)$.

Bayes' Theorem

- Bayes' theorem is a fundamental principle in probability theory and statistics that relates conditional probabilities.
- It describes how to update the probability of a hypothesis (an event or statement) based on new evidence or observations.
- The theorem can be expressed mathematically as:

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)}$$

Where:

- $P(H|E)$ is the posterior probability of hypothesis H being true given evidence E.
- $P(E|H)$ is the probability of observing evidence E given that hypothesis H is true (the likelihood).
- $P(H)$ is the prior probability of hypothesis H being true before considering the evidence.
- $P(E)$ is the probability of observing evidence E, which can be calculated as a marginal probability by considering all possible hypotheses.

Naïve Bayes (example)

Ex.	Crust size	shape	Filling size	class
e1	big	circle	small	**pos**
e2	small	circle	small	**pos**
e3	big	sq.	big	**pos**
e4	small	sq.	big	**pos**
e5	big	circle	big	**pos**
e6	big	sq.	small	**neg**
e7	big	tri.	small	**neg**
e8	small	sq.	big	**neg**

P(neg)=3/8;
P(pos)=5/8

x=(small,square,small)

- P(crust=small |pos)=2/5
- P(shape=square |pos)=2/5
- P(filling=small | pos) = 2/5
- $P(\mathbf{x}|pos) = \frac{2}{5} \times \frac{2}{5} \times \frac{2}{5} = \frac{8}{125}$
- P(crust=small |neg)=1/3
- P(shape=square |neg)=2/3
- P(filling=small | neg) = 2/3
- $P(\mathbf{x}|neg) = \frac{1}{3} \times \frac{2}{3} \times \frac{2}{3} = \frac{4}{27}$
- $P(\mathbf{x}|pos)\ P(pos) = \frac{8}{125} \times \frac{5}{8} = 0.040$
- $P(\mathbf{x}|neg)\ P(neg) = \frac{4}{27} \times \frac{3}{8} = 0.056$
- *Verdict*: **neg** because 0.056 > 0.040

Naïve Bayes (another example)

- **x**=[square,thick,gray,thin,white]

P(shape=square|pos)	= 1/6	P(shape=square|neg)	= 2/6
P(crust-size=thick|pos)	= 5/6	P(crust-size=thick|neg)	= 5/6
P(crust-shade=gray|pos)	= 1/6	P(crust-shade=gray|neg)	= 2/6
P(filling-size=thin|pos)	= 3/6	P(filling-size=thin|neg)	= 1/6
P(filling-shade=white|pos)	= 1/6	P(filling-shade=white|neg)	= 2/6

We see that $P(\text{pos}) = P(\text{neg}) = 0.5$ and also:

$$P(\mathrm{x}|\text{pos}) = \prod_{i=1}^{n} P(x_i|\text{pos}) = \frac{1}{6} \cdot \frac{5}{6} \cdot \frac{1}{6} \cdot \frac{3}{6} \cdot \frac{1}{6} = \frac{15}{6^5}$$

$$P(\mathrm{x}|\text{neg}) = \prod_{i=1}^{n} P(x_i|\text{neg}) = \frac{2}{6} \cdot \frac{5}{6} \cdot \frac{2}{6} \cdot \frac{1}{6} \cdot \frac{2}{6} = \frac{40}{6^5}$$

Since $P(\mathbf{x}|\text{pos}) \cdot P(\textbf{pos}) < P(\mathbf{x}|\text{neg}) \cdot P(\textbf{neg})$, we label $\mathbf{x}$ with the negative class.

Naïve Bayes Classifier with Python

Imagine you are a data scientist working on a project to classify iris flowers into different species based on their attributes. You have a dataset with labeled examples for training and testing purposes. Your goal is to develop a machine learning model using the Gaussian Naive Bayes algorithm to classify iris flowers accurately.

Note : The iris dataset consists of continuous numerical features, including sepal length, sepal width, petal length, and petal width. Gaussian Naive Bayes is well-suited for datasets with continuous features because it assumes that these features follow a Gaussian (normal) distribution.

To import the GaussianNB class from Scikit-Learn's naive_bayes module, you can use the following import statement in Python:

```
from sklearn.naive_bayes import GaussianNB
```

This import statement allows you to create an instance of the GaussianNB classifier as follows:

```
nb = GaussianNB()
```

Upload Iris Dataset.

These steps will load the Iris dataset, create a DataFrame for it, perform a train/test split, and print the shapes of the resulting datasets.

```
# Import necessary libraries
from sklearn import datasets
import pandas as pd
import sklearn.model_selection as skms
# Load the Iris dataset
iris = datasets.load_iris()
# Create a DataFrame for the dataset
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
iris_df['target'] = iris.target
# Simple train/test split of the dataset
(iris_train_ftrs, iris_test_ftrs, iris_train_tgt, iris_test_tgt) = skms.train_test_split(iris.data,
iris.target, test_size=.25)
# Print the shapes of the training and testing sets
print("Train features shape:", iris_train_ftrs.shape)
print("Test features shape:", iris_test_ftrs.shape)
```

Naïve Bayes Classifier with Python

```
nb = naive_bayes.GaussianNB()
```

In this line, you create an instance of the Gaussian Naive Bayes classifier by calling the GaussianNB() constructor from the naive_bayes module of scikit-learn. This classifier assumes that the features (attributes) are continuous and normally distributed.

```
fit = nb.fit(iris_train_ftrs, iris_train_tgt)
```

Here, you use the fit method to train (fit) the Gaussian Naive Bayes classifier on your training data. iris_train_ftrs represents the training features (attribute values) from your dataset, and iris_train_tgt represents the corresponding target labels (class labels) for the training examples. This step involves estimating the class-conditional probabilities and other parameters required for classification..

```
preds = fit.predict(iris_test_ftrs)
```

Here, you use the fit method to train (fit) the Gaussian Naive Bayes classifier on your training data. iris_train_ftrs represents the training features (attribute values) from your dataset, and iris_train_tgt represents the corresponding target labels (class labels) for the training examples. This step involves estimating the class-conditional probabilities and other parameters required for classification..

Naïve Bayes (Summary)

- Robust to isolated noise points
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes