U O W

A Probabilistic Perspective of Classification

CSCI316 Big Data Mining Techniques and Implementation

UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Contents

Bayes' Theorem

Implementation of simple Naïve Bayes classifier

Bayesian Classification

- A probabilistic classifier: performs *probabilistic prediction*, *i.e.*, predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and other classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data
- Standard: Even though general Bayesian methods are computationally intractable, simple Bayesian methods can provide a baseline of optimal decision making against which other methods can be measured

Classification Concepts Recap

- Given a set of records, each of which is described by a sequence of attributes $X_1, \ldots, X_n, Y$. The last one is an attribute of interest, called a **class**. The rest are called **features**.
- Given a new record where Y is unknown, the task of classification is to predict which class this record falls into.
- **Probabilistic classifier**: the output of prediction is a class together with a *probabilistic score*
  - to what extent the new record falls into the output class
  - Provides the likelihood instead of a hard decision

Probability and Uncertainty

- Our main tool is the probability theory, which assigns to a numerical degree of belief between *0* and *1* to each event.
  - It provides a way of characterizing the uncertainty
- Random variables:
  - Boolean random variables: cavity might be true or false
  - Discrete random variables: weather might be sunny, rainy, cloudy, snow
    - $P(weather = sunny)$
    - $P(weather = rainy)$
    - $P(weather = cloudy)$
    - $P(weather = snow)$
  - Continuous random variables: the temperature has continuous values
    - Discretization: < 10, $[10, 20]$, > 20
    - Probability density function: e.g., Normal distribution.

Prior and Posterior Probabilities

- Before the evidence is obtained; prior probability
  - $P(a)$ the prior probability that the proposition is true
  - $P(rain) = 0.1$
- After the evidence is obtained; posterior probability
  - $P(a \mid b)$
  - The probability of *a* given that all we know is *b* (i.e., **conditional probability**)
  - $P(rain \mid cloudy) = 0.8$

Bayes' Theorem (Simple)

- The conditional probability of event C occurring, given event A, is
$$P(C|A) = \frac{P(A \cap C)}{P(A)}$$
  - E.g. A is an attribute and C is the class.
- Bayes' theorem for two events:
$$P(C|A) = \frac{P(A \cap C)}{P(A)} = \frac{P(C \cap A)}{P(A)} = \frac{P(A|C) \cdot P(C)}{P(A)}$$
  - It links the prior probabilities of two events and their posterior probabilities given each other.

Example

- Computing the probability that a patient carries a disease based on the result of a lab test.
- The test returns a positive result in 95% of the cases in which the disease is actually present, and it returns a positive result in 6% of the cases in which the disease is not present.
- Furthermore, 1% of the entire population has this disease.
- Let $C = \{\text{having the disease}\}$ and $A = \{\text{testing positive}\}$.
- From the above description, $P(C) = 0.01$, $P(\neg C) = 0.99$, $P(A|C) = 0.95$ and $P(A|\neg C) = 0.06$.

Reasoning with Bayes' Theorem

$$
\begin{aligned}
P\left(A\right) &= P\left(A \cap C\right) + P\left(A \cap \neg C\right) \\
&= P(C) \cdot P(A|C) + P(\neg C) \cdot P(A \mid \neg C) \\
&= 0.01 \times 0.95 + 0.99 \times 0.06 = 0.0689
\end{aligned}
$$

$$
P(C|A) = \frac{P(A|C)P(C)}{P(A)} = \frac{0.95 \times 0.01}{0.0689} \approx 0.1379
$$

Therefore, if some one has a test with positive result, he has 13.79% chance to carry the disease.

Bayes' Theorem (General)

- In a more general form, Bayes' theorem says that

$$P(Y|X_1, \dots, X_m) = \frac{P(X_1, \dots, X_m|Y) \cdot P(Y)}{P(X_1, \dots, X_m)}$$

- Linking it to classification: Y is the class and $X_1, \dots, X_m$ are attributes.
- E.g., features: *age*, *income*, *student_status*, *credit_rating*; class: *buys_computer*

RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes

Naïve Bayes Classifiers

- To apply Bayes theorem to classification, one main problem is *the number of combinations of attribute values*
  – If there are m attributes and each attribute has k values, there are m^k combinations! Impractical to keep track of their join probabilities.
- Recall $P(Y|X_1, \ldots, X_m) = \frac{P(X_1,\ldots,X_m|Y) \cdot P(Y)}{P(X_1,\ldots,X_m)}$
- We don't need to compute $P(X_1, \ldots, X_m)$ since we just want to find out *which class (value of Y) has the highest score by comparison*.
  – E.g., given *age=youth*, *income=high*, *student=no*, and *credit_rating=fair*, is *buys_computer=yes* more likely than *buys_computer=no* ?
    - In this case, we don't need to know the joint probability of *age=youth*, *income=high*, *student=no*, and *credit_rating=fair*
  – In other words, we just reply on

$$P(Y|X_1, \ldots, X_m) \propto P(X_1, \ldots, X_m|Y) \cdot P(Y)$$

where $\propto$ indicates "being propositional to".

Naïve Bayes Classifiers

- Still, $P(Y|X_1, \dots, X_m) = \frac{P(X_1,\dots,X_m|Y) \cdot P(Y)}{P(X_1,\dots,X_m)}$
- We use the **conditional independence** assumption.
  - Each attribute is conditionally independent of every other attribute given a class label
  - Namely, $P(X_1, \dots, X_m|Y) = P(X_1|Y) \cdots P(X_m|Y)$ which dramatically simplifies the computation of $P(X_1, \dots, X_m|Y)$
- Therefore, we are concerned with

$$P(Y|X_1, \dots, X_m) \propto P(X_1|Y) \cdots P(X_m|Y) \cdot P(Y)$$

Dataset Example

- Training tuples:

RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Illustration of Naïve Bayes Classifiers

- Let $\boldsymbol{X}$ denote

  (age = $youth$, $income$ = $medium$, $student$ = yes, $credit\ rating$ = $fair$)

- The objective is to determine which one is larger:

$P(\boldsymbol{X}|\ buys\_computer = yes\ ) \cdot P(buys\_computer = yes)$ OR $P(\boldsymbol{X}|\ buys\_computer = no) \cdot P(buys\_computer = no)$?

- We perform the following steps:
- First, the prior probability of each class can be computed based on the training tuples:

$$P(buys\_computer = yes) = 9/14 = 0.643$$
$$P(buys\_computer = no) = 5/14 = 0.357$$

Illustration of Naïve Bayes Classifiers

- Next, compute the conditional probabilities of attributes on the class labels:

$$
\begin{aligned}
P(age = youth \mid buys\_computer = yes) &= 2/9 = 0.222 \\
P(age = youth \mid buys\_computer = no) &= 3/5 = 0.600 \\
P(income = medium \mid buys\_computer = yes) &= 4/9 = 0.444 \\
P(income = medium \mid buys\_computer = no) &= 2/5 = 0.400 \\
P(student = yes \mid buys\_computer = yes) &= 6/9 = 0.667 \\
P(student = yes \mid buys\_computer = no) &= 1/5 = 0.200 \\
P(credit\_rating = fair \mid buys\_computer = yes) &= 6/9 = 0.667 \\
P(credit\_rating = fair \mid buys\_computer = no) &= 2/5 = 0.400
\end{aligned}
$$

Naïve Bayes Reasoning

- Next, using those probabilities, obtain:

$$
\begin{aligned}
P(\mathbf{X}|buys\_computer = yes) = {} & P(age = youth \mid buys\_computer = yes) \\
& \times P(income = medium \mid buys\_computer = yes) \\
& \times P(student = yes \mid buys\_computer = yes) \\
& \times P(credit\_rating = fair \mid buys\_computer = yes) \\
= {} & 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044.
\end{aligned}
$$

$$P(\mathbf{X}|buys\_computer = no) = 0.600 \times 0.400 \times 0.200 \times 0.400 = 0.019.$$

- Finally

$$P(\mathbf{X}|buys\_computer = yes)P(buys\_computer = yes) = 0.044 \times 0.643 = 0.028$$
$$P(\mathbf{X}|buys\_computer = no)P(buys\_computer = no) = 0.019 \times 0.357 = 0.007$$

- Therefore, the classifier predicts *buys_computer = yes*

Numerical Underflow

- If the number of attributes is large, the outputs of a Naïve Bayesian classifier are usually very small.
- In theory this is not a problem, because only the ratio between the outputs matters; however, in practical, the difference may be close or rounded off to 0 (this is unknown as the ***underflow*** problem).
- To avoid this, one widely used treatment is to manipulate a logarithm of a number rather than the number itself. Therefore,
- Thus $p_* = p_1 \cdots p_m$ becomes $\log(p_*) = \log(p_1) + \cdots + \log(p_m)$
  - The ratio between the output values of the classifier is not distorted!
  - As multiplication becomes $+$, the underflow is avoided.

Smoothing Zero Count

- Another problem is the ***zero count***: the count of records with a value of an attribute is zero when some class label is given
- If the zero count occurs, then one of $P(X_1|Y), \ldots, P(X_m|Y)$ is zero, and their multiplication is zero (no matter how large the rest are)
  - This is certainly counter-intuitive
  - Also, applying the log function to a zero probability, log(0) is negative infinite
- One common technique to overcome this is the ***Laplace smoothing*** (or add-one) technique: it adds 1 to all counts.
  - Because usually the training dataset is large (i.e., the total count is large), adding 1 to each count causes minimum effect
  - But if it would cause effect, add a very small number $\varepsilon > 0$ instead of 1.

Smoothing Zero Count

- Suppose that for the class *buys computer* = *yes* in some training database, *D*, containing 1000 tuples. We have 0 tuple with *income* = *low*, 990 tuples with *income* = *medium*, and 10 tuples with *income* = *high*.
- Without the Laplacian smoothing, the probabilities of those events are 0, 0.990 (from 990/1000) and 0.010 (from 10/1000), respectively.
- If a tuple has *income* = *low*, the probability of falling into the class *buys computer* = *yes* is 0, no matter what values for other attributes!
- With the Laplacian smoothing for the three quantities, adding 1 more tuple for each income value: the probabilities become 0.001 (from 1/1003), 0.988 (from 991/1003) and 0.011 (form 11/1003).
- The above phenomenon won't happen.

NB Implementation with Scikit-Learn

- NB implementation by using the **CategoricalNB** API in Scikit-learn
  - See the supplementary materials

Continuous-Value Features

- We now consider an extension to Naïve Bayesian classifiers which are able to handle continuous-value features.
- If X is continuous, there are two common approaches to compute $P(X = a \mid Y = c)$:
  - ***Discretization/bucketing/binning***: The range of X is $(-\infty, a_1], [a_2, b_1], \dots, [a_k, b_{k-1}], [b_k, +\infty)$ for some k.
  - Assume that X has a **Gaussian distribution** (a.k.a. normal distribution).
- The following is the *probability density function* (PDF) of a Gaussian distribution with mean μ and variance σ^2:

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

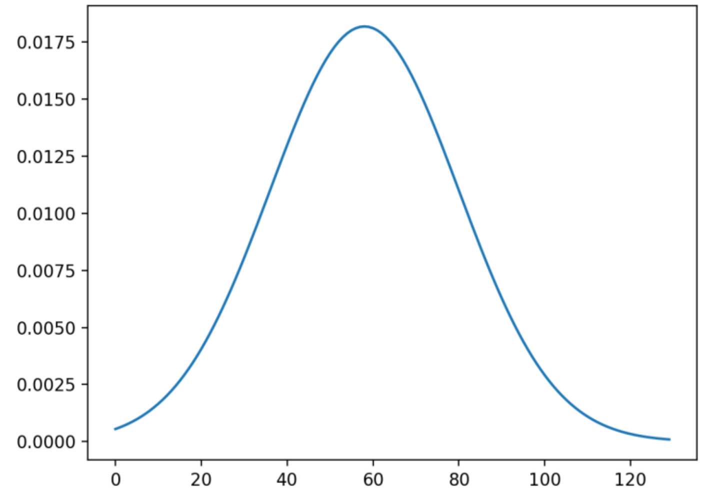
  - ➢ If we compute the mean value μ_0 and standard deviation σ_0 based on the training data for X when $Y = c$, then $P(X = x|Y = c)$ *is* $f(x, \mu_0, \sigma_0)$

Continuous-Value Features

- Estimation of mean and variance: Given observations $[x_1, \dots, x_N]$
  - mean $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$
  - Variance $\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2 = \left(\frac{1}{N}\sum_{i=1}^{N} {x_i}^2\right) - \mu^2$
- For example, if the incomes are not discretized in the costumer data and are 30, 36, 47, 50, 56, 60, 63, 70, 110 (K dollars) when *buys_computers* = *yes*, then
  - the mean is 58K and
  - the variance is 481.56
- Then

  $P(income = 47 | \; buys\_computers = yes)$

  $is \; \frac{1}{\sqrt{2\pi}\cdot 21.94} e^{-\frac{(47-58)^2}{2\cdot 481.56}} = \; 0.016$

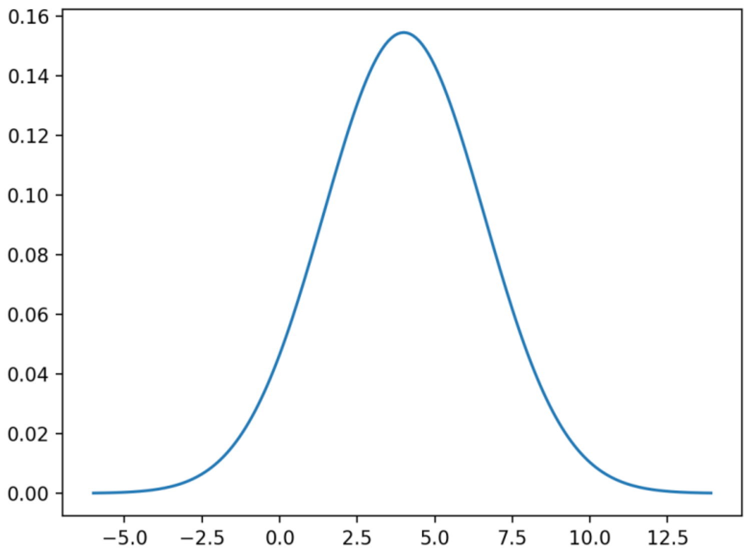


Continuous-Value Features

- To reason about PDF of the Gaussian distribution, we can use the norm package of the scipy.stats libarary :

  https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.norm.html#scipy.stats.norm

```
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as pyplt
a = range(9)
mu = np.mean(a) # mean
sigma = (np.var(a))**0.5
        # standard deviation
x = np.arange(-6, 14, 0.1)
y = norm.pdf(x, mu, sigma)
pyplt.plot(x,y)
pyplt.show()
```



*Continuous-Value Features

- The previous example provides an interpretation is somehow “over simplistic”, since the probability that a continuous random variable takes a particular value is zero.
- Instead, we should compute the conditional probability that X lies within some interval, say, $[r, r+\epsilon]$, where ϵ is a small constant:

$$P(r \leq X \leq r+\epsilon) = \int_{r}^{r+\epsilon} f\,(X, \mu, \sigma) dX \approx f(X, \mu, \sigma) \cdot \epsilon$$

- Since ϵ appears as a constant multiplicative factor for each class, it *cancels out* when normalizing the target probability, leaving just the $f(X, \mu, \sigma)$ part.

NB Implementation with Scikit-Learn

- NB implementation by using the **GaussianNB** API in Scikit-learn
  - See the supplementary materials

NB Classifier: Advantages / Disadvantages

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayes Classifier
  - How to deal with these dependencies?
    - Bayesian Belief Network
  - From correlation to causality?