Applied Data Science

Session 22: Naïve Bayes Classifier

Dr. Soharab Hossain Shaikh

1

Probability Basics

- Prior, conditional and joint probability for random variables
  - Prior probability: $P(x)$
  - Conditional probability: $P(x_1 \mid x_2), P(x_2 \mid x_1)$
  - Joint probability: $\mathbf{x} = (x_1, x_2),\ P(\mathbf{x}) = P(x_1, x_2)$
  - Relationship: $P(x_1, x_2) = P(x_2 \mid x_1)P(x_1) = P(x_1 \mid x_2)P(x_2)$
  - Independence: $P(x_1, x_2) = P(x_1)P(x_2 \mid x_1) = P(x_2)P(x_1 \mid x_2) = P(x_1)P(x_2)$
- Bayesian Rule

$$P(c \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid c)P(c)}{P(\mathbf{x})} \qquad Posterior = \frac{Likelihood \times Prior}{Evidence}$$

2

2

Conditional Probability

It is the probability that an event has occurred (not yet observed) given another event has occurred. e.g.

i) given a card drawn is red P(Y)(an event has occurred)

ii) what is the probability that the card is a king P(X|Y) (event not yet observed)

Since the card is red - there are 26 such red cards in a deck of cards.

Of these, 26 possible cards we are interested in a king - 2 such kings are there (one of heart one of diamond).

Thus the conditional probability is, P(X|Y) = 2/26=1/13.

3

3

Joint Probability

Joint probability is the probability of multiple events occurring together (we are not talking of causality here i.e one event leads to another).

For e.g.

i) probability of drawing a red colored card from a deck of cards is 26/52 = 1/2 (P(Y))

ii) probability of drawing a **king** from a deck of all cards is 4/52 = 1/13 (P(X))

iii) Probability of drawing a **king** given that the deck contains only the **red colored** cards, conditional probability P(X|Y) = 2/26 = 1/13

4

4

Joint Probability

Compare this with the joint probability P(**X**,**Y**) = probability of drawing a **king** from the deck of only **red colored** cards.

Relation between joint probability and conditional probability (when two events are not independent):

P(**X**,**Y**) **=** P(**X**|**Y**) * P(**Y**) **=** 1/13 * 1/2 = 1/26

5

Joint Probability

When two events are **not dependent (independent events)**:

P(X,Y) = P(X) * P(Y)

The joint probability for two events, A and B, is expressed mathematically as P(A,B). Joint probability is calculated by multiplying the probability of event A, expressed as P(A), by the probability of event B, expressed as P(B).

Example:

Suppose we wish to know the probability that the number five will occur twice when two six-sided dice are rolled at the same time. Since each die has six possible outcomes, the probability of a five occurring on each die is 1/6 or 0.1666.

P(A)=0.1666 and P(B)=0.1666

P(A,B) = 0.1666 x 0.1666 = 0.02777

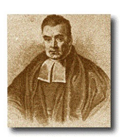
This means the joint probability that a five will be rolled on both dice at the same time is 0.02777.

6

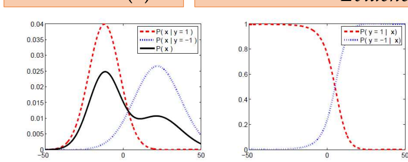
6

Bayes Formula

Bayes, Thomas (1763) An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London,* **53:370-418**



$$P(c\,|\,\mathbf{x}) = \frac{P(\mathbf{x}\,|\,c)P(c)}{P(\mathbf{x})} \qquad Posterior = \frac{Likelihood \times Prior}{Evidence}$$



9

7

Probabilistic Classification Principle

- Maximum A Posterior (MAP) classification rule
  - For an input $\boldsymbol{x}$, find the largest one from L probabilities output by a **probabilistic classifier** $P(c_1\,|\,\mathbf{x}), \ldots, P(c_L\,|\,\mathbf{x})$.
  - Assign $\boldsymbol{x}$ to label c^* if $P(c^*\,|\,\mathbf{x})$ is the largest.
- **Classification with the MAP rule**
  - Apply Bayesian rule to convert them into posterior probabilities

$$P(c_i\,|\,\mathbf{x}) = \frac{P(\mathbf{x}\,|\,c_i)P(c_i)}{P(\mathbf{x})} \propto P(\mathbf{x}\,|\,c_i)P(c_i)$$

$$\text{for } i = 1, 2, \cdots, L$$

Common factor for all L probabilities

  - Then apply the MAP rule to assign a label

8

8

Naïve Bayes Classifier

- Bayes classification

$$P(c \mid \mathbf{x}) \propto P(\mathbf{x} \mid c)P(c) = P(x_1, \cdots, x_n \mid c)P(c) \text{ for } c = c_1, \ldots, c_L.$$

Difficulty: learning the joint probability $P(x_1, \cdots, x_n \mid c)$ is often infeasible!

- **Naïve Bayes classification**
  - Assume all input features are class conditionally independent!

$$P(x_1, x_2, \cdots, x_n \mid c) = P(x_1 \mid x_2, \cdots, x_n, c)P(x_2, \cdots, x_n \mid c)$$
$$= P(x_1 \mid c)P(x_2, \cdots, x_n \mid c)$$
$$= P(x_1 \mid c)P(x_2 \mid c) \cdots P(x_n \mid c)$$

Applying the independence assumption

  - Apply the MAP classification rule: assign $\mathbf{x}' = (a_1, a_2, \cdots, a_n)$ to c^* if

$$\underbrace{[P(a_1 \mid c^*) \cdots P(a_n \mid c^*)]}_{\text{estimate of } P(a_1, \cdots, a_n \mid c^*)} P(c^*) > \underbrace{[P(a_1 \mid c) \cdots P(a_n \mid c)]}_{\text{esitmate of } P(a_1, \cdots, a_n \mid c)} P(c), \quad c \neq c^*, c = c_1, \cdots, c_L$$

9

9

Naïve Bayes Algorithm for Classification

- Algorithm: Discrete-Valued Features
  - Learning Phase: Given a training set S of F features and L classes,

For each target value of c_i $(c_i = c_1, \cdots, c_L)$
$\hat{P}(c_i) \leftarrow$ estimate $P(c_i)$ with examples in S;
For every feature value x_{jk} of each feature x_j $(j = 1, \cdots, F; k = 1, \cdots, N_j)$
$\hat{P}(x_j = x_{jk} \mid c_i) \leftarrow$ estimate $P(x_{jk} \mid c_i)$ with examples in S;

Output: $F * L$ conditional probabilistic (generative) models

  - Test Phase: Given an unknown instance $\mathbf{x}' = (a'_1, \cdots, a'_n)$

"Look up tables" to assign the label c^* to $\mathbf{X}'$ if

$$[\hat{P}(a'_1 \mid c^*) \cdots \hat{P}(a'_n \mid c^*)]\hat{P}(c^*) > [\hat{P}(a'_1 \mid c_i) \cdots \hat{P}(a'_n \mid c_i)]\hat{P}(c_i), \quad c_i \neq c^*, c_i = c_1, \cdots, c_L$$

10

10

Example

- Example: Play Tennis

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

11

Example

- Learning Phase

Outlook	Play=*Yes*	Play=*No*
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=*Yes*	Play=*No*
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=*Yes*	Play=*No*
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=*Yes*	Play=*No*
Strong	3/9	3/5
Weak	6/9	2/5

P(Play=*Yes*) = 9/14 $\quad$ P(Play=*No*) = 5/14

12

12

Example

- Test Phase
  - Given a new instance, predict its label
    x'=(Outlook=*Sunny*, Temperature=*Cool*, Humidity=*High*, Wind=*Strong*)
  - Look up tables achieved in the learning phrase

P(Outlook=*Sunny*|Play=*Yes*) = 2/9	P(Outlook=*Sunny*|Play=*No*) = 3/5
P(Temperature=*Cool*|Play=*Yes*) = 3/9	P(Temperature=*Cool*|Play==*No*) = 1/5
P(Huminity=*High*|Play=*Yes*) = 3/9	P(Huminity=*High*|Play=*No*) = 4/5
P(Wind=*Strong*|Play=*Yes*) = 3/9	P(Wind=*Strong*|Play=*No*) = 3/5
P(Play=*Yes*) = 9/14	P(Play=*No*) = 5/14

$P(Yes|\mathbf{x}') \approx [P(Sunny|Yes)P(Cool|Yes)P(High|Yes)P(Strong|Yes)]P(\text{Play}=Yes) = 0.0053$
$P(No|\mathbf{x}') \approx [P(Sunny|No)\,P(Cool|No)P(High|No)P(Strong|No)]P(\text{Play}=No) = 0.0206$

Given the fact P(*Yes*|**x**') < P(*No*|**x**'), we label **x**' to be "No".

13

Naïve Bayes Classifier

- Algorithm: Continuous-valued Features
  - Numeric values taken by a continuous-valued feature
  - Conditional probability is often modelled with the normal distribution

$$\hat{P}(x_j|c_i) = \frac{1}{\sqrt{2\pi}\sigma_{ji}} \exp\left(-\frac{(x_j-\mu_{ji})^2}{2\sigma_{ji}^2}\right)$$

μ_{ji} : mean (avearage) of feature values x_j of examples for which $c = c_i$
σ_{ji} : standard deviation of feature values x_j of examples for which $c = c_i$

  - **Learning Phase:** for $\mathbf{X} = (X_1, \cdots, X_F),\ C = c_1, \cdots, c_L$
    Output: $F \times L$ normal distributions and $P(C = c_i)\ i = 1, \cdots, L$
  - **Test Phase:** Given an unknown instance $\mathbf{X}' = (a_1', \cdots, a_n')$
    - **Instead of looking-up tables, calculate conditional probabilities with all the normal distributions achieved in the learning phrase**
    - **Apply the MAP rule to assign a label (the same as done for the discrete case)**

14

14

Naïve Bayes Classifier

- Example: Continuous-valued Features
  - **Temperature** is naturally of continuous value.
    Yes: 25.2, 19.3, 18.5, 21.7, 20.1, 24.3, 22.8, 23.1, 19.8
    No: 27.3, 30.1, 17.4, 29.5, 15.1
  - Estimate mean and variance for each class

$$\mu = \frac{1}{N}\sum_{n=1}^{N} x_n, \quad \sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu)^2$$

$\mu_{Yes} = 21.64,\ \sigma_{Yes} = 2.35$
$\mu_{No} = 23.88,\ \sigma_{No} = 7.09$

N: no. of samples belonging to that class. N=9 for 'Yes' N=5 for 'No'

  - **Learning Phase**: output two Gaussian models for P(temp|C)

$$\hat{P}(x|Yes) = \frac{1}{2.35\sqrt{2\pi}}\exp\left(-\frac{(x-21.64)^2}{2\times 2.35^2}\right) = \frac{1}{2.35\sqrt{2\pi}}\exp\left(-\frac{(x-21.64)^2}{11.09}\right)$$

$$\hat{P}(x|No) = \frac{1}{7.09\sqrt{2\pi}}\exp\left(-\frac{(x-23.88)^2}{2\times 7.09^2}\right) = \frac{1}{7.09\sqrt{2\pi}}\exp\left(-\frac{(x-23.88)^2}{50.25}\right)$$

15

15

Zero Conditional Probability

- If no example contains the feature value
  - In this circumstance, we face a zero conditional probability problem during test

$$\hat{P}(x_1|c_i)\cdots\hat{P}(a_{jk}|c_i)\cdots\hat{P}(x_n|c_i) = 0 \quad \text{for } x_j = a_{jk},\ \hat{P}(a_{jk}|c_i) = 0$$

  - For a remedy, class conditional probabilities re-estimated with

$$\hat{P}(a_{jk}|c_i) = \frac{n_c + mp}{n + m} \quad \textbf{(m-estimate)}$$

n_c : number of training examples for which $x_j = a_{jk}$ and $c = c_i$
n : number of training examples for which $c = c_i$
p : prior estimate (usually, $p = 1/t$ for t possible values of x_j)
m : weight to prior (number of "virtual" examples, $m \geq 1$)

16

16

Zero Conditional Probability

- Example: P(outlook=overcast|no)=0 in the play-tennis dataset
  - Adding ***m*** "virtual" examples (m: tunable but up to 1% of #training examples)
    - In this dataset, # of training examples for the "no" class is 5.
    - Assume that we add ***m=1*** "virtual" example in our m-estimate treatment.
  - The "outlook" feature can take only 3 values. So ***p=1/3***.
  - Re-estimate P(outlook|no) with the m-estimate

$$P(\text{overcast}|\text{no}) = \frac{0+1*\left(\frac{1}{3}\right)}{5+1} = \frac{1}{18}$$

$$P(\text{sunny}|\text{no}) = \frac{3+1*\left(\frac{1}{3}\right)}{5+1} = \frac{5}{9} \qquad P(\text{rain}|\text{no}) = \frac{2+1*\left(\frac{1}{3}\right)}{5+1} = \frac{7}{18}$$

17

17

Computational Consideration

- Multiplying lots of probabilities, which are between 0 and 1 by definition, can result in floating-point underflow.
- Since $\log(xy) = \log(x) + \log(y)$, it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities.
- Class with highest final un-normalized log probability score is still the most probable.

$$c_{NB} = \underset{c_j \in Classes}{\operatorname{argmax}} \left(\log P(c_j) + \sum_{i \in features} \log P(x_i | c_j) \right)$$

18

18

Computational Consideration

Assume that instance X described by n-dimensional vector of attributes $X = \langle x_1, x_2, \ldots, x_n \rangle$

then

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}}\ P(c | x_1, x_2, \ldots, x_n)$$

$$= \underset{c \in C}{\operatorname{argmax}} \frac{P(x_1, x_2, \ldots, x_n | c) P(c)}{P(x_1, x_2, \ldots, x_n)}$$

$$= \underset{c \in C}{\operatorname{argmax}}\ P(x_1, x_2, \ldots, x_n | c) P(c) = \arg\max_c P(c) \prod_{i=1}^{N} P(x_i | c)$$

$$c_{NB} = \underset{c_j \in Classes}{\operatorname{argmax}} \left(\log P(c_j) + \sum_{i \in features} \log P(x_i | c_j) \right)$$

Log

19

19

How to ***build*** a Naïve Bayes Model in ***Scikit-Learn***

20

Naïve Bayes in Scikit-Learn

sklearn.naive_bayes.GaussianNB

class `sklearn.naive_bayes.GaussianNB`(**, priors=None, var_smoothing=1e-09*) [source]

Gaussian Naive Bayes (GaussianNB)

Can perform online updates to model parameters via `partial_fit`. For details on algorithm used to update feature means and variance online, see Stanford CS tech report STAN-CS-79-773 by Chan, Golub, and LeVeque:

- Build an estimator model like **GaussianNB()**
- Use **fit()** function to ***Train*** the model with **Training Dataset**
- Use **predict()** function to ***Test/Evaluate*** the model with **Test Dataset**
- Use **predict()** function to make ***prediction/inference*** on **New Unseen Data**

21

Gaussian Naive Bayes

`GaussianNB` implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

The parameters σ_y and μ_y are estimated using maximum likelihood.

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.naive_bayes import GaussianNB
>>> X, y = load_iris(return_X_y=True)
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
>>> gnb = GaussianNB()
>>> y_pred = gnb.fit(X_train, y_train).predict(X_test)
>>> print("Number of mislabeled points out of a total %d points : %d"
...       % (X_test.shape[0], (y_test != y_pred).sum()))
Number of mislabeled points out of a total 75 points : 4
```

22

Multinomial Naive Bayes

`MultinomialNB` implements the naive Bayes algorithm for multinomially distributed data, and is one of the two classic naive Bayes variants used in text classification (where the data are typically represented as word vector counts, although tf-idf vectors are also known to work well in practice). The distribution is parametrized by vectors $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$ for each class y, where n is the number of features (in text classification, the size of the vocabulary) and θ_{yi} is the probability $P(x_i \mid y)$ of feature i appearing in a sample belonging to class y.

The parameters θ_y is estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

where $N_{yi} = \sum_{x \in T} x_i$ is the number of times feature i appears in a sample of class y in the training set T, and $N_y = \sum_{i=1}^{n} N_{yi}$ is the total count of all features for class y.

The smoothing priors $\alpha \geq 0$ accounts for features not present in the learning samples and prevents zero probabilities in further computations. Setting $\alpha = 1$ is called Laplace smoothing, while $\alpha < 1$ is called Lidstone smoothing.

23

Complement Naive Bayes

`ComplementNB` implements the complement naive Bayes (CNB) algorithm. CNB is an adaptation of the standard multinomial naive Bayes (MNB) algorithm that is particularly suited for imbalanced data sets. Specifically, CNB uses statistics from the *complement* of each class to compute the model's weights. The inventors of CNB show empirically that the parameter estimates for CNB are more stable than those for MNB. Further, CNB regularly outperforms MNB (often by a considerable margin) on text classification tasks. The procedure for calculating the weights is as follows:

$$\hat{\theta}_{ci} = \frac{\alpha_i + \sum_{j:y_j \neq c} d_{ij}}{\alpha + \sum_{j:y_j \neq c} \sum_k d_{kj}}$$
$$w_{ci} = \log \hat{\theta}_{ci}$$
$$w_{ci} = \frac{w_{ci}}{\sum_j |w_{cj}|}$$

where the summations are over all documents j not in class c, d_{ij} is either the count or tf-idf value of term i in document j, α_i is a smoothing hyperparameter like that found in MNB, and $\alpha = \sum_i \alpha_i$. The second normalization addresses the tendency for longer documents to dominate parameter estimates in MNB. The classification rule is:

$$\hat{c} = \arg\min_c \sum_i t_i w_{ci}$$

i.e., a document is assigned to the class that is the *poorest* complement match.

24

Bernoulli Naive Bayes

`BernoulliNB` implements the naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions; i.e., there may be multiple features but each one is assumed to be a binary-valued (Bernoulli, boolean) variable. Therefore, this class requires samples to be represented as binary-valued feature vectors; if handed any other kind of data, a `BernoulliNB` instance may binarize its input (depending on the `binarize` parameter).

The decision rule for Bernoulli naive Bayes is based on

$$P(x_i \mid y) = P(i \mid y) x_i + (1 - P(i \mid y))(1 - x_i)$$

which differs from multinomial NB's rule in that it explicitly penalizes the non-occurrence of a feature i that is an indicator for class y, where the multinomial variant would simply ignore a non-occurring feature.

In the case of text classification, word occurrence vectors (rather than word count vectors) may be used to train and use this classifier. `BernoulliNB` might perform better on some datasets, especially those with shorter documents. It is advisable to evaluate both models, if time permits.

25

Categorical Naive Bayes

`CategoricalNB` implements the categorical naive Bayes algorithm for categorically distributed data. It assumes that each feature, which is described by the index i, has its own categorical distribution.

For each feature i in the training set X, `CategoricalNB` estimates a categorical distribution for each feature i of X conditioned on the class y. The index set of the samples is defined as $J = \{1, \dots, m\}$, with m as the number of samples.

The probability of category t in feature i given class c is estimated as:

$$P(x_i = t \mid y = c \,;\, \alpha) = \frac{N_{tic} + \alpha}{N_c + \alpha n_i},$$

where $N_{tic} = |\{j \in J \mid x_{ij} = t, y_j = c\}|$ is the number of times category t appears in the samples x_i, which belong to class c, $N_c = |\{j \in J \mid y_j = c\}|$ is the number of samples with class c, α is a smoothing parameter and n_i is the number of available categories of feature i.

`CategoricalNB` assumes that the sample matrix X is encoded (for instance with the help of `OrdinalEncoder`) such that all categories for each feature i are represented with numbers $0, \dots, n_i - 1$ where n_i is the number of available categories of feature i.

26

27

Naïve Bayes Classifier
Case-study

28

Case Study on Naive Bayes Classifier

Objective:

To predict whether income exceeds 50K/yr based on census data.

Variable description:

Age: continuous

Workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.

Fnlwgt: continuous.

Education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.

Education-num: continuous.

Marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.

29

29

Case Study on Naive Bayes Classifier

Occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.

Relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.

Race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.

Sex: Female, Male.

Capital-gain: continuous.

Capital-loss: continuous.

Hours-per-week: continuous.

Native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, etc.

30

30

Let's go to the Coding Demo…

31

To be continued in the next session…..

32