Lesson B-4

Bayesian Classification

Probability Basics for Data Mining

- The branch of mathematics that deals with measuring the likelihood (or uncertainty) around events to predict future events based on their likelihood
- Computing the likelihood of a future event
  - Use the relative frequency of the event in the past
  - In a predictive analytics context, the <u>*relative frequency*</u> is a standard approach

Terminology of Probability

- Probability
- Experiment
- Sample space
- Outcome
- Event
- Random variable
- Probability (mass/density) function

Map the Terminology of Probability into the Language of Predictive Analysis

- Random variables → features (attributes)
- Sample space → the set of all possible combinations of assignments of values to features
- An experiment whose outcome has been already been stored → a row in dataset
- An experiment whose outcome we do not yet know but would like to predict → the prediction task for which we are building a model
- An event is any subset of an experiment → an event may describe an assignment of values to all the features in the domain (a full row in the dataset) or an assignment to one or more features in the domain
- The value returned by a probability function for an event → the **relative frequency** of that event in the dataset

A Dataset of Instances from The Sample Space of Two Random Variables

ID	DICE 1	DICE2
1	3	4
2	1	5
3	6	5
4	3	3
5	1	1

Bayes' Theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

– Where A and B are events and $P(B) \neq 0$

Properties of Probability Mass Function

All probability should satisfy the following axioms:

- For any event A, $0 \leq P(A) \leq 1$
- If A_1, A_2, …. , A_n is a finite collection of mutually exclusive events, then

$$P(A_1 \bigcup A_2 \bigcup \cdots \bigcup A_n) = \sum_{i=1}^{n} P(A_i)$$

Union, Intersection, Complement, Mutually Exclusive of Events

- The **union** of two events *A* and *B*, denoted by *A* U *B* is the event consisting of all outcomes that either in *A* or in *B* or in both events.
- The **intersection** of two events *A* and *B*, denoted by $A \cap B$ is the event consisting of all outcomes that are in both *A* and *B*.
- The **complement** of an event, denoted by *A′*, is the set of all outcomes in *S* that are not contained in *A*.
- When *A* and *B* have no outcomes in common, they are said to be **mutually exclusive** or **disjoint** events.

Random Variables

- A **random variable** represents the outcome of a probabilistic experiment.
  - Each random variable has a range of possible values (outcomes).
- A random variable is said to be **discrete** if its set of possible values is discrete set.
  - Possible outcomes of a random variable `Mood`: *Happy* and *Sad*
  - Each outcome has a probability.
    - The probabilities for all possible outcomes must sum to 1.
  - For example:
    - P(Mood=*Happy*) = 0.7
    - P(Mood=*Sad*) = 0.3

Joint Probability

- Joint probability is the probability of two events happening together.
- The two events are usually designated event A and event B.
  - In probability terminology, it can be written as:
  - p(A and B) or
  - $p(A \cap B)$
- Joint probability can also be described as the probability of the intersection of two (or more) events.
  - The intersection can be represented by a Venn diagram

Joint Probability

- The `Mood` can take 2 possible values: *happy, sad*.
- The `Weather` can take 3 possible vales: *sunny, rainy, cloudy.*
- Lets say we know:
  - P(Mood=*happy* ∩ Weather=*rainy*) = 0.25
  - P(Mood=*happy* ∩ Weather=*sunny*) = 0.4
  - P(Mood=*happy* ∩ Weather=*cloudy*) = 0.05

Joint Probabilities

	Sunny	Rainy	Cloudy
Happy	0.4	0.25	0.05
Sad	0.05	0.1	0.15

- P(Mood=*Happy*) = 0.25 + 0.4 + 0.05 = 0.7
- P(Mood=*Sad*) = ?
- Two random variables *A* and *B*
  - A has *m* possible outcomes $A_1, \ldots, A_m$
  - B has *n* possible outcomes $B_1, \ldots, B_n$

$$P(A = A_i) = \sum_{j=1}^{n} P((A = A_i) \bigcap (B = B_j))$$

Joint Probabilities

	Sunny	Rainy	Cloudy
Happy	0.4	0.25	0.05
Sad	0.05	0.1	0.15

- P(Weather=*Sunny*)=?
- P(Weather=*Rainy*)=?
- P(Weather=*Cloudy*)=?

$$P(B = B_i) = \sum_{j=1}^{m} P((A = A_j) \bigcap (B = B_i))$$

Joint Probabilities

ID	DICE 1	DICE2
1	3	4
2	1	5
3	6	5
4	3	3
5	1	1

- *In terms of rows in a dataset*, the probability of a joint event computation is simply the number of rows where the set of assignments listed in the joint event holds divided by the total number of rows in the dataset.

Conditional Probability

- In probability theory, conditional probability is a measure of the probability of an event occurring, given that another event (by assumption, presumption, assertion or evidence) has already occurred
- For any two events *A* and *B* with P(*B*) > 0, the conditional probability of *A* given that *B* has occurred is defined by

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

Conditional Probability

- $P(A = A_i \mid B = B_j)$ represents the probability of $A = A_i$ given that we know $B = B_j$. This is called **conditional probability**.

$$P(A = A_i \mid B = B_j) = \frac{P((A = A_i) \bigcap (B = B_j))}{P(B = B_j)}$$

Conditional Probability

	Sunny	Rainy	Cloudy
Happy	0.4	0.25	0.05
Sad	0.05	0.1	0.15

- P(Mood=Happy|Weather=Sunny) = ?
- P(Mood=Happy|Weather=Rainy) = ?
- P(Mood=Happy|Weather=Cloudy) = ?
- P(Weather=Cloudy|Mood= Sad) = ?

Conditional Probabilities

ID	DICE 1	DICE2
1	3	4
2	1	5
3	6	5
4	3	3
5	1	1

- $P\ (DICE1 = 6 \mid DICE2 = 5) =$

Conditional Probability

- $P(A \mid B) = 1$ is equivalent to $B \Rightarrow A$.
  - Knowing the value of B exactly determines the value of A.
- For example, suppose my dog rarely howls:

$$P(\text{MyDogHowls}) = 0.01$$

- But when there is a full moon, he always howls:

$$P(\text{MyDogHowls} \mid \text{FullMoon}) = 1.0$$

Conditional Probability and Product Rule

Conditional probability:

$$P(A \mid B) = \frac{P(A \bigcap B)}{P(B)}$$

Product rule:

$$P(A \bigcap B) = P(A \mid B)P(B) = P(B \mid A)P(A)$$

Independence

- Two random variables *A* and *B* are said to be **independent** if and only if P(*A* ∩ *B*) = P(*A*)P(*B*).
- Conditional probabilities for independent A and B:

$$P(A \mid B) = \frac{P(A \bigcap B)}{P(B)} = \frac{P(A)P(B)}{P(B)} = P(A)$$

$$P(B \mid A) = \frac{P(A \bigcap B)}{P(A)} = \frac{P(A)P(B)}{P(A)} = P(B)$$

- Product rule for independent events
  - If A and B are independent, P(A ∩ B)=P(A)P(B)

Bayes Theorem

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Where A and B are events and P(B) ≠ 0

The Law of Total Probability

Let $A_1, A_2, \ldots, A_n$ be a collection of n mutually exclusive and exhaustive events with $P(A_i) > 0$ for $i = 1, \ldots, n$. Then for any other event B for which $P(B) > 0$

$$P(B)$$

$$= P(B \mid A_1)P(A_1) + \ldots. + P(B \mid A_n)P(A_n)$$

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

Conditional Probability and The Law of Total Probability

Let $A_1, A_2, \ldots, A_n$ be a collection of n mutually exclusive and exhaustive events with $P(A_i) > 0$ for $i = 1, \ldots, n$. Then for any other event B for which $P(B) > 0$

$$P(A_k \mid B) = \frac{P(B \mid A_k)P(A_k)}{P(B)} = \frac{P(B \mid A_k)P(A_k)}{\sum_{i=1}^{n} P(B \mid A_i)P(A_i)} \quad k = 1, \ldots, n$$

Prior Probabilities and Posterior Probabilities

- The type of probabilities we have calculated so far are known as **prior probabilities** or **unconditional probabilities**.
- We want to know the probability of an event in the context where one or more other events are known to have happened.
  - This type of probability, where we take one or more events to already hold, is known as a **posterior probability**, because it is calculated *after* other events have happened.
  - It is also commonly known as a **conditional probability**, because the probability calculated is valid conditional on the given events (or evidence).

Bayesian Classification

- A statistical 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 selected neural network classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data

Bayes' Theorem in Data Analytics

- Bayes' Theorem:

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$

  – Let **X** be a data sample ("*evidence*"): class label is unknown
  – Let H be a *hypothesis* that X belongs to class C
  – Classification is to determine P(H|**X**), (i.e., *posteriori probability):* the probability that the hypothesis holds given the observed data sample **X**
  – P(H) (*prior probability*): the initial probability
    - E.g., **X** will buy computer, regardless of age, income, …
  – P(**X**): probability that sample data is observed
  – P(**X**|H) (likelihood): the probability of observing the sample **X**, given that the hypothesis holds
    - e.g., Given that **X** will buy computer, the prob. that X is youth, medium income

Prediction Based on Bayes' Theorem

- Given training data **X**, *posteriori probability of a hypothesis* H, P(H|**X**), follows the Bayes' theorem

$$P(H \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid H) P(H)}{P(\mathbf{X})} = P(\mathbf{X} \mid H) \times P(H) / P(\mathbf{X})$$

- Informally, this can be viewed as

  posteriori = likelihood x prior/evidence

- Predicts **X** belongs to C_i iff the probability $P(C_i|\mathbf{X})$ is the highest among all the $P(C_k|X)$ for all the *k* classes
- Practical difficulty: It requires initial knowledge of many probabilities, involving significant computational cost

Classification is to Derive the Maximum Posteriori

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector **X** = (x_1, x_2, …, x_n)

$$P(\mathbf{X} \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i) = P(x_1 \mid C_i) \times P(x_2 \mid C_i) \times \ldots \times P(x_n \mid C_i)$$

- Suppose there are *m* classes C_1, C_2, …, C_m.
- Classification is to derive the maximum posteriori, i.e., the maximal $P(C_i|\mathbf{X})$
- This can be derived from Bayes' theorem

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- Since P(X) is constant for all classes, only

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

needs to be maximized

Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \ldots \times P(x_n|C_i)$$

- This greatly reduces the computation cost: Only counts the class distribution
- If A_k is categorical, $P(x_k|C_i)$ is the # of tuples in C_i having value x_k for A_k divided by $|C_{i,D}|$ (# of tuples of C_i in D)
- If A_k is continuous-valued, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

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

and $P(x_k|C_i)$ is

$$P(\mathbf{X}|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

Naïve Bayesian Classifier: Training Dataset

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

Class-labeled training tuples from AllElectronics customer database.

Naïve Bayesian Classifier: Training Dataset

- Class:

  C1:*buys_computer* = yes

  C2:*buys_computer* = no
- Data sample

  X = (*age =youth, income = medium, student = yes, credit_rating = fair*)

Naïve Bayesian Classifier: An Example

- $P(C_i)$: P(*buys_computer* = yes) = 9/14 = 0.643

  P(*buys_computer* = no) = 5/14= 0.357

- Compute $P(\mathbf{X}|C_i)$ for each class

P(*age* = youth | *buys_computer* = yes) = 2/9 = 0.222

P(*income* = medium | *buys_computer* = yes) = 4/9 = 0.444

P(*student* = yes | *buys_computer* = yes) = 6/9 = 0.667

P(*credit_rating* = fair | *buys_computer* = yes) = 6/9 = 0.667

P(*age* = youth | *buys_computer* = no) = 3/5 = 0.6

P(*income* = medium | *buys_computer* = no) = 2/5 = 0.4

P(*student* = yes | *buys_computer* = no) = 1/5 = 0.2

P(*credit_rating* = fair | *buys_computer* = no) = 2/5 = 0.4

Naïve Bayesian Classifier: An Example

X = *(age* = youth*, income* = medium*, student* = yes*, credit_rating* = fair)

P(**X**|C_i) : P(**X**|*buys_computer* = yes) = 0.222 x 0.444 x 0.667 x 0.667 = 0.044

P(**X**|*buys_computer* = no) = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

P(**X**|C_i)xP(C_i) :

P(**X**|*buys_computer* = yes) x P(*buys_computer* = yes) = 0.044 x 0.643 = 0.028

P(**X**|*buys_computer* = no) x P(*buys_computer* = no) = 0.019 x 0.357 = 0.007

Therefore, **X** belongs to class (*buys_computer* = yes) !!

Naïve Bayesian Classifier: Pros and Cons

- Pros
  - The assumption that all features are independent makes naive bayes algorithm very fast compared to complicated algorithms.
    - In some cases, speed is preferred over higher accuracy. Good results obtained in most of the cases
  - It works well with high-dimensional data such as text classification, email spam detection.
- Cons
  - The assumption that all features are independent is not usually the case in real life so it makes naive bayes algorithm less accurate than complicated algorithms.

```python
# split dataset
X = dataset.iloc[:,0:8]
y = dataset.iloc[:, 8]
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, test_size=0.3)
```

```python
from sklearn.model_selection import train_test_split
```

```python
# hold-back
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,  random_state=7)
```

```python
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
```

```python
model.fit(X_train,y_train)
```

9]:

```
GaussianNB()
```

```python
result = model.score(X_test, y_test)
```

```python
# Evaluate Model
cm = confusion_matrix(y_test, y_pred)
print (cm)
```

```
[[115  32]
 [ 30  54]]
```

```python
print(f1_score(y_test, y_pred))
```

```
0.6352941176470588
```

```python
print(accuracy_score(y_test, y_pred))
```

```
0.7316017316017316
```

```python
print("Accuracy: %.3f%%" % (result*100.0))
```

```
Accuracy: 73.160%
```

```
# Pregnancies: 3,Glucose: 130, BloodPressure: 100, SkinThickness: 31;
# Insulin: 145, BMI: 25.3, DiabetesPedigreeFunction: 0.325, Age: 45
```

```
new_patient = [[3, 130, 100, 31, 145, 25.5, 0.325, 45]]
```

```
# Predict the new data result
y_pred = model.predict(new_patient)
y_pred
```

45]: array([0], dtype=int64)

```
new_patient = [[3, 330, 100, 31, 145, 56.5, 0.325, 45]]
```

```
# Predict the new data result
y_pred = model.predict(new_patient)
y_pred
```

47]: array([1], dtype=int64)