COMP7630 – Web Intelligence and its Applications

Basic Probability Theory

Valentino Santucci

(valentino.santucci@unistrapg.it)

Multinomial Distribution

- A multinomial distribution is essentially a probability vector
  - Example:

```
#probability of selecting a fruit

fruits = ['apple', 'banana', 'organge', 'mango']

probs = [0.3, 0.2, 0.1, 0.4]
```

Sampling from a Multinomial Distribution

- Roulette wheel sampling:
  - create the cumulative distribution corresponding to the probability vector;
  - generate a random number *r* in [0,1];
  - find the largest entry *i* in the cumulative vector which is smaller than *r*
  - *i* is the index sampled

Roulette Wheel Sampling in Python

in plain Python

```
In [42]: import random

In [43]:

In [43]: # Define the probability distribution

In [44]: probs = [0.3, 0.2, 0.1, 0.4]

In [45]: fruits = ['apple','banana','orange','mango']

In [46]:

In [46]: # Create the cumulative distribution

In [47]: cumulative_probs = [probs[0]]

In [48]: for i in range(1, len(probs)):
    ...:         cumulative_probs.append(cumulative_probs[i-1] + probs[i])
    ...:

In [49]: # Generate a random number between 0 and 1

In [50]: rand = random.random()

In [51]:

In [51]: # Find the index of the interval that the random number falls into

In [52]: index = 0

In [53]: while rand > cumulative_probs[index]:
    ...:         index += 1
    ...:

In [54]: # Output the fruit based on the sampled index

In [55]: print(fruits[index])
mango
```

using numpy

```
In [57]: import numpy as np

In [58]:

In [58]: # Define the probability distribution

In [59]: probs = [0.3, 0.2, 0.1, 0.4]

In [60]: fruits = ['apple','banana','orange','mango']

In [61]:

In [61]: # Create the cumulative distribution

In [62]: cumulative_probs = np.cumsum(probs)

In [63]:

In [63]: # Generate a random number between 0 and 1

In [64]: rand = np.random.rand()

In [65]:

In [65]: # Find the index of the interval that the random number falls into

In [66]: index = np.searchsorted(cumulative_probs, rand)

In [67]:

In [67]: # Output the fruit based on the sampled index

In [68]: print(fruits[index])
mango
```

Joint Probability of Independent Events

- Suppose $E_1, E_2, \dots, E_n$ are *n* <u>independent</u> events occurring with probabilities $p_1, p_2, \dots, p_n$

- Then, the joint probability that all the events $E_1, E_2, \dots, E_n$ occur altogether is $\prod_{i=1}^{n} p_i$

- In the case $p = p_1 = p_2 = \cdots = p_n$, then the joint probability (of all events occurring simultaneously) is p^n

Conditional Probability

$$P(Y|X) = \frac{P(X,Y)}{P(X)}$$

$$\Rightarrow$$

$$P(X,Y) = P(Y|X)P(X) = P(X|Y)P(Y)$$

Bayes Inference for Learning

- Bayes Theorem:

$$\underbrace{p(H|\boldsymbol{D})}_{\text{Posterior probability}} = \frac{\overbrace{p(\boldsymbol{D}|H)}^{\text{Likelihood}}\,\overbrace{p(H)}^{\text{Prior probability}}}{\underbrace{p(\boldsymbol{D})}_{\text{Evidence}}}$$

- where:
  - *D* usually refer to observed data (i.e. a data-matrix)
  - *H* usually refer to the parameters of a model (e.g. some probability distribution)
  - Bayes theorem is usually applied iteratively by setting posterior as prior in the next iteration. This will allow to iteratively increase the accuracy of how the model fits to the data
  - When there are two models to compare, the evidence in the denominator can be omitted (in fact, it only depends from the data)