Bike sharing previsions through Bayesian Networks

Fundamentals of Artificial Intelligence Module 3

Pietro Epis, Michele Milesi, Anna Valanzano

March 27, 2022

Abstract

1 London bike sharing dataset

We chose a dataset of bike sharing in London. For further references see London bike sharing dataset

Variables	Type	Cardinality	Domain
cnt	integer	8	0, 1, 2, 3, 4, 5, 6, 7
temperature	integer	4	0, 1, 2, 3
temperature1_feels	integer	4	0, 1, 2, 3
wind	string	2	"yes", "no"
hum	integer	4	0, 1, 2, 3
weather_code	string	8	1, 2 , 3, 4, 7, 10, 26, 94
time	integer	2	"morning", "afternoon", "evening", "night"
is_holiday	string	2	yes, no
is_weekend	string	2	yes, no

Table 1: Variables and domains of the dataset

Season	Values
1	Clear
2	scattered clouds
3	Broken clouds
4	Cloudy
7	Rain
10	thunderstorm
26	snowfall
94	Freezing Fog

Table 2:

Season	Values
spring	0
summer	1
fall	2
winter	3

Table 3:

Description of the variables:

- cnt: count of a new bike shares

- temperature: real temperature in C
- temperature_feels: temperature in C "feels like"
- wind
- hum: humidity in percentage
- time:
- is_holiday
- is_weekend
- weather_code: category of the weather

2 Construction of the Bayesian Network

The Bayesian Network is a data structure which represents the dependencies among the variables: it represents the full joint probability distribution in a compact way.
To define the Network we used the Python library `pgmpy`: we used the method `BayesianNetwork` to define the network, and the methods `add_nodes_from` and `add_edge` to add respectively nodes and edges. We defined the Conditional Probability tables trough the method `TabularCPD`.

basing on our intuition about the dataset.

3 Exact Inference

The basic task for any probabilistic inference system is to compute the posterior probability distribution for a set of query variables, given some observed event, that is some assignment of values to a set of evidence variables. Given a set of evidence variables E and a query variable X a typical query asks for the posterior probability distribution

$$P(X|e) = \frac{P(X,e)}{P(e)} = \alpha P(X|e) \tag{1}$$

It turns out that inference is a challenging task. For many probabilities of interest, it is NP-hard to answer any of these questions exactly. However, the Variable Elimination Algorithm allows a more efficient computation, basing on the simple idea of performing calculations once and saving results for later use.
To exploit this efficient algorithm we use the `pgmpy` method `VariableElimination`.

```
from pgmpy.inference import VariableElimination
exact_inference = VariableElimination(network)
```

Queries

After instantiating the object `exact_inference` we can use it to answer queries. If we ask ourselves what number of bikes will be rented if the temperature is low and it's windy:

```
exact_inference.query(variables=["cnt"],
               evidence = {"temperature": 0, "wind": 1, "weather": 4, "season": 3})
```

4 Approximate Inference

Unfortunately, the time complexity of exact inference is exponential in the number of variables: given the intractability of exact inference in large and multiple connected networks, it is essential to consider approximate methods. In this report we examined two Direct Sampling methods: the Rejection Sampling and the Likelihood weighting algorithm.
To implement approximate inference we the `pgmpy` method `BayesianModelSampling`.

```
from pgmpy.sampling import BayesianModelSampling
inference = BayesianModelSampling(network)
```

4.1 Rejection Sampling

To compute $P(X|e)$ in an approximate way we generate samples and reject those samples that are not consistent with the evidence e. After generating all samples that matches the evidence, the algorithm compute the approximation of $P(X|e)$ as the ration between the number of times in which X occurs and the total number of samples:

$$\widehat{P}(X|e) = \frac{N_{PS}(X,e)}{N_{PS}(e)} \approx \frac{P(X,e)}{P(e)} = P(X|e) \tag{2}$$

If the probability of the evidence $P(e)$ is small, Rejection Sampling can be very expensive, because it generates a lot of samples and rejects most of them.

```
samples = inference.rejection_sample(evidence = evidence, size = size, show_progress =
    False)
```

4.2 Likelihood Weighting

Likelihood weighting avoids the inefficiency of rejection sampling by generating only events that are consistent with the evidence e. Each event is weighted by the likelihood that the event accords to evidence : each event in which the evidence appears unlikely should be given less weight [1].
To compute $P(X|e)$ in an approximate way it sums the weights of samples in which X occurs and divide by the sum of all the weights.

```
samples = inference.likelihood_weighted_sample(evidence = evidence, size = size,
    show_progress = False)
```

5 Conclusion

[1] Artificial Intelligence. A Modern Approach, by Stuart Russel and Peter Norvig,3rd Ed.