Project

Algorithms for Handling Multivariate Poisson Distributions

Why?

Poisson distributions are widely used to model “the probability of a given number of events occurring in a fixed period of time if these events occur at a fixed rate and independently of the time the previous event occurred.” Poisson distributions are used for a wide variety of use cases, but sometimes, the variables aren’t independent, especially in some of the most popular use cases:

* Text analysis:
* Genomics
* Crime statistics

Multivariate Poisson distributions are not widely used despite plenty of use cases like the ones aforementioned. This is mostly due to the fact that current implementations of Multivariate Poisson Distributions can only represent positive dependencies between random variables (considerably narrows the scope of applications of the MPD)

I will study a new way to implement MPD: copulas. It’s a “cumulative distribution function for which the marginal probability of each random variable is uniform”. They are used to describe the dependence between variables.

Sklar’s theorem tells us that we can represent valid joint probability distributions when coupling a copula to marginal distributions.

**What will I do?**

I will be studying different types of copulas:

* Vine copulas 🡺 used for high-dimensional data as they allow to represent richer dependency structure between pairs of random variables. One can combine different types of bivariate copulas.
* Gaussian copulas => derived from the multivariate normal distribution. It’s also very flexible in the multidimensional case. Gaussian random variables are used instead of uniform ones, as other types of copulas use.

The word “copula” is Latin for “link”, so copulas link marginal distributions together to form the joint distributions.

The reason behind that choice is that these two are the most popular choices of copulas and therefore there’s a lot of documentation and different implementations available online, mostly in R, for me to rely on.

Then I would have to do to thing:

* Infer the parameters for the copulas (ground truths)
* Infer the parameters for the marginal distributions using techniques such as Inference Function for Marginals (IFM) or Maximum Likelihood Estimation.

Then compare the different resulting models using sample data and answering the following questions:

1. How well does the model fit the underlying data distribution? (accuracy measures from ML)
2. How well does the model capture the dependency structure between variables? 🡺 use tools like Maximum Mean Discrepancy or Spearman metric to measure dependency.

Simulate data:

Generating our own data is oftentimes a better solution than using a publicly available dataset If one wants to test the implementation of a new algorithm as fixed datasets probably can’t help us confirm or deny all the intricacies of our algorithm (class imbalance, various degrees of class dependence, outliers and biases…). Therefore being able to be in control of the ground truths of the dataset allows us to set up a robust test battery to possibly measure its performance against as many different situations as possible (🡺 be more representative of all the data available)

**SymPy 🡺** Computer Algebra System (CAS) library for Python. Create symbolic functions (using formulae previously seen) and SymPy will automatically generate data samples that fit this function with full control of the underlying ground truths: adding random noise, complexity, variable dependence…