Why Moments (and Generalized Moments) Are Used in Statistics and Why Expected Utility Is Used in Decision Making: A Possible Explanation

R. Noah Padilla and Vladik Kreinovich
Department of Computer Science, University of Texas at El Paso
El Paso, TX 79968, USA, rnpadilla2@miners.utep.edu, vladik@utep.edu

Formulation of the problem. In this talk, we provide a new explanation of two seemingly unrelated phenomena: that moments, and, more generally, generalized moments, $M = \sum_{i=1}^{n} p_i \cdot f(x_i)$ are effectively used in statistics, e.g., to decide which approximate model is more accurate, and that expected utility $\sum_{i=1}^{n} p_i \cdot u(x_i)$ is effectively used in decision making to decide which action is better.

For moments, x_i are different observed values; for decision making, x_i are the outcomes of different situations. In both cases, p_i are the probabilities of these values or situations.

Our explanation. In many computational problems, computation time is still a big problem. A natural way to speed up computations is to parallelize computations: first, several processors process data, then they process the results of the first-stage processing, etc. To speed up computations, we need to minimize the number of stages and minimize the time needed for each stage.

On each stage, what each processor in a deterministic computer computes is a function of its inputs. The simplest – and thus fastest to compute – are linear functions. However, if only use linear stages, the result will still be a linear function of its inputs, and many real-world dependencies are nonlinear. Thus, on some stages, we need to compute nonlinear functions. In general, the fewer inputs, the faster it is to compute a function. Thus, the fastest to compute are functions of one variable. So, on each stage, each processor computes either a linear function, or a function of one variable.

It makes no sense to have two consequent stages computing linear functions – since the resulting composition of linear functions is also linear, and it can this be computed in a single stage. It also does not make sense to have two consequent stages computing functions of one variable – since the composition of such functions is still a function of one variable. Thus, to speed up computations, we need consequent stages to be different.

The fastest is when we use only one stage, but in this case, we compute either a linear function – while many real-life functions are nonlinear – or use only one input – while we want to take all the inputs x_i into account. So, we need at least two stages. Due to the above, these stage must be different.

If the first stage is linear and the following one nonlinear, then, in general, we compute a function $f\left(a_0 + \sum_{i=1}^{n} a_i \cdot x_i\right)$. Comparing such values is equivalent to comparing the corresponding linear combinations $a_0 + \sum_{i=1}^{n} a_i \cdot x_i$. If the first stage is nonlinear and the second one linear, then we compute expressions $a_0 + \sum_{i=1}^{n} a_i \cdot f_i(x_i)$, which provide a more general opportunities for comparison.

In particular, if a priori, we have no reason to prefer some i's, then it makes sense to use the same nonlinear function $f_i(x) = f(x)$ to process all the inputs. Thus, we get the expression $a_0 + \sum_{i=1}^{n} a_i \cdot f(x_i)$, which is exactly what is used when we use generalized moments or expected utility. Thus, we have indeed explained these two expressions.