CAB203 Problem solving

CAB203 teaching team

1 Topic

Please note: For this example the question is included with the report for your convenience only. **Please do not include the question in your report** (it causes false positives for plagiarism detection.)

In many cases, concepts from one area of mathematics can be matched up with concepts from other areas. For example, there is a natural correspondence between sets and vectors over $\{0, 1\}$ (via the characteristic vector, see week 4) or between vectors and functions over a subset of the integers. For example, suppose that we have a probability space $S = \{1, 2, \ldots n\}$. Then the probabilities $P(j)$ can be interpreted as an n-dimensional vector $\vec{p}$ where $p_j = P(j)$.

Use the ideas of characteristic vectors and vectors as probabilities to interpret the following probability concepts and calculations in the language of linear algebra:

- What are the properties of a probability distribution (now probability vector)?
- What is an event?
- How to calculate the probability of an event
- Conditional probability
- Utility functions
- Expected utility

For your Python implementation, re-implement the following functions from `probability.py` found on the CAB203 Blackboard site:

- `isprobDist(P)`
- `probEvent(P, E)`
- `conditionalProb(P, E, C)`
- `utility(P, u)`
- `decide(P, ulist)`

Probability functions, events and utility functions should all be numpy `np.ndarray()` arrays. `ulist` is a list of utility functions. You should make use of numpy functions, such as `ndarray.dot()` and `ndarray.prod()`. See `https://numpy.org/doc/stable/reference/arrays.ndarray.html` for documentation on these and other useful functions.

The following section gives the solution to this topic.

2 Probabilities from linear algebra

Let $S = \{1, 2, \ldots n\}$ (note, in the Python implementation all indices start from 0 rather than 1 and go to $n-1$). A probability function/distribution on S is a vector $\vec{p} \in \mathbb{R}^n$ with the properties:

- The entries satisfy $0 \leq p_j \leq 1$ for all $j \in S$
- The sum of the entries is 1, i.e. $\vec{1} \cdot \vec{p} = 1$ where $\vec{1}$ is the vector of all 1's.

We will use the notation $P_{\vec{p}}(\cdot)$ to indicate the probability function corresponding to $\vec{p}$.

An event is a subset of S, which we can represent by a characteristic vector $\vec{e}$ which has a 1 in e_j if outcome j is in the event, and has a 0 in p_j otherwise. For example, with $n = 4$, the event corresponding to outcomes $\{1, 3\}$ is

$$\begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

The probability of an event $\vec{e}$ can be easily calculated as

$$P_{\vec{p}}(\vec{e}) = \sum_{j \in S; e_j = 1} p_j \; = \vec{p} \cdot \vec{e}.$$

In Python, this can be found using the `ndarray.dot()` function.

To find the intersection of two events, we can take advantage of the fact that $1 \times 1 = 1$ and $1 \times 0 = 0$. Let $\vec{e}$ and $\vec{f}$ be events. Then for the intersection (say, $\vec{g}$) we want $g_j = 1$ if $e_j = f_j = 1$ and 0 otherwise. This is satisfied by $g_j = e_j f_j$. Hence we can form $\vec{g}$ as the *entry-wise* product[1] of $\vec{e}$ and $\vec{f}$, often notated $e \odot f$. In Python, this can be found using the `ndarray.prod()` function. Thus

$$P_{\vec{p}}\left(\vec{e}, \vec{f}\right) = \vec{p} \cdot \left(\vec{e} \odot \vec{f}\right)$$

With the above in mind, we can calculate the conditional probability of an event $\vec{e}$ given $\vec{c}$ like so:

$$P_{\vec{p}}(\vec{e} \,|\, \vec{c}) = \frac{\vec{p} \cdot (\vec{e} \odot \vec{c})}{\vec{p} \cdot \vec{c}}$$

Utility functions are, like probability distributions, n-dimensional real vectors $\vec{u} \in \mathbb{R}^n$. The expected utility is the weighted sum of the entries of the utility vector with the probabilities as weights. Hence

$$\mathcal{E}_{\vec{p}}(\vec{u}) = \vec{u} \cdot \vec{p}.$$

Gives some utility functions $\vec{u}_1 \ldots \vec{u}_m$ we can easily calculate the expected utility of all of them at once. Recall that when multiplying a vector by a matrix, the entries of the resulting vector are the dot product between the vector and the *rows* of the matrix. We form a matrix U whose rows are the vectors $\vec{u}_1 \ldots \vec{u}_m$, i.e. $U_{ij} = (\vec{u}_i)_j$. Then we calculate $U\vec{p} = \vec{v}$ and v_k will be $\vec{u}_k \cdot \vec{p}$, the expected utility for utility function k. The maximum entry in $\vec{v}$ then corresponds to the utility with the highest expected utility.

[1] Horn, Roger A.; Johnson, Charles R. (2012). Matrix analysis. Cambridge University Press.