How to apply advantage to a PDF?

In the old ProbDist implementation, this was done with the following code:

```
fn apply_positive_advantage(&mut self, adv: usize) {
        for _ in 0..adv {
            let mut total_above = 0.0;
            let mut next = BTreeMap::new();
            let cumulative = self.get_cumulative_prob();
            let mut rev_iter = self.keys().rev();
            let first = rev_iter.next().unwrap();
            // The cumulative probabilities, in descending order, with the first one
skipped and a zero appended to the end
            // Because we always need the cumulative probability of the X <= x - 1
            let mut rev_cumul =
cumulative.values().rev().skip(1).chain([0.0].iter());
            // Pa(X = max) = 1 - P(X <= max - 1)^2
            total_above += 1.0 - rev_cumul.next().unwrap().powi(2);
            next.insert(*first, total_above);

            // Now for the rest of them
            for (outcome, cumul_below) in rev_iter.zip(rev_cumul) {
                // Pa(X=x<max) = 1 - P(X <= x - 1)^2 - Pa(X > x)
                let new_prob = 1.0 - cumul_below.powi(2) - total_above;
                // Add new probability to total
                total_above += new_prob;
                // Insert new entry
                next.insert(*outcome, new_prob);
            }

            // Next should now contain the probability weight function with advantage
applied
            *self = ProbDist(next);
        }
    }
```

First of all, it is good to take note of the function signature. The keys don't have to change, and the ProbDist remains valid, so it can be 'mut &self'. The process the function follows to apply advantage is as follows:

1. Create new BTreeMap (P_a)
2. Get the cumulative PDF ($P(X \leq x)$)
3. Create a reverse iterator over the keys and cumulative pdf, then take out the first element of both
4. Add a zero to the end of the reverse cumulative pdf
5. Then calculate $P_a(X = \text{max}) = 1 - P(X \leq \text{max} - 1)^2$ and insert in into the new BTreeMap
6. Then for the rest of the elements, calculate $P_a(X = x < \text{max}) = 1 - P(X \leq x - 1)^2 - P_a(X > x)$ and insert into the new BTreeMap.

From this is follows that:

$$P_a(X = x) = 1 - P(X \leq x - 1)^2 - P_a(X > x) \qquad 1.$$

The following results from facts and logic (and Demorgan: $A \cup B = \overline{\overline{A} \cap \overline{B}}$):

$$P(X \leq x - 1) = P(X < x)$$

$$P_a(X > x) = \text{Chance that one die rolls higher} \overset{\text{Demorgan}}{=} \text{Chance that both don't roll higher} = 1 - P(X \leq x)^2 \qquad 2.$$

Now substituting these relations in $P_a(X = x)$:

$$\begin{aligned} P_a(X = x) &= 1 - P(X \leq x)^2 - 1 + P(X \leq x)^2 = P(X \leq x)^2 - P(X < x)^2 \\ &= \text{Chance that both roll x or lower} - \text{Chance that both roll lower than x} \end{aligned} \quad 3.$$

This suggests that this is generalizable to higher order, and possibly negative, advantages. So where n is the level of advantage, the following may hold:

$$P_a(X = x) = P(X \leq x)^{n+1} - P(X < x)^{n+1} \quad 4.$$

The most efficient way of calculating this is probably this:
1. Let a = $P(X < x)^{n+1}$ with a trailing 1
2. Then $P_a(X = x) = a[x + 1] - a[x]$

The logic behing Equation 3 suggests that it is generalizeable to the following:

$$P(X = \max(x_1, ..., x_n)) = \prod_{i=1}^{n} P(X \leq x_i) - \prod_{i=1}^{n} P(X < x_i) \quad 5.$$

This is more difficult to implement for arbitrary random variables than the simple case of $x_1 = x_2$ (AKA advantage).

Disadvantage

From Equation 5, it follows that disadvantage can be based on the following

$$P(X = \min(x_1, ..., x_n)) = \prod_{i=1}^{n} P(X \geq x_i) - \prod_{i=1}^{n} P(X > x_i) \quad 6.$$

, and the algorithm for $x_1 = x_2$ is:
1. Let a = $P(X > x)^{n+1}$ with a leading 1
2. Then $P_a(X = x) = a[x] - a[x + 1]$

Small note on composable operations

It feels like it should be possible to just use a lazy-iterator-like system to perform all these computations with great efficiency. This would be tricky to implement, because some algorithms require access to more than pointwise operations.