Improvements to 1-Dice-Dudo

Skipping extremely negative regret actions (Regret-based Pruning)

In large games, computing the regrets associated with a single iteration can be slow. There are many clearly suboptimal actions in the game and repeatedly exploring them wastes computational resources that could be better used to improve the strategy elsewhere. (Brown, 2017)

- An action a with regret R(a) that is below a threshold C (where C is negative) is sampled with probability $K/[K + C - R(a)]$, where K is a positive constant. There is additionally a floor on the sample probability. This sampling is only done for about the last half of iterations to be run; the first half is conducted using traditional external-sampling

- Another way is to set a threshold C, and not explore the subtree with regret below C with a fixed probability. (Pluribus used this strategy) MCCFR. Other formulas can also be used.

TODO: Implement partial pruning in Brown, 2017

Prune histories whose opponent's reach is zero.

Discounted CFR

The first iteration of CFR plays uniformly at random, as a consequence many suboptimal strategies are played. The regrets accumulated in these early iterations can go a long way in the algorithm.(Brown, 2019) We can discount this by assigning different weights across iterations.

- Linear CFR.

  On iteration t the updates to the regrets and average strategies are given weight t. That is, the iterations are weighed linearly. (Equivalently, one could multiply the accumulated regret by $\frac{t}{t+1}$ on each iteration. We do this in our experiments to reduce the risk of numerical instability.) This means that after T iterations of LCFR, the first iteration only has a weight of $\frac{2}{T^2+T}$ on the regrets rather than a weight of $\frac{1}{T}$, which would be the case in vanilla CFR.

TODO: Parallel Computing

Need a parallel implementation that speeds up computation. Read parallel computing workshop repos.

Experiments log

In 1DD, `dt-3.5MPr` used linear CFR for the first 100k iterations only, trained 3.5m iteratons at about 17 iterations per second.

Extensive testing was conducted(see training log) to find a conservative pruning threshold. Child nodes with regret sum less than -10000 are pruned, at total of 3282 child nodes pruned.

Afterwards training is based on the pruned tree `dt-3.5MPr@-10k`, 95% of the time pruned nodes are not visited, at training speed about 42 iterations per second.

At 7 million iterations, examining previously pruned nodes, no pruned nodes had regret sum greater than -10k. Tested pruning `dt-7MPr` again at regret sum -10k. This time, an additional 808 nodes were pruned, creating `dt-7MPrPr`. Training for 2M iterations on top of both, the difference between the two trees was -2.0262 e-07. The only difference in strategySum is the choice '1*1' in node ['1', '1*4'], might have been wrongly pruned.