1 Counterfactual Regret Minimization

We have seen in Lecture 2 that a sequence-form strategy space can be characterized recursively by composing convex hulls and Cartesian products operations. By applying the regret circuits described above, we can then construct a regret minimizers for any sequence-form strategy space. The resulting regret minimizer is called CFR. In a nutshell, CFR decomposes the problem of minimizing regret on the whole tree-form decision problem into local regret minimization problems at each of the individual decision points $j \in \mathcal{J}$. Any regret minimizer $\mathcal{R}_j$ for simplex domains can be used to solve the local regret minimization problems. Popular options are the regret matching algorithm, and the regret matching plus algorithm (Lecture 4).

Before giving pseudocode, we recall and introduce a bit of notation to deal with tree-form sequential decision processes.

Notation for tree-form sequential decision processes We use the following notation for dealing with tree-form sequential decision processes (TFSDPs), most of which was already introduced in Lecture 2. The notation is also summarized in **??**.

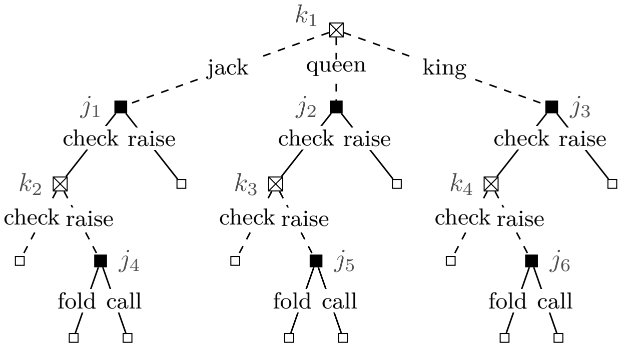

Figure 1: Tree-form sequential decision making process of the first acting player in the game of Kuhn poker.

- We denote the set of decision points in the TFSDP as $\mathcal{J}$, and the set of observation points as $\mathcal{K}$. At each decision point $j \in \mathcal{J}$, the agent selects an action from the set A_j of available actions. At each observation point $k \in \mathcal{K}$, the agent observes a signal s_k from the environment out of a set of possible signals S_k.
- (**new!**) We denote by ρ the transition function of the process. Picking action $a \in A_j$ at decision point $j \in \mathcal{J}$ results in the process transitioning to $\rho(j, a) \in \mathcal{J} \cup \mathcal{K} \cup \{\perp\}$, where $\perp$ denotes the end of the decision process. Similarly, the process transitions to $\rho(k, s) \in \mathcal{J} \cup \mathcal{K} \cup \{\perp\}$ after the agent observes signal $s \in S_k$ at observation point $k \in \mathcal{K}$.
- A pair (j, a) where $j \in \mathcal{J}$ and $a \in A_j$ is called a *sequence*. The set of all sequences is denoted as $\Sigma \coloneqq \{(j, a) : j \in \mathcal{J}, a \in A_j\}$. For notational convenience, we will often denote an element (j, a) in Σ as ja without using parentheses.
- Given a decision point $j \in \mathcal{J}$, we denote by p_j its *parent sequence*, defined as the last sequence (that is, decision point-action pair) encountered on the path from the root of the decision process to j. If the agent does not act before j (that is, j is the root of the process or only observation points are encountered on the path from the root to j), we let $p_j = \varnothing$.

Symbol	Description
$\mathcal{J}$	Set of decision points
A_j	Set of legal actions at decision point $j \in \mathcal{J}$
$\mathcal{K}$	Set of observation points
S_k	Set of possible signals at observation point $k \in \mathcal{K}$
ρ	Transition function: • given $j \in \mathcal{J}$ and $a \in A_j$, $\rho(j, a)$ returns the next point $v \in \mathcal{J} \cup \mathcal{K}$ in the decision tree that is reached after selecting legal action a in j, or $\perp$ if the decision process ends; • given $k \in \mathcal{K}$ and $s \in S_k$, $\rho(k, s)$ returns the next point $v \in \mathcal{J} \cup \mathcal{K}$ in the decision tree that is reached after observing signal s in k, or $\perp$ if the decision process ends
Σ	Set of sequences, defined as $\Sigma \coloneqq \{(j, a) : j \in \mathcal{J}, a \in A_j\}$
p_j	Parent sequence of decision point $j \in \mathcal{J}$, defined as the last sequence (decision point-action pair) on the path from the root of the TFSDP to decision point j; if the agent does not act before j, $p_j = \varnothing$

Table 1: Summary of notation in TFSDPs.

As an example, consider the TFSDP faced by Player 1 in the game of Kuhn poker (**?**), depicted in **??**, which we already introduced in Lecture 2. There, we have that $\mathcal{J} = \{j_1, \ldots, j_6\}$ and $\mathcal{K} = \{k_1, \ldots, k_4\}$. $A_{j_1} = S_{k_4} = \{\text{check}, \text{raise}\}$. $A_{j_5} = \{\text{fold}, \text{call}\}$. $S_{k_1} = \{\text{jack}, \text{queen}, \text{king}\}$. $\rho(k_3, \text{check}) = \rho(j_2, \text{raise}) = \perp$. $\rho(k_1, \text{king}) = j_3$. $\rho(j_2, \text{check}) = k_3$. $p_{j_4} = (j_1, \text{check})$. $p_{j_6} = (j_3, \text{check})$. $p_{j_1} = p_{j_2} = p_{j_3} = \varnothing$.

Notation for the components of vectors Any vector $\boldsymbol{x} \in \mathbb{R}^{|\Sigma|}$ has, by definition, as many components as sequences Σ. The component corresponding to a specific sequence $ja \in \Sigma$ is denoted as $\boldsymbol{x}[ja]$. Similarly, given any decision point $j \in \mathcal{J}$, any vector $\boldsymbol{x} \in \mathbb{R}^{|A_j|}$ has as many components as the number of actions at j. The component corresponding to a specific action $a \in A_j$ is denoted $\boldsymbol{x}[a]$.

CFR algorithm Pseudocode for CFR is given in **??**. Note that the implementation is parametric on the regret minimization algorithms $\mathcal{R}_j$ run locally at each decision point. Any regret minimizer $\mathcal{R}_j$ for simplex domains can be used to solve the local regret minimization problems. Popular options are the regret matching algorithm, and the regret matching plus algorithm (Lecture 4).

It can be shown that the regret cumulated by the CFR algorithm satisfies the following bound.

Proposition 1.1. Let R_j^T ($j \in \mathcal{J}$) denote the regret cumulated up to time T by each of the regret minimizers $\mathcal{R}_j$. Then, the regret R^T cumulated by **??** up to time T satisfies

$$R^T \le \sum_{j \in \mathcal{J}} \max\{0, R_j^T\}.$$

It is then immediate to see that if each R_j^T grows sublinearly in T, then so does R^T.

Algorithm 1: CFR regret minimizer

```
Data: $\mathcal{R}_j$, one regret minimizer for $\Delta^{|A_j|}$; one for each decision point $j \in \mathcal{J}$ of the TFSDP.
function NextStrategy()
    [▷ Step 1: we ask each of the $\mathcal{R}_j$ for their next strategy local at each decision point]
    for each decision point $j \in \mathcal{J}$ do
        $\boldsymbol{b}_j^t \in \Delta^{|A_j|} \leftarrow \mathcal{R}_j$.NextStrategy()
    [▷ Step 2: we construct the sequence-form representation of the strategy that plays according to
       the distribution $\boldsymbol{b}_j^t$ at each decision point $j \in \mathcal{J}$]
    $\boldsymbol{x}^t = \mathbf{0} \in \mathbb{R}^{|\Sigma|}$
    for each decision point $j \in \mathcal{J}$ in top-down traversal order in the TFSDP do
        for each action $a \in A_j$ do
            if $p_j = \varnothing$ then
                $\boldsymbol{x}^t[ja] \leftarrow \boldsymbol{b}_j^t[a]$
            else
                $\boldsymbol{x}^t[ja] \leftarrow \boldsymbol{x}^t[p_j] \cdot \boldsymbol{b}_j^t[a]$
    [▷ You should convince yourself that the vector $\boldsymbol{x}^t$ we just filled in above is a valid sequence-form
       strategy, that is, it satisfies the required consistency constraints we saw in Lecture 2. In symbols,
       $\boldsymbol{x}^t \in Q$]
    return $\boldsymbol{x}^t$
------------------------------------------------------------------------------------------------------
function ObserveUtility($\boldsymbol{\ell}^t \in \mathbb{R}^{|\Sigma|}$)
    [▷ Step 1: we compute the expected utility for each subtree rooted at each node $v \in \mathcal{J} \cup \mathcal{K}$]
    $V^t \leftarrow$ empty dictionary            [▷ eventually, it will map keys $\mathcal{J} \cup \mathcal{K} \cup \{\bot\}$ to real numbers]
    $V^t[\bot] \leftarrow 0$
    for each node in the tree $v \in \mathcal{J} \cup \mathcal{K}$ in bottom-up traversal order in the TFSDP do
        if $v \in \mathcal{J}$ then
            Let $j = v$
            $V^t[j] \leftarrow \sum_{a \in A_j} \boldsymbol{b}_j^t[a] \cdot \big(\boldsymbol{\ell}^t[ja] + V^t[\rho(j,a)]\big)$
        else
            Let $k = v$
            $V^t[k] \leftarrow \sum_{s \in S_k} V^t[\rho(k,s)]$
    [▷ Step 2: at each decision point $j \in \mathcal{J}$, we now construct a local utility vector $\boldsymbol{\ell}_j^t$ called
       counterfactual utility]
    for each decision point $j \in \mathcal{J}$ do
        $\boldsymbol{\ell}_j^t \leftarrow \mathbf{0} \in \mathbb{R}^{|A_j|}$
        for each action $a \in A_j$ do
            $\boldsymbol{\ell}_j^t[a] \leftarrow \boldsymbol{\ell}^t[ja] + V^t[\rho(j,a)]$
        $\mathcal{R}_j$.ObserveUtility($\boldsymbol{\ell}_j^t$)
```