Formal Algorithms for Constitutional AI

Notation and Conventions

Sequences: - $\mathcal{L} \equiv [M_V]$: Vocabulary of size M_V - $\mathcal{L}^*$: Set of all token sequences over vocabulary $\mathcal{L}$ - $x, y \in \mathcal{L}^*$: Prompt and response sequences - $[s_1; s_2; \ldots]$: String concatenation operator

Models: - θ: Neural network parameters - DTransformer($\cdot \mid \theta$): Decoder-only transformer (Algorithm 10 from Formal Algorithms) - $\hat{P}_\theta(y \mid x)$: Model's conditional probability distribution

Constitutional AI Specific: - $\mathcal{C} = \{c_1, c_2, \ldots, c_K\}$: Set of constitutional principles (natural language strings) - θ_{helpful}: Parameters of helpful-only RLHF model - θ_{SL}: Parameters of SL-CAI model (after Stage 1) - θ_{RL}: Parameters of final RL-CAI model (after Stage 2)

Functions: - ∇: Gradient operator (computed via automatic differentiation) - $\sigma(z) = 1/(1 + e^{-z})$: Sigmoid function - $\text{softmax}(z)_i = e^{z_i} / \sum_j e^{z_j}$: Softmax function

---

Stage 1: Supervised Learning (SL-CAI)

Algorithm 1: Critique Generation

Input: $x \in \mathcal{L}^*$, a prompt
Input: $y \in \mathcal{L}^*$, a response to critique
Input: $c \in \mathcal{C}$, a constitutional principle
Input: θ_{helpful}, helpful-only model parameters
Output: critique $\in \mathcal{L}^*$, the generated critique

Algorithm:

1. critique_prompt $\leftarrow [x; \text{``Response: "}; y; \text{``\textbackslash n\textbackslash nCritique based on: "}; c]$
2. critique $\leftarrow$ DTransformer(critique_prompt $\mid \theta_{\text{helpful}}$) $\triangleright$ Generate critique
3. **return** critique

Algorithm 2: Revision Generation

Input: $x \in \mathcal{L}^*$, a prompt
Input: $y \in \mathcal{L}^*$, original response
Input: critique $\in \mathcal{L}^*$, the critique of y
Input: θ_{helpful}, helpful-only model parameters
Output: $y_{\text{revised}} \in \mathcal{L}^*$, the revised response

Algorithm:

1. revision_prompt $\leftarrow [x; \text{``Response: "}; y; \text{``\textbackslash n\textbackslash nCritique: "}; \text{critique}; \text{``\textbackslash n\textbackslash nRevise: "}]$

2. $y_{\text{revised}} \leftarrow \text{DTransformer}(\text{revision\_prompt} \mid \theta_{\text{helpful}})$ ▷ Generate revision
3. **return** y_{revised}

Algorithm 3: Iterative Critique-Revision

Input: $x \in \mathcal{L}^*$, a red-team prompt designed to elicit harmful behavior
Input: θ_{helpful}, parameters of helpful-only RLHF model
Input: $\mathcal{C} = \{c_1, c_2, \ldots, c_K\}$, constitutional principles
Output: $y_{\text{final}} \in \mathcal{L}^*$, a harmless revised response
Hyperparameters: $N_{\text{revisions}} \in \mathbb{N}$, number of critique-revision iterations (typically 4)

Algorithm:

1. $y_0 \leftarrow \text{DTransformer}(x \mid \theta_{\text{helpful}})$ ▷ Generate initial (likely harmful) response
2. **for** $n = 1, 2, \ldots, N_{\text{revisions}}$ **do**
3. $\quad c_n \leftarrow \text{sample\_uniform}(\mathcal{C})$ ▷ Randomly sample a principle
4. $\quad \text{critique}_n \leftarrow \text{CritiqueGeneration}(x, y_{n-1}, c_n, \theta_{\text{helpful}})$
5. $\quad y_n \leftarrow \text{RevisionGeneration}(x, y_{n-1}, \text{critique}_n, \theta_{\text{helpful}})$
6. **end for**
7. **return** $y_{\text{final}} = y_{N_{\text{revisions}}}$

Key Property: Each iteration refines the response to be progressively less harmful according to sampled principles.

Algorithm 4: SL-CAI Training

Input: $\{x_i^{\text{harm}}\}_{i=1}^{M_{\text{harm}}}$, red-team prompts
Input: $\{x_j^{\text{help}}\}_{j=1}^{M_{\text{help}}}$, helpful prompts
Input: θ_0, initial pretrained model parameters
Input: θ_{helpful}, helpful-only RLHF model parameters
Input: $\mathcal{C}$, constitutional principles
Output: $\hat{\theta}_{\text{SL}}$, trained SL-CAI model parameters
Hyperparameters: $M_{\text{epochs}} \in \mathbb{N}$, $\eta \in (0, \infty)$ (learning rate)

Algorithm:

1. $\mathcal{D}_{\text{harmless}} \leftarrow \emptyset$
2. **for** $i = 1, 2, \ldots, M_{\text{harm}}$ **do**
3. $\quad y_i \leftarrow \text{IterativeCritiqueRevision}(x_i^{\text{harm}}, \theta_{\text{helpful}}, \mathcal{C})$
4. $\quad \mathcal{D}_{\text{harmless}} \leftarrow \mathcal{D}_{\text{harmless}} \cup \{(x_i^{\text{harm}}, y_i)\}$
5. **end for**
6. $\mathcal{D}_{\text{helpful}} \leftarrow \emptyset$
7. **for** $j = 1, 2, \ldots, M_{\text{help}}$ **do**
8. $\quad y_j \leftarrow \text{DTransformer}(x_j^{\text{help}} \mid \theta_{\text{helpful}})$ ▷ Sample helpful responses

9. $\mathcal{D}_{\text{helpful}} \leftarrow \mathcal{D}_{\text{helpful}} \cup \{(x_j^{\text{help}}, y_j)\}$
10. **end for**
11. $\mathcal{D}_{\text{train}} \leftarrow \mathcal{D}_{\text{harmless}} \cup \mathcal{D}_{\text{helpful}}$ ▷ Mix harmless and helpful data
12. $\theta \leftarrow \theta_0$
13. **for** epoch $= 1, 2, \ldots, M_{\text{epochs}}$ **do**
14. **for** $(x, y) \in \mathcal{D}_{\text{train}}$ **do**
15. $T \leftarrow \text{length}(y)$
16. $\omega(\theta) \leftarrow \text{DTransformer}(x; y[1 : T-1] \mid \theta)$ ▷ Forward pass
17. $\text{loss}(\theta) = -\sum_{t=1}^{T-1} \log \omega(\theta)[y[t+1], t]$ ▷ Cross-entropy loss
18. $\theta \leftarrow \theta - \eta \cdot \nabla \text{loss}(\theta)$ ▷ Gradient descent
19. **end for**
20. **end for**
21. **return** $\hat{\theta}_{\text{SL}} = \theta$

Memory Complexity: Same as standard DTransformer training (Algorithm 13)
Key Innovation: Training data is self-generated via critique-revision, not human-labeled

---

Stage 2: Reinforcement Learning from AI Feedback (RL-CAI)

Algorithm 5: AI Feedback Generation (Standard)

Input: $x \in \mathcal{L}^*$, a prompt
Input: θ_{SL}, SL-CAI model parameters
Input: θ_{feedback}, feedback model parameters (pretrained LM)
Input: $c \in \mathcal{C}$, a constitutional principle
Output: $(y_{\text{chosen}}, y_{\text{rejected}})$, a preference pair
Hyperparameters: $T_{\text{sample}} \in (0, \infty)$, sampling temperature

Algorithm:

1. $y_A \leftarrow \text{DTransformer}(x \mid \theta_{\text{SL}}, \text{temp} = T_{\text{sample}})$ ▷ Sample first candidate
2. $y_B \leftarrow \text{DTransformer}(x \mid \theta_{\text{SL}}, \text{temp} = T_{\text{sample}})$ ▷ Sample second candidate
3. $x_{\text{compare}} \leftarrow$ [“Prompt: "; x; “\n"]
4. $x_{\text{compare}} \leftarrow$ [x_{compare}; “Which is better per ‘"; c; “’?\n"]
5. $x_{\text{compare}} \leftarrow$ [x_{compare}; “(A) "; y_A; “\n(B) "; y_B; “\nAnswer: "]
6. $\omega \leftarrow \text{DTransformer}(x_{\text{compare}} \mid \theta_{\text{feedback}})$ ▷ Get distribution over next token
7. $p_A \leftarrow \exp(\log \text{prob}(\omega, \text{“(A)"}))$ ▷ Log probability of token “(A)”
8. $p_B \leftarrow \exp(\log \text{prob}(\omega, \text{“(B)"}))$ ▷ Log probability of token “(B)”
9. **if** $p_A/(p_A + p_B) > 0.5$ **then**
10. **return** $(y_{\text{chosen}} = y_A, y_{\text{rejected}} = y_B)$
11. **else**
12. **return** $(y_{\text{chosen}} = y_B, y_{\text{rejected}} = y_A)$
13. **end if**

Key Property: Uses normalized log probabilities as well-calibrated preference labels

---

Algorithm 6: AI Feedback with Chain-of-Thought

Input: $x \in \mathcal{L}^*$, a prompt
Input: θ_{SL}, SL-CAI model parameters
Input: θ_{helpful}, helpful RLHF model for reasoning
Input: $c \in \mathcal{C}$, a constitutional principle
Output: $(y_{\text{chosen}}, y_{\text{rejected}})$, a preference pair
Hyperparameters: $p_{\min}, p_{\max} \in (0, 1)$, probability clamping bounds (typically 0.4, 0.6)

Algorithm:

1. $y_A \leftarrow \text{DTransformer}(x \mid \theta_{\text{SL}})$
2. $y_B \leftarrow \text{DTransformer}(x \mid \theta_{\text{SL}})$
3. $x_{\text{CoT}} \leftarrow$ [“Human: Prompt: "; x; “\n"]
4. $x_{\text{CoT}} \leftarrow$ [x_{CoT}; “Evaluate per: "; c; “\n"]
5. $x_{\text{CoT}} \leftarrow$ [x_{CoT}; “(A) "; y_A; “\n(B) "; y_B; “\n"]
6. $x_{\text{CoT}} \leftarrow$ [x_{CoT}; “Assistant: Let’s think step-by-step: "]
7. reasoning $\leftarrow \text{DTransformer}(x_{\text{CoT}} \mid \theta_{\text{helpful}})$ ▷ Generate CoT reasoning
8. **if** “(A)" appears last in reasoning **then**
9. $\quad p_{\text{chosen}} \leftarrow \min(\max(0.9, p_{\min}), p_{\max})$ ▷ Clamp probability
10. $\quad$ **return** $(y_{\text{chosen}} = y_A, y_{\text{rejected}} = y_B)$
11. **else**
12. $\quad p_{\text{chosen}} \leftarrow \min(\max(0.9, p_{\min}), p_{\max})$
13. $\quad$ **return** $(y_{\text{chosen}} = y_B, y_{\text{rejected}} = y_A)$
14. **end if**

Key Innovation: Explicit reasoning makes AI decision process transparent and improves accuracy
Clamping Rationale: Prevents overconfident labels that destabilize RL training

---

Algorithm 7: Preference Model Training

Input: $\{x_i^{\text{harm}}\}_{i=1}^{M_{\text{harm}}}$, red-team prompts
Input: $\{(x_j^{\text{help}}, y_j^{\text{chosen}}, y_j^{\text{rejected}})\}_{j=1}^{M_{\text{help}}}$, human helpfulness preferences
Input: θ_{SL}, SL-CAI model parameters
Input: θ_{feedback}, feedback model parameters
Input: $\mathcal{C}$, constitutional principles
Output: $\hat{\theta}_{\text{PM}}$, trained preference model parameters
Hyperparameters: $M_{\text{epochs}} \in \mathbb{N}$, $\eta \in (0, \infty)$

Algorithm:

1. $\mathcal{D}_{\text{AI}} \leftarrow \emptyset$ ▷ AI-generated harmlessness preferences
2. **for** $i = 1, 2, \ldots, M_{\text{harm}}$ **do**
3. $\quad c \leftarrow \text{sample\_uniform}(\mathcal{C})$
4. $\quad (y_{\text{chosen}}, y_{\text{rejected}}) \leftarrow \text{AIFeedback}(x_i^{\text{harm}}, \theta_{\text{SL}}, \theta_{\text{feedback}}, c)$
5. $\quad \mathcal{D}_{\text{AI}} \leftarrow \mathcal{D}_{\text{AI}} \cup \{(x_i^{\text{harm}}, y_{\text{chosen}}, y_{\text{rejected}})\}$
6. **end for**
7. $\mathcal{D}_{\text{human}} \leftarrow \{(x_j^{\text{help}}, y_j^{\text{chosen}}, y_j^{\text{rejected}})\}_{j=1}^{M_{\text{help}}}$ ▷ Human helpfulness data
8. $\mathcal{D}_{\text{PM}} \leftarrow \mathcal{D}_{\text{AI}} \cup \mathcal{D}_{\text{human}}$ ▷ Mix AI and human labels
9. $\theta_{\text{PM}} \leftarrow \theta_{\text{SL}}$ ▷ Initialize from SL-CAI
10. **for** epoch $= 1, 2, \ldots, M_{\text{epochs}}$ **do**
11. $\quad$ **for** $(x, y_{\text{chosen}}, y_{\text{rejected}}) \in \mathcal{D}_{\text{PM}}$ **do**
12. $\quad\quad r_{\text{chosen}} \leftarrow \text{PreferenceScore}(x, y_{\text{chosen}} \mid \theta_{\text{PM}})$
13. $\quad\quad r_{\text{rejected}} \leftarrow \text{PreferenceScore}(x, y_{\text{rejected}} \mid \theta_{\text{PM}})$
14. $\quad\quad \text{loss}(\theta_{\text{PM}}) = -\log(\sigma(r_{\text{chosen}} - r_{\text{rejected}}))$ ▷ Ranking loss
15. $\quad\quad \theta_{\text{PM}} \leftarrow \theta_{\text{PM}} - \eta \cdot \nabla \text{loss}(\theta_{\text{PM}})$
16. $\quad$ **end for**
17. **end for**
18. **return** $\hat{\theta}_{\text{PM}}$

Key Property: Distills both AI (harmlessness) and human (helpfulness) preferences into single model

---

Algorithm 8: Preference Scoring Function

Input: $x \in \mathcal{L}^*$, a prompt
Input: $y \in \mathcal{L}^*$, a response
Input: θ_{PM}, preference model parameters
Output: $r \in \mathbb{R}$, scalar reward/preference score

Algorithm:

1. $T \leftarrow \text{length}(y)$
2. $H \leftarrow \text{DTransformer}(x; y \mid \theta_{\text{PM}})$ ▷ Get hidden states (shape: $d_e \times T$)
3. $h_{\text{final}} \leftarrow H[:, T]$ ▷ Extract final hidden state
4. $r \leftarrow W_r^\top h_{\text{final}} + b_r$ ▷ Linear projection to scalar
5. **return** r

Parameters: $W_r \in \mathbb{R}^{d_e}$, $b_r \in \mathbb{R}$ (learned during PM training)

Algorithm 9: RL-CAI Training (Proximal Policy Optimization)

Input: $\{x_i\}_{i=1}^{M_{\text{prompts}}}$, training prompts (harmfulness + helpfulness)
Input: θ_{SL}, SL-CAI model parameters (initial policy)

Input: $\hat{\theta}_{\text{PM}}$, trained preference model
Output: $\hat{\theta}_{\text{RL}}$, final RL-CAI model parameters
Hyperparameters: $M_{\text{epochs}} \in \mathbb{N}$, $\eta \in (0, \infty)$, $\beta \in (0, \infty)$ (KL penalty coefficient)

Algorithm:

$\theta_{\text{policy}} \leftarrow \theta_{\text{SL}}$ ▷ Initialize policy from SL-CAI
for epoch $= 1, 2, \ldots, M_{\text{epochs}}$ **do**
  for $i = 1, 2, \ldots, M_{\text{prompts}}$ **do**
    $y \leftarrow \text{DTransformer}(x_i \mid \theta_{\text{policy}})$ ▷ Sample response from policy
    $r_{\text{PM}} \leftarrow \text{PreferenceScore}(x_i, y \mid \hat{\theta}_{\text{PM}})$ ▷ Get PM reward
    $T \leftarrow \text{length}(y)$
    $\text{KL}_{\text{penalty}} \leftarrow 0$
    for $t = 1, 2, \ldots, T$ **do**
      $p_{\text{policy}} \leftarrow \hat{P}_{\theta_{\text{policy}}}(y[t] \mid x_i; y[1 : t-1])$ ▷ Current policy prob
      $p_{\text{SL}} \leftarrow \hat{P}_{\theta_{\text{SL}}}(y[t] \mid x_i; y[1 : t-1])$ ▷ Reference policy prob
      $\text{KL}_{\text{penalty}} \leftarrow \text{KL}_{\text{penalty}} + p_{\text{policy}} \cdot \log(p_{\text{policy}}/p_{\text{SL}})$
    end for
    $r_{\text{total}} \leftarrow r_{\text{PM}} - \beta \cdot \text{KL}_{\text{penalty}}$ ▷ Total reward with KL penalty
    $\text{loss}(\theta_{\text{policy}}) = -r_{\text{total}} \cdot \sum_{t=1}^{T} \log \hat{P}_{\theta_{\text{policy}}}(y[t] \mid x_i; y[1 : t-1])$
    $\theta_{\text{policy}} \leftarrow \theta_{\text{policy}} - \eta \cdot \nabla \text{loss}(\theta_{\text{policy}})$ ▷ Policy gradient
  end for
end for
return $\hat{\theta}_{\text{RL}} = \theta_{\text{policy}}$

KL Penalty Rationale: Prevents policy from drifting too far from SL-CAI initialization, maintaining quality
Note: Simplified for clarity; production implementations use PPO clipping and advantage estimation