Learning to Prune: Speeding up Repeated Computations
Daniel Alabi, Adam Tauman Kalai, Katrina Ligett, Cameron Musco, Christos Tzamos, and Ellen Vitercik
COLT 2019

Speeding up Repeated Computations

**Goal:** Solve sequence of similar computational problems, exploiting common structure

Typically, large swaths of search space never optimal

*Learn to ignore them!*

- Shortest path always in specific region of road network
- Only handful of LP constraints ever bind
- Large portions of DNA never contain patterns of interest

---

Model

Function \( f: X \rightarrow Y \) maps problem instances \( x \) to solutions \( y \)

Learning algorithm receives sequence \( x_1, \ldots, x_T \in X \)

*E.g., each \( x_i \) equals edge weights for a fixed graph*

**Goal:**

Correctly compute \( f \) on most rounds, minimizing runtime

*Worst-case algorithm would compute \( f(x_i) \) for each \( x_i \)*

Assume access to other functions mapping \( X \rightarrow Y \)

- Faster to compute
- Defined by subsets (prunings) \( S \) of universe \( \mathcal{U} \)
- Universe \( \mathcal{U} \) represents entire search space
- Denote corresponding function \( f_S: X \rightarrow Y \)
- \( f_u = f \)

*Example:*  

\( \mathcal{U} = \) all edges in fixed graph  

\( S = \) subset of edges

Assume exists \( S^*(x) \subseteq \mathcal{U} \) where \( f_S(x) = f(x) \) iff \( S^*(x) \subseteq S \)

- "Minimally pruned set"
- *E.g., the shortest path*

---

Algorithm

1. Initialize pruned set \( S_1 \leftarrow \emptyset \)
2. For each round \( i \in \{1, \ldots, T\} \):
   a. Receive problem instance \( x_i \)
   b. With probability \( 1/\sqrt{i} \), **exploit**:
      i. Output \( f(x_i) \)
      ii. Compute minimally pruned set \( S^*(x_i) \)
      iii. Update pruned set: \( S_{i+1} = S_i \cup S^*(x_i) \)
   c. Otherwise (with probability \( 1 - 1/\sqrt{i} \)), **exploit**:
      i. Output \( f_S(x_i) \)
      ii. Don’t update pruned set: \( S_{i+1} = S_i \)

---

Experiments

- **Linear programming**
  - Top line: Simplex  
  - Bottom line: Our algorithm
  - 204 variables, 946 constraints  
  - Fraction of mistakes: 0.018 over 5000 runs with \( T = 30 \)

- **Shortest path routing**
  - Top line: Dijkstra’s algorithm  
  - Bottom line: Our algorithm
  - Fraction of mistakes: 0.068 over 5000 runs with \( T = 30 \)

---

Guarantees

Recap: At round \( i \), algorithm outputs \( f_S(x_i) \)

\( S_i \) depends on \( x_{1:i} \)

**Goal 1:** Minimize \( |S_i| \)

*Time it takes to compute \( f_S(x_i) \) typically grows with \( |S_i| \)*

**Theorem:**

\[
\mathbb{E}\left[ \sum_{i=1}^T |S_i| \right] \leq |S^*| + \frac{|\mathcal{U}| - |S^*|}{\sqrt{T}}
\]

where \( S^* = \bigcup_{i=1}^T S^*(x_i) \)

*Proof:*

\[
\mathbb{E}[|S_i|] = \frac{1}{\sqrt{T}}|\mathcal{U}| + \left(1 - \frac{1}{\sqrt{T}}\right) \mathbb{E}[|S_i|] \leq \frac{1}{\sqrt{T}} |\mathcal{U}| + \left(1 - \frac{1}{\sqrt{T}}\right) |S^*|
\]

**Goal 2:** Minimize # of mistakes

*Rounds where \( f_S(x_i) \neq f(x_i) \)*

**Theorem:**

\[
\mathbb{E}[\text{# of mistakes}] \leq \frac{|S^*|}{\sqrt{T}}
\]

where \( S^* = \bigcup_{i=1}^T S^*(x_i) \)

\( S^* \) is smallest set \( S \) where \( f_S(x_i) = f(x_i) \) for all \( i \)

*Proof sketch:*

- For \( e \in S^* \), let \( N_T(e) \) be # of times \( e \notin S_i \) but \( e \in S^*(x_i) \)
- When makes mistake, must be \( e \in S^*(x_i) \) with \( e \notin S_i \)
    - Otherwise, \( S_i \supsetneq S^*(x_i) \), so no mistake
    - This means \( N_T(e) += 1 \)
- Therefore, \( \mathbb{E}[\text{# of mistakes}] \leq \sum_{e \in S^*} \mathbb{E}[N_T(e)] \)
- We prove \( \mathbb{E}[N_T(e)] \leq \sum_{i=1}^T \left(1 - \frac{1}{\sqrt{T}}\right)^i \leq \frac{1}{\sqrt{T}} \)
  - If \( e \notin S_i \), then \( e \notin S_j \) for \( j \leq i \)
  - This means \( \mathbb{E}[\text{# of mistakes}] \leq \frac{|S^*|}{\sqrt{T}} \)