COMS 4995-2: Advanced Algorithms (Spring'20) — Apr 9, 2020

Lecture 20: MWU Application, Models for Large-scale Computation

Instructor: *Alex Andoni* — Scribes: *Ruoshi Liu, Hongjin Wang*

1 Recap: Multiplicative Weights Update

Problem Definition We have n "experts" and an event which lasts from 1 to T. At each timestamp, each expert will make a prediction of the event. We define the following terms:

- $f_i^t \in [-1,+1]$: the error of expert i at time t
- $m_i^T = \sum_{t=1}^T f_i^t$: number of errors of expert i
- $M^T = \sum_{t=1}^T f_{e_i}^t$: number of our errors, where e_i is the expert we chose to follow at time t

A Randomized Algorithm

1. Keep weights $w_i^t > 0$, where i is in index for expert, and t is for time
2. Initialize $w_i^t = 1$
3. At time t, make prediction: i p, $p_i^t = \frac{w_i^t}{\sum_j w_j^t}$ and update: $w_i^{t+1} = w_i^t(1 - \varepsilon f_i^t)$

Theorem 1. $M^T \leq min_i m_i^T + \frac{\ln n}{\varepsilon} + \varepsilon T$

Corollary 2. *If* $T > \frac{\ln n}{\varepsilon^2} \implies M^T \leq min_i m_i^T + 2\varepsilon T$

Note that the theorem has been proved in last lecture.

2 Application of MWU: feasibility of LP

Problem Definition: feasibility of a standard-form LP: $\exists x \in \mathbb{R}$, s.t. $Ax \geq b$

Relaxed Version: distinguish between

1. $\exists x \in \mathbb{R}^n$, s.t. $Ax \geq b - \varepsilon \mathbb{1}_m$
2. $\exists! x$ s.t. $Ax \geq b$

To solve the relaxed version of the problem, we assume an access to the following oracle:

Definition 3. *$\mathcal{O}$ is an oracle the given an input of $p \in \mathbb{R}^m$, if $\exists x \in \mathbb{R}^m$ s.t. $p^T Ax \geq p^T b$ and $max_i|A_i x - b_i| \leq 1$, then output x; output 'NO' otherwise*

Theorem 4. *We can solve the relaxed version of the LP-feasibility problem using $O(\frac{\ln n)}{\varepsilon^2}$ oracle calls*

We give an algorithm solving the feasibility problem using the oracle specified above:

1. Initilize $w_i^1 = 1$, $\forall i \in \{1, ..., m\}$, where m is the number of experts
2. at iteration t $\in 1, ..., \mathbb{O}(\frac{\ln n}{\varepsilon^2})$
   (a) $p_i^t = \frac{w_i^t}{\sum_j w_j^t}$; $p^t = [p_1^t, p_2^t, ..., p_m^t]$
   (b) $x^t = \mathcal{O}(p^t)$
   (c) if $\exists ! x \implies$ return *Infeasible*
   (d) $w_i^{t+1} = w_i^t(1 - \varepsilon f_i^t)$, $f_i^t = A_i x - b_i$
3. return the average: $\bar{x} = \sum_{t=1}^{T} \frac{x^t}{T}$

Intuition of the algorithm: in step 2.(d), if f_i^t is bigger than 0, or in other words, the constraint i is well satisfied from expert i perspective, then the weight corresponding to that expert will decrease. According to step 2.(a), the probability of choosing that specific expert will also decrease. Therefore in the next iteration, the oracle will focus more on other constraints that are less satisfied.

Proof. First, If the algorithm returns 'inf' $\implies$ the relaxed problem is infeasible.
Here we draw connection to MWU: expert i is the same as constraint $A_i x \geq b_i$
We define f_i^t as the error at constraint i:

$$f_i^t = A_i x^t - b_i; f^t = Ax^t - b \tag{1}$$

which satisfies $f_i^t \in [-1, +1]$ by $\mathcal{O}$
By definition,

$$\begin{aligned}
M^T &= \sum_{t=1}^{T} < p^t, f^t > \\
&= \sum_{t=1}^{T} \sum_{i=1}^{m} p_i^t \cdot f_i^t \\
&= \sum_{t} < p^t, Ax^t - b > \\
&= \sum_{t} < p^t, Ax^t > - \sum_{t} < p^t, b > \\
&= \sum_{t} [(p^t)^T Ax^t - (P^t)^T \cdot b] \geq 0
\end{aligned}$$

From corallary 2 , we know that,

$$M^t \leq m_i^T + 2\varepsilon T, \forall i \in [m] \tag{2}$$

Which implies,

$$\begin{aligned} m_i^T \geq -2\varepsilon T &\implies \sum_{t=1}^{T} f_i^t \geq -2\varepsilon T \\ &\implies \sum_{t=1}^{T} [A_i x^t - b_i] \geq -2\varepsilon T \\ &\implies A_i \cdot \sum_{t=1}^{T} x^t \geq T \cdot b_i - 2\varepsilon t \\ &\implies A_i \cdot \frac{\sum_{t=1}^{T} x^t}{T} \geq b_i - 2\varepsilon \end{aligned}$$

□

Student question: How can we guarantee $\max_i |A_i x - b_i| \leq 1$?
Suppose we can solve $\mathcal{O}$:

$$\max_i |A_i x - b_i| \leq k, Ax \geq b$$

$$\Rightarrow \max_i |\frac{A_i}{k} x - \frac{b_i}{k}| \leq 1$$

This can be solve approximately:

$$\frac{A}{k} x \geq \frac{b}{k} - \epsilon \mathbb{1} m$$
$$\Rightarrow Ax \geq b - \epsilon k \mathbb{1} m$$

Define $\epsilon' = \epsilon k$
by above can be solved in $T = O(\frac{\lg n}{\epsilon^2}) = O(k^2 \frac{\lg n}{(\epsilon')^2})$

3 Large Scale Model

From now on we design algorithm taking into account the architecture.

3.1 Usual Model(RAM Model)

See Figure 1

3.2 Modern Model

See Figure 2

3.3 I/O Model

See Figure 3

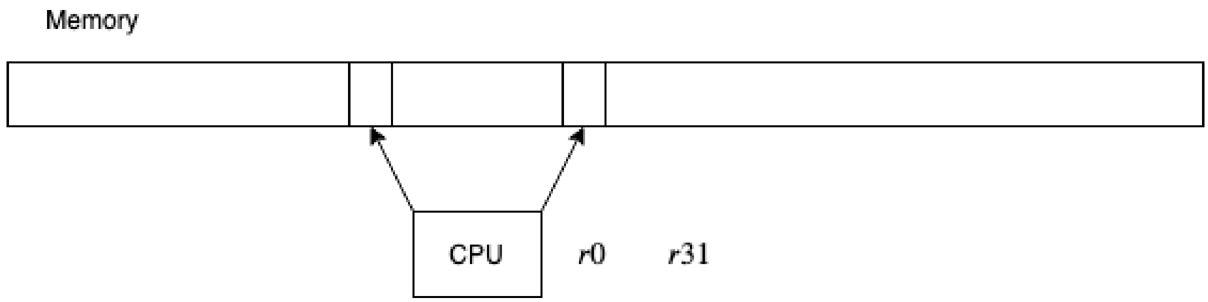

Figure 1: Usual Model(RAM Model)

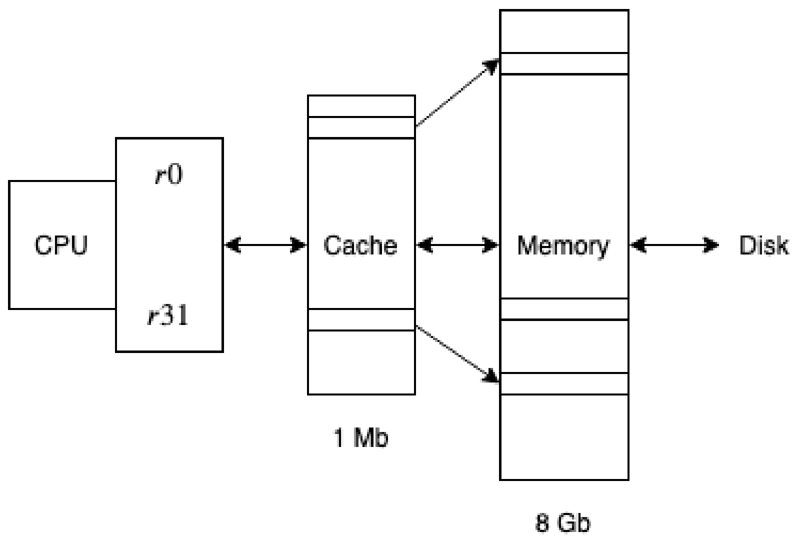

Figure 2: Modern Model

- CPU with O(1) registers(or cell of memory/word)
- cache of size M
- unlimited memory
- cache/memory are composed of cache lines, each of B words
- when accessing a new location x, if it not in cache, we need to bring entire cache line
- cache line: Starts at $B \cdot i, i \in N$, ends at $B \cdot (i+1) - 1$
- when cache is full, throw one cache line out(replace), either "manually" or Least Recently Used(LRU) policy

Since accessing cache is cheap, while accessing memory is expensive, we can define **cost** as number of times bring a cache line from memory into cache.

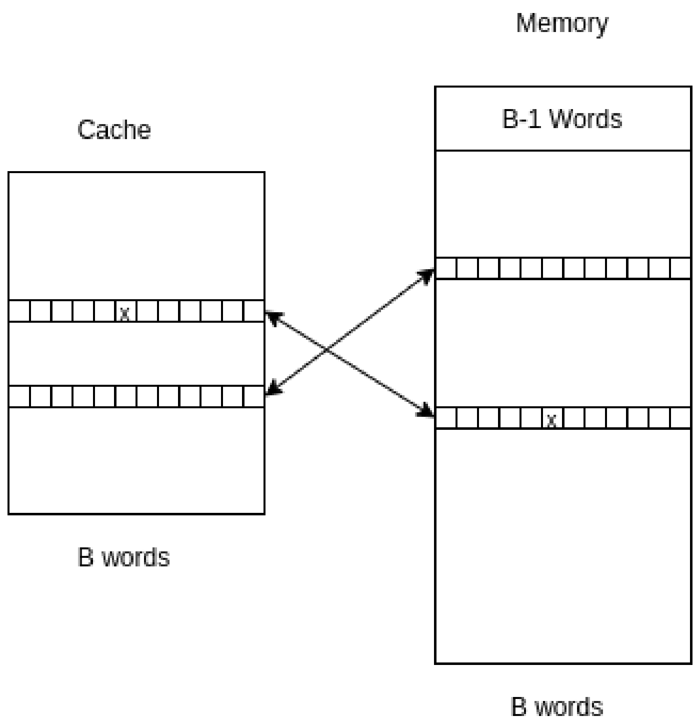

Figure 3: I/O model

3.4 Problem: searching memory

Searching x in an unsorted array of size N.

- RAM model: O(N)
- I/O model: Assume array stored linearly. O(N/B+1)

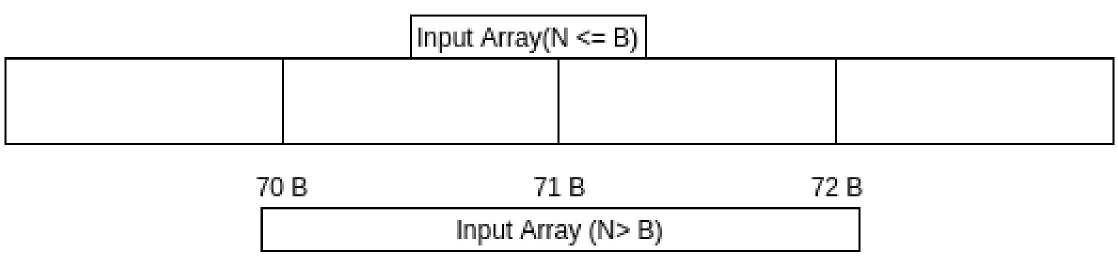

Figure 4: Search x in an unsorted array

Proof. The proof is illustrated in Figure 4.

When $N \leq B$, it depends on whether the array is divided by the cache line boundary. $cost = 2$ if divided, $cost = 1$ if not.

When $N > B$, the array straddles $\leq N/B + 2$ cache lines $\Rightarrow O(N/B + 1)$ "runtime"

□