CS 229r: Algorithms for Big Data Fall 2013

Lecture 7 — September 24, 2013

Prof. Jelani Nelson *Scribe: David Liu*

1 Overview

In this lecture, we cover a randomized approximate algorithm to solve the DTM (distance to monotonicity) problem.

2 Distance to Monotonicty

For a given sequence $\{x_i\}$, the distance to monotonicity is the fewest number of elements we must remove from the sequence such that the remaining subsequence is an increasing subsequence. If we have n elements in our sequence, this number is precisely $n - K$ where K is the length of the longest increasing subsequence in our original sequence.

2.1 DP algorithm

Below is the DP algorithm to solve this problem. Essentially, we remember for each position i in the sequence the fewest number of elements we can remove from $\{1, 2, ..., i-1\}$ such that we have a subsequence ending at i that is an increasing subsequence.

1. Give each item i a weight $w(i) = 1$
2. Prepend and postpend items of value $-\infty$ and ∞, with weights $w(0) = w(n+1) = \infty$
3. Initialize $R = \{0\}$, $W(0) = 0$, $s(0) = 0$. $W(t)$ will be just the total weight of the first t elements.
4. For $t = 1$ to $n + 1$, do the following:
   - Set $W(t) = W(t-1) + w(t)$
   - Set $s(t) = min\{s(i) + W(t-1) - W(i) : i \in R, x[i] \leq x[t]\}$
   - Add t to the set R
5. Output $s(n+1)$

This algorithm uses $O(n)$ space.

3 Streaming algorithms

Suppose that for each i that $x[i] \in [m]$.
A deterministic 1-pass streaming algorithm for the LIS (longest increasing subsequence) problem that gives a $(1+\epsilon)$ approximation to the solution and uses space $O(\sqrt{\frac{n}{\epsilon}} \log m)$ was demonstrated in a paper by Gopalan, Jayram, Krauthgamer, and Kumar [3]. Furthermore, in papers by Ergün and Jahari [1] and Gál and Gopalan [2] it was shown that any deterministic 1-pass algorithm needs $\Omega(\sqrt{\frac{n}{\epsilon}} \log(\frac{m}{n\epsilon}))$ space. An open question is whether there exists a polylogarithmic space algorithm for a 2-approximation for the LIS problem.

4 DTM algorithm

Below, we will present an 1-pass streaming algorithm discovered by Saks and Seshadri [4] that provides a $(1+\epsilon)$ approximation to the DTM problem, uses $O(\frac{log^2 n}{\epsilon})$ space, and succeeds with probability $1 - \frac{1}{n^3}$.

4.1 Description

Essentially, the algorithm is the same as the one presented on the previous page, except now we sometimes forget values with a certain probability.

1. Give each item i a weight $w(i) = 1$
2. Prepend and postpend items of value $-\infty$ and ∞, with weights $w(0) = w(n+1) = \infty$
3. Initialize $R = \{0\}$, $W(0) = 0$, $S(0) = 0$.
4. For $t = 1$ to $n+1$, do the following:
   - Set $W(t) = W(t-1) + w(t)$
   - Set $r(t) = min\{r(i) + W(t-1) - W(i) : i \in R, x[i] \leq x[t]\}$
   - Add t to the set R
   - For each $i \in R$, remove i from R with probability $1 - p(i,t)$
5. Output $r(n+1)$

It is clear that $r(n+1)$ (the output value for this algorithm) is always greater than or equal to $s(n+1)$ (the true optimal solution). Thus, we need to show the following about our algorithm -

1. With probability greater than $1 - \frac{1}{\text{poly } n}$, $r(n+1) \leq (1+\epsilon)s(n+1)$
2. With probability greater than $1 - \frac{1}{\text{poly } n}$, $|R| \leq O(\frac{log^2 n}{\epsilon})$ at all times during the algorithm

Write the above $1 - \frac{1}{polyn}$ as $1 - \frac{\delta}{2}$. $p(i,t)$ is defined as follows. First, we define

$$q(i,t) = \min\{1, \frac{\epsilon}{1+\epsilon}\ln(\frac{4t^3}{\delta})\frac{w(i)}{W([i,t])}\}$$

We then define

$$p(i,i) = 1$$

$$p(i,t)_{t>i} = \frac{q(i,t)}{q(i,t-1)}$$

Thus, this tells us that defining R^t to be the set R after t steps, that $\mathbb{P}(i \in R^t) = q(i,t)$.

4.2 Proof

Let C be the indices of some optimal solution to the LIS problem. We say that $i \in C$ is unsafe at time t if the following is true:

$$(C \cap [i,t]) \cap R^t = \emptyset$$

The above is the same as saying that we have forgotten everything in C between i and t at time t. Let U^t bet the set of unsafe indices at time T. Define

$$U = \bigcup_{t=1}^{n+1} U_t$$

By definition $U \subset C$

Lemma 1. $r(n+1) \leq W(\overline{C} \cup U)$

Note: $\overline{C}$ is the complement of C

We induct to show that $r(t) \leq W(\overline{C}_{\leq t-1} \cup U^{t-1})$, where $\overline{C}_{\leq t-1} = \overline{C} \cap [1, t-1]$. The base case is clear as $r(1) = 0$.
For the inductive step, there are two cases. The first is that $U^{t-1} = C_{\leq t-1}$, i.e. the case when everything is unsafe. In this case $W(\overline{C}_{\leq t-1} \cup U^{t-1}) = W([1, t-1]) \geq r(t)$. For the second case, $U^{t-1} \backslash C \neq \emptyset$. Choose $j \in C, j < t$ to be the largest index of C remembered in R^t. We see that

$$r(t) \leq r(j) + W([j+1, t-1]) \leq W(\overline{C}_{\leq j-1} \cup U^{j-1}) + W([j+1, t-1]) = W(\overline{C}_{\leq t-1} \cup U^{t-1})$$

thus completing the lemma. $\square$

We define $I \subset [n]$ to be dangerous if

$$|C \cap I| \leq \frac{\epsilon}{1+\epsilon}|I|$$

We then define $i \in [n]$ to be dangerous if there exists a dangerous interval I such that i is the left endpoint of I.
Now, we greedily construct a set of disjoint dangerous intervals I_j as follows - at each point, we

look at the left-most point we have not tested that is not currently in one of our intervals I_j. If that point is a dangerous point, we choose a dangerous interval with that point as a left endpoint and add it to our set of I_j. Otherwise, we move on to the next point. We define then $B = \cup I_j$ and D to be the set of dangerous indices.
Consider the following set of three statements.

1. $\overline{C} \subset D \subset B$
2. $W(B) \leq (1+\epsilon)W(\overline{C})$
3. With probability $\geq 1 - \frac{\delta}{2}$, $U \subset B$

If all of the above claims are true, we are done, because

$$r(n+1) \leq W(\overline{C} \cup U) \leq W(B) \leq (1+\epsilon)W(\overline{C})$$

with high probability.

1. $\overline{C} \subset D \subset B$
   $\overline{C} \subset D$ is clear, as we take the interval of size 1 for any point in $\overline{C}$, and C intersect this set is empty. $D \subset B$ follows from the way we greedily constructed B.

2. $W(B) \leq (1+\epsilon)W(\overline{C})$
   Recall $B = \bigcup_j I_j$ for dangerous I_j

$$W(I_j \cap C) \leq \frac{\epsilon}{1+\epsilon} W(I_j) \implies W(I_j \cap \overline{C}) \geq \frac{1}{1+\epsilon} W(I_j)$$

$$W(B \cap \overline{C}) \geq \frac{1}{1+\epsilon} W(B)$$

   Using that $\overline{C} \subset B$, meaning $W(B \cap \overline{C}) = W(\overline{C})$, we thus have the desired statement.

3. With probability $\geq 1 - \frac{\delta}{2}$, $U \subset B$
   Fix $t \in [n]$, $i \in \overline{B} \cap [t]$. We want to show that $\mathbb{P}(i \in U^t) \leq \frac{\delta}{4t^3}$, which suffices by union bound, because

$$\mathbb{P}(U \subset B) = 1 - \mathbb{P}(\overline{B} \cap U \neq \emptyset) \geq 1 - \sum_{t=1}^{n} \mathbb{P}(\overline{B} \cap U^t \neq \emptyset)$$

$$\geq 1 - \sum_{t=1}^{n} \sum_{i \in \overline{B} \cap [t]} \mathbb{P}(i \in U^t) \geq 1 - \frac{\delta}{4} \sum_{t} \frac{1}{t^2} \geq 1 - \frac{\delta}{2}$$

   To show this, we need now to show that $i \in \overline{B} \cap [t] \implies \mathbb{P}(i \in U^t) \leq \frac{\delta}{4t^3}$. As we know that i is not dangerous, we know that $[i,t]$ is not dangerous, so thus $W(C \cap [i,t]) \geq \frac{\epsilon}{1+\epsilon} W([i,t])$. We know $i \in U^t$ only if we have forgotten everything in $C \cap [i,t]$ at time t. Thus,

$$\mathbb{P}(i \in U^t) = \prod_{j \in C \cap [i,t]} (1 - q(j,t)) \leq \prod_{j \in C \cap [i,t]} (1 - ln(\frac{4t^3}{\delta}) \frac{w(j)}{W(C \cap [j,t])})$$

   Using $1 - x \leq e^{-x}$, we thus get that

$$\mathbb{P}(i \in U^t) \leq \prod_{j \in C \cap [i,t]} e^{-ln(\frac{4t^3}{\delta}) \frac{w(j)}{W(C \cap [j,t])}} = e^{-ln(\frac{4t^3}{\delta})(\sum_{j \in C \cap [i,t]} \frac{w(j)}{W(C \cap [j,t])})} = \frac{\delta}{4t^3}$$

This thus completes the proof of the correctness of the algorithm.

4.3 Space

The space used during the algorithm is proportional to $\max_{t\in[n]}|R^t|$. We need to show that the probability that $|R^t$ is large can be bounded by $\frac{\delta}{2n}$, which would give us that $\mathbb{P}(\exists t \text{ s.t. } |R^t| \text{ large}) < \frac{\delta}{2}$. We then let Z_i^t be the indicator random variable for the event that $i \in R^t$, which we see to be 1 with probability $q(i,t)$. Using Chernoff, we can get that the probability we use more space than allowed is less than $\frac{\delta}{2n}$ for each $|R^t|$. Note then that if we ever would use more space than allotted, we can simply abort the algorithm and return an arbitrary value, ensuring that we never exceed our space limit.

References

[1] Funda Ergün, Hossein Jowhari. On distance to monotonicity and longest increasing subsequence of a data stream. *SODA*, 730-736, 2008.

[2] Anna Gál, Parikshit Gopalan. Lower Bounds on Streaming Algorithms for Approximating the Length of the Longest Increasing Subsequence. *SIAM J. Comput.*, 39(8):3463-3479, 2010.

[3] Parikshit Gopalan, T.S. Jayram, Robert Krauthgamer, Ravi Kumar. Estimating the sortedness of a data stream. *SODA*, 318-327, 2007.

[4] Michael Saks, C. Seshadri. Space efficient streaming algorithms for the distance to monotonicity and asymmetric edit distance. *SODA*, 1698-1709, 2013.