CS-450 Advanced algorithms

Homework 2

Jannik Moritz Reichert (300198)
Martin Vold (300510)
Sacha Wattel (235735)

École polytechnique fédérale de Lausanne

24th May 2019

Problem 1: Weighted Frequency Estimation

We will design a one-pass streaming algorithm to estimate the weighted frequency vector

$$W_2 = \sum_{i=1}^{n} w_i f_i^2$$

by starting with picking a 4-wise independent hash function: $h : [n] \to \{-1, +1\}$ such that $Pr_{h \in H}[h(x_1) = u_1 \wedge \cdots \wedge h(x_4) = u_4]$. With this hash function we will let every element in the stream be equal to the hash of it's position, $\sigma_i = h(i)$ so $\sigma \in \{-1, 1\}^n$. With the hash function and modified stream we can now estimate the value W_2 by processing elements in the stream with value i: $W = W + \sigma_i$, W is initialized to 0. When the stream is processed we will have $W = \sum_{i=1}^{n} \sqrt{w_i} f_i \sigma_i$ and the algorithm will output W^2. We will call this algorithm $\mathcal{X}$ and we need that it is an unbiased estimator.

$$\begin{aligned}
\mathbb{E}[W^2] &= \mathbb{E}[(\sum_{i\in[n]} \sqrt{w_i} f_i \sigma_i)^2] \\
&= \mathbb{E}[\sum_{i,j\in[n]} \sqrt{w_i}\sqrt{w_j} f_i f_j \sigma_i \sigma_j] \\
&= \mathbb{E}[\sum_{i\in[n]} w_i f_i^2 \sigma_i^2] + \mathbb{E}[\sum_{i\neq j\in[n]} \sqrt{w_i}\sqrt{w_j} f_i f_j \sigma_i \sigma_j] \\
&= \mathbb{E}[\sum_{i\in[n]} w_i f_i^2] + \sum_{i\neq j\in[n]} \sqrt{w_i}\sqrt{w_j} f_i f_j \mathbb{E}[\sigma_i]\mathbb{E}[\sigma_j] \\
&= ||\vec{f}||_2^2
\end{aligned}$$

Where $||\vec{f}||_2^2$ is the frequency vector where all the frequencies have been multiplied with their respective wight. This shows that the estimator is unbiased as it's expected value is the same as W_2.

Now we want to bound the variance of the estimate.

$$Var[W^2] = \mathbb{E}[W^4] - (\mathbb{E}[W^2])^2$$

Looking at $\mathbb{E}[W^4]$ one can see that W^4 is the product of 4 sums of $\sqrt{w_i}$, f_i and σ_i giving us more than the two cases $i \in [n]$ and $i \neq j \in [n]$ as in the calculation of the expected value.

- All four indexes are equal which gives us: $\sum_{i\in[n]} \sigma_i^4 f_i^4 w_i^2 = \sum_{i\in[n]} f_i^4 w_i^2$
- Two and two indices are matching which gives us: $\binom{4}{2} \sum_{i<j} (\sqrt{w_i}\sqrt{w_j} f_i f_j \sigma_i \sigma_j)^2 = 6 \sum_{i<j} w_i w_j f_i^2 f_j^2$
- Three indices are match and one is unmatched. In this case we will end up with a $\mathbb{E}[\sigma_i]$ which is equal to zero for any i. Since our hash function is 4-wise independent it will make it possible that these terms equal to 0.

Now we can calculate $Var[W^2]$

$$\begin{aligned}
Var[W^2] &= \mathbb{E}[W^4] - (\mathbb{E}[W^2])^2 \\
&= \sum_{i\in[n]} f_i^4 w_i^2 + 6\sum_{i<j} w_i w_j f_i^2 f_j^2 - (\sum_i w_i f_i^2)^2 \\
&= \sum_i f_i^4 w_i^2 + 6\sum_{i<j} w_i w_j f_i^2 f_j^2 - \sum_i w_i^2 f_i^4 - 2\sum_{i<j} w_i w_j f_i^2 f_j^2 \\
&= 4\sum_{i<j} w_i w_j f_i^2 f_j^2 \\
&\leq 2(\sum_i w_i f_i^2)^2 \\
&= 2||\vec{f}||_2^4
\end{aligned}$$

Where $||\vec{f}||_2^2$ is the frequency vector where all the frequencies have been multiplied with their respective wight.

We have now that the expected value of algorithm $\mathcal{X}$ is the value of W_2 which we want to estimate, but the variance is quite high. To get the wanted bound on the estimate we will make r i.i.d copies of the estimator $\mathcal{X}$ and run them all in parallel and output the average over all the outputs. We will call this algorithm $\mathcal{Y}$, output is $\widetilde{W}^2 = \frac{1}{r}\sum_{i=1}^{r} W^2$. By linearity of expectation, we have $\mathbb{E}[\widetilde{W}^2] = \mathbb{E}[W^2] = ||\vec{f}||_2^2$. Looking at the variance of the new estimate we can see that it will be lower, $\frac{Var(\widetilde{W}^2)}{r} \leq \frac{2}{t}||\vec{f}||_2^4$. By Chebyshev's inequality we now have that the probability that the output of $\mathcal{Y}$ will deviate with more than $\epsilon||\vec{f}||_2^2$ from the expected value is

$$Pr[|\widetilde{W}^2 - ||\vec{f}||_2^2| \geq \epsilon||\vec{f}||_2^2] \leq \frac{\frac{2}{r}||\vec{f}||_2^4}{\epsilon^2||\vec{f}||_2^4} \leq \frac{2}{r\epsilon^2}$$

By choosing $r = \frac{6}{\epsilon^2}$, we get $Pr[|\widetilde{W}^2 - ||\vec{f}||_2^2| > \epsilon||\vec{f}||_2^2] \leq \frac{1}{3}$. This means that algorithm $\mathcal{Y}$ will return a $1 \pm \epsilon$ approximation with a probability of at least $\frac{2}{3}$.

Now we want to decrease the probability of failing to δ, so that we can get the desired probability for our bound on $\hat{W}_2$ which is $1 - \delta$. To accomplish this we will create a new algorithm, $\mathcal{W}$ which will make s i.i.d copies of $\mathcal{Y}$, run them in parallel and output the median over the outputs of the s copies. We wish to show $Pr[|\hat{W}_2 - ||\vec{f}||_2^2| \geq \epsilon||\vec{f}||_2^2|] \leq \delta$ which means that we want to analyze the failure probability of $\mathcal{W}$. The way we will do this is by defining a indicator variable $Z_i \in \{0, 1\}$ that takes the value 1 if the output value $\widetilde{W}_i^2$ of $\mathcal{Y}_i$ satisfies the inequality $|\widetilde{W}_i^2 - ||\vec{f}||_2^2| \geq \epsilon ||\vec{f}||_2^2$. The probability of this happening is $Pr[|\widetilde{W}_i^2 - ||\vec{f}||_2^2| \geq \epsilon ||\vec{f}||_2^2] = Pr[Z_i = 1] \leq \frac{1}{3}$ and since we are running s i.i.d copies of $\mathcal{Y}$ the expected value of the sum of all Z_i is $\mathbb{E}[Z] \leq \frac{s}{3}$. Since we choose $\hat{W}_2$ by taking the median of all $\widetilde{W}^2$ the probability for that $\hat{W}_2$ will get to high or low so it will satisfy the bound is: $Pr[|\hat{W}_2 - ||\vec{f}||_2^2| \geq \epsilon ||\vec{f}||_2^2] \leq Pr[Z \geq s/2]$. Since Z is the sum of independent random variables which take the value 0 or 1 we can use the Chernoff Bounds to estimate their probability. We have:

$$Pr[Z \geq \frac{s}{2}] \leq Pr[Z > \frac{3\mathbb{E}[Z]}{2}] \leq e^{-\frac{(\frac{1}{2})^2}{2+\frac{1}{2}}\mathbb{E}[Z]} = e^{\frac{1}{10}\mathbb{E}[Z]} = e^{\frac{1}{10}10\log(\frac{1}{\delta})} = \delta$$

This result is achieved by setting $s = 30\log(\frac{1}{\delta})$ as we have that $\mathbb{E}[Z] \leq s/3$.

We can see that the probability that over half of the $\widetilde{W}^2$ values satisfies $|\widetilde{W}_i^2 - ||\vec{f}||_2^2| \geq \epsilon ||\vec{f}||_2^2$ is less than δ, which makes the probability that less than half of them do is greater or equal to $1 - \delta$. Since $|\widetilde{W}_i^2 - ||\vec{f}||_2^2| \geq \epsilon ||\vec{f}||_2^2$ satisfies both the case where $\widetilde{W}_i^2 - ||\vec{f}||_2^2 \geq \epsilon ||\vec{f}||_2^2$ and $\widetilde{W}_i^2 - ||\vec{f}||_2^2 \leq -\epsilon ||\vec{f}||_2^2$ and $\hat{W}_2$ will return a $1 \pm \epsilon$ approximation with probability $1 - \delta$ we have satisfied

$$Pr[(1-\epsilon)W_2 \leq \hat{W}_2 \leq (1+\epsilon)W_2] \geq 1 - \delta$$

For the memory usage: we run $s = 30\log(\frac{1}{\delta})$ $\mathcal{Y}$ estimators and $r = \frac{6}{\epsilon}$ $\mathcal{X}$ estimators for each $\mathcal{Y}$ estimator. This gives us $O(\frac{1}{\epsilon^2}\log(\frac{1}{\delta}))$ $\mathcal{X}$ estimators. Each of these estimators uses $O(\log(n) + \log(m))$ space. This because each one of the estimator has to calculate each element i the stream with the hash function, which takes $O(\log(n))$ space, and store the element in the stream, which takes $O(\log(m))$. Giving us a final memory usage of $O((\log(n) + \log(m))\frac{1}{\epsilon^2}\log(\frac{1}{\delta}))$.

Problem 2: Submodular function maximization

2.1 $|S| = k$

From the definition, we know:

$$f(S_k) = f(S_{k-1} \cup \{e_k\}) = f(e_k \mid S_{k-1}) + f(S_{k-1})$$

and

$$f(e_k \mid S_{k-1}) \geq \frac{OPT}{2k}$$

$$f(S_k) \geq \frac{OPT}{2k} + f(S_{k-1} = \frac{OPT}{2k} + f(e_{k-1} \mid S_{k-2}) + f(S_{k-2}) \geq \frac{OPT}{2k} + \frac{OPT}{2k} + f(S_{k-2})$$

Since $|S| = k$ and $f(\emptyset) = 0$, it follows that:

$$f(S_k) \geq k \cdot \frac{OPT}{2k} = \frac{OPT}{2}$$

2.2 $|S| < k$

We have that some elements from the optimal set, $O = \{o_1, \ldots, o_k\}$, has been rejected from the set $S = \{o_1, \ldots, o_l\}$. From lemma 1 in lecture 21 we have that

$$\sum_{o \in O \setminus S} f(o|S) \geq f(O \cup S) - f(S)$$

As $S \subset O$ the union of O and S will be the optimal solution so we can replace $f(O \cup S)$ by OPT.

Looking at the left hand side of the formula we have the sum of every element in $O \setminus S$. By looking at how the algorithm accepts elements in the stream the elements in $O \setminus S$ has to be strictly smaller than $\frac{OPT}{2k}$. Since we are summing over all elements in $O \setminus S$ and $|O| = K$ the cardinality of this set can be written like $|O \setminus S| \leq k$. This gives us

$$\sum_{o \in O \setminus S} f(o|S) < \sum_{o \in O \setminus S} \frac{OPT}{2k} = |O \setminus S| \frac{OPT}{2k} \leq \frac{OPT}{2}$$

Now we can put this into the inequeality from lemma 1

$$\frac{OPT}{2} \geq f(O \cup S) - f(S) = OPT - f(S)$$

Rearranging this gives of the inequality we are after

$$f(S) \geq \frac{OPT}{2}$$

Problem 3: Exact matching in bipartite graph

Objective Given an n-by-n bipartite graph $G = (V, E)$, an integer k and a subset $R \subseteq E$ of red edges, find a randomized polynomial time algorithm that outputs a k-red perfect matching, i.e. a perfect matching M such that $|M \cap R| = k$ or NONE if a k-red perfect matching doesn't exist. The outputs should output be correct with a probability at least p.

Algorithm $\mathcal{A}(G, k, R)$ for finding the existence of a k-red perfect matching

1. Draw weight $w_{\{u,v\}}$ at random (uniformly and independently) from the the set $S = \{1, 2, ..., n^2\}$ for each edge in the graph G.
   - This step is polynomial because we have not more than $|V|^2$ edges in the graph.
2. Assign $X_{\{u,v\}} = w_{\{u,v\}}$ to the elements of the matrix A with dimensions $|V| \times |V|$:

$$A_{u,v} = \begin{cases} Y \cdot X_{\{u,v\}} & \text{if } \{u, v\} \in R, \\ X_{\{u,v\}} & \text{if } \{u, v\} \in E \setminus R, \\ 0 & \text{otherwise} \end{cases}$$

   - This step is polynomial because the matrix has $|V|^2$ elements (one for each edge that could exist in it).
3. Compute $\det(A)$ (polynomial) which can be interpreted as a polynomial $p(Y)$
   - This step is polynomial because we can compute the determinant in $O(n^3)$ with LU decomposition.
4. Find a_k the coefficient of Y^k of $p(Y)$ (polynomial) determined by the given k.
   - This step is polynomial as given in hint 2 of the exercise.
5. If $a_k = 0$, return NONE. Otherwise, return True.
   - This step is polynomial because we can check scalar equality in $O(1)$ and return a simple value in $O(1)$.

Hence, this algorithm runs in polynomial time.

Claim 1 The probability of returning NONE when a k-red perfect matching exists is less than $\frac{1}{n}$.

Proof The determinant is

$$\det(A) = \sum_{i=0}^{n} Y^i (\sum_{\sigma \in \sigma_i} \operatorname{sgn}(\sigma) \prod_{m=0}^{n} A_{m,\sigma(m)})$$

Where σ_i contains all the permutations $\sigma : [n] \to [n]$ with i red edges.

If a k-red perfect matching exists, then $g(i) = \sum_{\sigma \in \sigma_i} \operatorname{sgn}(\sigma) \prod_{m=0}^{n} X_{m,\sigma(m)}$ should be nonzero. $g(i)$ is a polynomial of degree n with variables $\{X_{i,\sigma(i)} \mid \sigma \in \sigma_i\}$. Since we draw at random from the set S, $|S| = n^2$, by the Schwartz-Zippel lemma, the probability that $g(i) = 0$ is less than $\frac{1}{n}$.

Identify which edges are part of of the matching First, the algorithm is always correct when outputting True because, assuming correct computation, a zero polynomial will always evaluate to zero no matter of the variables.

Run the described algorithm for $\mathcal{A}(G, k, E)$. If it outputs NONE, no k-red perfect matching exists with a probability at least $1 - \frac{1}{n}$. Repeat at least $m > -\frac{\log(p)}{\log(n)}$ times to achieve the desired probability p of not being wrong.

If still NONE was returned every time, return NONE to the user. Otherwise, find the matching.

Algorithm $\mathcal{B}(G, k, R)$ to find an exact k-red perfect bipartite matching We state the algorithm that we use afterwards.

Let $Z' = \emptyset$.

Repeat for each edge $e \in E$:

- Let $G' = (V, E')$ with $E' = E \setminus \{e\}$. Let R' be the accordingly updated set of red edges ($R' = R' \cap R$).
- Run the algorithm for $\mathcal{A}(G', k, R')$.
- If the algorithm returns True, e is not in all k-red matchings. Hence, update Z' to $Z' \leftarrow Z' \cup \{e\}$

- Otherwise, if it returns NONE, there is a probability at least $1 - \frac{1}{n}$ that it is part of all k-red perfect matchings. We do not add it to Z.

Return Z'.

Complexity This is done at most $|V|^2$ times as this is the highest possible number of edges in a graph. Thus $\mathcal{B}(G, k, R)$ is polynomial.

Finding the matching Afterwards, let Z be the set of edges to be removed. Initialise Z as $Z = \emptyset$. Let $E_{\text{new}} = E$ and $R_{\text{new}} = E_{\text{new}} \cap R$.

Now, repeat

- Let $G_{\text{new}} = (V, E_{\text{new}})$.
- Run the algorithm $\mathcal{B}$ with $\mathcal{B}(G_{\text{new}}, k, R_{\text{new}})$.
- Update Z with the returned Z' to $Z \leftarrow Z \cup Z'$.
- Update $E_{\text{new}} \leftarrow E_{\text{new}} \setminus Z$ and $R_{\text{new}} \leftarrow E_{\text{new}} \cap R$.

until $|E_{\text{new}}| = n$.

Then, G_{new} is a k-red perfect matching. Return E_{new} to the user.

Probability of correct return At the end of one full run, the remaining number of edges not part of a k-red perfect matching is upper bounded by the binomial law $B(|E| - n, \frac{1}{n})$. After $m - 1$ runs, in average, less than $L = (|E| - n) \cdot (1 - \frac{1}{n})^{m-1}$ edges which are not part of all k-red perfect matchings remain. The probability of eliminating all these edges during the m-th run is at least $(1 - \frac{1}{n})^L$. Thus, the number of runs m^* that is necessary to return a k-red perfect matching with at least a probability p is:

$$m^* = \frac{\log\left(\frac{\log(p)}{b \cdot \log(a)}\right)}{\log(a)} + 1$$

with

$$a = 1 - \frac{1}{n}, \quad b = |E| - n$$

The problem is solved to desired probability with $O(m^*)$ runs of a polynomial algorithm, thus the problem is solved in polynomial time.

Problem 4: Implementation

The submission #54523350 submitted by szw was accepted.