10-801: Advanced Optimization and Randomized Methods

Homework 4: Streaming algorithms, sketching

(March 1, 2014)

Instructor: *Suvrit Sra, Alex Smola* ***Due:*** *March 8, 2014*

1. A distribution P is said to be μ-stable, if for any constant $a, b \in \mathbb{R}$, and $X, Y, Z \sim P$, we have $aX + bY \sim (|a|^p + |b|^p)^{\frac{1}{p}} Z$.

   (a) Show that the standard Cauchy distribution ($\mu = 0, \gamma = 1$) is 1-stable and standard Gaussian distribution is 2-stable.

   (b) Given a data stream drawn iid from an even mixture of two categorical distributions p_X and p_Y supported on $\{x_1, ..., x_n\}$ and $\{y_1, ..., y_n\}$, where $x_i = y_i$ for all $i = 1, ..., n$ (there is a label that comes with each item indicating whether it is from p_X or p_Y). An estimator of the total variation distance (defined here) between the two distributions is

   $$\delta_1 = \frac{1}{m} \sum_{i=1}^{n} |f_i - g_i| \tag{4.1}$$

   where f_i, g_i are count of the number of times x_i and y_i appearing in the data stream. Show that this is a consistent estimator and provide the number of count one needs to track to compute this exactly for all stream length m.

   (c) Now consider another estimator

   $$\delta_2 = \frac{1}{m} \text{median}(|Af - Ag|) \tag{4.2}$$

   where $A \in \mathbb{R}^{p \times n}$ Cauchy random matrix (A_{ij} is drawn iid from standard Cauchy distribution). Again, show that this is also a consistent estimator (assume p and m both go to ∞) and it requires tracking only p values.

   (d) Show that (4.2) approximates (4.1). More precisely, show that if $p = O(\frac{1}{\epsilon^2})$ for probability greater than 0.5

   $$\delta_1(1 - \epsilon) \leq \delta_2 \leq \delta_1(1 + \epsilon).$$

   (Hint: use the part (a) and an appropriate concentration inequality)

2. One simple way to track the median of a stream of values is by the following algorithm

```
med=0
while (x arrives)
    if (x<med)
        med -=1
    elseif (x>med)
        med +=1
    endif
    yield med
endwhile
```

   Note that this takes only $O(1)$ memory and it should work reasonably well if the input data are well-normalized and not "pathological".

   (a) Find one "pathological" example that the algorithm fails. In particular, the sequence of median estimates gets worse as the number of data gets larger.

   (b) Using static increments 1 and -1 are likely to produce either a slow convergence, or an unstable output. This is analogous to the problem of choosing a proper step size in gradient descent.

   Reformulate the task in the machine learning setting as an ℓ_1 norm minimization problem, make sure that the median is an optimal solution of the optimization program you formulated. Derive a stochastic subgradient optimization algorithm.

(c) Find the conditions in an optimization textbook (or otherwise) upon which stochastic subgradient methods converge to a local minimum and explain whether the conditions are satisfied in this case.

3. Assume we are observing the stream $x_1 \dots x_i \dots$ with $x_i \in \mathbb{R}$. Unlike the settings discussed in class, these observations are not iid. Instead, the entries in the stream belong to one of the models below. You may assume that $x_1 = 0$.

   (a) $x_{i+1} = x_i + e_i$, where each e_i is independently drawn and can take the values $\{-1, 0, 1\}$, with probability distribution p_e.

   (b) $x_{i+1} = \sum_{j=1}^{\min(k,i)} \alpha_j x_{i-j+1} + e_i$, where e_i is defined as above and $\alpha_1 \dots \alpha_k$ are constants $\alpha_j \in \{+1, -1\}$.

   For each of the models: Provide an algorithm to approximate the counts for each unique element in the stream - only P counters are available at any given time. What is the expected error of your procedure for a unique value u after n samples are observed? The expectation is taken over the variables e_i. You may start by considering the cases $u = 0$ and $u = n$. Compute the expected error averaged over all possible variants. What is the improvement you obtain over an algorithm that makes no assumptions about the structure of the stream?

4. Assume we are observing the stream of integers $u_1 \dots u_i \dots u_n$ in the range $[-2^b, 2^b - 1]$. Let's say that, at some point after collecting the data, we would like to find which of the elements satisfy a set of properties given in the form of constraints $f_j(u) = \beta_j$, where $f_1 \dots f_m$ are smooth functions and β_j are constants. Note that the functions are not available during data collection, so it is not possible to simply evaluate the function values and store the results. Devise a procedure which will summarize a stream of data in a way that makes the retrieval of elements satisfying the constraints possible. What is the number of bits B that have to be used for this task? This number will depend on the number of elements in the stream and their distribution p.

5. We've seen the min-hash sampler in class. Reservoir sampling is another useful technique to sample s uniformly from a data stream $\{x[1], ..., x[m]\}$ (Vitter,1985). Algorithmically, it is very simple:

```
while (x[t] arrives)
    s = x[t] with probability 1/t
    yield s
endwhile
```

   (a) Show that at every $t = 1, ..., m$, the $\mathbb{P}(s = x[i]) = 1/t$.

   (b) Suppose we want to get k uniform samples from the data stream instead, how can we change the algorithm to achieve that?

   (c) It is often desirable to do non-uniform sampling instead. For instance, in graph sparsifier, we would like to sample the edges with probability proportional to its effective resistance, or in the sparsification of general matrices, one needs to sample the entries with probability proportional to their statistical leverage scores; and in the exponential mechanism for private unlimited supply auctions, one needs to sample the potential decisions a with probability proportional to $\exp(q(a|X))$ where q is the utility function for a particular action a conditioned on data X.
   Specifically, if we augment the data stream with a list of weights $\{w[1], ..., w[m]\}$ (e.g., $w[i] = \|x[i]\|$), design an algorithm by modifying what you have in (b) (or otherwise) to sample k items from the data stream such that at all $t = k, k+1, ..., m$, the probability that any item $x[i]$ being picked in sample S is proportional to its weight $w[i]$, i.e., for any pair of item $(x[i], x[j])$, the

$$\frac{\mathbb{P}(x[i] \in S)}{\mathbb{P}(x[j] \in S)} = \frac{w[i]}{w[j]}.$$

   To avoid exceptions, you may assume at any t, $\frac{w(t)}{\sum_{i=1}^{t} w_i} \leq \frac{1}{k}$.

   (d) (Bonus question) It is mostly likely that both your algorithms in (b) and (c) need $O(k)$ active memory and $O(k)$ randomized operations when each item arrives. This is very undesirable in practice when k is large. The question is can we do better (e.g., $O(\log k)$ active memory and $O(1)$ operation at every step) by using some additional disk space? Make any assumptions that you deem necessary and justify the your answer.