Scalable Machine Learning CS281B, STAT241B
http://alex.smola.org/teaching/berkeley2012
UC Berkeley

January 21, 2012
due on March 6, 2012
Alexander J. Smola

Problem Set 2 — Streams and Optimization

1 Quantile Regression (15)

Quantile regression provides a mechanism for estimating not only the average behavior but rather also more precisely what constitutes normal or unusual behavior. This is useful, e.g. when estimating children's growth curves, as conditioned on age, gender, race, and societal background. Likewise it helps when estimating the *value at risk* of traded securities, e.g. in the context of option pricing. Overall, it helps whenever we want to obtain not necessarily the mean of a random variable but a reliable upper or lower bound on it.

1.1 Properties of Losses

Given a set of numbers $Y := \{y_1, \ldots, y_m\} \subset \mathbb{R}$ prove that the following properties hold:

1. The solution of the following problem is the mean of Y

$$g(Y) := \operatorname*{argmin}_g \sum_{i=1}^{m} \frac{1}{2}(y_i - g)^2 \tag{1}$$

2. The solution of the following problem is the mean of Y

$$g(Y) := \operatorname*{argmin}_g \sum_{i=1}^{m} |y_i - g| \tag{2}$$

3. The solution of the following problem computes a quantile of Y

$$g(Y) := \operatorname*{argmin}_g \sum_{i=1}^{m} [a \max(0, y_i - g) + b \max(0, g - y_i)] \text{ for } a, b > 0. \tag{3}$$

That is, it finds some g such that for $\nu \in (0, 1)$ we have $\sum_i \{y_i < g\} \leq \nu m$ and $\sum_i \{y_i > g\} \leq (1-\nu)m$. Express a and b in terms of ν? Hint, the problem is invariant under certain transformations in (a, b). Fix that parameter. Use the ν (new) parametrization for the remainder of the problem.

1.2 Optimization Problem

We now use the above loss functions to derive a convex optimization problem for quantile estimation. Here we assume that we observe m pairs (x_i, y_i) with $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. Use a linear function class

$$f(x) = \langle w, x \rangle + b, \tag{4}$$

and the loss function in (3), i.e.

$$l(f(x), y) := [a \max(0, y - f(x)) + b \max(0, f(x) - y)]$$

and a quadratic penalty $\frac{1}{2} \|w\|^2$ on w as in SVMs.

1. Write out an equivalent constrained quadratic optimization problem.
2. Compute the dual objective function. Hint — you will observe that the problem depends on x only in terms of inner products $\langle x_i, x_j \rangle$.
3. Write out the kernelized version of the algorithm.

Problem Set 2 — Streams and Optimization

2 Efficient Stochastic Gradient (10)

2.1 Quantile Regression updates

Derive a stochastic gradient descent update equation for the quantile regression problem. The learning rate η_t is given by $\frac{1}{\sqrt{\alpha t+\beta}}$ and the regularization penalty is $\frac{1}{2}\|w\|^2$.

2.2 Sparse updates

Assume that x_i is sparse. How can you avoid updating all coordinates at each stochastic gradient descent step?

2.3 p-norm regularization

Now assume that we change the capacity penalty from $\frac{1}{2}\|w\|^2$ to

$$\Omega[w] := \frac{1}{p}\sum_{i=1}^{p} |w_i|^p \text{ for } p \geq 1. \tag{5}$$

Show that the problem remains convex.

2.4 Lazy updates

Find an adjusted efficient coefficient updating rule that avoids updating each coefficient at each iteration for sparse x_i.

Hint — since the x_i are sparse you only need to know a coordinate of w whenever there is a corresponding nonzero term in x_i. Hence you can be lazy about updating w. Moreover, you may replace the discrete update steps by a differential equation and approximate sums over learning rates by an integral.

3 Weighted SpaceSaving (10)

SpaceSaving has many desirable properties, such as rate optimality. That said, the algorithm described by Metwally et al. discusses only unweighted inserts. Your goal is to modify the update algorithm to allow for weighted updates. In other words, the algorithm needs to be able to deal with (x_i, w_i) with $w_i \geq 0$ insert operations, rather than simply $(x_i, 1)$.

3.1 Update Equations

Derive update equations for weighted inserts and prove that the $\frac{n}{k}$ accuracy bound remains valid, where now $n = \sum_i w_i$. That is, for the estimate $\hat{n}_x$ we have

$$n := \sum_i w_i \text{ and } n_x := \sum_{i:x_i=x} w_i \text{ we have } n_x \leq \hat{n}_x \leq n_x + \frac{n}{k} \tag{6}$$

whenever x is in the list of the top k instances.

Problem Set 2 — Streams and Optimization

3.2 Properties

Denote by $\hat{n}_x$ the count of the smallest element in the list of SpaceSaving. Prove that any item with count $n_x > \hat{n}_x$ must be in the list.

3.3 Weighted Decay

In some case we may want to compute a Laplace transform of the data stream we observe. That is, we attach a weight $e^{(t-T)\lambda}$ to an instance arriving at time t. Use weighted SpaceSaving to compute an estimate of item frequencies under this weighted stream.

3.4 Weighted decay with unknown time horizon

Now assume that T is unknown. How do you need to modify the sketch such that the insert (and query) operations are not significantly more expensive than an unweighted estimate. In other words, you should not have to down-weight all coefficients at every iteration. Hint — you can assume that the exponent in floating point numbers is rather large but not infinite.

4 Load Balancing (10)

Bennett's inequality provides exponential tail bounds sums of random variables as long as their range and variance is well controlled.

Theorem 1 *Denote by X_i independent zero-mean random variables with $|X_i| \leq a$ almost surely for all i. Furthermore, let*

$$X := \sum_i X_i \text{ and } \sigma^2 := \sum_i \operatorname{Var}[X_i]. \tag{7}$$

Then the following tail bound holds

$$\Pr\{X > t\} \leq \exp\left(-\sigma^2 a^{-2} h(\sigma^{-2} a t)\right) \text{ where } h(\xi) = (1+\xi)\log(1+\xi) - \xi. \tag{8}$$

We now study the efficacy of randomized load balancing schemes. For this purpose we use consistent hashing. That is, we distribute keys x over a pool of machines $\mathcal{M}$ according to

$$m(x, \mathcal{M}) = \operatorname*{argmin}_{m \in \mathcal{M}} h(x, m). \tag{9}$$

4.1 Expected load and variance

Assume that the total number of symbols is n, that n_x is the number of occurrences of a single symbol and that m is the number of distinct symbols. Furthermore let $l := |\mathcal{M}|$.

1. Compute the expected load per machine.
2. Compute the variance in the load distribution for a machine.

4.2 Simplifying Bennett

The scaling term $h(\xi)$ is inconvenient since it is not easily invertible. Derive an upper bound on Bennett's tail inequality by using a Taylor expansion on $h(\xi)$. Hint — for the subsequent calculations it is advantageous to expand at $e^a - 1$ for $a \in \mathbb{R}$.

4.3 Tail bound

1. Derive a tail bound for largest number of symbols arriving at a given machine when using consistent hashing.
2. Derive a bound for the largest machine load for all l machines. Hint — apply the union bound over all l machines.

5 Sketches (20)

The CountMin sketch is well suited to turnstile statistics. That is, it allows the insertion and *removal* of items while keeping the same accuracy as if we had never inserted the item in the first place. In some cases this is not necessary. Your goal is to derive and test a better sketch in the case where we have only inserts.

5.1 Update equations

Modify the update equation $M[i, h_i(x)] \leftarrow M[i, h_i(x)] + 1$ to obtain a tighter bound. Hint — you do not need to increment all counters (to an equal amount). Think of how the Bloom filter was modified.

5.2 Properties

Prove that your improved sketch sill satisfies.

$$n_x \leq \min_i M[i, h_i(x)] \tag{10}$$

Bonus points if you prove that it still satisfies the tail bounds of the CountMin sketch paper (you do not need to prove the Zipfian distribution bounds).

5.3 Implementation

Implement the following sketches:

1. CountMin sketch.
2. Improved CountMin sketch for insert-only operations.
3. SpaceSaving sketch. Hint — the data structure you might want is a bimap. You can find efficient implementations thereof in Boost (for C++ and with R bindings) and Java. Alternatively look at *bisect* for Python (you'll need to keep two sorted lists with pointers to the actual objects).

5.4 Top k words

Download the preprocessed English Wikipedia corpus from

`http://alex.smola.org/teaching/berkeley2012/assignments/enwiki.txt.bz2`

This is the latest Wikipedia dump as of January 4, 2012. I removed all XML metadata and additional structure in it. Only section headers are denoted by some minor markup. Compressed the file is over 2GB, uncompressed over 7GB. It is compressed using `bzip`. Decompressors can be found at `http://bzip.org/`.

1. Find the approximate top 1000 words and their counts using SpaceSaving with $k = 2000$, i.e. only use the top 1000 words from your list in the end. Hint — almost certainly you will be unable to load the Wikipedia file into memory. You need to stream things from disk using basic file IO.

2. Using this list, compute their exact word counts. Hint — you can probably reuse most of the code. You only need to run the algorithm with the 'correct' words seeded and without the option of removing them from the list.

5.5 CountMin comparison

1. Compute the CountMin and improved CountMin statistics.
2. Use the list of top 1000 words obtained by the SpaceSaver sketch to query their corresponding item frequencies.
3. Plot the relative overestimates for the three sketches when ordered by actual frequencies of occurrence for the top 1000 words. Hint — since all ratios will be slightly larger than 1 it is advantageous to subtract 1 from the ration when plotting.

Note, the file is 2GB compressed and a bit over 7GB uncompressed. This is the latest dump of the English Wikipedia as per January 4, 2012. The data has been reduced from XML to text format with some minor annotation remaining.

Hint - you will probably not be able to load the entire file into memory. Use basic file I/O routines to deal with this problem.

Your task is: Pick the words in this obscure and self referential phrase and compute their item true and estimated item frequencies. Using a 100MB to 1GB sketch is enough.

1. Correct item frequencies.
2. Using the CountMin sketch.
3. Using the improved CountMin sketch.

5.6 Space Saving

Compute a sketch of the $k = 1000$ most frequent items (it's sufficient to find the find an approximate list using your sketch). Hint — the Boost bimap will make your life much easier in C++. `http://www.boost.org/doc/libs/1_42_0/libs/bimap/doc/html/index.html`

Compare the accuracy between the following four statistics:

1. Accurate counts.
2. Upper bounds using SpaceSaving.
3. Upper bounds using the CountMin sketch.
4. Upper bounds using the improved CountMin sketch.

Reminder for Project Presentations

Project presentations are due on **March 13**. The presentations will start at 4pm, tentatively until 8pm (i.e. until all presentations are done). Please e-mail slides and supporting documentation to Dapo by Sunday night, i.e. by **March 11 midnight**. Also let him know if you cannot stay late. By default the order is alphabetical (i.e. a sorted list of sorted lists of team members), e.g. $\{A, D, E\}$ comes before $\{B, C, F\}$. Criteria:

- A maximum of 6 pages for the 10 minute presentation. Only **PDF** is acceptable. Maximum file size is **3MB**. Slides must be readable from the last row of the lecture theater.
- Up to 10 pages supporting documentation using the formatting prescribed by the ACM (they have LaTeXand Word templates)
  `http://www.acm.org/sigs/publications/proceedings-templates`

Problem Set 2 — Streams and Optimization

- Your presentation and write up should address the following issues
  1. What are you planning to do (goal, data, algorithm, etc.)
  2. What are the challenges and risks?
  3. What have you achieved so far?
  4. What is different or new?
  5. Why do you think that you'll succeed?
- By the presentation time you should definitely have resolved the data issue and have some preliminary experiments that show that what you're planning to do might actually work. Think of this as good practice for pitching to a VC or writing a grant proposal.