CSE 584A Class 25

Jeremy Buhler

April 25, 2018

1 How Big a Sketch is Needed to Estimate J_k?

- Min-hashing estimates the overlap of S with T by sampling m k-mers uniformly from $S \cup T$ and measuring the fraction w/m of the sample that falls in $S \cap T$.
- How accurate is w/m as an estimate of $J_k(S,T)$?
- Clearly, each of the m sampled k-mers is shared with probability $J_k(S,T)$ (since we sample uniformly from the union).
- Hence, $E[w] = m \cdot J_k(S,T)$; that is, the estimate has the right mean.
- But if J_k is really small, then the value of w in a small sample is likely to have *high variance*, i.e. it is very noisy.
- Since w is a sum of independent events (is each randomly sampled k-mer in the intersection?), we can use a Chernoff bound to quantify the chance that w/m has a large deviation from its mean.
- In particular, Koslicki and Zabeti (2017) note that

$$\Pr\left(\left|\frac{w/m - J_k(S,T)}{J_k(S,T)}\right| \geq \delta\right) \leq 2e^{-\delta^2 m J_k(S,T)/3}.$$

- If we know the approximate value of J_k and can set an error bound for δ (say, 0.1 for 10% relative error), we can find the smallest sketch size m needed to obtain this error with probability at most, say, $1 - \epsilon$.
- In particular, we need

$$m \geq \frac{-3\ln(\epsilon/2)}{\delta^2 J_k(S,T)}.$$

- To plug in some numbers, suppose we want fairly high accuracy ($\delta \leq 0.1$, i.e. 10% error) with probability at least 0.99.
- Then if we expect J_k to be about 0.9 (high overlap), we get a minimum sketch size $m = 1777$.
- But if J_k is only 0.5 (moderate overlap), the same accuracy requires $m = 3179$.
- And if J_k is 0.1 (not much overlap), we need $m = 15895$.
- In general, the required sketch size for fixed ϵ, δ grows as $1/J_k$.

2 Containment – a Relative of Similarity

The error behavior of min-hashing is a serious problem for applications with small expected overlap between S and T.

- If we have two big sequences with small anticipated overlap, oh well – we're stuck using big sketches (and may want to skip sketching altogether if the space cost, i.e., m, becomes unreasonable).
- But what if we want to measure overlap between a big S and a small T?
- For example, suppose we have a big collection S of DNA (e.g. a metagenome), and we want to know if it contains a particular (say) microbial genome T.
- S is billions of bases, while T is likely just a few thousand to a few million.
- Even if $\Sigma_k[T]$ is a subset of $\Sigma_k[S]$, $J_k(S,T)$ is bounded by $|\Sigma_k[T]|/|\Sigma_k[S]|$, which in this case is tiny.
- Hence, we will still need a ridiculously big sketch size to estimate J_k with any sort of accuracy.

Let's consider an alternate approach that needs smaller sketches.

- Let S and T be sequences with k-spectra $\Sigma_k[S]$ and $\Sigma_k[T]$, such that S has many more k-mers than T.
- We want to measure *how much of T is present in S*.
- The *containment index* $C_k(S,T)$ is the fraction of unique k-mers in $\Sigma_k[T]$ that are present in $\Sigma_k[S]$.
- Formally,
$$C_k(S,T) = \frac{|\Sigma_k[S] \cap \Sigma_k[T]|}{|\Sigma_k[T]|}.$$
- Like J_k, C_k lies between 0 and 1.
- If T is completely contained in S (e.g., T is a substring of S), $C_k(S,T) = 1$.
- Just as for J_k, C_k is insensitive to rearrangement.

How can we use min-hashing to estimate C_k?

- Compute a min-hash sketch of size m for the smaller sequence T.
- Now count the number c of k-mers in this sketch that occur in S.
- As before, we can argue that $E[c] = mC_k(S,T)$, since our sketch samples k-mers uniformly from T.

- And, applying the same Chernoff bound, we get

$$\Pr\left(\left|\frac{c/m - C_k(S,T)}{C_k(S,T)}\right| \geq \delta\right) \leq 2e^{-\delta^2 m C_k(S,T)/3},$$

  and hence

$$m \geq \frac{-3\ln(\epsilon/2)}{\delta^2 C_k(S,T)}.$$

- Hence, for moderately large values of C_k, we can get accurate estimates with much smaller sketches than it would take to estimate the (much smaller) J_k by the previous approach.

We can in fact convert our estimate of C_k into one of J_k.

- Observe that

$$\begin{aligned} J_k(S,T) &= \frac{C_k(S,T)|\Sigma_k[T]|}{|\Sigma_k[S] \cup \Sigma_k[T]|} \\ &= \frac{C_k(S,T)|\Sigma_k[T]|}{|\Sigma_k[S]| + (1 - C_k(S,T))|\Sigma_k[T]|}. \end{aligned}$$

- To estimate J_k, we use the empirical counts c and $m - c$ as our estimates of $C_k(S,T)|\Sigma_k(T)|$ and $(1 - C_k(S,T))|\Sigma_k[T]|$, respectively .

- The error of this estimate for J_k is higher than for C_k alone, but it is typically much lower than that obtained by estimating J_k by min-hash intersection when T is much smaller than S. (See Koslicki and Zabeti for the bound.)

- *NB*: to compute J_k from our estimate of C_k, we need to know $|\Sigma_k[S]|$. We can either count unique k-mers of S exactly or use an approximate counting method such as, e.g., the HyperLogLog algorithm (Flajolet et al. 2007).

3 Implementing C_k Estimation

- When we estimated J_k, we sketched both S and T.

- In contrast, estimating C_k sketches only the smaller sequence T.

- We need to check for each k-mer in the sketch whether it appears in S.

- How expensive is this check?

How to implement the check depends on the application.

- Which is the fixed reference sequence, S or T?

- If T is fixed and S varies, we can build any sort of efficient index from our sketch of T and scan S in time $O(|S|)$ to mark all the k-mers in the sketch that appear in S.

- The best indices we've discussed take space $O(m)$, with various constant factors, and $O(k)$ lookup time.

- *Example*: T is a reference microbial genome, S is a metagenome, and we want to know if S contains T.
- In practice, we do this not for a single reference T but for *lots* of references (think every microbial genome in GenBank).
- If instead S is fixed and T varies, we index S and then compute c in time proportional to m.
- In this case, our best indices take space $O(|\Sigma_k[S]|)$, again with various constant factors, and $O(k)$ lookup time.
- *Example*: S is an archived metagenome, T is a newly discovered virus, and we want to know if T is present in S.
- Here again, we may have a lot of references S.

What if we can't afford the space needed to keep around large indices (e.g. if S is big) or numerous indices (e.g. if we have many reference genomes T)?

- If S is too big to index conveniently, and we are willing to accept some error in our computation of c, we can use *Bloom filters* to create an approximate index.
- A Bloom filter is an array A of q bits, together with a bunch of hash functions $h_1 \dots h_r$ that map k-mers to the range $[0, q-1]$.
- Initially, A contains only zero bits.
- To index a k-mer s, we set bits $h_1(s) \dots h_r(s)$ of A to one (if they aren't already set).
- To check if a k-mer t is in the index, we check whether all of bits $h_1(t) \dots h_r(t)$ of A are ones.
- This approach has no false negatives (i.e. it always identifies k-mers that are in S), but it can have false positives if all the bits $h_1(t) \dots h_r(t)$ were set by other k-mers from S.
- It's not hard to compute the false positive rate of a Bloom filter for given q and r, assuming that the hash functions are uniform and independent.
- In general, we can get usefully low false positive rates even with $q \ll |\Sigma_k(S)|$.
- Moreover, we can lower the false positive rate by "stacking" several independent Bloom filters and requiring that a k-mer hits in all of them.
- Again, see Koslicki and Zabeti for details, including how to adjust the predicted error rate of C_k estimation to account for the false positives generated by a Bloom filter.

OK, that helps for large, fixed S. But what about the case of numerous small, fixed T?

- We can merge many small indices for the sketches of different T into fewer, bigger indices.

- For example, we could build a joint hash map on the k-mers in the sketches of all T, where each k-mer maps to the list of sequences T in whose sketches it occurs.
- Whenever a k-mer of S matches a k-mer t in the table for the first time, we mark s as "seen" and increment counts $c(T)$ for all T whose sketches contain t.
- Something like this approach is used in Mash.
- Alternatively, if an exact index of all T takes too much space, we can use *hierarchical Bloom filters* (Solomon and Kingsford, 2016).
- Build a tree on the references T. At each node v of the tree, create a Bloom filter from all the k-mers appearing in any sequence in v's subtree.
- To determine whether a query k-mer $s \in S$ appears in any reference, we check if s hits the root Bloom filter.
- If s hits at the root, we check each of its children, and so forth until we've identified every reference (leaf) T that (probably) contains s.
- We want to build the tree so that more similar genomes end up in the same subtree, to minimize the loading of the internal nodes' Bloom filters.
- If the sequences T are genomes, we can group them into a tree by their taxonomy.
- The *sourmash* library (Brown and Irber, 2017) uses hierarchical Bloom filters.

4 Some Important Open Problems

- In the above applications, we are trying to answer the question "Is T homologous to a subset of S"?
- For any S and T, we can compute $C_k(S, T)$, with small error if C_k is not too small.
- But how large a value of C_k is biologically meaningful?
- In particular, how large must C_k be before we can reject the null hypothesis that T is not homologous to any part of S?
- Ondov et al. offer one way to estimate p-values for this null hypothesis.
- Moreover, suppose references T_1 and T_2 both match in S with large enough C_k to reject the null hypothesis.
- If T_1 and T_2 are themselves similar (e.g. genomes of different bacterial strains in one species, or of closely related species), can we tell if S really contains T_1, T_2, or both?
- *Example*: *E. coli* strains K-12 and 0157:H7 have fairly similar genomes. One is benign; the other is pathogenic. If both exhibit a high containment score in a metagenome S derived from a sample of lettuce, should the CDC recommend a recall of that lettuce?
- As a related question, what if S contains sequences similar to some reference genome T, but not T itself? (E.g. a new, undocumented bacterial strain.)

- Can you tell from C_k or other statistics how closely related T is to the sequences in S?