6.889 Sublinear Time Algorithms	October 20, 2020

Lecture 14: Poissonization and Closeness Testing

Lecturer: Ronitt Rubinfeld *Scribe: Pachara Sawettamalya*

1 Overview

Today we will show how to test closeness of two *unknown* distributions in a sublinear number of samples. The Plug-in Estimator that we saw in previous lectures posits two major concerns: the dependency and the L_2-norm. We will introduce a few procedures that can help us resolve these issues.

2 Setting

2.1 Probability Distribution

For a probability distribution p on a discrete domain $\mathcal{D}$, for any $i \in \mathcal{D}$, we refer to $\mathbf{Pr}_{x\in\mathcal{D}}(x = i)$ as p_i or $p(i)$ depending on the context.

2.2 Distances and Norm

Suppose we have two distributions p and q on the same discrete domain $\mathcal{D}$, we define two distances functions as follows.

- L_1-distance : $||p - q||_1 = \sum_{i\in\mathcal{D}} |p_i - q_i|$
- L_2-distance : $||p - q||_2 = \sqrt{\sum_{i\in\mathcal{D}} (p_i - q_i)^2}$

In addition, we denote an L_2-norm of a distribution p as $\sqrt{\sum_{i\in\mathcal{D}} p_i^2}$.

2.3 Closeness Testing

Given two *unknown* distributions p and q on the same size-n domain $\mathcal{D}$. This means we cannot make any prior assumption of p or q. The only operation allowed is to sample from either distributions. We want to determine a tester with the following behaviors.

- If $p = q$, then output PASS
- If $||p - q||_1 \leq \varepsilon$, then output FAIL with probability at least $\frac{3}{4}$

The rest of this note is dedicated to showing the tester with $O(n^{\frac{2}{3}}\varepsilon^{-\frac{4}{3}})$ samples. It is also worth recalling the *typical strategy*, such as the one used in the Plug-in Estimator, is as follows. We take a multiset S of m samples and count x_i = the number of i's occurrences in S. We note here that there are two major concerns about this approach: the dependency of x_i's and the magnitude of p's L_2-norm.

3 Resolving Dependency via Poissonization

The first concern that arises is the dependency of x_i's. This is because we only limit the total number of samples to m. For example, $x_i > \frac{m}{2}$ infers $x_j < \frac{m}{2}$ for the rest of the domain. To resolve this problem, we introduce the Poisson Distribution.

Definition 1 *A random variable X is said to have a Poisson Distribution* $\mathbf{Pois}(\lambda)$ *if for any non-negative integer k,* $\mathbf{Pr}(x = k) = \frac{e^{-\lambda}\lambda^k}{k!}$*. In addition, we have* $\mathbb{E}[X] = \mathrm{Var}[X] = \lambda$.

We now propose two algorithms for sampling x_i's. In Algorithm 1, we first fix $\hat{m}$ drawn from a Poisson distribution and do $\hat{m}$ sampling, while in Algorithm 2, we individually sample the number of occurrences of x_i based on a Poisson distribution.

Algorithm 1 : Single-Poissonization$(p, \mathcal{D})$

```
1: m̂ ← Pois(m)
2: S_1 ← m̂ samples from distribution p
3: for i ∈ [n] do
4:     x_i^(1) ← occurrences of i in S_1
```

Algorithm 2 : Individual-Poissonization$(p, \mathcal{D})$

```
1: S_2 ← φ
2: for i ∈ [n] do
3:     Sample x_i^(2) ∈ Pois(mp_i) and add x_i^(2) copies of i to S_2
4: Randomly permute S_2
```

Claim 2 *The distributions of x_i's in Algorithm 1 and Algorithm 2 are equivalent.*

Proof Let's consider

$$\begin{aligned}
\mathbf{Pr}(x_i^{(1)} = c) &= \sum_{k \geq c} \mathbf{Pr}(\hat{m} = k) \cdot \binom{k}{c} p_i^k (1-p_i)^{k-c} \\
&= \sum_{k \geq c} \frac{e^{-m} m^k}{k!} \cdot \frac{k!}{c!(k-c)!} \cdot p_i^c (1-p_i)^{k-c} \\
&= \frac{e^{-m} \cdot (mp_i)^c}{c!} \cdot \sum_{k \geq c} \frac{m^{k-c}(1-p_i)^{k-c}}{(k-c)!} \\
&= \frac{e^{-m} \cdot (mp_i)^c}{c!} \cdot e^{m(1-p_i)} = \frac{e^{-mp_i} \cdot (mp_i)^c}{c!} = \mathbf{Pr}(x_i^{(2)} = c)
\end{aligned}$$

In addition, we can check that the joint distributions are equivalent in a similar analysis. ■

A key observation is that the sampling of $x_i^{(2)}$'s from Algorithm 2 is independent. Since the distribution from both algorithms are equivalent, it must also be independent in Algorithm 1.

4 Resolving L_2-norm via Reduction

It is also known that the magnitude of distribution's L_2-norm affects the sample complexity. In particular, distributions with large L_2-norm requires more samples to check. For this reason, we can now state our goal. We need a way to transform the initial distributions (p, q) to (p', q') such that

- If $p = q$, then $p' = q'$
- If $||p - q||_1 \geq \varepsilon$, then $||p' - q'||_1 \geq \varepsilon$
- $||p'||_2$ is small.

It is crucial to note that this reduction does not require $||q'||_2$ to be small though it would be ideal. We will discuss in the next section that the small $||q'||_2$ is unnecessary as we can circumvent it.

We propose a procedure to transform (p, q) into (p', q') as follows.

Algorithm 3 : Transform(p, q)

1: $\mathcal{D}' \leftarrow \phi$
2: $\hat{m} \leftarrow \text{Pois}(m)$
3: $S \leftarrow \hat{m}$ samples drawn from p over domain $[n]$
4: **for** $i \in [n]$ **do**
5:     $b_i \leftarrow$ occurrences of i in S
6:     add $b_i + 1$ elements $(i, 1), (i, 2), ...(i, b_i + 1)$ to $\mathcal{D}'$
7: $p' \leftarrow$ a distribution on $\mathcal{D}'$ where $p'(i, j) = \frac{p_i}{b_i+1}$
8: $q' \leftarrow$ a distribution on $\mathcal{D}'$ where $q'(i, j) = \frac{q_i}{b_i+1}$
9: Output (p', q')

Since p_i's and q_i's are unknown, we cannot directly construct p' and q' based on the probability distributions given in Line 7-8; however, there is an easy fix. For p', we can sample i from the distribution p and sample j uniformly from $\{1, 2, ..., b_i + 1\}$, and likewise for q'.

Furthermore, notice that $\hat{m} = O(m)$ with high probability. This means that with high probability, Algorithm 3 only uses $O(m)$ samples. Plus, we have $|\mathcal{D}'| = \sum_{i\in[n]} b_i + 1 = n + \sum_{i\in[n]} b_i = n + |S| = n + \hat{m}$.

We will now show that the output (p', q') has the desired behaviors. The first behavior is trivially true – if $p = q$ then the probability distribution of p' and q' will be identical which is equivalent to $p' = q'$.

Claim 3 $||p - q||_1 = ||p' - q'||_1$

This is true as we can see that

$$||p' - q'||_1 = \sum_{i\in[n]} \sum_{j\in[b_i+1]} |p'(i, j) - q'(i, j) = \sum_{i\in[n]} \sum_{j\in[b_i+1]} \frac{|p_i - q_i|}{b_i + 1} = \sum_{i\in[n]} |p_i - q_i| = ||p - q||_1$$

Claim 4 $\mathbb{E}\left[||p'||_2^2\right] \leq \frac{1}{m}$

Proof First of all, recall that as we discussed in Section 3, the distribution of b_i is indeed Pois(λ_i) when $\lambda_i = mp_i$. This implies

$$\mathbb{E}\left[\frac{1}{b_i + 1}\right] = \sum_{k\geq 0} \frac{1}{k + 1} \cdot \frac{e^{-\lambda_i}\lambda_i^k}{k!} = \frac{1}{\lambda_i} \cdot \sum_{k\geq 1} \frac{e^{-\lambda_i}\lambda_i^{k+1}}{(k + 1)!} \leq \frac{1}{\lambda_i} = \frac{1}{mp_i}$$

Therefore we will have

$$\begin{aligned}\mathbb{E}\left[||p'||_2^2\right] &= \mathbb{E}\left[\sum_{i\in[n]} \sum_{j\in[b_i+1]} p'(i, j)^2\right] = \mathbb{E}\left[\sum_{i\in[n]} \sum_{j\in[b_i+1]} \left(\frac{p_i}{b_i + 1}\right)^2\right] = \mathbb{E}\left[\sum_{i\in[n]} \frac{p_i^2}{b_i + 1}\right] \\ &= \sum_{i\in[n]} p_i^2 \cdot \mathbb{E}\left[\frac{1}{b_i + 1}\right] \leq \sum_{i\in[n]} \left(p_i^2 \cdot \frac{1}{mp_i}\right) = \frac{1}{m} \cdot \sum_{i\in[n]} p_i = \frac{1}{m}\end{aligned}$$

This completes the proof. ■

5 Circumventing Large $||q'||_2$

Theorem 5 *Given distributions p, q on a discrete domain of size n and $b \geq \max(||p||_2, ||q||_2)$. Then we can distinguish the case of $p = q$ from $||p - q||_1 \geq \varepsilon$ in $O\left(\frac{bn}{\varepsilon^2}\right)$ samples.*

We will not prove this theorem, but we rather give a sketch proof to a useful corollary.

Corollary 6 *Given distributions p, q on a discrete domain of size n and $b \geq \min(||p||_2, ||q||_2)$. Then we can distinguish the case of $p = q$ from $||p - q||_1 \geq \varepsilon$ in $O\left(\frac{bn}{\varepsilon^2} + \sqrt{n}\right)$ samples.*

Sketch of Proof The following procedure gives the desired tester.

1. Estimate $||p||_2$ and $||q||_2$ within a multiplicative factor of C when $C > 1$ is a constant. This can be done within $O(\sqrt{n})$ samples.
2. If the estimated $||p||_2$ and $||q||_2$ are more than a multiplicative factor of C away, output FAIL. We can do this because if it is the case, we can guarantee that $||p||_2 \neq ||q||_2$ which implies $p \neq q$.
3. Run the tester from Theorem 5 with $b' = Cb$, and gives the same output. We only need to verify that $b' = Cb \geq C \cdot \min(||p||_2, ||q||_2) \geq \max(||p||_2, ||q||_2)$ where the last inequality follows from the fact that $||p||_2$ and $||q||_2$ are at most a multiplicative factor of C away. This only uses $O\left(\frac{bn}{\varepsilon^2}\right)$ samples according to Theorem 5.

This completes the proof. ■

The key takeaway from Corollary 6 is that we only need $||p'||_2$ to be small but not necessary for $||q'||_2$. This because we can set $b = ||p'||_2$ regardless of q'.

6 The Closeness Tester

Now we can give a tester for closeness between two distributions as follows.

Algorithm 4 : Closeness-Tester(p, q)

1: $k \leftarrow n^{\frac{2}{3}}\varepsilon^{-\frac{4}{3}}$
2: $(p', q') \leftarrow (p, q)$ transformed by Algorithm 3 with $m = k$
3: Run the tester from Corollary 6 on (p', q') and give the same output

To begin with, we notice that Algorithm 4 can distinguish $p' = q'$ and $||p' - q'||_1 \geq \varepsilon$ due to Corollary 6. Plus, in section 4 we have shown that (p, q) and (p', q') are equivalent in closeness testing. Therefore Algorithm 4 successfully distinguishes $p = q$ and $||p - q||_1 \geq \varepsilon$.

Last but not least, let's analyze the number of samples needed.

Claim 7 *With high probability, Algorithm 4 uses $O(n^{\frac{2}{3}}\varepsilon^{-\frac{4}{3}})$ samples*

Proof We have established that with high probability, transforming (p, q) into (p', q') uses at most $O(k) = O(n^{\frac{2}{3}}\varepsilon^{-\frac{4}{3}})$ samples.

In addition, Claim 4 gives $\mathbb{E}(||p'||_2^2) \leq \frac{1}{k}$ which implies $||p'||_2 = O\left(\frac{1}{\sqrt{k}}\right)$ with high probability. This also means we can set $b = O\left(\frac{1}{\sqrt{k}}\right)$ in Corollary 6, and the tester will use $O(\frac{n}{\varepsilon^2\sqrt{k}} + \sqrt{n}) = O(n^{\frac{2}{3}}\varepsilon^{-\frac{4}{3}})$ samples.

In total, Algorithm 4 uses $O(n^{\frac{2}{3}}\varepsilon^{-\frac{4}{3}})$ samples. ■