Testing probability distributions using conditional samples

(when testers get to be picky)

Clément Canonne* Dana Ron† Rocco Servedio*

* Columbia University
† Tel-Aviv University

March 8th, 2013
Plan of the talk

1. Introduction

2. Testing Uniformity

3. Tools and subroutines

4. Back to uniformity

5. Conclusion
Background and motivation
What is distribution testing?

Property testing
Given a big, hidden “object” $X$ one can only access by local, expensive inspections (e.g., oracle queries), and a property $P$, the goal is to check in sublinear number of inspections if (a) $X$ has the property or (b) $X$ is “far” from all objects having the property.\(^1\)

\(^1\)wrt to some specified metric, and parameter $\epsilon > 0$ given to the tester.
What is distribution testing?

Property testing

Given a big, hidden “object” $X$ one can only access by local, expensive inspections (e.g., oracle queries), and a property $\mathcal{P}$, the goal is to check in sublinear number of inspections if (a) $X$ has the property or (b) $X$ is “far” from all objects having the property.\footnote{wrt to some specified metric, and parameter $\varepsilon > 0$ given to the tester.}

Testing distributions (standard model)

$X$ is an unknown probability distribution $D$ over some $N$-element set; the testing algorithm has blackbox sample access to $D$. 
Distribution testing (1)
In more details.

Distance criterion: total variation distance ($\propto L_1$ distance)

$$d_{TV}(D_1, D_2) \stackrel{\text{def}}{=} \frac{1}{2} \| D_1 - D_2 \|_1 = \frac{1}{2} \sum_{i \in [N]} |D_1(i) - D_2(i)|.$$  

Definition (Testing algorithm)

Let $\mathcal{P}$ be a property of distributions over $[N]$, and ORACLE$_D$ be some type of oracle which provides access to $D$. A $q(\varepsilon, N)$-query ORACLE testing algorithm for $\mathcal{P}$ is an algorithm $T$ which, given $\varepsilon, N$ as input parameters and oracle access to an ORACLE$_D$ oracle, and for any distribution $D$ over $[N]$, makes at most $q(\varepsilon, N)$ calls to ORACLE$_D$, and:

- if $D \in \mathcal{P}$ then, w.p. at least $2/3$, $T$ outputs ACCEPT;
- if $d_{TV}(D, \mathcal{P}) \geq \varepsilon$ then, w.p. at least $2/3$, $T$ outputs REJECT.
A few remarks

- tester is randomized;
A few remarks

- tester is randomized;
- “gray” area for $d_{TV}(D, \mathcal{P}) \in (0, \varepsilon);$
A few remarks

- tester is randomized;
- “gray” area for $d_{TV}(D, P) \in (0, \varepsilon)$;
- $2/3$ is completely arbitrary;
A few remarks

- tester is randomized;
- “gray” area for $d_{TV}(D, \mathcal{P}) \in (0, \varepsilon)$;
- $2/3$ is completely arbitrary;
- extends to several oracles and distributions;
A few remarks

- tester is randomized;
- “gray” area for $d_{TV}(D, \mathcal{P}) \in (0, \varepsilon)$;
- 2/3 is completely arbitrary;
- extends to several oracles and distributions;
- our measure is the sample complexity \textit{(not} the running time).
Distribution testing (3)
Concrete example: testing uniformity

Property $\mathcal{P}$ (“being $\mathcal{U}$, the uniform distribution over $[N]$”) $\Leftrightarrow$ set $S_{\mathcal{P}}$ of distributions with this property ($S_{\mathcal{P}} = \{\mathcal{U}\}$)
Distance to $\mathcal{P}$:
$$d_{TV}(D, S_{\mathcal{P}}) = \min_{D' \in S_{\mathcal{P}}} d_{TV}(D, D') = d_{TV}(D, \mathcal{U})$$

General outline

1. Draw a bunch of samples from $D$;
2. “Process” them, for instance by counting the number of points drawn more than once ($collisions$);
3. Compare the result to what one would expect from the uniform distribution $\mathcal{U}$;
4. Reject if it differs too much; accept otherwise.
Background and motivation
Well, it’s more or less settled.

Fact

In the standard sampling model, most (natural) properties are “hard” to test; that is, require a strong dependence on $N$ (at least $\Omega(\sqrt{N})$).
Background and motivation
Well, it’s more or less settled.

**Fact**

In the standard sampling model, most (natural) properties are “hard” to test; that is, require a strong dependence on $N$ (at least $\Omega(\sqrt{N})$).

**Example**

Testing *uniformity* has $\Theta(\sqrt{N}/\varepsilon^2)$ sample complexity [GR00, BFR+10, Pan08], *equivalence to a known distribution* $\tilde{\Theta}(\sqrt{N}/\varepsilon^2)$ [BFF+01, Pan08]; *equivalence of two unknown distributions* $\Omega(N^{2/3})$ [BFR+10, Val11] (and essentially matching upperbound)...
Our model

More power to the tester

In a lot of natural applications, the tester has more control over the “experiment” it is running – e.g., by tuning the conditions or the settings to influence the outcome, effectively restricting its range. *This is not captured by the SAMP model;* to mend this, we consider a new model where the testing algorithm can ask for a specific range of outcomes, and get a draw conditioned on it being in that domain.
Our model

More power to the tester

In a lot of natural applications, the tester has more control over the “experiment” it is running – e.g., by tuning the conditions or the settings to influence the outcome, effectively restricting its range. *This is not captured by the SAMP model*; to mend this, we consider a new model where the testing algorithm can ask for a specific range of outcomes, and get a draw conditioned on it being in that domain.

Definition (COND oracle)

Fix a distribution $D$ over $[N]$. A **COND oracle for $D$**, denoted $\text{COND}_D$, is defined as follows: The oracle is given as input a query set $S \subseteq [N]$ that has $D(S) > 0$, and returns an element $i \in S$, where the probability that element $i$ is returned is $D_S(i) = D(i)/D(S)$, independently of all previous calls to the oracle.
Remark

- generalizes the SAMP oracle \((S = [N])\), but allows adaptiveness;
Our model

Remark

- generalizes the SAMP oracle ($S = [N]$), but allows adaptiveness;
- variants of the (general) COND oracle, which only allow some specific types of subsets to be queried: PCOND (either $[N]$ or sets $\{i,j\}$) and ICOND (only intervals);
Our model

Remark

- generalizes the SAMP oracle \((S = [N])\), but allows adaptiveness;
- variants of the (general) COND oracle, which only allow some specific types of subsets to be queried: PCOND (either \([N]\) or sets \(\{i,j\}\)) and ICOND (only intervals);
- not defined for sets \(S\) with zero probability under \(D\);
Our model

Remark

- generalizes the SAMP oracle ($S = [N]$), but allows adaptiveness;
- variants of the (general) COND oracle, which only allow some specific types of subsets to be queried: PCOND (either $[N]$ or sets $\{i, j\}$) and ICOND (only intervals);
- not defined for sets $S$ with zero probability under $D$;
- similar model independently introduced by Chakraborty et al. [CFGM13].
Our model

Remark

- generalizes the SAMP oracle ($S = [N]$), but allows adaptiveness;
- variants of the (general) COND oracle, which only allow some specific types of subsets to be queried: PCOND (either $[N]$ or sets $\{i,j\}$) and ICOND (only intervals);
- not defined for sets $S$ with zero probability under $D$;
- similar model independently introduced by Chakraborty et al. [CFGM13].

Question

Do COND oracles enable more efficient testing algorithms than SAMP oracles? And what does it reveal about testing distributions?
Our results

Question

Do COND oracles enable more efficient testing algorithms than SAMP oracles?
Our results

Question

Do COND oracles enable more efficient testing algorithms than SAMP oracles? Yes, they do.
### Our results

Comparison of the COND and SAMP models on several testing problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Our results</th>
<th>Standard model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is $D$ uniform?</td>
<td>COND$_D$ $\Omega\left(\frac{1}{\varepsilon^2}\right)$</td>
<td>$\Theta\left(\frac{\sqrt{N}}{\varepsilon^2}\right)$ [GR00, BFR$^+$10, Pan08]</td>
</tr>
<tr>
<td></td>
<td>PCOND$_D$ $\tilde{O}\left(\frac{1}{\varepsilon^2}\right)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ICOND$_D$ $\tilde{O}\left(\frac{\log^3 N}{\varepsilon^3}\right)$ $\Omega\left(\frac{\log N}{\log \log N}\right)$</td>
<td></td>
</tr>
<tr>
<td>Is $D = D^*$?</td>
<td>COND$_D$ $\tilde{O}\left(\frac{1}{\varepsilon^4}\right)$</td>
<td>$\tilde{O}\left(\frac{\sqrt{N}}{\varepsilon^2}\right)$ [BFF$^+$01, Pan08]</td>
</tr>
<tr>
<td></td>
<td>PCOND$_D$ $\Omega\left(\frac{\log N}{\log \log N}\right)$</td>
<td></td>
</tr>
<tr>
<td>Are $D_1, D_2$ equivalent?</td>
<td>COND$_{D_1, D_2}$ $\tilde{O}\left(\frac{\log^5 N}{\varepsilon^4}\right)$</td>
<td>$\tilde{O}\left(\frac{N^{2/3}}{\varepsilon^{8/3}}\right)$ [BFR$^+$10]</td>
</tr>
<tr>
<td></td>
<td>PCOND$_{D_1, D_2}$ $\tilde{O}\left(\frac{\log^6 N}{\varepsilon^{21}}\right)$</td>
<td>$\Omega\left(N^{2/3}\right)$ [BFR$^+$10, Val11]</td>
</tr>
<tr>
<td>How far is $D$ from $\mathcal{U}$?</td>
<td>PCOND$_D$ $\tilde{O}\left(\frac{1}{\varepsilon^{20}}\right)$</td>
<td>$O\left(\frac{1}{\varepsilon^2} \frac{N}{\log N}\right)$ [VV11, VV10b] $\Omega\left(\frac{N}{\log N}\right)$ [VV11, VV10a]</td>
</tr>
</tbody>
</table>

Table: The upper bounds for the first 3 problems are for testing the property, while the last one involves estimating the total variation distance to uniformity to within an additive $\pm \varepsilon$. 

Clément Canonne (Columbia University)  
Testing distributions with a COND oracle  
March 8th, 2013  
11 / 35
Plan for rest of talk:

- testing uniformity: an upper bound (with pairwise queries)
- testing uniformity: a lower bound
- introducing tools: \texttt{Estimate-Neighborhood} and \texttt{Approx-Eval}
- testing uniformity, again: a (glimpse at) interval queries.
Theorem (Testing Uniformity with PCOND)

There exists a $\tilde{O}(1/\varepsilon^2)$-query $\text{PCOND}_D$ tester for uniformity, i.e. it accepts w.p. at least $2/3$ if $D = \mathcal{U}$ and rejects w.p. at least $2/3$ if $d_{\text{TV}}(D, \mathcal{U}) \geq \varepsilon$. 

High-level idea

Intuitively, if $D$ is $\varepsilon$-far from uniform, it must have (a) a lot of points "very light"; and (b) a lot of weight on points "very heavy". Sampling $\tilde{O}(1/\varepsilon^4)$ points both uniformly and according to $D$, we obtain whp both light and heavy ones; and use PCOND to compare them.

Not good enough ($O(1/\varepsilon^4)$ queries) $\Rightarrow$ refine this approach to get $\tilde{O}(1/\varepsilon^2)$. 

Why bother with $N$?
Testing Uniformity (1)
Why bother with $N$?

Theorem (Testing Uniformity with PCOND)

There exists a $\tilde{O}(1/\varepsilon^2)$-query PCOND$_D$ tester for uniformity, i.e. it accepts w.p. at least $2/3$ if $D = \mathcal{U}$ and rejects w.p. at least $2/3$ if $d_{TV}(D, \mathcal{U}) \geq \varepsilon$.

High-level idea

Intuitively, if $D$ is $\varepsilon$-far from uniform, it must have (a) a lot of points “very light”; and (b) a lot of weight on points “very heavy”. Sampling $O(1/\varepsilon)$ points both uniformly and according to $D$, we obtain whp both light and heavy ones; and use PCOND to compare them.
Theorem (Testing Uniformity with PCOND)

There exists a $\tilde{O}(1/\varepsilon^2)$-query $\text{PCOND}_D$ tester for uniformity, i.e. it accepts w.p. at least $2/3$ if $D = \mathcal{U}$ and rejects w.p. at least $2/3$ if $d_{TV}(D, \mathcal{U}) \geq \varepsilon$.

High-level idea

Intuitively, if $D$ is $\varepsilon$-far from uniform, it must have (a) a lot of points “very light”; and (b) a lot of weight on points “very heavy”. Sampling $O(1/\varepsilon)$ points both uniformly and according to $D$, we obtain whp both light and heavy ones; and use PCOND to compare them.

Not good enough ($O(1/\varepsilon^4)$ queries) $\leadsto$ refine this approach to get $\tilde{O}(1/\varepsilon^2)$. 
Testing Uniformity (2)
Getting our hands dirty.

Algorithm 1: $\text{PCOND}_D$-Test-Uniform

1: Set $t = \log \left( \frac{4}{\epsilon} \right) + 1.$
2: Select $q = \Theta(1)$ points $i_1, \ldots, i_q$ uniformly \{Reference points\}
3: for $j = 1$ to $t$ do
4: Call the $\text{SAMP}_D$ oracle $s_j = \Theta(2^j t)$ times to obtain points $h_1, \ldots, h_{s_j}$ distributed according to $D$ \{Try to get a heavy point\}
5: Draw $s_j$ points $\ell_1, \ldots, \ell_{s_j}$ uniformly from $[N]$ \{Try to get a light point\}
6: for all pairs $(x, y) = (i_r, h_{r'})$ and $(x, y) = (i_r, \ell_{r'})$ do
7: Call $\text{COMPARE}_D(\{x\}, \{y\}, \Theta(\epsilon 2^j), 2, \exp^{-\Theta(t)}).$
8: if it does not return a value in $[1 - 2^{j-5} \frac{\epsilon}{4}, 1 + 2^{j-5} \frac{\epsilon}{4}]$ then
9: output REJECT (and exit).
10: end if
11: end for
12: end for
13: Output ACCEPT
Proof (Outline).

Sample complexity by the setting of $t$, $q$ and the calls to COMPARE

Completeness unless COMPARE fails to output a correct value, no rejection

Soundness Suppose $D$ is $\varepsilon$-far from $U$; refinement of the previous approach by bucketing low and high points:

$$H_j \overset{\text{def}}{=} \left\{ h \mid \left(1 + 2^{j-1} \frac{\varepsilon}{4}\right) \frac{1}{N} \leq D(h) < \left(1 + 2^j \frac{\varepsilon}{4}\right) \frac{1}{N} \right\}$$

$$L_j \overset{\text{def}}{=} \left\{ \ell \mid \left(1 - 2^{j} \frac{\varepsilon}{4}\right) \frac{1}{N} < D(\ell) \leq \left(1 - 2^{j-1} \frac{\varepsilon}{4}\right) \frac{1}{N} \right\}$$

for $j \in [t-1]$, with also $H_0, L_0, H_t, L_t$ to cover everything; each loop iteration on l.3 “focuses” on a particular bucket.

+ Chernoff and union bounds.
Theorem (Testing Uniformity with COND)

Any \( \text{COND}_D \) algorithm for testing whether \( D = \mathcal{U} \) versus \( d_{\text{TV}}(D, \mathcal{U}) \geq \varepsilon \) must make \( \Omega(1/\varepsilon^2) \) queries.

Theorem (Testing Uniformity with COND)

Any $\text{COND}_D$ algorithm for testing whether $D = \mathcal{U}$ versus $d_{\text{TV}}(D, \mathcal{U}) \geq \varepsilon$ must make $\Omega(1/\varepsilon^2)$ queries.

Remark

As PCOND is a restriction of COND, the previous upper bound was essentially optimal.
High-level idea.

Reduce it to the problem of **distinguishing between a fair and a biased coin**, by defining a “no-instance” $D_{\text{no}}$ s.t.

1. $D_{\text{no}}$ is $\varepsilon$-far from $\mathcal{U}$;
2. any $q$-query tester $A$ which distinguishes $D_{\text{no}}$ from $\mathcal{U}$ can be turned into a tester $A'$ distinguishing between (1) a sequence of $q$ fair coin tosses and (2) a sequence of $q$ $(4\varepsilon)$-biased coin tosses.

However, it is known that distinguishing between these two scenarios requires $\Omega(1/\varepsilon^2)$ coin tosses.
Figure: The no-instance $D_{\text{no}}$. 
Testing Uniformity – Lower Bound (3)

The reduction: how to simulate $\text{COND}_D$ from coin tosses

To run $\mathcal{A}$ from $\mathcal{A}'$, we must simulate $\text{COND}_D$ ($D$ either $\mathcal{U}$ or $D_{\text{no}}$) to provide the former with samples, given the corresponding coin tosses.

At step $1 \leq t \leq q$, $\mathcal{A}$ chooses to query $S \subset [N]$ (according to the $(t - 1)$ previous answers it got from the simulation). $\mathcal{A}'$ behaves as follows:

- sets $S_0 \overset{\text{def}}{=} S \cap [1, \frac{N}{2}]$, $S_1 \overset{\text{def}}{=} S \cap [\frac{N}{2} + 1, N]$;
- gets bit $b_t$, and draws $\sigma \sim \begin{cases} \text{Bern}(u_t) & \text{if } b_t = 1 \\ \text{Bern}(v_t) & \text{o.w.} \end{cases}$
- draws $s$ u.a.r. from $S_\sigma$;
- gives $(S, s)$ to $\mathcal{A}$.

\((\dagger)\) for a right choice of $u_t$, $v_t$ depending on $|S_0|$, $|S_1|$, $\varepsilon$
Testing Uniformity – Summary

- $\Theta\left(\sqrt{N}/\varepsilon^2\right)$ with SAMP: counting collisions [GR00, BFR$^+10$, Pan08]
- $\tilde{O}(1/\varepsilon^2)$ with PCOND: comparing random pairs of points
- $\Omega(1/\varepsilon^2)$ with COND: reducing to fair vs. biased coin
Testing Uniformity – Summary

- $\Theta\left(\sqrt{N}/\varepsilon^2\right)$ with SAMP: counting collisions [GR00, BFR+10, Pan08]
- $\tilde{O}(1/\varepsilon^2)$ with PCOND: comparing random pairs of points
- $\Omega(1/\varepsilon^2)$ with COND: reducing to fair vs. biased coin

Remark

Testing with ICOND will require a logarithmic dependence on $N$. 
Building tools (1)

- **Compare**
  Low-level procedure: compares the relative weight of sets $X$, $Y$, given some accuracy parameter $\eta$.

- **Estimate-Neighborhood**
  On input a point $i \in [N]$ and parameter $\gamma$, estimates the weight under $D$ of the $\gamma$-neighborhood of $i$ – that is, points with probability mass within a factor $(1 + \gamma)$ of $D(i)$.

- **Approx-Eval**
  Given $i \in [N]$ and accuracy parameter $\eta$, returns an approximation of $D(i)$ – succeeds whp for most points $i$. 
Building tools (2)

“Comparison is the death of joy.” – Mark Twain.

The low-level tool \texttt{COMPARE}

Given as input two disjoint subsets \(X, Y\), parameters \(\eta \in (0, 1]\), \(K \geq 1\), and 
\(\delta \in (0, 1/2]\), and \texttt{COND} access to \(D\), the procedure \texttt{COMPARE} either 
outputs a value \(\rho > 0\), High or Low, s.t:

- If \(D(X)/K \leq D(Y) \leq K \cdot D(X)\) then w.p. \(1 - \delta\) it outputs a value 
  \(\rho \in [1 - \eta, 1 + \eta]D(Y)/D(X)\);

- If \(D(Y) > K \cdot D(X)\) then w.p. \(1 - \delta\) it outputs either High or a value 
  \(\rho \in [1 - \eta, 1 + \eta]D(Y)/D(X)\);

- If \(D(Y) < D(X)/K\) then w.p. \(1 - \delta\) it outputs either Low or a value 
  \(\rho \in [1 - \eta, 1 + \eta]D(Y)/D(X)\).

\texttt{COMPARE} performs \(O\left(\frac{K \log(1/\delta)}{\eta^2}\right)\) \texttt{COND} queries on \(X \cup Y\).
Building tools (3)

\[ \rho D(X) \ll \rho D(Y) \]

\[ \rho D(X) \approx \rho D(Y) \]

\[ \rho D(X) \gg \rho D(Y) \]
Building tools (4)

**Definition (γ-Neighborhood)**

\[ U_\gamma(x) \overset{\text{def}}{=} \left\{ y \in [N] : \frac{1}{1 + \gamma} D(x) \leq D(y) \leq (1 + \gamma) D(x) \right\}, \quad \gamma \in [0, 1] \]
Building tools (4)

**Definition ($\gamma$-Neighborhood)**

\[ U_\gamma(x) \overset{\text{def}}{=} \left\{ y \in [N] : \frac{1}{1+\gamma} D(x) \leq D(y) \leq (1+\gamma) D(x) \right\}, \quad \gamma \in [0,1] \]

**Goal**

Given a point \( x \in [N] \) and a parameter \( \gamma \), get an approximation of \( D(U_\gamma(x)) \) – i.e., “how much weight does \( D \) put on points like \( x \)?"
The (slightly) higher-level subroutine **Estimate-Neighborhood**

Given as input a point $x$, parameters $\gamma, \beta, \eta, \delta \in (0, 1/2]$ and PCOND$_D$ access, the procedure **Estimate-Neighborhood** outputs a pair $(\hat{w}, \alpha) \in [0, 1] \times (\gamma, 2\gamma)$ such that, for $\theta$ small:

1. If $D(U_\alpha(x)) \geq \beta$, then w.p. $1 - \delta$ we have
   \[ \hat{w} \in [1 - \eta, 1 + \eta] \cdot D(U_\alpha(x)), \text{ and } D(U_{\alpha + \theta}(x) \setminus U_\alpha(x)) \leq \eta \beta / 16; \]

2. If $D(U_\alpha(x)) < \beta$, then w.p. $1 - \delta$ we have $\hat{w} \leq (1 + \eta) \cdot \beta$, and
   \[ D(U_{\alpha + \theta}(x) \setminus U_\alpha(x)) \leq \eta \beta / 16. \]

**Estimate-Neighborhood** performs $\tilde{O}\left(\frac{\log(1/\delta)}{\gamma^2 \eta^4 \beta^3 \delta^2}\right)$ queries.
The (slightly) higher-level subroutine $\textsc{Estimate-Neighborhood}$

Given as input a point $x$, parameters $\gamma, \beta, \eta, \delta \in (0, 1/2]$ and $\text{PCOND}_D$ access, the procedure $\textsc{Estimate-Neighborhood}$ outputs a pair $(\hat{w}, \alpha) \in [0, 1] \times (\gamma, 2\gamma)$ such that, for $\theta$ small:

1. If $D(U_\alpha(x)) \geq \beta$, then w.p. $1 - \delta$ we have $\hat{w} \in [1 - \eta, 1 + \eta] \cdot D(U_\alpha(x))$, and $D(U_{\alpha + \theta}(x) \setminus U_\alpha(x)) \leq \eta\beta/16$;

2. If $D(U_\alpha(x)) < \beta$, then w.p. $1 - \delta$ we have $\hat{w} \leq (1 + \eta) \cdot \beta$, and $D(U_{\alpha + \theta}(x) \setminus U_\alpha(x)) \leq \eta\beta/16$.

$\textsc{Estimate-Neighborhood}$ performs $\tilde{O}\left( \frac{\log(1/\delta)}{\gamma^2 \eta^4 \beta^3 \delta^2} \right)$ queries.

Remark

Does not estimate exactly $D(U_\gamma(x))$. 
A $\delta$-EVAL$_D$ simulator for $D$ is a randomized procedure ORACLE such that w.p. $1 - \delta$ the output of ORACLE on input $i^* \in [N]$ is $D(i^*)$. 
Building tools (6)

(Approximate) EVAL oracle

An \((\varepsilon, \delta)\)-approximate \(\text{EVAL}_D\) simulator for \(D\) is a randomized procedure \text{ORACLE} such that w.p. \(1 - \delta\) the output of \text{ORACLE} on input \(i^* \in [N]\) is a value \(\alpha \in [0, 1]\) such that \(\alpha \in [1 - \varepsilon, 1 + \varepsilon]D(i^*)\).
(Approximate) EVAL oracle

An \((\varepsilon, \delta)\)-approximate \(\text{EVAL}_D\) simulator for \(D\) is a randomized procedure \(\text{ORACLE}\) s.t for each \(\varepsilon\), there is a fixed set \(S^{(\varepsilon)} \subset \mathbb{N}\) with \(D(S^{(\varepsilon)}) < \varepsilon\) for which the following holds. For all \(i^* \in \mathbb{N}\), \(\text{ORACLE}(i^*)\) is either a value \(\alpha \in [0, 1]\) or Unknown, and furthermore:

(i) If \(i^* \notin S^{(\varepsilon)}\) then w.p. \(1 - \delta\) the output of \(\text{ORACLE}\) on input \(i^*\) is a value \(\alpha \in [0, 1]\) such that \(\alpha \in [1 - \varepsilon, 1 + \varepsilon]D(i^*)\);

(ii) If \(i^* \in S^{(\varepsilon)}\) then w.p. \(1 - \delta\) the procedure either outputs Unknown or outputs a value \(\alpha \in [0, 1]\) such that \(\alpha \in [1 - \varepsilon, 1 + \varepsilon]D(i^*)\).
(Approximate) EVAL oracle

An \((\varepsilon, \delta)\)-approximate \(\text{EVAL}_D\) simulator for \(D\) is a randomized procedure \(\text{ORACLE}\) s.t for each \(\varepsilon\), there is a fixed set \(S^{(\varepsilon)} \subseteq [N]\) with \(D(S^{(\varepsilon)}) < \varepsilon\) for which the following holds. For all \(i^* \in [N]\), \(\text{ORACLE}(i^*)\) is either a value \(\alpha \in [0, 1]\) or Unknown, and furthermore:

(i) If \(i^* \notin S^{(\varepsilon)}\) then w.p. \(1 - \delta\) the output of \(\text{ORACLE}\) on input \(i^*\) is a value \(\alpha \in [0, 1]\) such that \(\alpha \in [1 - \varepsilon, 1 + \varepsilon]D(i^*)\);

(i) If \(i^* \in S^{(\varepsilon)}\) then w.p. \(1 - \delta\) the procedure either outputs Unknown or outputs a value \(\alpha \in [0, 1]\) such that \(\alpha \in [1 - \varepsilon, 1 + \varepsilon]D(i^*)\).

The high-level blackbox \texttt{APPROX-EVAL}

There is an algorithm \texttt{APPROX-EVAL} which uses \(\tilde{O}\left(\frac{(\log N)^5 \cdot (\log(1/\delta))^2}{\varepsilon^3}\right)\) calls to \(\text{COND}_D\), and is an \((\varepsilon, \delta)\)-approximate \(\text{EVAL}_D\) simulator.
Approx-Eval \( \varepsilon \)

\( \hat{D}(i) \)

"Unknown"

or \( \hat{D}(i) \)

\( i^* \in [N] \)

\( \text{COND}_D \)

\( i^* \in S(\varepsilon) \)

\( i^* \notin S(\varepsilon) \)
Applications

Testing equivalence of two unknown distributions $D_1$, $D_2$

Blackbox access to $D_1$ and $D_2$ (two oracles); distinguish $D_1 = D_2$ vs. $d_{TV}(D_1, D_2) \geq \varepsilon$.

---

3 (extension of the original results)
Applications

Testing equivalence of two unknown distributions $D_1$, $D_2$

Blackbox access to $D_1$ and $D_2$ (two oracles); distinguish $D_1 = D_2$ vs. $d_{TV}(D_1, D_2) \geq \varepsilon$.

In the language of property testing: $S_{\mathcal{P}} = \{ (D, D) \mid D \text{ distribution } \}$, with metric over pairs of distributions $d(((D, D'), (P, P'))) \overset{\text{def}}{=} d_{TV}(D, P) + d_{TV}(D', P')$.

\(^3\) (extension of the original results)

Clément Canonne
Columbia University
Testing distributions with a COND oracle
March 8\(^{th}\), 2013
Testing equivalence of two unknown distributions $D_1$, $D_2$

Blackbox access to $D_1$ and $D_2$ (two oracles); distinguish $D_1 = D_2$ vs. $d_{TV}(D_1, D_2) \geq \varepsilon$.

In the language of property testing: $S_P = \{(D, D) \mid D \text{ distribution}\}$, with metric over pairs of distributions $d(((D, D'), (P, P'))) \overset{\text{def}}{=} d_{TV}(D, P) + d_{TV}(D', P')$.

Two different approaches:

1. with PCOND and \textsc{Estimate-Neighborhood} – finding “representatives” points for both distributions;
2. with COND and \textsc{Approx-Eval} – adapting an EVAL algorithm from [RS09].

Other uses: estimating distance to uniformity (\textsc{Estimate-Neighborhood}), testing monotonicity\(^3\) (\textsc{Approx-Eval}). . .

\(^3\) (extension of the original results)
Main message

ICOND algorithms are weaker than PCOND ones for this: while \( \text{poly}(\log N, 1/\varepsilon) \) queries are enough, \( \tilde{\Omega}(\log N) \) are necessary.
Main message

ICOND algorithms are weaker than PCOND ones for this: while \(\text{poly}(\log N, 1/\varepsilon)\) queries are enough, \(\tilde{\Omega}(\log N)\) are necessary.

Overview

Upper bound: sort of binary descent on random points (custom-tailored version of \textsc{Approx-Eval}), to spot deviations from \(1/N\);

Lower bound: family of “no-instances” + LB against non-adaptive + hybrid argument to get LB against adaptive.
**Upper Bound**

![Diagram of binary descent]

**Figure:** Idea of the “binary descent” on $i$: get an estimate of $D(i)$ by multiplying estimates at each branching, each time rejecting if ratio between weight of two subintervals is far from $\frac{1}{2}$. Repeat for $\Theta(1/\varepsilon)$ points drawn from $D$. 
new model for studying probability distributions
arises naturally in a number of settings
allows significantly more query-efficient algorithms

generalizing to other structured domains? (e.g., the Boolean hypercube \( \{0, 1\}^n \))
what about distribution learning in this framework
more properties? (entropy, independence, monotonicity\(\dagger\) \ldots)
The end.

Thank you.


______, *Testing closeness of discrete distributions*, Tech. Report abs/1009.5397, 2010, This is a long version of [BFR+00].


______, *Estimating the unseen: an n/ log(n)-sample estimator for entropy and support size, shown optimal via new CLTs*, Proceedings of STOC, 2011, See also [VV10a] and [VV10b], pp. 685–694.