Testing equivalence between distributions using conditional samples

(when testers get to be picky)

Clément Canonne* Dana Ron† Rocco Servedio*

*Columbia University

†Tel-Aviv University

January 6, 2014

Plan of the talk

1. Introduction
2. Testing Uniformity and Identity
3. Tools and subroutines
4. Conclusion

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.[1]

[1] wrt to some specified metric, and parameter $\varepsilon > 0$ given to the tester.

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 $\mathcal{P}$, 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.[1]

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.

[1] wrt to some specified metric, and parameter $\varepsilon > 0$ given to the tester.

Distribution testing (1)

In more detail.

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

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

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 a (randomized) algorithm T which, given ε, 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 $d_{\mathrm{TV}}(D, \mathcal{P}) \geq \varepsilon$ then, w.p. at least $2/3$, T outputs REJECT.

Distribution testing (2)

Comments

A few remarks

- "gray" area for $d_{\mathrm{TV}}(D, \mathcal{P}) \in (0, \varepsilon)$;

Distribution testing (2)

Comments

A few remarks

- "gray" area for $d_{\mathrm{TV}}(D, \mathcal{P}) \in (0, \varepsilon)$;
- $2/3$ is completely arbitrary;

Distribution testing (2)

Comments

A few remarks

- "gray" area for $d_{\mathrm{TV}}(D, \mathcal{P}) \in (0, \varepsilon)$;
- $2/3$ is completely arbitrary;
- extends to several oracles and distributions;

Distribution testing (2)

Comments

A few remarks

- "gray" area for $d_{\mathrm{TV}}(D, \mathcal{P}) \in (0, \varepsilon)$;
- $2/3$ is completely arbitrary;
- extends to several oracles and distributions;
- our measure is the sample complexity (*not* the running time).

Distribution testing (3)

Concrete example: testing uniformity

Property $\mathcal{P}$ ("being $\mathcal{U}$, the uniform distribution over $[N]$") $\Leftrightarrow$ set $\mathcal{S}_\mathcal{P}$ of distributions with this property ($\mathcal{S}_\mathcal{P} = \{\mathcal{U}\}$)
Distance to $\mathcal{P}$:

$$\mathrm{d_{TV}}(D, \mathcal{S}_\mathcal{P}) = \min_{D' \in \mathcal{S}_\mathcal{P}} \mathrm{d_{TV}}(D, D') \underset{\text{here}}{=} \mathrm{d_{TV}}(D, \mathcal{U})$$

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: collision-based tester)*;
3. Output ACCEPT or REJECT based on the result.

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, CDVV13] (and essentially matching upperbound)...

Our model

More power to the tester

We consider a new model where the tester can specify a subset of the domain, and then get a draw conditioned on it landing in that subset. Models natural applications where a scientist/experimenter has some control over an 'experiment' to restrict the range of possible outcomes – e.g., by tuning the conditions or the setting: *this is not captured by the* SAMP *model*.

Our model

More power to the tester

We consider a new model where the tester can specify a subset of the domain, and then get a draw conditioned on it landing in that subset. Models natural applications where a scientist/experimenter has some control over an 'experiment' to restrict the range of possible outcomes – e.g., by tuning the conditions or the setting: *this is not captured by the* SAMP *model*.

Definition (COND oracle)

Fix a distribution D over $[N]$. A COND oracle for D, denoted 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.

Our model

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

Problem	Our results		Standard model
Is $D = D^*$ for a known D^*?	COND_D	$\tilde{O}\left(\frac{1}{\varepsilon^4}\right)$	$\tilde{\Theta}\left(\frac{\sqrt{N}}{\varepsilon^2}\right)$ [BFF+01, Pan08]
	PCOND_D	$\tilde{O}\left(\frac{\log^4 N}{\varepsilon^4}\right)$ $\Omega\left(\sqrt{\frac{\log N}{\log\log N}}\right)$	
Are D_1, D_2 (both unknown) equivalent?	COND_{D_1,D_2}	$\tilde{O}\left(\frac{\log^5 N}{\varepsilon^4}\right)$	$\Theta\left(\max\left(\frac{N^{2/3}}{\varepsilon^{4/3}}, \frac{\sqrt{N}}{\varepsilon^2}\right)\right)$ [BFR+10, Val11, CDVV13]
	PCOND_{D_1,D_2}	$\tilde{O}\left(\frac{\log^6 N}{\varepsilon^{21}}\right)$	

Table : Comparison between the COND model and the standard model for these problems. The upper bounds are for testing $d_{\mathrm{TV}} = 0$ vs. $d_{\mathrm{TV}} \geq \varepsilon$.

Rest of the talk

Plan for rest of talk:

- sketch of testing uniformity and testing D vs. D^* (with pairwise queries)
- introducing tools: ESTIMATE-NEIGHBORHOOD and APPROX-EVAL
- using them: testing equivalence of two unknown distributions

Testing Uniformity (1)

Special case of testing identity to D^*

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* $d_{\mathrm{TV}}(D,\mathcal{U}) \geq \varepsilon$.

Testing Uniformity (1)

Special case of testing identity to D^*

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 $d_{\mathrm{TV}}(D,\mathcal{U}) \geq \varepsilon$.*

High-level idea

Intuitively, if D is ε-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.

Testing Uniformity (1)

Special case of testing identity to D^*

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* $d_{\mathrm{TV}}(D, \mathcal{U}) \geq \varepsilon$.

High-level idea

Intuitively, if D is ε-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) $\rightsquigarrow$ refine this approach to get $\tilde{O}(1/\varepsilon^2)$.

Testing Uniformity (2) – generalizing to D^*

From uniform to arbitrary distribution: poly$(1/\varepsilon)$-query algorithm

Approach does not work for general D^*...

The ratios can be arbitrarily big or small: e.g., if $D^*(x)/D^*(y) = \sqrt{N}$, need $\Omega(\sqrt{N})$ calls to $\mathrm{PCOND}_D(\{x, y\})$ to distinguish $D(x)/D(y) = \sqrt{N}$ from $D(x)/D(y) = 2\sqrt{N}$

Testing Uniformity (2) – generalizing to D^*

From uniform to arbitrary distribution: poly($1/\varepsilon$)-query algorithm

Approach does not work for general D^*...

The ratios can be arbitrarily big or small: e.g., if $D^*(x)/D^*(y) = \sqrt{N}$, need $\Omega(\sqrt{N})$ calls to $\mathrm{PCOND}_D(\{x, y\})$ to distinguish $D(x)/D(y) = \sqrt{N}$ from $D(x)/D(y) = 2\sqrt{N}$

...but it can be adapted.

Idea: compare points with carefully chosen *comparable sets* $\rightsquigarrow D(x)/D(Y)$ instead of $D(x)/D(y)$

Testing Uniformity (2) – generalizing to D^*

From uniform to arbitrary distribution: poly$(1/\varepsilon)$-query algorithm

Approach does not work for general D^*...

The ratios can be arbitrarily big or small: e.g., if $D^*(x)/D^*(y) = \sqrt{N}$, need $\Omega(\sqrt{N})$ calls to $\mathrm{PCOND}_D(\{x,y\})$ to distinguish $D(x)/D(y) = \sqrt{N}$ from $D(x)/D(y) = 2\sqrt{N}$

...but it can be adapted.

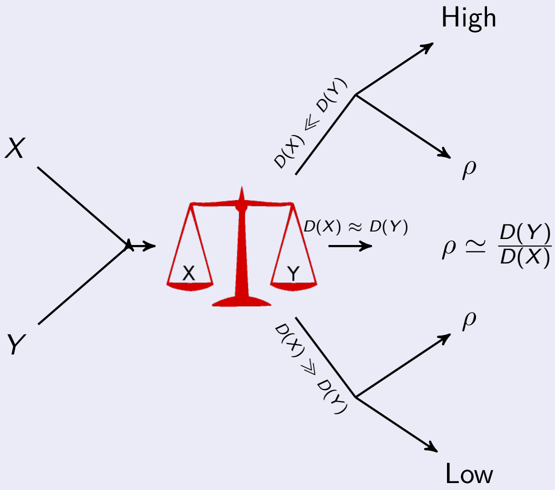
Idea: compare points with carefully chosen *comparable sets* $\rightsquigarrow D(x)/D(Y)$ instead of $D(x)/D(y)$

However, cannot do this with PCOND (Lower bound: $\log^{\Omega(1)} N$ samples)): a COND oracle is needed.

Building tools (1)

- COMPARE
  Low-level procedure: compares the relative weight of disjoint sets X, Y, given some accuracy parameter η.
- ESTIMATE-NEIGHBORHOOD
  On input a point $i \in [N]$ and parameter γ, estimates the weight under D of the γ-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 η, returns an approximation of $D(i)$ – succeeds whp for most points i.

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



Building tools (3)

Second tool: ESTIMATE-NEIGHBORHOOD procedure

Definition (γ-Neighborhood)

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

Building tools (3)

Second tool: Estimate-Neighborhood procedure

Definition (γ-Neighborhood)

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

Goal

Given a point $x \in [N]$ and a parameter γ, Estimate-Neighborhood gives a multiplicative approximation of $D(U_\gamma(x))$ – i.e., "how much weight does D put on points like x?"

Building tools (4)

Third tool: APPROXIMATE-EVAL oracle

EVAL oracle

A δ-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 (4)

Third tool: APPROXIMATE-EVAL oracle

(Approximate) EVAL oracle

Ideally, an (ε, δ)-approximate EVAL_D simulator for D *would be* a randomized procedure ORACLE such that w.p. $1-\delta$ the output of 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

Actually, an (ε, δ)-approximate EVAL_D simulator for D *is* a randomized procedure ORACLE s.t for each ε, there is a fixed set $S^{(\varepsilon)} \subsetneq [N]$ with $D(S^{(\varepsilon)}) < \varepsilon$ for which the following holds. For all $i^* \in [N]$, 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 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 $\alpha \in [1 - \varepsilon, 1 + \varepsilon]D(i^*)$.

Building tools (4)

Third tool: APPROXIMATE-EVAL oracle

(Approximate) EVAL oracle

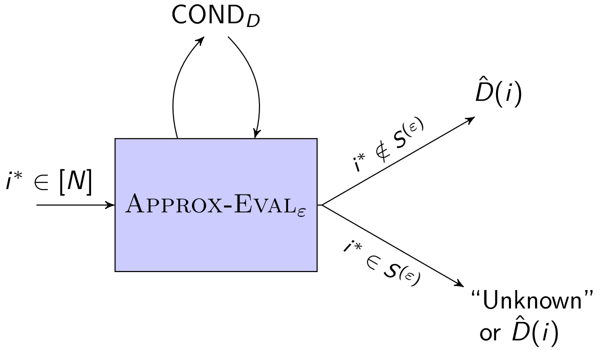
Actually, an (ε, δ)-approximate EVAL_D simulator for D *is* a randomized procedure ORACLE s.t for each ε, there is a fixed set $S^{(\varepsilon)} \subsetneq [N]$ with $D(S^{(\varepsilon)}) < \varepsilon$ for which the following holds. For all $i^* \in [N]$, ORACLE(i^*) is either a value $\alpha \in [0, 1]$ or Unknown, and furthermore:

(i) If $i^* \notin S^{(\varepsilon)}$ then w.p. $1 - \delta$ the output of 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 APPROX-EVAL

There is an algorithm APPROX-EVAL which uses $\tilde{O}\left(\frac{(\log N)^5 \cdot (\log(1/\delta))^2}{\varepsilon^3}\right)$ calls to COND_D, and is an (ε, δ)-approximate EVAL_D simulator.



COND_D
$\hat{D}(i)$
$i^* \notin S^{(\varepsilon)}$
$i^* \in [N]$
$\textsc{Approx-Eval}_\varepsilon$
$i^* \in S^{(\varepsilon)}$
"Unknown"
or $\hat{D}(i)$

Building tools (5)

Third tool: APPROXIMATE-EVAL oracle

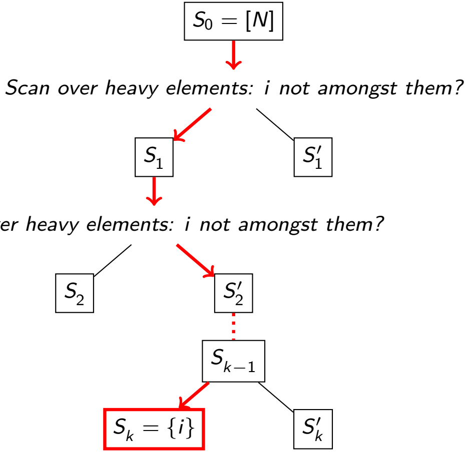


Figure : Execution of APPROX-EVAL on some i: scan over heavy elements, randomly partition the light ones, recurse; finally get an estimate of $D(i)$ by multiplying estimates at each branching.

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. $d_{TV}(D_1, D_2) \geq \varepsilon$.

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. $d_{TV}(D_1, D_2) \geq \varepsilon$.

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. $\mathrm{d_{TV}}(D_1, D_2) \geq \varepsilon$.

Two different approaches:

1. with PCOND and ESTIMATE-NEIGHBORHOOD – finding "representatives" points for both distributions;
2. with COND and APPROX-EVAL – adapting an EVAL algorithm from [RS09].

Other uses: estimating distance to uniformity (ESTIMATE-NEIGHBORHOOD), testing monotonicity[2] (APPROX-EVAL)...

[2](extension of the original results)

Idea: get a *succinct representation*

- Get a "cover for D_1" in $\tilde{O}(\log N/\varepsilon^2)$ *representatives* $r_1, \ldots, r_\ell$;

Applications

Testing $D_1 \equiv D_2$ with PCOND and ESTIMATE-NEIGHBORHOOD

Idea: get a *succinct representation*

- Get a "cover for D_1" in $\tilde{O}(\log N/\varepsilon^2)$ *representatives* $r_1, \ldots, r_\ell$;
- If $D_1 = D_2$, cover perfect for D_2;

Applications

Testing $D_1 \equiv D_2$ with PCOND and ESTIMATE-NEIGHBORHOOD

Idea: get a *succinct representation*

- Get a "cover for D_1" in $\tilde{O}(\log N/\varepsilon^2)$ *representatives* $r_1, \dots, r_\ell$;
- If $D_1 = D_2$, cover perfect for D_2; but
- If $d_{\mathrm{TV}}(D_1, D_2) \geq \varepsilon$, then for one of the representatives r^* (covering a set of points R^* under D_1), either
  1. "many" $y \in R^*$ are *not* covered by r^* *under* D_2 (mismatching representative); or
  2. $D_2(R^*)$ differs significantly from $D_1(R^*)$ (mismatching neighborhoods)

45

Idea: get a *succinct representation*

- Get a "cover for D_1" in $\tilde{O}(\log N/\varepsilon^2)$ *representatives* $r_1, \dots, r_\ell$;
- If $D_1 = D_2$, cover perfect for D_2; but
- If $d_{\mathrm{TV}}(D_1, D_2) \geq \varepsilon$, then for one of the representatives r^* (covering a set of points R^* under D_1), either
  1. "many" $y \in R^*$ are *not* covered by r^* *under* D_2 (mismatching representative); or
  2. $D_2(R^*)$ differs significantly from $D_1(R^*)$ (mismatching neighborhoods)

Both can be detected efficiently; try it for each $r_i \rightsquigarrow \mathrm{poly}(\log N, 1/\varepsilon)$ sample and time complexity.

Conclusion

- 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†...)

The end.

Thank you.

An extended version of this work [CRS12] is available online (`arXiv:1211.2664`).

References I

T. Batu, E. Fischer, L. Fortnow, R. Kumar, R. Rubinfeld, and P. White, *Testing random variables for independence and identity*, Proceedings of FOCS, 2001, pp. 442–451.

T. Batu, L. Fortnow, R. Rubinfeld, W. D. Smith, and P. White, *Testing that distributions are close*, Proceedings of FOCS, 2000, pp. 189–197.

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

S.-O. Chan, I. Diakonikolas, G. Valiant, and P. Valiant, *Optimal Algorithms for Testing Closeness of Discrete Distributions*, ArXiv e-prints (2013).

S. Chakraborty, E. Fischer, Y. Goldhirsh, and A. Matsliah, *On the power of conditional samples in distribution testing*, Proceedings of ITCS, 2013, Arxiv posting http://arxiv.org/abs/1210.8338 31 Oct 2012.

C. Canonne, D. Ron, and R. Servedio, *Testing probability distributions using conditional samples*, Tech. Report http://arxiv.org/abs/1211.2664, 12 Nov 2012.

O. Goldreich and D. Ron, *On testing expansion in bounded-degree graphs*, Tech. Report TR00-020, ECCC, 2000.

L. Paninski, *A coincidence-based test for uniformity given very sparsely sampled discrete data*, IEEE-IT **54** (2008), no. 10, 4750–4755.

R. Rubinfeld and R. A. Servedio, *Testing monotone high-dimensional distributions*, RSA **34** (2009), no. 1, 24–44.

P. Valiant, *Testing symmetric properties of distributions*, SICOMP **40** (2011), no. 6, 1927–1968.

Backup slides

Testing Uniformity (3)

Getting our hands dirty.

Algorithm 1: PCOND_D-TEST-UNIFORM

Set $t = \Theta(\log(\frac{1}{\varepsilon}))$.
Select $q = \Theta(1)$ points $i_1, \dots, i_q$ uniformly {Reference points}
for $j = 1$ to t **do**
  Call the oracle $s_j = \Theta(2^j t)$ times to get $h_1, \dots, h_{s_j} \sim D$ {Heavy points?}
  Draw s_j points $\ell_1, \dots, \ell_{s_j}$ uniformly from $[N]$ {Light points?}
  for all pairs $(x, y) = (i_r, h_{r'})$ and $(x, y) = (i_r, \ell_{r'})$ **do**
    Get a good estimate of $D(x)/D(y)$. {Ideally, should be 1}
    Reject if the value is not in $[1 - 2^{j-5}\frac{\varepsilon}{4}, 1 + 2^{j-5}\frac{\varepsilon}{4}]$
  end for
end for
Accept

Testing Uniformity (4)

Proof (Outline).

Sample complexity by the setting of t, q and the calls to $\textsc{Compare}$

Completeness unless $\textsc{Compare}$ fails to output a correct value, no rejection

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

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

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

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. □

Building tools (6)

The (slightly) higher-level subroutine Estimate-Neighborhood

Given as input a point x, parameters $\gamma, \beta, \eta \in (0, 1/2]$ and PCOND_D access, the procedure Estimate-Neighborhood outputs a pair $(\hat{w}, \alpha) \in [0,1] \times (\gamma, 2\gamma)$ such that w.h.p

1. If $D(U_\alpha(x)) \geq \beta$, then $\hat{w} \in [1-\eta, 1+\eta] \cdot D(U_\alpha(x))$, and $(\ldots)$
2. If $D(U_\alpha(x)) < \beta$, then $\hat{w} \leq (1+\eta) \cdot \beta$, and $(\ldots)$

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

Building tools (6)

The (slightly) higher-level subroutine Estimate-Neighborhood

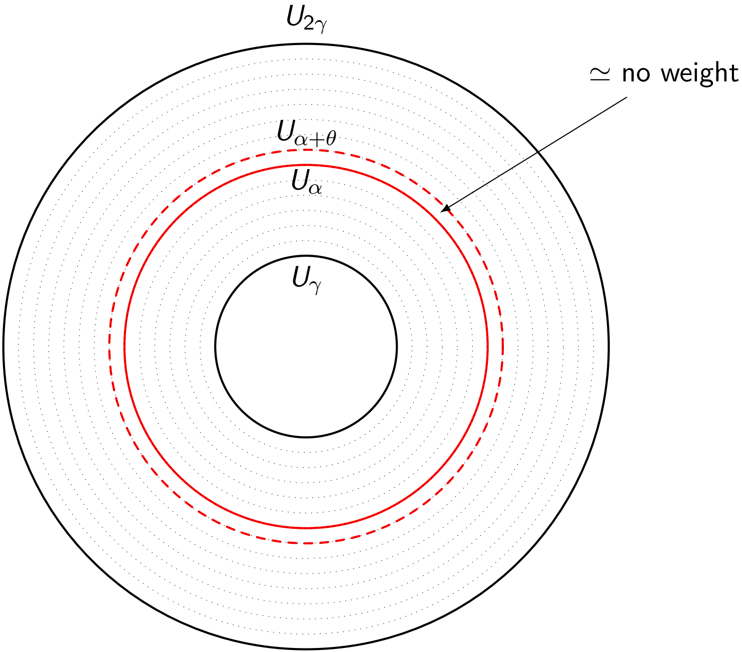
Given as input a point x, parameters $\gamma, \beta, \eta \in (0, 1/2]$ and PCOND_D access, the procedure Estimate-Neighborhood outputs a pair $(\hat{w}, \alpha) \in [0,1] \times (\gamma, 2\gamma)$ such that w.h.p

1. If $D(U_\alpha(x)) \geq \beta$, then $\hat{w} \in [1-\eta, 1+\eta] \cdot D(U_\alpha(x))$, and $(\ldots)$
2. If $D(U_\alpha(x)) < \beta$, then $\hat{w} \leq (1+\eta) \cdot \beta$, and $(\ldots)$

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

Remark

Does not estimate exactly $D(U_\gamma(x))$.


$U_{2\gamma}$
$\simeq$ no weight
$U_{\alpha+\theta}$
U_α
U_γ

Building tools (7)

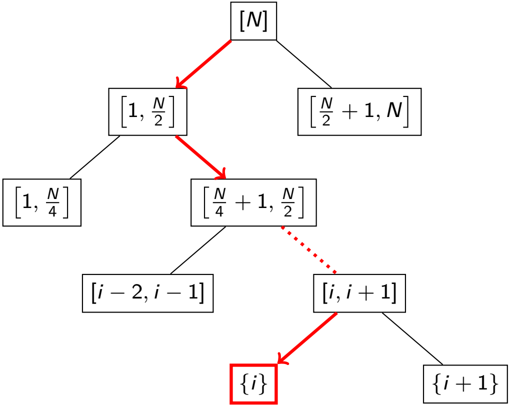


Figure : (Rough) idea of the "binary descent" on i for APPROX-EVAL: get an estimate of $D(i)$ by multiplying estimates at each branching.