Testing probability distributions using conditional samples

C. Canonne D. Ron R. Servedio

March 20, 2015

Plan of the talk

- The model
- Some results and tools in the model
- Application of the tools

Background and motivation

Property testing for probability distributions – the basic setup

- There is an unknown distribution D over $[N]$ that one can only access via (expensive) calls to some "oracle"
- There is a property $\mathcal{P}$ of interest that D may or may not have
- The goal is to distinguish, using a sublinear number of oracle calls, between the two cases:
  (a) D has property $\mathcal{P}$;
  (b) D is "ϵ-far" from all distributions that have property $\mathcal{P}$ (w.r.t. some chosen metric).

Background and motivation

Property testing for probability distributions – the basic setup

- There is an unknown distribution D over $[N]$ that one can only access via (expensive) calls to some "oracle"
- There is a property $\mathcal{P}$ of interest that D may or may not have
- The goal is to distinguish, using a sublinear number of oracle calls, between the two cases:
  (a) D has property $\mathcal{P}$;
  (b) D is "ϵ-far" from all distributions that have property $\mathcal{P}$ (w.r.t. some chosen metric).

Standard model of testing probability distributions:

The oracle is SAMP_D: at each call, it returns an i.i.d. draw from D.

Distribution testing (1)

In more detail.

Our metric: 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 $d_{\mathrm{TV}}(D, \mathcal{P}) \in (0, \varepsilon)$;
- $2/3$ is completely arbitrary;
- extends to several oracles and distributions;
- our measure is the # of oracle calls (*not* the running time).

Distribution testing (3)

A 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 $\mathcal{P}$:

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

Distribution testing in the standard model:

1. Draw a bunch of points 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.

The lay of the land in the standard model

Fact

In the standard SAMP_D *model, many basic properties are "expensive" to test: any tester requires* $\Omega(\sqrt{N})$ *queries to test, even to accuracy* $\varepsilon = 1/10$.

The lay of the land in the standard model

Fact

In the standard SAMP_D *model, many basic properties are "expensive" to test: any tester requires* $\Omega(\sqrt{N})$ *queries to test, even to accuracy* $\varepsilon = 1/10$.

Examples:

- Testing *uniformity*: $\Theta(\sqrt{N}/\varepsilon^2)$ sample complexity [GR00, BFR$^+$10, Pan08]
- Testing *equivalence to a known distribution*: $\tilde{\Theta}(\sqrt{N}/\varepsilon^2)$ [BFF$^+$01, Pan08];
- Testing *equivalence of two unknown distributions*: $\Theta\left(\max\{\frac{N^{2/3}}{\varepsilon^{4/3}}, \frac{\sqrt{N}}{\varepsilon^2}\}\right)$ [BFR$^+$10, Val11, CDVV13]

Our model: a different oracle

More power to the tester

We consider a new model. Each oracle call:

- Testing algorithm specifies a subset S of the domain $[N]$;
- In response, gets a draw from D *conditioned on it landing in* S.

Models scenarios where a scientist/experimenter has some control over an 'experiment' to restrict the range of possible outcomes (e.g., by altering the conditions).

Our model: a different oracle

More power to the tester

We consider a new model. Each oracle call:

- Testing algorithm specifies a subset S of the domain $[N]$;
- In response, gets a draw from D *conditioned on it landing in* S.

Models scenarios where a scientist/experimenter has some control over an 'experiment' to restrict the range of possible outcomes (e.g., by altering the conditions).

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 ($S = [N]$);
- Provides a richer "algorithmic playground" (adaptiveness);
- Natural variants of the COND model only allow certain specific types of subsets to be queried:
  - PCOND: can query $[N]$ or 2-element sets $\{i, j\}$;
  - ICOND can query intervals $[i, \ldots, j]$;
- similar model independently introduced by Chakraborty et al. [CFGM13].

Our model

Remark

- Generalizes the SAMP oracle ($S = [N]$);
- Provides a richer "algorithmic playground" (adaptiveness);
- Natural variants of the COND model only allow certain specific types of subsets to be queried:
  - PCOND: can query $[N]$ or 2-element sets $\{i, j\}$;
  - ICOND can query intervals $[i, \ldots, j]$;
- similar model independently introduced by Chakraborty et al. [CFGM13].

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 — in many cases, *exponentially* more efficient.

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^*
- introduce some tools: Estimate-Neighborhood and Approx-Eval
- use the tools: test equivalence of two unknown distributions

Testing Uniformity (1)

Special case of testing identity to D^*

Recall the standard SAMP model uniformity testing bounds:

Theorem (Testing Uniformity with SAMP)

Given SAMP_D*, testing whether* $D = \mathcal{U}$ *versus* D *is* ϵ*-far from uniform requires* $\Theta(\sqrt{N}/\varepsilon^2)$ *calls to* SAMP_D*.*

Testing Uniformity (1)

Special case of testing identity to D^*

Recall the standard SAMP model uniformity testing bounds:

Theorem (Testing Uniformity with SAMP)

Given SAMP_D*, testing whether* $D = \mathcal{U}$ *versus* D *is* ϵ*-far from uniform requires* $\Theta(\sqrt{N}/\varepsilon^2)$ *calls to* SAMP_D*.*

Lower bound sketch: Suppose D is either uniform over $[N]$, or uniform over a random subset of $[N]$ (hence far from uniform over $[N]$). In either case, $\sqrt{N}/100$ calls to SAMP_D will w.v.h.p. result in a uniform random subset of $\sqrt{N}/100$ distinct elements of $[N]$ (birthday paradox), so can't distinguish.

Testing Uniformity (2)

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 (2)

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" (noticeably less than $1/N$); and

(b) a *lot of weight* on "very heavy" points (noticeably more than $1/N$).

Sampling $\tilde{O}(1/\varepsilon)$ points uniformly, w.v.h.p. we get a type-(a) point, and sampling $\tilde{O}(1/\varepsilon)$ points according to D, w.v.h.p. we get a type-(b) point. Use PCOND to compare them, and get evidence that D is far from uniform.

Testing D versus D^* (1)

A more general problem: have COND_D access to D, want to test equality to a *fixed, known* distribution D^*.

The approach for uniformity does not work for general D^*.

- For testing uniformity, $O(1/\varepsilon)$ calls to PCOND will reveal when two points' weights differ by at least a multiplicative $1+\varepsilon$.
- But with a general D^*, the actual 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 D versus D^* (2)

Towards a fix: Adapt the algorithm to compare *points* with (carefully chosen) comparable *sets*. I.e., for carefully chosen $x \in [N]$ and $Y \subset [N]$, check (using COND_D) that $D(x)/D(Y)$ (approximately matches) the (known) value $D^*(x)/D^*(Y)$.

Theorem (Testing Equivalence to a known D^* with COND)

For any fixed known distribution D^*, there is a $\tilde{O}(1/\varepsilon^4)$-query COND_D *tester for equivalence to D^* (accepts w.p. at least $2/3$ if $D = D^*$ and rejects w.p. at least $2/3$ if $d_{\mathrm{TV}}(D, D^*) \geq \varepsilon$).*

Our $\tilde{O}(1/\varepsilon^4)$-query algorithm uses COND_D, not just PCOND_D. In fact, *no* PCOND_D algorithm can have query complexity independent of N:

Theorem (Lower bound for testing equivalence to D^* with PCOND)

There exists a distribution D^* over $[N]$ such that for $\varepsilon = 1/2$, any PCOND_D tester that ε-tests equivalence to D^* must make $\Omega\left(\sqrt{\frac{\log N}{\log\log N}}\right)$ queries to PCOND_D.

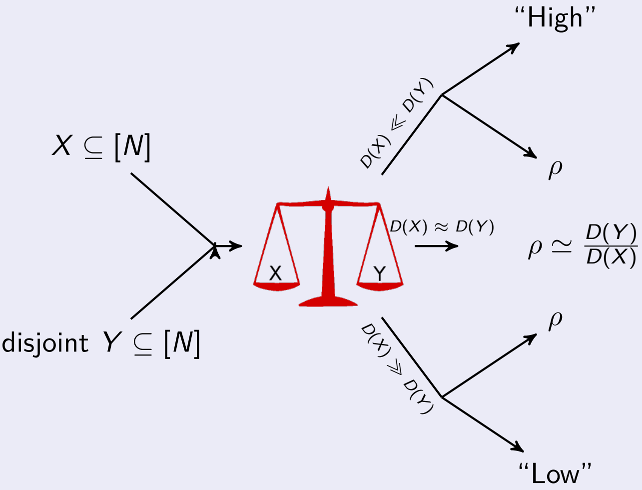
Intuition: construct D^*, and distributions D far from D^*, so that any two points either have equal weight in both cases, or very skewed weights in both cases. This means pairwise comparisons don't give new information.

Rest of the talk

Plan for rest of talk:

- sketch of testing uniformity and testing D vs. D^*
- introduce some tools: ESTIMATE-NEIGHBORHOOD and APPROX-EVAL
- use the tools: test equivalence of two unknown distributions

"Comparison is the death of joy." – Mark Twain.



Some useful tools (2)

Second tool: Estimate-Neighborhood procedure

Fix D. Given a point x, the γ*-neighborhood of* x is the set $U_\gamma(x)$ of points that have "roughly" the same weight as x (up to multiplicative $1+\gamma$):

Definition (γ-Neighborhood)

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

Some useful tools (2)

Second tool: ESTIMATE-NEIGHBORHOOD procedure

Fix D. Given a point x, the γ*-neighborhood of* x is the set $U_\gamma(x)$ of points that have "roughly" the same weight as x (up to multiplicative $1+\gamma$):

Definition (γ-Neighborhood)

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

How much weight does D put on the γ-neighborhood of x?

"Theorem"

There is an algorithm ESTIMATE-NEIGHBORHOOD which, given a point $x \in [N]$ and a parameter γ, gives a multiplicative $(1 \pm \varepsilon)$-approximation of $D(U_\gamma(x))$, and makes $\mathrm{poly}(1/\varepsilon, 1/\gamma)$ many PCOND_D queries.

Some useful tools (3)

Third tool: APPROXIMATE-EVAL oracle

EVAL oracle

An EVAL$_D$ simulator for D is a procedure ORACLE such that the output of ORACLE on input $i^* \in [N]$ is $D(i^*) \in [0, 1]$, the amount of probability D puts on i^*.

Some useful tools (3)

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 $\hat{D}(i^*) \in [0, 1]$ such that $\hat{D}(i^*) \in [1 - \varepsilon, 1 + \varepsilon]D(i^*)$.

- **Bad news:** Can't achieve this efficiently for every i^*: how to tell whether $D(i^*)$ is $1/2^{2^N}$ or $1/2^{2^{2^N}}$?
- **Good news:** *Can* achieve this for every i^* except for an "error set" of mass at most ε (where we may say "don't know").

(Approximate) EVAL oracle

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 $\hat{D}(i^*) \in [0,1]$ 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 $\hat{D}(i^*) \in [0,1]$ such that $\hat{D}(i^*) \in [1-\varepsilon, 1+\varepsilon]D(i^*)$;

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

Some useful tools (3)

Third tool: APPROXIMATE-EVAL oracle

(Approximate) EVAL oracle

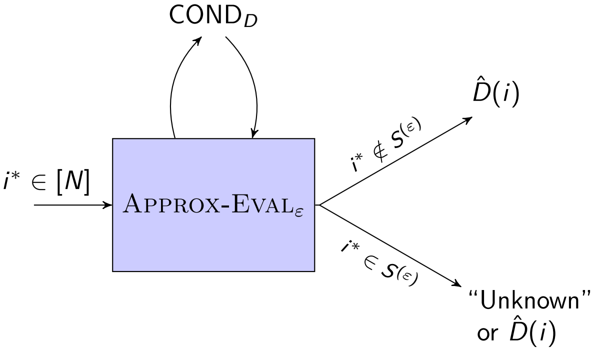
An (ε, δ)-approximate EVAL_D simulator for D is a randomized procedure ORACLE s.t for each ε, there is a fixed set $S^{(\varepsilon)} \subsetneq [N]$ with $D(S^{(\varepsilon)}) < \varepsilon$ for which the following holds. For all $i^* \in [N]$, ORACLE(i^*) is either a value $\hat{D}(i^*) \in [0, 1]$ 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 $\hat{D}(i^*) \in [0, 1]$ such that $\hat{D}(i^*) \in [1 - \varepsilon, 1 + \varepsilon]D(i^*)$;

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

Theorem

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

Some useful tools (4)

Third tool: APPROXIMATE-EVAL oracle

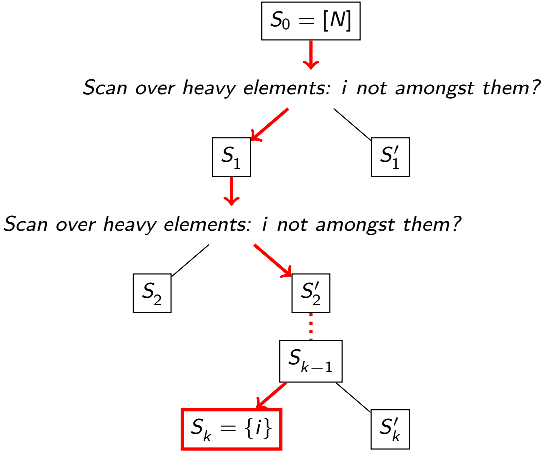


Figure: Execution of APPROX-EVAL on some i: scan over heavy elements, randomly partition the light ones, recurse; finally get an estimate of $D(i)$ by multiplying estimates at each branching.

Applications

Testing equivalence of two unknown distributions D_1, D_2

Given an oracle for D_1 and a separate oracle for D_2, distinguish $D_1 = D_2$ vs. $\mathrm{d}_{\mathrm{TV}}(D_1, D_2) \geq \varepsilon$.

Applications

Testing equivalence of two unknown distributions D_1, D_2

Given an oracle for D_1 and a separate oracle for D_2, distinguish $D_1 = D_2$ vs. $\mathrm{d}_{\mathrm{TV}}(D_1, D_2) \geq \varepsilon$.

Applications

Testing equivalence of two unknown distributions D_1, D_2

Given an oracle for D_1 and a separate oracle for D_2, distinguish $D_1 = D_2$ vs. $\mathrm{d_{TV}}(D_1, D_2) \geq \varepsilon$.

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].

Both approaches use $\mathrm{poly}(\log N, 1/\varepsilon)$ calls to the oracle.

Applications

Testing equivalence of two unknown distributions D_1, D_2

Given an oracle for D_1 and a separate oracle for D_2, 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].

Both approaches use $\mathrm{poly}(\log N, 1/\varepsilon)$ calls to the oracle.

Theorem

[Acharya, Canonne, Kamath '14] Any tester for this problem must make $\Omega(\sqrt{\log\log N})$ COND_D *queries.*

Conclusion

- new model for studying probability distributions
- attempt to capture aspects of real-world settings where experimenter can do more than just SAMP
- 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, ...)

The end.

Thank you.

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.

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 $i_1, \ldots, i_q$ uniformly {Reference points}

for $j = 1$ to t **do**

  Call the oracle $s_j = \Theta(2^j t)$ times to get $h_1, \ldots, h_{s_j} \sim D$ {Heavy points?}

  Draw s_j points $\ell_1, \ldots, \ell_{s_j}$ 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 COMPARE

Completeness unless 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. □

Some useful tools (5)

The (slightly) higher-level subroutine $\textsc{Estimate-Neighborhood}$

Given as input a point x, parameters $\gamma, \beta, \eta \in (0, 1/2]$ and PCOND_D access, the procedure $\textsc{Estimate-Neighborhood}$ 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 $(\dots)$
2. If $D(U_\alpha(x)) < \beta$, then $\hat{w} \leq (1 + \eta) \cdot \beta$, and $(\dots)$

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

Some useful tools (5)

The (slightly) higher-level subroutine ESTIMATE-NEIGHBORHOOD

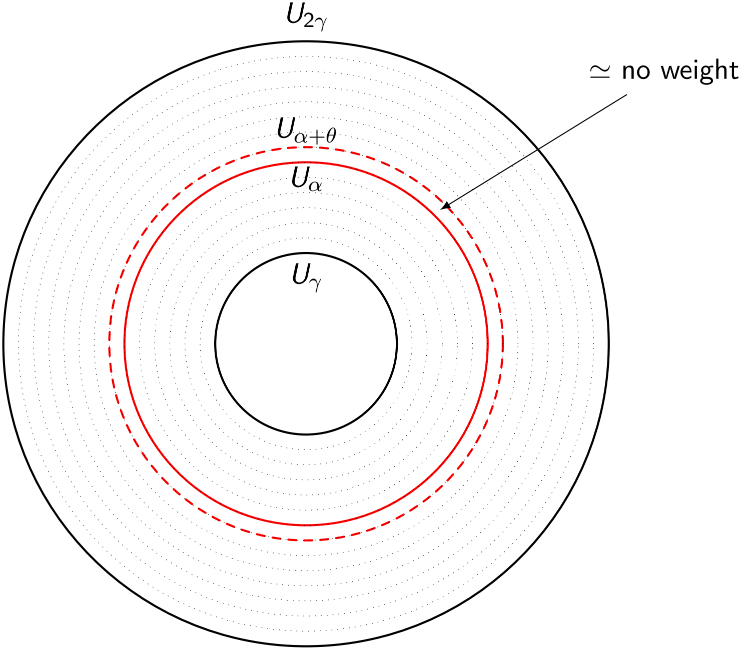
Given as input a point x, parameters $\gamma, \beta, \eta \in (0, 1/2]$ and PCOND_D access, the procedure ESTIMATE-NEIGHBORHOOD outputs a pair $(\hat{w}, \alpha) \in [0, 1] \times (\gamma, 2\gamma)$ such that w.h.p

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

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}

Some useful tools (6)

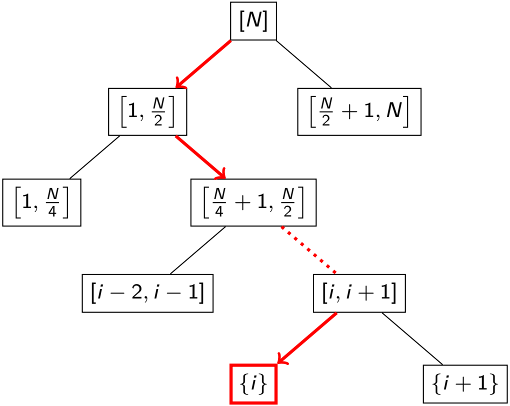


Figure: (Rough) idea of the "binary descent" on i for APPROX-EVAL: get an estimate of $D(i)$ by multiplying estimates at each branching.