A Subgradient Algorithm For Computational Distances and Applications to Cryptography

Maciej Skorski
University of Warsaw

Abstract

The task of finding a constructive approximation in the computational distance, while simultaneously preserving additional constrains (referred to as "simulators"), appears as the key difficulty in problems related to complexity theory, cryptography and combinatorics.

In this paper we develop a general framework to *efficiently* prove results of this sort, based on *subgradient-based optimization applied to computational distances*. This approach is simpler and natural than KL-projections already studied in this context (for example the uniform min-max theorem from CRYPTO'13), while simultaneously may lead to quantitatively better results.

Some applications of our algorithm include:

- Fixing an erroneous boosting proof for simulating auxiliary inputs from TCC'13 and much better bounds for the EUROCRYPT'09 leakage-resilient stream cipher
- Deriving the unified proof for Impagliazzo Hardcore Lemma, Dense Model Theorem, Weak Szemeredi Theorem (CCC'09)
- Showing that "dense" leakages can be efficiently simulated, with significantly improved bounds

Interestingly, our algorithm can take advantage of small-variance assumptions imposed on distinguishers, that have been studied recently in the context of key derivation.

1 Introduction

Many important objects in modern cryptography are built on the common concept of relaxing the way we define similarity or indistinguishability, which allows us to go beyond information-theoretic limits. Typically, when defining them, we refer to some *ideal object* whose behavior we want to mimic, that is approximate but keeping certain restrictions, however with better efficiency (for example reduced complexity or reduced amount of randomness needed). For illustrative purposes, we mention some of them below

(a) *Pseudorandom generators*: we want to generate a distribution close to the uniform one, but using much less auxiliary randomness. A probabilistic argument shows that this is indeed possible if we don't care about the complexity[7]. It remains an open question whether such a transformation can be polynomial in the input size.

(b) *Simulating leakages in cryptography*: the goal of leakage-resilient cryptography is to extend classical results (encryption schemes, key derivation and so on) to handle cases where some auxiliary information leaks to adversaries. For any X which is leakage from a secret state Z we can think of X as a randomized function of Z, that is $X \stackrel{d}{=} h(Z)$. A non-trivial problem is to answer whether we can find an efficient h which approximates X well [10, 17].

(c) *Key derivation*: The classical Leftover Hash Lemma [8] states that universal hash functions are randomness extractors. The only problem is with the inherited entropy loss, where one needs to sacrifice $2\log(1/\epsilon)$ bits of randomness, in order to achieve ϵ-closeness (security for any adversary). For practical applications where $\epsilon = 2^{-80}$ this gives 160 bits, which is quite a lot. It has been shown [1, 4] that this loss can be overcome if we modify the closeness by considering only certain computationally bounded adversaries (which is possible for a wide class of applications, so called square-friendly applications)

1.1 Question

We are interested in approximating some reference object p^* when keeping certain constraints satisfied and with possibly low complexity. Informally this reads as follows

Given the target p^*, consider p satisfying the following conditions

(a) *computational closeness*: p is close to p^* under some class of test functions
(b) *constraints preserved*: p satisfies required constraints

Question: how to construct p with *minimal complexity* with respect to $\mathcal{D}$?

We intentionally emphasize the word *construct* to mention that we want a constructive (preferably iterative) solutions. In Section 1.2 we will give a more precise meaning to what are constrains, computational distances, and computational efficiency. In Section 1.3 we discuss the related literature, and in Section 1.5 we present our algorithm. After providing necessary definitions in Section 2, we present applications of our results in Section 3.

1.2 Abstract problem statement

We start by discussing main components of optimization tasks we study in this paper.

1.2.1 Key ingredients

1.2.1.1 Domain

We consider optimization problems over the set of all bounded functions on a finite set X, which we denote by $B(\mathcal{X})$. In most cases we will be interested in *probability distributions* or *positive measures*, however even for these applications it is convenient to work with the extended definition (similarly to related works [16, 10]). We can think of functions on $\mathcal{X}$ as vectors with components enumerated by elements of $\mathcal{X}$, and $B(\mathcal{X})$ as a subset of $\mathbb{R}^N$ where $N = |\mathcal{X}|$[1] (For example, probability distributions correspond to vectors with positive components summing up to 1). This allows us to use multidimensional optimization tools.

1.2.1.2 Reference function

By p^* we denote the reference function/vector, which is the *model object* we want to approximate. Typically it satisfies certain bounds (inequalities) we want to preserve.

1.2.1.3 Constraints

The constraints we impose usually come directly from the problem setting and in most cases translate some simple information-theoretic restrictions, for example probability distributions with certain entropy. Often the constraints are simply *linear inequalities and equations*, perhaps composed with the maximum function. We will see that this sort of functions is particularly easy to deal with, basically because it allows very simple subgradient calculations (see Section 2). For example, to consider probability distributions $p(\cdot)$ over

[1] Where the vector entries correspond to the values of the probability mass function on particular points, according to some fixed order

$\mathcal{X}$ that have min-entropy at least k we write the following inequalities

$$p(x): \quad \begin{cases} \max_{x\in\mathcal{X}}(-p(x)) \leqslant 0 \\ \max_{x\in\mathcal{X}}(p(x)) - 2^{-k} \leqslant 0 \\ \sum_{x\in\mathcal{X}} p(x) - 1 = 0 \end{cases}$$

It suffices to consider inequalities only, as every equation can be written as two inequalities. We follow the "zero right-hand side" convention, when constraints are of the form $C(p) \leqslant 0$.

1.2.1.4 (Computational) Distance

Fix a class $\mathcal{D}$ of real functions on $\mathcal{X}$, understood as "distinguishers". In most cases $\mathcal{D}$ consists of bounded functions of small complexity[2]; later we will discuss a specific case of restrictions imposed on the variance of members of $\mathcal{D}$. Distinguishers are used to measure "similarity" or "closeness" between functions on $\mathcal{X}$ as follows

$$\delta^{\mathcal{D}}(p_1;p_2) = \max_{\mathrm{D}\in\mathcal{D}} \left| \sum_{x\in\mathcal{X}} \mathrm{D}(x)(p_1(x)-p_2(x)) \right|$$

For example, if we take $\mathcal{D}$ to be all $[-1,1]$ valued function, we retrieve the notion of the L_1-norm. More generally, we can rewrite and extend this definition using the standard inner product between D and $p_1 - p_2$ interpreted as vectors (see Theorem 7), as well as extend to any convenient inner product. This observation is crucial for out technique, because it shows explicitly that *distinguishers are subgradients* of the computational distance functions.

1.2.2 Optimization tasks

As in every optimization problem we attempt to minimize a loss function given some constraints

$$\begin{aligned} \underset{p}{\text{minimize}} \quad & \delta^{\mathcal{D}}(p,p^*) \\ \text{s.t.} \quad & C_j(p) \leqslant 0, \quad j\in J \end{aligned} \tag{1}$$

where in our case constraints come from information-theoretic restrictions and the loss equals the statistical distance. We expect to achieve $\delta^{\mathcal{D}}(p,p^*) \leqslant \epsilon$ making at most $\mathrm{poly}(\epsilon^{-1})$ oracle queries to $\mathcal{D}$ and arithmetical operations in the definition of p[3].

1.3 Related works

Below we give a short overview of how the problem has been addressed so far in the literature.

1.3.1 Nonconstructive min-max theorem

Interestingly the best tool to prove results of this sort non-constructively, if we don't care much about efficiency (improving bounds of this sort depends on convex approximation results and may be tricky), is the min-max theorem. Being interested in constructive techniques, however, we omit the detailed discussion of applications of the classical min-max theorem, mentioning only that eventually, iterative methods turn out to be equally good or better in terms of efficiency. See for example [16].

---

[2] Often functions computable by circuits of size logarithmic in the domain size.
[3] Sometimes we also allow the use of small storage space,like the standard RAM model.

1.3.2 Naive iterative optimization (descent)

1.3.2.1 Description

Presumably the most convenient way to attack the problem is by a *naive iterative optimization*. In this approach we simply check if our candidate solution is good enough, and - if not - optimize it based on a "certificate" of the negative answer. This certificate is simply a distinguisher D which contradicts to the computational distance being (supposedly) small; by definition, such a certificate must exists. The pseudocode of this meta-algorithm is provided in Algorithm 1. We hope to find update rules which are relatively simple and such that the sequence of updates converges[4]. While the general theory equips us with concrete updates and guarantees the convergence under general convexity conditions, the complexity of computing updates for concrete problems may be quite large, especially with constraints.

1.3.2.2 Naive iteration pseudocode

Algorithm 1: Boosting for computational distance

Data: starting point p_0, steps number N, update rules, distinguisher oracle
Result: a point p close to p_0

$i \leftarrow 0$
$p_i \leftarrow p_0$ `// initialize`
while $i < N$ **do**
  $i \leftarrow i + 1$
  `// check if the current solution is too far`
  if *there is a distinguisher* D *for* p_{best} *and* p^* **then**
    $p_i \leftarrow \text{update}\,(p_{i-1}, \text{D})$ `// update based on the new distinguisher`
  else
    $p_{\text{best}} \leftarrow p_i$
    return p_{best}
`// Lookup for the best answer (not always the last one)`
$p_{\text{best}} \leftarrow \text{best}(p_1, \ldots, p_N)$
return p_{best}

1.3.2.3 Note on uniformness

The sequence of updates may be less or more "uniform", for example it might be dependent on calculating some auxiliary numbers. We stress that when building a solution we do not attempt to find this distinguisher - it simply exists and is considered as an oracle answer. While this yields no problems in the non-uniform setting, it looks like a little bit of cheating whenever we talk about uniform iterative algorithms. This is, however, how uniform iterative algorithms for computational distance are defined in the literature [2, 17].

1.3.2.4 [11]

Possible applications of boosting technique for crypto-related problems (though not related directly to computational-distance) were investigated for the first time in [11]. The au-

[4] The method is referred to as a "descent" because the proof of a convergence usually goes by showing that a certain function, called "potential" decreases with every iteration

thors observed that the proof of the Hardcore Lemma [9] can be seen as a specific iterative optimization algorithm.

1.3.2.5 [16]

The first application of iterative optimization techniques to cryptography-related problems concerning computational distance goes back to [16], where the authors reproved Dense Model Theorem, Weak Szemeredi Lemma and proved that any distribution of high-min entropy is computationally close to samplable distributions with the same entropy (see Theorem 11). The only constraints covered in the proof were *rectangular constraints*, that is inequalities of the form $-1 \leqslant p(x) \leqslant 1$. The technique uses an ad-hoc potential function.

1.3.2.6 [2]

A different iterative algorithm was proposed by Barak et al. [2]. The proof uses *Bregman projections* projections, and is restricted for distributions with certain density constraints.

1.3.2.7 [10]

An extension of the method from [16], aiming to to cover constraints induced by conditional distributions, that is $0 \leqslant p(x|z)$ and $\sum_x p(x|z) = 1$, was proposed in [10]. However this proof is flawed and doesn't seem to be fixable [13] (we will return to this later in Section 3.2).

1.3.2.8 [17]

The uniform min-max theorem from [17] gives a result that solves the problem in general, under some (weak) convexity assumptions. However, as we point out later, there are some interesting cases where, due to generic approximations, it fails to achieve best bounds.

1.4 Advantages of our approach

Compared to related works, in particular to [17], our approach is

1. *More modular and explicit*: it captures several factors, see e.g. the remark after Theorem 2 and Section 3.4
2. *Simpler*: we do not need complicated projections, just subgradients which are easy to calculate for practical constraints; also the convergence is proved in the second norm)
3. *May be quantitatively better*: see Section 3.2 and in particular Table 2.
4. *Illustrative*: it demonstrates that distinguishers are nothing more than subgradients for computational distance, and how the subgradient calculus is powerful in this context.

1.5 Our results

We consider the problem in two settings, depending on whether the constraints involve all vector components or only a subset of indexes (the later being referred to as the conditional setting). This distinction allows us to better handle constraints in specific cases where we can take advantage of a smaller domain, but the algorithm is essentially the same. Our technique is a *subgradient algorithm*, in which we follow the negative subgradient direction in every round, similarly to the classical gradient/subgradient descent technique [3]. However, we apply it to the specifically designed objective function. It captures not only the computational distance but also *penalty terms* for violating constraints. Technically speaking, we

convert our problem into an *unconstrained problem with penalty terms* and solve it by the standard subgradient optimization. The iterative process ends on a point at which *penalties are small*, which allows us to simply build the final answer by making it fit the constraints. A comparison of out technique with related works is discussed in Section 1.5.3

1.5.1 Unconditional setting (general)

Suppose that the constraints are functions of *all components* of the input vector:

$$\begin{array}{ll} \underset{p=p(x)\in\mathbb{R}^N}{\text{minimize}} & \max_{\mathrm{D}\in\mathcal{D}}\langle \mathrm{D};\ p-p^*\rangle \\ \text{s.t.} & C_j(p)\leqslant 0, \quad j\in J \end{array} \tag{2}$$

In order to solve problem (2), we attempt to minimize the following objective L which captures *weighted penalty terms* for violating constraints:

$$L(p)=\max_{\mathrm{D}\in\mathcal{D}}\langle \mathrm{D};\ p-p^*\rangle+\sum_j \alpha_j(\max(C_j(p(x)),0))^2 \tag{3}$$

The pseudocode is given in Algorithm 2 below.

Algorithm 2: Subgradient Algorithm, General

Data: starting point p_0, loss function L, step size γ, steps number N
// use L defined as in Equation (12)
// L is the comp. distance to p^* + constraints penalty terms
Result:
$i \leftarrow 0$
$p_{\text{best}} \leftarrow p_0$ // the best answer found so far
while $i < N$ **do**
  $i \leftarrow i+1$
  $w \leftarrow \partial_{p_i} L(p_i)$ // compute a subgradient of L at p_i (see Equation (4))
  $p_i = p_{i-1} - \gamma w$ // go towards the negative subgradient direction
$p_{\text{best}} \leftarrow p_i^{\text{proj}}$
return p_{best}

The subgradient of L can be computed by basic rules (see Lemma 9 and 10) as shown below.

▶ **Lemma 1** (Objective subgradient). *For some* D *in* $\mathcal{D}$ *and any* $v_j \in \partial_p C_j(p)$ *we have that*

$$w=\mathrm{D}+\sum_j \alpha_j \max(C_j(p),0)\cdot v_j \tag{4}$$

is a subgradient of L *at* p.

▶ Remark (Non-uniqueness). For any j we need to choose v_j from possibly many subgradients $\partial_p C_j$. This flexibility might help us to choose possibly simple v_j (in terms of complexity).

In Theorem 2 below we estimate the convergence of Algorithm 2.

▶ **Theorem 2** (Subgradient algorithm for computational distance, convergence). *Let* p^* *be the optimal point and* p^0 *be the starting point. Let* $\langle , \rangle$ *be an inner product and* $\|\cdot\|$ *be the induced norm. Let* D, R *and* M_j *for* $j \in J$ *be constants such that*

$$D \geqslant \max_{\mathrm{D}\in\mathcal{D}} \|\mathrm{D}\|^2 \qquad R \geqslant \left\|p^0-p^*\right\|^2 \qquad M_j \geqslant \max_{p:\|p-p^*\|<R^2} \|\partial_p C_j(p)\|^2 \quad \text{for } j\in J \tag{5}$$

and L be the objective

$$L(p) = \max_{\mathrm{D}\in\mathcal{D}} \langle \mathrm{D};\ p - p^* \rangle + \sum_{j\in J} \alpha_j \left(\max\left(C_j(p), 0\right)\right)^2 . \tag{6}$$

Then setting $\gamma = \frac{\epsilon}{(|J|+1)D}$ and $\alpha_j = \frac{2}{(4|J|+1)M_j\gamma}$ in Algorithm 1, after T steps where

$$T = O\left(\frac{D\cdot R\cdot |J|}{\epsilon^2}\right) \tag{7}$$

for $p = p^T$ we get

$$\max_{\mathrm{D}\in\mathcal{D}} \langle \mathrm{D};\ p - p^* \rangle + \sum_{j\in J} \alpha_j C_j(p) < \epsilon. \tag{8}$$

In particular, at the end we have

$$\delta^{\mathcal{D}}(p, p^*) < \epsilon \tag{9}$$

$$C_j(p) < \sqrt{\frac{M_j\epsilon^2}{D}} \quad \forall j \in J. \tag{10}$$

▶ Remark (Intuitions). Our result captures the following intuitions:

(a) the closer to the target we start (smaller R), the better convergence
(b) the closer are distinguishers to being constant (small variance D), the better convergence
(c) the less regular constrains (big numbers M_j) the worse convergence
A short comparison with the related works is given in Table 1.

1.5.2 Average-case version (conditional)

We consider the following generalization of Equation (2)

$$\begin{array}{ll} \underset{p=p(x,z)}{\text{minimize}} & \max_{\mathrm{D}\in\mathcal{D}} \langle \mathrm{D};\ p - p^* \rangle \\ \text{s.t.} & C_{j,z}(p(\cdot,z))) \leqslant 0, \quad \forall j \in J, \quad \forall z \in \mathcal{Z} \end{array} \tag{11}$$

where $p(\cdot, z)$ denotes the vector of size $|\mathcal{X}|$ with components $(p(x,z))_{x\in\mathcal{X}}$ for fixed z. Note that although formally the domain of the problem is the product $\mathcal{X}\times\mathcal{Z}$, the constraints are restricted to only components from $\mathcal{X}$ and technically, when optimizing, we work only over the set $\mathcal{X}$. This distinction is meaningful when the constraints are imposed on *conditional distributions*, for example distributions (X, Z) where X has high entropy conditioned on Z. In order to solve problem (2), we attempt to minimize *for every single* z the following objective L which captures *penalty terms* for violated constraints (when z is fixed).

$$L_z(p) = \max_{\mathrm{D}\in\mathcal{D}} \langle \mathrm{D}(\cdot,z);\ p(\cdot,z) - p^*(\cdot,z) \rangle + \sum_{j\in J} \alpha_j (\max(C_j(p(\cdot,z), 0))^2 \tag{12}$$

The corresponding convergence result is similar, the only difference is that we prove that the constraint are (almost) satisfied on average over $\mathcal{Z}$. The proof is also similar, but we need to make sure not only the convergence but in applications also that the step size doesn't depend on z (which would massively increase the complexity).

▶ **Theorem 3** (Subgradient algorithm for computational distance). *Let p^* be the optimal point and p^0 be the starting point. Let $\langle , \rangle$ be an inner product and $\|\cdot\|$ be the induced norm. Let D, R and M_j for $j \in J$ be constants such that*

$$D = \max_{z\in\mathcal{Z}} \max_{\mathrm{D}\in\mathcal{D}} \|\mathrm{D}\|^2 \tag{13}$$

$$R = \max_{z\in\mathcal{Z}} \left\|p^0(\cdot,z) - p^*(\cdot,z)\right\|^2 \tag{14}$$

$$M_j = \max_{z\in\mathcal{Z}} \max_{p:\|p(\cdot,z)-p^*(\cdot,z)\|<R^2} \left\|\partial_{p(\cdot,z)} C_j(p(\cdot,z))\right\|^2 \quad \text{for } j \in J \tag{15}$$

and, for any single z, let L_z be the objective

$$L_z(p) = \max_{\mathrm{D}\in\mathcal{D}} \langle \mathrm{D}(\cdot,z);\ p(\cdot,z) - p^*(\cdot,z)\rangle + \sum_j \alpha_j (\max(C_j(p(\cdot,z),0))^2 \tag{16}$$

Then setting $\gamma = \frac{\epsilon}{(|J|+1)D}$ and $\alpha_j = \frac{2}{(4|J|+1)M_j\gamma}$ in Algorithm 1, after T steps where

$$T = O\left(\frac{D\cdot R\cdot |J|}{\epsilon^2}\right) \tag{17}$$

for $p = p^T$ and any z we get $L_z(p) < \epsilon$, which means that (in particular)

$$\delta^{\mathcal{D}}(p, p^*) < \epsilon \tag{18}$$

$$C_{j,z} < \sqrt{\frac{M_j\epsilon^2}{D}} \quad \forall z \in \mathcal{Z}\ \forall j \in J \tag{19}$$

We skip the proof as it goes along the lines of the proof of Theorem 2.

1.5.3 Comparison with related works (overview)

Author	Technique	Constrains	Initial distance	Distinguisher structure	Comments
[16]	ad-hoc potential + proj.	rectangular	Not exploited	Not exploited	
[2]	Bregman proj.	density	Not exploited	Not exploited	
[10]	L_2-porential	restricted subsums			An error in the proof
[17]	Bergman Projections	any convex	No	No	
this paper	subgradients	any convex	speed up (L_2-distance)	speed up (variance)	

Table 1 Comparison of different techniques with our approach.

2 Preliminaries

▶ **Definition 4** (Convex hull). For any subset S of $\mathbb{R}^N$ by $\mathrm{conv}(S)$ we denote the set which contains all convex combinations of elements of S.

▶ **Definition 5** (Inner products and induced norms). The standard inner product on $\mathbb{R}^N$ is

$$\langle p; q\rangle = \sum_{i=1}^{N} p(i)q(i) \tag{20}$$

where $p(i)$ denotes the i-th component of the vector p. In general, $\langle\cdot;\ \cdot\rangle$ is an inner product when is bilinear and positive definite. The norm $\|\|$ associated with the inner product is

$$\|p\| = \langle p;\ p\rangle. \tag{21}$$

▶ **Definition 6** (Bounded functions)**.** Given a finite set $\mathcal{X}$, we denote the set of all finite real-valued function on $\mathcal{X}$ by $B(\mathcal{X})$. Enumerating elements $\mathcal{X}$ as $x_1, \ldots, x_N$, with respect to some fixed order[5], we can identify it with a vector $(p(x_1), \ldots, p(x_N))$ that has N non-negative components. In particular, the inner product of two functions p and q is defined as the inner product of the corresponding vectors

$$\langle p_1; p_2 \rangle = \langle (p(x_1), \ldots, p(x_N);\ q(x_1), \ldots, q(x_N)) \,. \tag{22}$$

▶ **Definition 7** (Computational distance)**.** The computational distance of functions p_1 and p_2 under the class of functions $\mathcal{D}$ and fixed scalar product $\langle\rangle$ is equal to

$$\delta^{\mathcal{D}}(p_1, p_2) = \max_{\mathrm{D} \in \mathcal{D}} \langle \mathrm{D};\ p_1 - p_2 \rangle \tag{23}$$

▶ **Definition 8** (Subgradients and subdifferentials)**.** A *subgradient* of a function $L : \mathbb{R}^N \to \mathbb{R}$ at a point $p \in \mathbb{R}^N$ is any vector $\alpha \in \mathbb{R}^N$ such that

$$\forall q \in \mathbf{dom}(L) : \quad L(q) \geqslant L(p) + \langle \alpha; q - p \rangle.$$

where $\langle \cdot; \cdot \rangle$ is any (fixed) scalar product on $\mathbb{R}^N$. The set of all such vectors α is called *subdifferential* of L at p and denoted by $\partial_p L(p)$.

▶ Remark (Subgradients existence and uniquness). Subgradients may not exist or be non-unique. However, if L is convex, finite-valued and continuous they always exist [6] with any inner product. There may be many subgradients at a given point, however for the standard inner product and differentiable L, the subgradient is unique and equal to the gradient.

Linearity and chain rules are what subgradients share with gradients (first order derivatives).

▶ **Lemma 9.** *For convex and finite functions L_1, L_2, any real constants α_1, α_2 we have*

$$\partial \left(\alpha_1 L_1 + \alpha_2 L_2 \right) = \alpha_1 \partial L_1 + \alpha_2 \partial L_2$$

Also, for any convex L_1 and L_2 we have the chain rule as for the first-order derivatives

$$\partial_p \left(L_1 \circ L_2(p) \right) = \langle \partial_{L_2(p)} \left(L_1 \right);\ \partial_p L_2(p) \rangle$$

The next lemma shows how to compute subgradients of pointwise maximums.

▶ **Lemma 10** (Pointwise maximum subgradients, [3])**.** *Let $\{L_j\}_{j \in J}$ be a finite collection of real convex functions on $\mathbb{R}^N$. Then for any point p*

$$\partial \left(\max_{j \in J} L_j(p) \right) = \mathrm{conv} \left(\partial L_j(p) : \quad j \in J^{\mathrm{act}} \right) \tag{24}$$

where J is the so called set of active indexes *at p, that is*

$$j \in J^{\mathrm{act}} \iff L_j(p) = \max_{j=1 \in J} L_j(p).$$

In particular, for any active j any subgradient of L_j at p is the subgradient of $\partial \max_{j \in J} L_j$.

[5] For $\mathcal{X} = \{0,1\}^n$ we take the lexicographical order

3 Applications

3.1 Hardcore Lemma, Dense Model Theorem, and simulating high min-entropy

As shown in [16], several important results: Hardcore Lemma, Dense Model Theorem, Weak Szemeredi Lemma and an original result about simulating high-min entropy distributions can be derived from one single theorem. We show that it easily follows from Theorem 2.

▶ **Theorem 11** ([16]). *Let $\mathcal{X}$ be a finite set, and let $\mathcal{D}$ be a class of functions $\mathrm{D} : \mathcal{X} \to [-1, 1]$. Then for any function $f : \mathcal{X} \to [-1, 1]$ there exists a function $h : \mathcal{X} \to [-1, 1]$ such that*

(a) Low complexity with respect to $\mathcal{D}$
(b) Indistinguishability from f
More precisely, with complexity $O\left(\epsilon^{-2}\right)$ one gets ϵ-indistinguishability by $\mathcal{D}$.

▶ Remark. The result is provided for the extended notion of computational indistinguishability, with a weight function μ involved: $\left|\sum_{x\in\mathcal{X}} \mathrm{D}(x)h(x)\mu(x) - \sum_{x\in\mathcal{X}} \mathrm{D}(x)f(x)\mu(x)\right| \leqslant \epsilon$ (the standard case is $\mu(x) = 1$). We will also conclude this slightly more general version, by choosing a non-standard inner product.

Proof. Suppose for now that the weight function μ equals 1. Consider the space $\mathcal{X} = \{0, 1\}^n$ and the restrictions $C_1(p) = \sum_{x\in\mathcal{X}} \max(0, p(x) - 1)$, $C_2(p) = \sum_{x\in\mathcal{X}} \max(0, -p(x) - 1)$, $C_3(p) = 1 - \sum_{x\in\mathcal{X}} p(x)$ and $C_4(p) = \sum_{x\in\mathcal{X}} p(x) - 1$. By Lemma 9 and 10 for every p and $j = 1, 2$ we have

$$\frac{1}{|\mathcal{X}'|} \sum_{x\in\mathcal{X}'} \mathbf{1}_{p(x)>1}\mathbf{e}_x \in \partial_p C_1(p), \quad \frac{1}{|\mathcal{X}'|} \sum_{x\in\mathcal{X}'} -\mathbf{1}_{p(x)<-1}\mathbf{e}_x \in \partial_p C_2(p)$$

where we use the following notational convention: the vector $\mathbf{e}_x$ has 1 at position x and 0 elsewhere, and for any predicate A the function $\mathbf{1}_A$ outputs 1 on instances where it is satisfied and 0 elsewhere. We also conclude that $\partial_p C_3(p)$ and $\partial_p C_4(p)$ are vectors with all -1 and respectively 1 entries. Now we easily compute necessary constants for Theorem 2.

▶ Claim 1. In Theorem 2 we can put $R = 1$, $D = 2^n$, $M_1 = M_2 = 1$ and $M_3 = M_4 = 2^n$.

Clearly, C_j they can be computed in complexity $O(1)$ from p[6]. By Theorem 2 we obtain $\left|\sum_{x\in\mathcal{X}} \mathrm{D}(x)p(x) - \sum_{x\in cX} \mathrm{D}(x)p^*(x)\right| \leqslant \epsilon$ and

$$\sum_{x\in\mathcal{X}} \max\left(p(x) - 1, 0\right) \leqslant \epsilon, \quad \sum_{x\in\mathcal{X}} \max\left(-p(z, x) - 1, 0\right) \leqslant \epsilon, \quad 1 - \epsilon \leqslant \sum_{x\in\mathcal{X}} p(x) \leqslant 1 + \epsilon$$

Note that the constraints are only "slightly" violated - a mass shifting argument shows that they can be fixed, in the sense that p can be transformed, with an only additive overhead $O(1)$, into a feasible point p', such that we lose only $O(\epsilon)$ in the computational distance. This final "shift" is essentially the same as in the proof of Theorem 14. To obtain the statement with the weight function μ we need to *change the inner product* to

$$\langle p;\ q\rangle = \sum_x p(x)q(x)\mu(x)$$

and proceed as before. It remains only to prove that we can compute subgradients efficiently. ◀

[6] This is trivial for $j = 3, 4$. In turn for $j = 1, 2$ we only compare $p(x)$ with -1 or 1 at one point, and hardcoding the constants $|\mathcal{X}'|, |\mathcal{X}''|$

3.2 Simulating auxiliary inputs and better security for leakage-resilient stream ciphers

For every joint distribution (Z, X) the marginal X can be represented as a randomized (and inefficient) function of Z. If X is relatively *short* we intuitively hope to approximately compute if from Z. Indeed, this is possible at a price of a distance-efficiency tradeoff, as shown in [10]. Applying such a "simulator" to X being leakage from a secret Z, one achieves better and simpler bounds for leakage-resilient stream ciphers [12] as explained in [10]. We use our Theorem 3 to beat best known bounds by a factor $O\left(\epsilon^{-2}\right)$ in running time.

▶ **Theorem 12** ([10], corrected). *For every distribution (Z, X) on $\{0,1\}^n \times \{0,1\}^\ell$ and every ϵ, s, there exists a "simulator" $h : \{0,1\}^n \to \{0,1\}^\ell$ such that*

(a) $(Z, h(Z))$ and (Z, X) are (ϵ, s)-indistinguishable[7]
(b) h is of complexity $s_{\mathsf{h}} = O\left(s \cdot 2^{3\ell}\epsilon^{-2}\right)$

▶ Remark (Flaws in the original proof). The original proof achieves only $s_{\mathsf{h}} = O\left(s \cdot 2^{4\ell}\epsilon^{-4}\right)$ due to flaws. Here we correct a fixable flaw (a missing factor of 2^ℓ), but also report that there is a more serious error (which seems to be unfixable) in the alternative boosting proof, which was claimed to offer much better bounds. The flaws are discussed in Appendix A.

▶ Remark (Alternative bounds in [17]). We note that the uniform min-max theorem due to Vadhan and Zheng [17] implies incomparable bounds $s_{\mathsf{h}} = O\left(s \cdot 2^{\ell}\epsilon^{-2} + \epsilon^{-4}\right)$ in Theorem 14. However, our result may be much better, in particular for small-or moderate values of ℓ, as demonstrated in Table 2, and in terms of time-advantage ratio (where we lose only a factor of 4 versus a factor of 6 in [17])
Using the above result and following the argument in [10], we improve the security bounds of the Eurocrypt'09 leakage-resilient stream cipher [12]. A necessary background on leakage-resilient stream ciphers is given in Appendix B, here we only mention that for the first time we get meaningful bounds for a weak PRF instantiated with a standard 256-bit block cipher like AES. See Table 2 for an illustrative example.

▶ **Theorem 13** (Proving Security of Stream Ciphers [10]). *If F is a $(\epsilon_F, s_F, 2)$-secure weak PRF then SC^F is a $(\epsilon', s', q, \lambda)$-secure leakage resilient stream cipher where*

$$\epsilon' = 4q\sqrt{\epsilon_F 2^\lambda}, \quad s' = \Theta(1) \cdot \frac{s_F \epsilon'^2}{2^{3\lambda}}.$$

▶ Remark (The exact complexity loss). The inspection of the proof in [10] shows that s_F equals the complexity of the simulator h in Theorem 14 applied to the class of all circuits of size s' and ϵ replaced by ϵ'. Thus, our result is better by a factor of $O(\epsilon'^{-2})$.

Analysis/Authors	wPRF security	Leakage	Advantage ϵ'	Size s'
[10] (a variant of Theorem 14)	256	$\lambda = 3$	2^{-40}	0
[17] (a variant of Theorem 14)				0
this paper (Theorem 14)				2^{66}

Table 2 The security of the EUROCRYPT'09 stream cipher, instantiated with AES256 as a weak PRF of $k = 256$ bits of security, that is $s_F/\epsilon_F \approx 2^{256}$. In this setting only our new bounds provide non-trivial bounds even if we aim for security $\epsilon' \approx 2^{-40}$.

[7] That is indistinguishable by the class $\mathcal{D}$ of circuits of size s

Proof. We start by observing, similarly to [10], that it suffices to prove the following result

▶ Theorem. Let $\mathcal{Z}, \mathcal{X}$ be finite sets, and let $\mathcal{D}$ be a class of functions $\mathrm{D} : \mathcal{Z} \times \mathcal{X} \to [-1, 1]$. Then for any function $f : \mathcal{Z} \times \mathcal{X} \to [-1, 1]$ there exists a function $h : \mathcal{X} \to [-1, 1]$ with the following properties:

(a) h uses $O\left(s \cdot 2^{2\ell}\epsilon^{-2}\right)$ calls to $\mathcal{D}$ and $O\left(2^{\ell}\right)$ auxiliary memory[8],
(b) h is ϵ-indistinguishable from f by $\mathcal{D}$,
(c) $\sum_x h(z, x) = \sum_x f(z, x)$ for every $z \in \mathcal{Z}$.

Note that constraint (c) makes the only difference to the proof of Theorem 11. Define $\beta(z) = \sum_x f(z, x)$ and consider the variables $p(z, x)$, 2^{ℓ}-dimensional vectors $p = p(z, \cdot)$ and the following constraints: $C_{1,z}(p) = \sum p(z, x) - \beta(z)$ for $z \in \mathcal{Z}$, $C_{2,z}(p) = -\sum p(z, x) + \beta(z)$ for $z \in \mathcal{Z}$, $C_{3,z}(p) = \sum_x \max(-p(z, x) - 1, 0)$ for $z \in \mathcal{Z}$, $C_{4,z}(p) = \sum_x \max(p(z, x) - 1, 0)$ for $z \in \mathcal{Z}$. The numbers $\beta(z)$ stands for the probabilities $\Pr[Z = z]$, and numbers $p(z, x)$ for the probabilities $\Pr[h(z) = x]$. Constraints $C_{3,z}$, if positive, are penalties for breaking the inequality $-1 \leqslant p(x, z)$ and similarly $C_{4,z}$ penalize (if positive) breaking the inequality $p(z, x) \leqslant 1$. The result easily follows from Theorem 3, for details see Appendix D. ◀

3.3 Simulating high min-entropy leakage

We obtain the following extension for Theorem 14 and Theorem 11. It essentially says that not the length of leakage makes it simulatable, but *high density* in the uniform distribution.

▶ **Theorem 14** ([10], corrected)**.** *For every distribution (Z, X) on $\{0,1\}^n \times \{0,1\}^m$ and every ϵ, s, there exists a "simulator" $h : \{0,1\}^n \to \{0,1\}^m$ such that*

(a) $(Z, h(Z))$ and (Z, X) are (ϵ, s)-indistinguishable
(b) h is of complexity $s_{\mathsf{h}} = O\left(s \cdot 2^{3\ell}\epsilon^{-2}\right)$
where

$$\ell = n - \mathbf{H}_{\infty}(X|Z)$$

is the min-entropy[9] deficiency.

This result beats by over $O(\epsilon^{-2})$ the recent bound from [15].

3.4 Simulating for square-friendly applications

As the final remark, note that we have not exploited yet the constant D in Theorem 2. Indeed, small variance assumptions imposed on distinguishers appear in recent works on key derivation [1, 5, 4]. It has been shown that they are already satisfied by a large class of applications, so called square-friendly applications. With our Theorem 2 it is possible to derive simulators that gain in the complexity a factor of the variance, which could be quite large (for example, of order $\epsilon \approx 2^{-80}$ bits). We discuss this problem in another paper.

[8] We use the standard RAM model of computations as it is more convenient, the statement above for circuits can be obtained by general reductions [14].
[9] Recall, min-entropy is defined by $\mathbf{H}_{\infty}(X|Z) = \min_z(-\log \max_x \Pr[X = x|Z = z])$.

References

1 Boaz Barak, Yevgeniy Dodis, Hugo Krawczyk, Olivier Pereira, Krzysztof Pietrzak, Francois-Xavier Standaert, and Yu Yu. Leftover hash lemma, revisited. Cryptology ePrint Archive, Report 2011/088, 2011. http://eprint.iacr.org/.

2 Boaz Barak, Moritz Hardt, and Satyen Kale. The uniform hardcore lemma via approximate bregman projections. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 1193–1200, 2009.

3 Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

4 Yevgeniy Dodis, Krzysztof Pietrzak, and Daniel Wichs. Key derivation without entropy waste. In PhongQ. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 93–110. Springer Berlin Heidelberg, 2014.

5 Yevgeniy Dodis and Yu Yu. Overcoming weak expectations. In Amit Sahai, editor, *Theory of Cryptography*, volume 7785 of *Lecture Notes in Computer Science*, pages 1–22. Springer Berlin Heidelberg, 2013.

6 J. R. Giles. *Convex analysis with application in the differentiation of convex functions / John R. Giles*. Pitman Pub Boston, 1982.

7 Oded Goldreich. *Foundations of Cryptography: Volume 1*. Cambridge University Press, New York, NY, USA, 2006.

8 Johan Hastad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

9 Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *In 36th Annual Symposium on Foundations of Computer Science*, pages 538–545. IEEE, 1995.

10 Dimitar Jetchev and Krzysztof Pietrzak. How to fake auxiliary input. In Yehuda Lindell, editor, *TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, volume 8349 of *Lecture Notes in Computer Science*, pages 566–590. Springer, 2014.

11 Adam R. livans and Rocco A. Servedio. Boosting and hard-core set construction. *Machine Learning*, 51(3):217–238, 2003.

12 Krzysztof Pietrzak. A leakage-resilient mode of operation. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 462–482. Springer Berlin Heidelberg, 2009.

13 Krzysztof Pietrzak. private communication, may, 2015.

14 John E. Savage. *Models of Computation: Exploring the Power of Computing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1997.

15 Maciej Skorski. Efficiently simulating high min-entropy sources in the presence of side information. In *Progress in Cryptology - INDOCRYPT 2015 - 16th International Conference on Cryptology in India, Bangalore, India, December 6-9, 2015, Proceedings*, pages 312–325, 2015.

16 Luca Trevisan, Madhur Tulsiani, and Salil Vadhan. Regularity, boosting, and efficiently simulating every high-entropy distribution. In *Proceedings of the 2009 24th Annual IEEE Conference on Computational Complexity*, CCC '09, pages 126–136, Washington, DC, USA, 2009. IEEE Computer Society.

17 Salil Vadhan and ColinJia Zheng. A uniform min-max theorem with applications in cryptography. In Ran Canetti and JuanA. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 93–110. Springer Berlin Heidelberg, 2013.

A More on the flaw in [10]

In the original setting we have $\mathcal{Z} = \{0,1\}^\lambda$. In the proof of the claimed better bound $O\left(s \cdot 2^{3\lambda}\epsilon^{-2}\right)$ there is a mistake on page 18 (eprint version), when the authors enforce a signed measure to be a probability measure by a mass shifting argument. The number M defined there is in fact a function of x and is hard to compute, whereas the original proof amuses that this is a constant independent of x. During iterations of the boosting loop, this number is used to modify distinguishers class step by step, which drastically blows up the complexity (exponentially in the number of steps, which is already polynomial in ϵ). In the min-max based proof giving the bound $O\left(s \cdot 2^{3\lambda}\epsilon^{-4}\right)$ a fixable flaw is a missing factor of 2^λ in the complexity (page 16 in the eprint version), which is because what is constructed in the proof is only a probability mass function, not yet a sampler [13].

B Stream ciphers definitions

We start with the definition of weak pseudorandom functions, which are *computationally indistinguishable* from random functions, when queried on random inputs and fed with uniform secret key.

▶ **Definition 15** (Weak pseudorandom functions)**.** A function $\mathrm{F} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^m$ is an (ϵ, s, q)-secure weak PRF if its outputs on q random inputs are indistinguishable from random by any distinguisher of size s, that is

$$\left|\Pr\left[\mathrm{D}\left((X_i)_{i=1}^q, \mathrm{F}((K, X_i)_{i=1}^q\right) = 1\right] - \Pr\left[\mathrm{D}\left((X_i)_{i=1}^q, (R_i)_{i=1}^q\right) = 1\right]\right| \leqslant \epsilon$$

where the probability is over the choice of the random $X_i \leftarrow \{0,1\}^n$, the choice of a random key $K \leftarrow \{0,1\}^k$ and $R_i \leftarrow \{0,1\}^m$ conditioned on $R_i = R_j$ if $X_i = X_j$ for some $j < i$.

Stream ciphers generate a keystream in a recursive manner. The security requires the output stream should be indistinguishable from uniform[10].

▶ **Definition 16** (Stream ciphers)**.** A *stream-cipher* $\mathsf{SC} : \{0,1\}^k \to \{0,1\}^k \times \{0,1\}^n$ is a function that need to be initialized with a secret state $S_0 \in \{0,1\}^k$ and produces a sequence of output blocks $X_1, X_2, ...$ computed as

$$(S_i, X_i) := \mathsf{SC}(S_{i-1}).$$

A stream cipher SC is (ϵ, s, q)-secure if for all $1 \leqslant i \leqslant q$, the random variable X_i is (s, ϵ)-pseudorandom given $X_1, ..., X_{i-1}$ (the probability is also over the choice of the initial random key S_0).

Now we define the security of leakage resilient stream ciphers, which follow the "only computation leaks" assumption.

▶ **Definition 17** (Leakage-resilient stream ciphers)**.** A leakage-resilient stream-cipher is $(\epsilon, s, q, \lambda)$-secure if it is (ϵ, s, q)-secure as defined above, but where the distinguisher in the j-th round gets λ bits of arbitrary deceptively chosen leakage about the secret state accessed during this round. More precisely, before $(S_j, X_j) := \mathsf{SC}(S_{j-1})$ is computed, the distinguisher

[10] We note that in a more standard notion the entire stream $X_1, \ldots, X_q$ is indistinguishable from random. This is implied by the notion above by a standard hybrid argument, with a loss of a multiplicative factor of q in the distinguishing advantage.

can choose any leakage function f_j with range $\{0,1\}^\lambda$, and then not only get X_j, but also $\Lambda_j := f_j(\hat{S}_{j-1})$, where $\hat{S}_{j-1}$ denotes the part of the secret state that was modified (i.e., read and/or overwritten) in the computation $\mathsf{SC}(S_{j-1})$.

C Proof of Theorem 2

We use the following notational convention: for any function $C = C(p)$ we denote by C^+ its non-negative part $C^+(p) = \max(C(p), 0)$. Denote for shortness $L_j(p) = \left(C_j^+\right)^2$. According to Equation (4), the update is

$$p^{t+1} = p^t - \gamma \left(\mathrm{D}^t + 2 \sum_j \alpha_j \partial L_j \right). \tag{25}$$

Now we have

$$\begin{aligned}
\left\|p^{t+1} - p^*\right\|_2^2 &= \left\| p^t - p^* - \gamma \mathrm{D} - \gamma \sum_j \alpha_j \partial L_j \right\|_2^2 \\
&= \left\|p^t - p^*\right\|_2^2 - 2\gamma \sum_j \alpha_j \langle p^t - p^*, \partial L_j \rangle + \\
&\quad - 2\gamma \langle p^t - p^*, \mathrm{D} \rangle + 2\gamma^2 \sum_j \alpha_j \langle \mathrm{D}, \partial L_j \rangle + \\
&\quad + \gamma^2 \left\| \sum_j \alpha_j \partial L_j \right\|_2^2 + \gamma^2 \left\|\mathrm{D}\right\|_2^2
\end{aligned}$$

which by triangle inequality and the Jensen inequality $(\sum_{j\in J} u_j)^2 \leqslant |J| \cdot \sum_j u_j^2$ means that

$$\begin{aligned}
\left\|p^{t+1} - p^*\right\|_2^2 &\leqslant \left\|p^t - p^*\right\|_2^2 - 2\gamma \sum_j \alpha_j \langle p^t - p^*, \partial L_j \rangle + \\
&\quad - 2\gamma \langle p^t - p^*, \mathrm{D} \rangle + 2\gamma^2 \sum_j \alpha_j \langle \mathrm{D}, \partial L_j \rangle + \\
&\quad + \gamma^2 |J| \sum \alpha_j^2 \left\|\partial L_j\right\|_2^2 + \gamma^2 \left\|\mathrm{D}\right\|_2^2
\end{aligned}$$

Suppose that at the i-th step we still are ϵ-far from the target function p^*. That is,

$$\langle p^t - p^*, \mathrm{D}^t \rangle \geqslant \epsilon - \sum_j \alpha_j L_j. \tag{26}$$

Using this, we get

$$\begin{aligned}
\left\|p^{t+1} - p^*\right\|_2^2 &\leqslant \left\|p^t - p^*\right\|_2^2 - 2\gamma \sum_j \alpha_j \langle p^t - p^*, \partial L_j \rangle + 2\gamma^2 \sum_j \alpha_j \langle \mathrm{D}^t, \partial L_j \rangle \\
&\quad + \gamma^2 \left\|\mathrm{D}^t\right\|_2^2 + 2\gamma \sum_j \alpha_j L_j + \gamma^2 |J| \sum_j \alpha_j^2 \left\|\partial L_j\right\|_2^2 - 2\gamma\epsilon
\end{aligned} \tag{27}$$

▶ **Lemma 18.** *Let C_j be a convex function, L_j be the function defined by $L_j(p) = \left(C_j^+\right)^2$, p^* be a point such that $C_j(p^*) \leqslant 0$ and p be an arbitrary point. Then*

$$\langle p - p^*, \partial_p L_j(p) \rangle \geqslant 2L_j(p).$$

Proof of Lemma. Consider the case $L_j(p) = 0$. We have then $C_j^+(p) = 0$. Since $\partial_p L_j(p) = 2C_j^+ \partial_p C_j^+(p)$, we see that both sides are zero. Suppose now that $L_j(p^t) > 0$. In particular $C_j^+(p) > 0$. The assumption $C_j(p^*) \leqslant 0$ gives us $C_j^+(p^*) = 0$. Therefore, since C_j^+ is convex as a maximum of two convex functions: C_j and the zero function, we have

$$\begin{aligned}\langle p^t - p^*, \partial L_j(p^t)\rangle &= \langle p^t - p^*, 2C_j(p^t)\partial_{p^t} C_j^+\rangle \\ &= 2C_j(p)\langle p^t - p^*, \partial_{p^t} C_j^+\rangle \\ &\geqslant 2C_j(p)\left(C_j^+(p^t) - C_j^+(p^*)\right) \\ &= 2\left(C_j^+(p^t)\right)^2 \\ &= 2L_j(p^t)\end{aligned}$$

◀

By Theorem 18 we obtain

$$\left\|p^{t+1} - p^*\right\|_2^2 \leqslant \left\|p^t - p^*\right\|_2^2 + 2\gamma^2 \sum_j \alpha_j \langle \mathrm{D}^t, \partial L_j\rangle + \gamma^2 \left\|\mathrm{D}^t\right\|_2^2 + \\ - 2\gamma \sum_j \alpha_j L_j + \gamma^2 \sum_j \alpha_j^2 \left\|\partial L_j\right\|_2^2 - 2\gamma\epsilon. \tag{28}$$

▶ **Lemma 19.** *Let C_j be a convex function, and L_j be the function defined by $L_j(p) = \left(C_j^+\right)^2$. Then for any p*

$$\left\|\partial_p L_j(p)\right\|^2 = 4L_j(p) \cdot \left\|\partial_p C_j\right\|^2$$

Using Lemma 19 we can estimate

$$\left\|p^{t+1} - p^*\right\|_2^2 \leqslant \left\|p^t - p^*\right\|_2^2 + 2\gamma^2 \sum_j \alpha_j \langle \mathrm{D}^t, \partial L_j\rangle + \gamma^2 \left\|\mathrm{D}^t\right\|_2^2 + \\ + \sum_j \left(-2\gamma\alpha_j + 4|J|\gamma^2\alpha_j^2 \left\|\partial C_j\right\|^2\right) L_j - 2\gamma\epsilon. \tag{29}$$

where we denote $\partial C_j = \partial_{p_t} C_j(p^t)$. Now we get rid of tems $\langle \mathrm{D}^t, \partial L_j\rangle$. By the Cauchy-Schwarz Inequality, we have for any positive number θ_j

$$2\langle \mathrm{D}^t, \partial L_j\rangle \leqslant \frac{1}{\theta_j}\left\|\mathrm{D}^t\right\|^2 + \theta_j \left\|L_j\right\|^2$$

Using again Lemma 19 we finally obtain

$$\left\|p^{t+1} - p^*\right\|_2^2 \leqslant \left\|p^t - p^*\right\|_2^2 + \left(\gamma^2 + \sum_j \frac{\gamma^2\alpha_j}{\theta_j}\right)\left\|\mathrm{D}^t\right\|_2^2 + \\ + \sum_j \left(-2\gamma\alpha_j + 4|J|\gamma^2\alpha_j^2 \left\|\partial C_j\right\|^2 + \gamma^2\theta_j\alpha_j \left\|\partial C_j\right\|^2\right) L_j - 2\gamma\epsilon. \tag{30}$$

The coefficient of the term L_j is negative as long as

$$-2 + 4|J|\gamma\alpha_j M_j + \gamma\theta_j M_j < 0$$

Setting $\theta_j = \alpha_j$ and $\alpha_j = \frac{2}{(4|J|+1)M_j}$ we get

$$\left\|p^{t+1} - p^*\right\|_2^2 \leqslant \left\|p^t - p^*\right\|_2^2 + (|J|+1)\gamma^2 \left\|\mathrm{D}^t\right\|_2^2 - 2\gamma\epsilon.$$

To minimize the right-hand side we choose

$$\gamma = \frac{\epsilon}{(|J|+1)\|\mathrm{D}\|_2^2}$$

which yields

$$\left\|p^{t+1} - p^*\right\|_2^2 \leqslant \left\|p^t - p^*\right\|_2^2 - \frac{\epsilon^2}{(|J|+1)\|\mathrm{D}\|_2^2}. \tag{31}$$

From this equation we easily conclude the result.

D Proof of Theorem 14

For every $p = p(z,\cdot)$ we compute

$$\partial_p C_{1,z}(p) = [1,\ldots,1], \quad \partial_p C_{1,z}(p) = [-1,\ldots,-1]$$

where both vectors have $|\mathcal{X}| = 2^\ell$ components and that

$$-\sum_x \mathbf{1}_{p(z,x)<0}\mathbf{e}_x \in \partial_p C_{3,z}(p), \quad \sum_x \mathbf{1}_{p(z,x)>0}\mathbf{e}_x \in \partial_p C_{3,z}(p)$$

where for shortness we use the following notational convention: the vector $\mathbf{e}_x$ has 1 at position x and 0 elsewhere, and for any predicate A the function $\mathbf{1}_A$ outputs 1 on instances where it is satisfied and 0 elsewhere. In particular, since $\mathcal{X}$ has 2^ℓ elements, we have

▶ Claim 2. Calculating subgradients requires $O(2^\ell)$ extra computations in every round of Algorithm 2, given we store p_i from the previous round (for C_3 and C_4 we scan entire $\mathcal{X}$ and read all components of p_{i-1}).

It is now easy to calculate constants in Theorem 3

▶ Claim 3. We can take $D = 2^\ell$, $R = 1$, $M_1 = M_2 = 2^\ell$ and $M_3 = M_4 = 1$.

Since $|J| = 4$ by Theorem 3 we obtain $h = p(z,x)$ of complexity $O\left(2^{2\ell}\epsilon^{-2}\right)$ such that $\delta^{\mathcal{D}}(h;f) \leqslant \epsilon$, $\sum_x \max\left(p(z,x)-1,0\right) \leqslant \epsilon \quad \forall z \in \mathcal{Z}$, $\sum_x \max\left(-p(z,x)-1,0\right) \leqslant \epsilon, \quad \forall z \in \mathcal{Z}$ and $\beta(z) - \epsilon \leqslant \sum_x h(z,x) \leqslant \beta(z) + \epsilon \quad \forall z$ The constraints are almost satisfied, it suffices to apply a mass-shifting argument. Define $\theta(z,x) = 1 - h(z,x)$ if $h(z,x) > 1$ and $\theta(z,x) = -1 - h(z,x)$ if $h(z,x) < -1$. Setting $h'(z,x) = h(z,x) + \theta(z,x)$ we fix the constraints $C_{3,z} \leqslant 0$ and $C_{4,z} \leqslant 0$, the constraints $C_{1,z}$ and $C_{2,z}$ get violated at most by an extra additive term ϵ. Then we put $h''(z,x) = h'(z,x) \cdot \frac{\beta(z)}{\sum_{x'} h(z,x')}$ and get $\sum_x |h''(z,x) - h'(z,x)| \leqslant 2\epsilon$, and hence $\sum_x |h''(z,x) - h(z,x)| \leqslant 3\epsilon$ for every z. Thus, the constraints are satisfied and the computational distance is at most $\epsilon + 3\epsilon = 4\epsilon$, essentially the same up to a constant. Note that the last step can be realized with $O\left(2^\ell\right)$ operations given the vector p in the storage at the last step of the algorithm.