A Sequential Lightbulb Problem

Noah Bergam njb2154@columbia.edu
Arman Özcan ao2794@columbia.edu
Berkan Ottlik bto2106@columbia.edu
Department of Computer Science
Columbia University
New York, NY 10027, USA

Editor: Alex Andoni

Abstract

The light bulb problem aims to distinguishing correlated vectors among random vectors. In this report, we summarize existing algorithms and explore a novel sequential formulation of the light bulb problem. We provide a naive algorithm that uses linear space and a quadratic number of rounds. We use a bucketing technique, inspired by (Valiant, 2012), to achieve an algorithm that solves the sequential light bulb problem using *constant* space and $\tilde{O}(n)$ number of rounds.

Keywords: light bulb problem, learning correlations, k-juntas, fast matrix multiplication

1 Introduction

Distinguishing signal from noise is a fundamental challenge in unsupervised machine learning. In this report, we review a prototypical problem about learning correlations in the presence of noise, known as the light bulb problem.

Definition 1 (Light bulb Problem) *Given $x_1, ..., x_n \in \{\pm 1\}^d$ and $i^*, j^* \in [n]$ such that:*

- $\langle x_{i^*}, x_{j^*} \rangle \geq \rho d$.
- *For all $i \in [n] \setminus \{i^*, j^*\}$ and $l \in [d]$, $x_{i,l}$ is an i.i.d. Rademacher random variable.*

The goal is to detect (i^*, j^*) (the planted pair) from the rest (the noise vectors).

The idea here is that for large enough $d = O(\log n/\rho^2)$ (see Claim 1), we have a high-probability guarantee that the planted pair has the maximum inner product and hence is distinguishable from the noise vectors. This problem is easily solved in time $O(n^2 d)$ by finding the largest off-diagonal entry of $X^T X$ where $X = (x_1, ..., x_n)$. There are two known routes to obtaining subquadratic time algorithms: locality sensitive hashing and fast matrix multiplication. The former depends on the correlation ρ and works better for a highly correlated planted pair; the latter suite of methods is actually independent of ρ and hence works better for small values.

There is a natural (but to our knowledge, undiscussed) sequential analogue of the light bulb problem, which we formulate and study in this report.

Definition 2 (Sequential Light bulb Problem) *Every day $t \in [T]$, the learner observes $x_{i,t} \in \{\pm 1\}$ for $i \in [n]$. There exists $i^*, j^* \in [n]$ such that $\mathbb{E}(x_{i^*} \cdot x_{j^*}) = \rho$. For all other indices, $x_{i,t} \sim$ Unif(± 1) independently, such that $\mathbb{E}(y_i \cdot y_j) = 0$ for all other j (including j planted). How many days does it take for the learner to predict (i^*, j^*) with high probability?*

This of course, can be solved in a logarithmic number of days by just waiting until $T = O(\log n/\rho^2)$, storing the vectors, and then running the offline algorithm. If, however, the goal is to use limited space, say linear in n, then this approach no longer works. This is the starting point for our analysis.

The contributions of this report are as follows:

- We review the major ideas behind subquadratic-time algorithms for the light bulb problem, especially the fast matrix multiplication approaches of (Valiant, 2012) and (Alman, 2018).
- We analyze the light bulb problem in an online setting. We present a naive algorithm that uses linear space and quadratic number of rounds; we then analyze an enhanced algorithm that uses constant space and near-linear number of rounds.

We note briefly that the sequential light bulb problem is well-motivated by empirical observation. Apparently, when Leslie Valiant came up with the light bulb problem (Valiant, 1988), he was inspired in part by a psychology experiment which showed that, if you flash n light bulbs over time, all random except for a planted pair that shine together with positive correlation, a human can pick this up very quickly.

2 Background

For the sake of completeness, we first formalize the feasibility of the light bulb problem.

Claim 1 *Fix $\delta > 0$. If $d \geq \frac{1}{2}\rho^2 \log(n^2/\delta)$, then with probability $1 - \delta$, the planted pair has the maximum inner product.*

Proof Apply a Chernoff bound. Let x, y be random vectors.

$$\mathbb{P}(x \cdot y > \rho d) = \mathbb{P}\Big(\frac{1}{d}\sum_{i=1}^{d} x_i y_i > \rho\Big) \leq \exp(-\rho^2/2d) < \delta/n^2$$

Here we use the fact that $x_i y_i \in \{\pm 1\}$ is a Rademacher random variable, independent over i. By a union bound over uncorrelated pairs, we have with probability $1 - \delta$ that the correlations between any the random vectors is less than the correlation between the planted pair. For x a random vector and y a planted vector, the argument is almost identical: since y is deterministic, $x \cdot y$ is distributed as a sum of Bernoulli random variables. ■

We proceed to review the connections between the light bulb problem and other fundamental learning problems in theoretical computer science. Then we discuss the major fast matrix multiplication based methods for solving the light bulb problem.

2.1 Locality-Sensitive Hashing

The light bulb problem is easily recognized as a special case of the approximate closest pair problem. Since the planted pair has dot product $\geq \rho d$, the Hamming distance between them is $\|x_{i^*} - x_{j^*}\|_0 \leq (1-\rho) \cdot d/2$. This is seen by simply decomposing the dot product:

$$x_{i^*} \cdot x_{j^*} = \#\{\text{entries same}\} - \#\{\text{entries different}\} = d - 2d_{\mathrm{H}}(x_{i^*}, x_{j^*}) \geq \rho d \qquad (1)$$

For uncorrelated vectors, we have Hamming distance between vectors tightly concentrated around $d/2$ (easily checked using a Chernoff argument similar to that in Claim 1). So the light bulb problem reduces to distinguishing between Hamming distance $d/2$ and $(1-\rho)d/2$. This can be solved directly using a locality sensitive hash scheme (Indyk and Motwani, 1998). We recall the definition below.

Definition 3 *An (r, cr, P_1, P_2)-locality sensitive hash (LSH) family $\mathcal{H}$ mapping $\mathbb{R}^d$ to some universe U obeys the following rule:*

- $\|p - q\| \leq r \implies \mathbb{P}_h(h(p) = h(q)) \geq P_1$
- $\|p - q\| \geq cr \implies \mathbb{P}_h(h(p) = h(q)) \leq P_2$

Let $\eta = \log(P_1)/\log(P_2)$ denote the strength of the LSH gap (usually denoted ρ, but we don't want notation to overlap).

To solve the light bulb problem, we use a $(d/2, (1-\rho) \cdot d/2, P_1, P_2)$ LSH. To find the closest pair, we first apply the locality sensitive hash to every vector. Then, we iterate through the each vector and find it's nearest neighbor. Our LSH will be such that the size of the universe $|U| \asymp n^{\rho}$. Therefore, for each vector we only need to we only need to compute its distance to the $O(n^{1-\Theta(\rho)})$ points hashed to the same value. This gives an overall runtime of $O(dn^{2-\Theta(\rho)})$.

2.2 Connections to other Problem

The light bulb problem is connected to other fundamental problems in theoretical computing, aside from the approximate closest pair problem. We discuss some of these problems and sketch the reductions.

Definition 4 (Learning Parities) *We are given samples $(x_i, y_i)_{i \in [d]} \subset \{\pm 1\}^n \times \{\pm 1\}$, generated as follows: each $y_i = z \prod_{j \in S} x_{ij}$ where the "noise" $z_i \in \{\pm 1\}$ is chosen independently of x_i with $\mathbb{P}(z_i = 1) = 1 - \eta$. How many samples n are needed to predict the "parity set" S with high probability?*

If $\eta = 0$, then the problem has a very simple poly-time algorithm: convert to $\mathbb{F}_2$ (send $1 \to 0$ and $-1 \to 1$). Then solve the linear equation $r \cdot x = y$ (where r is the unknown indicator of S) using Gaussian elimination. Once $\eta > 0$, the problem exhibits a computational-statistical gap: efficient algorithms are hard to develop, though a straightforward Chernoff bound ensures that with $m = O(n)$ samples, the planted parity set S is the only set that satisfies all the examples.

The light bulb problem is connected to *sparse parities,* i.e. when the parity set S is of size $k \ll n$ (in this case, we seek to improve on the poly-time brute force over $\binom{n}{k} = O(n^k)$ size-k subsets). This connection is most easily seen for $k = 2$, $S = \{j_1^*, j_2^*\} \subset [n]$. Given such an instance of learning parity with noise, if one removes all examples where $y_i = 1$, then indeed this becomes the light bulb problem over $\{x_{*,j} : j \in [n]\} \subset \{\pm 1\}^d$ (i.e. the j-indices of the vectors) where $x_{j_1^*}$ and $x_{j_2^*}$ have $\rho = 1 - 2\eta$ correlation.

The problems are in fact equivalent for general small k, in that reductions go both ways. See (Valiant, 2012) for more details. There is also a known equivalency due to (Feldman et al., 2006) between learning parties and learning k-juntas, another classic problem in TCS which we define below.

Definition 5 (Learning k-juntas) *A k-junta is a Boolean function $f : \{\pm 1\}^n \mapsto \{\pm 1\}$ which only depends on $k \ll n$ of the inputs, indexed by $S \subset [n]$. How many samples $(x_i, f(x_i))$ with $x_i \sim \text{Unif}(\{\pm 1\}^n)$ are needed to learn S?*

2.3 Matrix Multiplication Approaches

(Valiant, 2012) was the first to use fast matrix multiplication to solve the light bulb problem. There are a few broad motivations for why fast matrix multiplication could help:

- The light bulb problem is clearly a special case of the approximate closest pair problem, but it has more structure to be exploited. Therefore, even if LSH were optimal for approximate closest pairs (which is unknown), it seems unlikely that LSH would be optimal for light bulb.

- The brute-force solution to the light bulb problem is a naive matrix multiplication.

- There are known hardness results for statistical query (SQ) learning for the learning parities problem, whose sparse version is equivalent to the light bulb problem. Fast matrix multiplication is a "highly non-SQ" algorithm, in the sense that it mixes information in a non-localized way.

	Runtime	Key Idea
Brute Force	$O(n^2 d)$	
LSH	$O(n^{2-O(\rho)} d)$	
(Valiant, 2012)	$O(n^{1.615} + nd)$	expand, aggregate, fast matrix mult.
(Karppa et al., 2018)	$O(n^{1.582} + nd)$	simultaneously expand and aggregate
(Alman, 2018)	$O(n^{1.582} + nd)$	polynomial method, easily derandomized

Table 1: A snapshot of the main algorithms for the light bulb problem.

Since Greg Valiant's breakthrough result, the application of fast matrix multiplication to this problem has been refined and derandomized. We sample this line of work in the following few sections.

2.3.1 Valiant's Algorithm

The first key observation for Valiant's algorithm is how matrix multiplication naturally arises in the light bulb problem. If you package $X = (x_1, ..., x_n)$, the brute-force approach to the problem is simply a matter of computing $X^\top X$ and isolating its largest off-diagonal entry. The problem, of course, is that this matrix multiplication takes $\Omega(n^2)$ even with fast matrix multiplication. We can reap the benefits of fast matrix multiplication if we somehow condense the matrix, and search for correlations between *groups of vectors.* This gives rise to the following aggregation construction.

Definition 6 *Aggregate the columns of X into $n^{1-\alpha}$ groups of size n^α; call these $G_1, ..., G_{n^{1-\alpha}}$. Define, for $k \in [n^{1-\alpha}]$,*

$$z_k = \sum_{l \in G_k} x_l$$

*We call $Z = (z_1, ..., z_{n^{1-\alpha}}) \in \mathbb{R}^{d \times n^{1-\alpha}}$ the α-**aggregation of** X.*

Observe that, with probability $1 - o(1)$, x_{i^*} and x_{j^*} lie in separate groups. This is because the probability of any two vectors landing in the same group is $\approx (n^\alpha/n)^2 \to 0$ as $n \to \infty$. We note that this aggregation preserves the signal but decays its strength. The following lemma makes this more precise.

Lemma 1 *$\mathbb{E}\langle z_{i'}, z_{j'}\rangle = \rho d$ if $G_{i'}, G_{j'}$ split the planted pair; otherwise $\mathbb{E}\langle z_{i'}, z_{j'}\rangle = 0$. In general, $\mathbb{E}\langle z_{i'}, z_{j'}\rangle^2 = O(n^{2\alpha} d)$.*

Proof It suffices to consider the variance term where both of the aggregated vectors contain completely independent entries (if the planted pair is inside, it only effects one entry).

$$\begin{aligned}
\mathbb{E}\langle z_{i'}, z_{j'}\rangle^2 &= \mathbb{E}\Big(\sum_{k=1}^{d}\Big(\sum_{l \in G_{i'}} x_{lk}\Big)\Big(\sum_{r \in G_{j'}} x_{rk}\Big)\Big)^2 \\
&= \mathbb{E}\Big(\sum_{k=1}^{d}\Big(\sum_{l \in G_{i'}}\sum_{r \in G_{j'}} x_{rk}x_{lk}\Big)\Big)^2 \\
&= \mathbb{E}\Big(\sum_{k,k'=1}^{d}\sum_{l,l' \in G_{i'}}\sum_{r,r' \in G_{j'}} x_{rk}x_{rk'}x_{lk}x_{lk'}x_{r'k}x_{r'k'}x_{l'k}x_{l'k'}\Big) \\
\text{(independence of } x\text{'s)} \quad &= \sum_{k=1}^{d}\sum_{l \in G_{i'}}\sum_{r \in G_{j'}} (x_{rk}x_{lk})^2 = d \cdot n^\alpha \cdot n^\alpha = dn^{2\alpha}
\end{aligned}$$

$\mathbb{E}\langle z_{i'}, z_{j'}\rangle = \rho d \cdot \mathbf{1}(i^* \in G_{i'}, j^* \in G_{j'})$ is clear enough (apply linearity of expectation). ■

Therefore, as long as $\rho d \gg \sqrt{n^{2\alpha} d}$, equivalently $d \gg n^{2\alpha}/\rho^2$, we can find the correlated columns of Z by isolating the largest off diagonal entry of $Z^\top Z$. This matrix multiplication takes time $n^{(1-\alpha)\omega}\text{poly}(1/\rho)$, where ω is the exponent of matrix multiplication (note that the use of fast matrix multiplication is crucial here; for $\omega = 3$ we do not reap benefits

from this approach). Once we know $G'_i, G_{j'}$ that split the planted pair, we can find nearest neighbors between elements in G'_i and G'_j in time $O(n^{2\alpha})$.

The problem with this approach is that is requires a much larger dimension than the $d = O(\log n/\rho^2)$ dimension that is information-theoretically necessary. The last key ingredient to Valiant's algorithm is the so-called "XOR/tensor embedding," which amplifies the dimension (at the expense of reducing the correlation of the planted pair) such that the aggregation method concentrates correctly.

The key motivation here is a certain identity relating the tensor power of a vector with inner products. Recall that, given a vector $x \in \mathbb{R}^d$, its nth tensor power (also known as Kronecker product) $x^{\otimes n} \in \mathbb{R}^{d^n}$ is such that $x_E^{\otimes n} = \prod_{i \in E} x_i$ for E is a size n multiset of $[d]$ (of which there are d^n). So the tensor power blows up dimension. Next, we check how it affects inner products:

$$\begin{aligned}\langle x^{\otimes n}, y^{\otimes n}\rangle &= \sum_E \prod_{i\in E} x_i y_i \\ &= \sum_{i_1=1}^{d} \cdots \sum_{i_n=1}^{d} x_{i_1} y_{i_1} \cdots x_{i_n} y_{i_n} \\ &= \Big(\sum_{i_1=1}^{d} x_{i_1} y_{i_1}\Big) \cdots \Big(\sum_{i_n=1}^{d} x_{i_n} y_{i_n}\Big) = \langle x, y\rangle^n\end{aligned}$$

The goal in Valiant's paper is to calculate an approximate tensor power via random sampling: so we have the benefits of the tensor embedding, without . Let us define this construction more carefully.

Definition 7 *Take $X = (x_1, ..., x_n) \in \mathbb{R}^{d\times n}$ and sample m multisets of length q uniformly and independently, denoted as $E_1, ..., E_m$. **The** (m, q)**-approximate tensor embedding** $Z = (z_1, ..., z_{n^{1-\alpha}}) \in \mathbb{R}^{m\times n^{1-\alpha}}$ is defined as follows:*

$$z_{i,j} = \prod_{l\in E_j} x_{i,l}$$

In expectation, the approximate tensor embedding has a very similar correlation decay feature as $x^{\otimes q}$.

$$\mathbb{E}\langle z_i, z_j\rangle = \mathbb{E}\sum_{k=1}^{m} z_{ik} z_{jk} = \sum_{k=1}^{m} \mathbb{E}\prod_{l\in E_k} x_{il} x_{jl} = m \cdot \rho^q$$

Note that for $m = d^q$ this corresponds precisely to the original tensor product. Recall that we want $m = O(n^{2\alpha})$, so for $d = O(\log m)$ and $q = o(\log n)$ we succeed on this front.

To summarize: modulo some details about concentration and elementary amplification techniques, Valiant's algorithm proceeds by: expanding vectors using the XOR embedding, randomly aggregating vectors, and then running fast matrix multiplication on the aggregated matrix.

2.3.2 Karppa, Kaski, and Kohonen's Algorithm

Karppa et al. (2018)'s algorithm is almost identical to that of Valiant (2012), except it expands and aggregates vectors simultaneously by replacing the uniform random sampling of dimensions in the (m, q)-approximate tensor embedding with *Cartesian product sampling*.

First aggregate the columns of X into $n^{1-\alpha}$ groups of size n^α; call these $G_1, ..., G_{n^{1-\alpha}}$. Draw independently and uniformly at random two $m^{1/2}$-tuples of elements of $[d^{q/2}]$ and let I_1 and I_2 be the two resulting multi-sets of size $m^{1/2}$. Let $I = I_1 \times I_2$ be the Cartesian product of size m with elements that are q tuples in $[d]^q$. We wish to efficiently compute the compressed matrix $Z \in \mathbb{R}^{m \times n^{1-\alpha}}$ consisting of elements

$$z_{\vec{i},k} = \sum_{g \in G_k} (x_g^{\otimes n})_{\vec{i}}$$

where $(x_g^{\otimes n})_{\vec{i}}$ denotes the product of the entries of $x_g^{\otimes n}$ in $\vec{i} \in I$.

$$z_{\vec{i},k} = z_{(\vec{i}_1, \vec{i}_2),k} = \sum_{g \in G_k} (x_g^{\otimes n/2})_{\vec{i}_1} (x_g^{\otimes n/2})_{\vec{i}_2}.$$

Define $L_k \in \mathbb{R}^{m^{1/2} \times n^\alpha}$ and $R_k \in \mathbb{R}^{m^{1/2} \times n^\alpha}$ as $l_{\vec{i}_1,g} = (x_g^{\otimes n/2})_{\vec{i}_1}$ and $r_{\vec{i}_2,g} = (x_g^{\otimes n/2})_{\vec{i}_2}$ respectively. Observe

$$(L_k R_k^\top)_{\vec{i}_1, \vec{i}_2} = \sum_{g \in G_k} l_{\vec{i}_1,g} r_{\vec{i}_2,g} = \sum_{g \in G_k} (x_g^{\otimes n/2})_{\vec{i}_1} (x_g^{\otimes n/2})_{\vec{i}_2} = z_{\vec{i},k}.$$

Therefore, to compute Z, for each $k \in n^{1-\alpha}$, we construct each entry of L_k and R_k and compute the product $L_k R_k^\top \in \mathbb{R}^{m^{1/2} \times m^{1/2}}$. This costs us $n^{1-\alpha}$ matrix multiplications with inner dimension n^α and outer dimension $m^{1/2}$. Valiant's algorithm requires time $O(qmn^{1-\alpha})$. This improvement enables Karppa et al. (2018) to achieve better performance trade-offs with other steps of the algorithm and improve the overall runtime.

2.3.3 Alman's Algorithm

Like the aforementioned algorithms, Alman's proceeds by amplifying vectors in some sense, aggregating them into groups, running fast matrix multiplication to detect planted groups, and then running a brute force search over the planted groups. The key innovation here is in the amplification step: it is best understood in terms of $\mathbb{F}_2$ polynomials.

- Let $G_1, ..., G_{n^{1-\alpha}}$ be the random partition into groups of size n^α.
- Define the following polynomial:

  $$C_{ij} = \sum_{x \in G_i} \sum_{y \in G_j} a_x a_y p(x_1 y_1, ..., x_d y_d)$$

  where $a_x, a_y \sim \{-1, 1\}$ independently and uniformly at random, and:

  $$p(z_1, ..., z_n) = \Big(\sum_{i=1}^{n} z_i \Big)^r$$

  The claim is that if C_{ij} exceeds some judiciously-chosen threshold

Since the inputs of the polynomial belong to ± 1, it can be understood as a polynomial over $\mathbb{F}_2$. Because of this, p is equivalent to its *multilinearization*: this is obtained by expressing the polynomial as a sum of monomials and then reducing every exponent modulo 2, such that it becomes linear in all variables. Furthermore, since p is a degree r polynomial, we can rewrite it in a basis consisting of r-sized subsets of $z = (z_1, ..., z_n)$. Index this basis by $M_1, ..., M_t$ for $t = \binom{d}{\leq r}$. We can rewrite $p(z) = \sum_{s=1}^{t} c_s z_{M_s}$ (and compute this decomposition efficiently, using a procedure discussed further in Alman (2018)). Since $z = x \odot y$ we can decompose this sum quite nicely.

Let $\alpha = 1/3$. So we have $m = n^{2/3}$ groups.

$$\begin{aligned} C_{ij} &= \sum_{x \in G_i} \sum_{y \in G_j} a_x a_y \sum_{s=1}^{f} c_s (x \odot y)_{M_s} \\ &= \sum_{x \in G_i} \sum_{y \in G_j} a_x a_y \sum_{s=1}^{f} c_s x_{M_s} y_{M_s} \\ &= \sum_{s=1}^{t} \Big[\underbrace{c_s \Big(\sum_{x \in G_i} a_x x_{M_s} \Big)}_{B_{i,s}} \underbrace{\Big(\sum_{y \in G_j} a_y y_{M_s} \Big)}_{A_{i,s}} \Big] \end{aligned}$$

Hence, $C = AB^T$ is the matrix of values we need. For small enough r, the $n^{2/3}$ dimension of the matrix multiplication dominates, and we achieve runtime $O(n^{2\omega/3+\epsilon})$ for arbitrarily small ϵ.

A priori, computing the entries of A and B would take time $\Omega(n^{2/3} \cdot n^{1/3} \cdot t) = \Omega(n^{5/3})$ time and hence be the bottleneck. They employ the following trick, found in (Karppa et al., 2018), to get time below $O(n^{2\omega/3+\epsilon})$

- Let $N_1, ..., N_u$ be an enumeration of subsets of size $\leq r/2$.
  Let $L^i_{s,x} = x_{N_s}$ and $\tilde{L}^i_{s,x} = a^x x_{N_s}$.
- Compute $P^i = L^i \tilde{L^i}^T$ where $P^i_{s,s'} = \sum_{x \in G_i} a^x x_{N_s \Delta N_{s'}}$, where Δ denotes symmetric difference.
- Every set of size r can be realized as such a symmetric difference; thus each element $A_{i,s}$ can be found as an element of P^i.
- With $u = O(n^{1/3+\epsilon})$, computing the entries of L^i takes $O(mugr) = \tilde{O}(n^{4/3+\epsilon})$ while computing P^i takes $O(m \cdot \max(u, g)^\omega) = O(n^{(2+\omega)/3+\epsilon})$

2.3.4 Generalized Tensor Method

In the paper "Generalizations of Matrix Multiplication can solve the Light Bulb Problem", Josh Alman and Hengjie Zhang proposed a new approach to designing faster algorithms for the light bulb problem by replacing fast matrix multiplication with other tensors that are generalizations of matrix multiplication, which may contain only some terms of matrix multiplication and some additional error terms but are computed faster than full matrix

multiplication. (Alman and Zhang, 2023) They also showed how hashing methods can be combined with their fast matrix multiplication approach to design even faster algorithms: while previous matrix multiplication-based algorithms for the light bulb problem have the same running time regardless of how large $\rho > 0$ is, their new approach yields algorithms which are faster as ρ gets larger. To state their main result, one must first define some terms. Here is a basic linear-algebraic definition of a tensor.

Definition 8 *For a vector space V over a field $\mathbb{F}$, a **tensor** $f : V^k \to \mathbb{F}$ is a k-linear map, i.e. an element of the dual $(V^k)^*$. It can therefore be written as follows:*

$$f(x_1, ..., v_n) = \sum_{i_1,...,i_n} A_{i_1,...,i_n} [v_1]_{i_1} \cdots [v_n]_{i_n}$$

A matrix is a rank-2 tensor (or, more precisely, they are in one-to-one correspondence[1]). The easiest way to see this is through the quadratic form. For $M \in \mathbb{R}^{n \times m}$, $x \in \mathbb{R}^n$, and $y \in \mathbb{R}^m$, M generates the following bilinear map:

$$f_M(x, y) = x^T M y = \sum_{i \in [n]} \sum_{j \in [m]} M_{ij} x_i y_j$$

The tensor is determined by the coefficients in the summation. Hence, an alternative way to view tensors is as formal polynomials where the multiplicity of any variable in a term is at most 1. One can think of the formal variables as basis elements in the tensor product vector space. In the above example, we can see $x_i \cdot y_j$ as $(x \otimes y)_{ij}$.

Definition 9 *One can define a three-dimensional tensor T to represent bilinear problems that take as input a $q_i \times q_k$ matrix X and a $q_j \times q_k$ matrix Y, and output a $q_i \times q_j$ matrix Z as follows:*

$$T = \sum_{i,i' \in [q_i], j,j' \in [q_j], k,k' \in [q_k]} T\left(\mathrm{X}_{i,k} \mathrm{Y}_{j,k'} \mathrm{Z}_{i',j'}\right) \cdot \mathrm{X}_{i,k} \mathrm{Y}_{j,k'} \mathrm{Z}_{i',j'}$$

where $T\left(\mathrm{X}_{i,k} \mathrm{Y}_{j,k'} \mathrm{Z}_{i',j'}\right) \in \mathbb{R}$ is the coefficient of $\mathrm{X}_{i,k} \mathrm{Y}_{j,k'}$ in the bilinear polynomial we output in entry $\mathrm{Z}_{i',j'}$.

Next, we will define two key properties of a tensor T: rank and efficacy. The latter is a new property originally introduced by the authors.

Definition 10 *A tensor T has **rank** 1 if it can be written in the form*

$$T = \left(\sum_{i \in [q_i], k \in [q_k]} \alpha_{i,k} \mathrm{X}_{i,k} \right) \left(\sum_{j \in [q_j], k \in [q_k]} \beta_{j,k} \mathrm{Y}_{j,k} \right) \left(\sum_{i \in [q_i], j \in [q_j]} \gamma_{i,j} \mathrm{Z}_{i,j} \right)$$

for coefficients $\alpha_{i,k}, \beta_{j,k}, \gamma_{i,j} \in \mathbb{R}$. More generally, the rank *of T, denoted* $\mathrm{rank}(T)$*, is the minimum nonnegative integer k such that there are rank 1 tensors $T_1, \ldots, T_k$ with $T = T_1 + \cdots + T_k$.*

1. Indeed, they are group-isomorphic, where matrix multiplication corresponds to a certain composition of tensors.

Rank is a measure of the complexity of a tensor. Upper bounding rank for tensors yield algorithms for applying that tensor to matrices. For example, Strassen showed in 1969 that the rank of the tensor $\langle 2,2,2\rangle$ for multiplying two 2×2 matrices is at most 7, and that one can multiply $n\times n$ matrices in time $O\left(n^{\log_2 7}\right)$.

Definition 11 *For $i\in[q_i]$ and $j\in[q_j]$, the efficacy of T at (i,j) is defined as:*

$$\underset{i,j}{\text{eff}}(T) := \frac{\sum_{k\in[q_k]} T\left(\mathrm{X}_{i,k}\mathrm{Y}_{j,k}\mathrm{Z}_{i,j}\right)}{\sqrt{\sum_{i'\in[q_i],j'\in[q_j],k,k'\in[q_k]} T\left(\mathrm{X}_{i',k}\mathrm{Y}_{j',k'}\mathrm{Z}_{i,j}\right)^2}}.$$

$\text{eff}_{i,j}(T)$ *is essentially the ratio of the 'signal' and the 'noise' of T for computing the (i,j) output entry of matrix multiplication. The efficacy of the whole tensor T is then defined as the ℓ_2 norm of the efficacies of all its output entries:*

$$\text{eff}(T) := \sqrt{\sum_{i\in[q_i],j\in[q_j]} \left(\text{eff}_{i,j}(T)\right)^2}.$$

The efficacy of a tensor T turns out to be an important measure for how useful T is for solving the light bulb problem, as stated in the main theorem below.

Theorem 1 *For any tensor T, if*

$$\frac{\log(\text{rank}(T))}{\log(\text{eff}(T))} < \frac{2\omega}{3}$$

then

$$\omega_\ell < \frac{2\omega}{3}.$$

where w is the smallest real number such that for any $\epsilon>0$, one can multiply $n\times n$ matrices over a field using $O(nw+\epsilon)$ field operations and w_l is similarly the smallest real number such that for any $\epsilon>0$, one can solve the light bulb problem with n vectors, for any constant correlation $\rho>0$ in time $O(n^{w_l+\epsilon})$. If T has negligible aggregation time, then

$$\omega_\ell < \frac{\log(\text{rank}(T))}{\log(\text{eff}(T))}.$$

The aggregation time assumption in the above theorem refers to the initial aggregation step that is necessary in all matrix multiplication-based light bulb algorithms including their algorithm. However, this aggregation time can be made negligible, at least for the tensors that the authors studied, through a technique introduced by (Karppa et al., 2018).

This result gives a way to use an algorithm for almost any bilinear problem T to solve the light bulb problem, even if T only computes some of the terms of matrix multiplication, and if T also computes other 'noise' terms. Hence, as long as T is easy to compute ($\text{rank}(T)$ is small) and it has a high ratio of signal to noise for computing matrix multiplication (eff (T) is large), one can use it to design a fast algorithm for the light bulb computation problem.

Algorithm Description Although their algorithm involves a number of subtle details, especially in the case when the tensor T is not 'symmetric enough', at a high level it follows the simple idea of bucketing vectors into groups, as first introduced in Valiant's paper. Taking input vectors $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ and a tensor T, to find the correlated pair (y_{j^*}, y_{j^*}), first it makes many copies of each input vector. Then, all these copies for n different x inputs are mapped into one of q_i independently random buckets, and all the copies for n different y inputs into one of q_j independently random buckets. (Note that q_i and q_j are the parameters for the size of the input tensor T, but they can be assumed equal $q_i = q_j = q_k = q$ for simplicity.) By aggregating the vectors in each bucket, the matrices A and B are formed whose rows are then multiplied by a random sign. Then, the tensor T is applied to A and $B^\intercal$ to 'multiply' them to get the matrix C. This procedure is repeated $100 \log n$ times with fresh inputs to get $C_1 \ldots C_{100 \log n}$. Finally, if $C_k[i, j]$ is larger than some optimized threshold for at least $20 \log n$ different $k \in [100 \log n]$ for some pair (i, j), that pair is reported as the correlated one. As their analysis shows, if the correlated pair were mapped into buckets $i \in [q_i]$ and $j \in [q_j]$, then the algorithm succeeds as long as $\text{eff}_{i,j}(T)$ is large enough, which happens with high probability as i and j are picked uniformly randomly.

Locality-Sensitive Hashing and Tensor $\mathbf{T_{2112}}$ As the tensor T is faster to compute than full matrix multiplication, the algorithm is already fast. However, the authors improved the algorithm even more by combining it with locality-sensitive hashing. The main idea is to place the vectors into buckets using (a variation on) bit sampling locality-sensitive hashing, instead of uniformly random hashing. Thinking of the buckets as $\{1, 1\}$ bit strings, the planted pair with correlation $\rho > 0$ will be likely put into buckets $i \in \{1, 1\}^{\log_2 q_i}$ and $j \in \{1, 1\}^{\log_2 q_j}$ which also have correlation close to ρ. If these buckets have larger $\text{eff}_{i,j}(T)$ than uniformly random buckets, then one can speed up the algorithm.

In fact, the new tensor the authors propose in the paper, named T_{2112}, has this property. This tensor is defined as:

$$\begin{aligned} T_{2112} = &(X_{1,1}Y_{1,1} + X_{1,2}Y_{2,1} + O(\epsilon))Z_{1,1} + (X_{1,2}Y_{2,2} + O(\epsilon))Z_{2,1} \\ &+(X_{2,1}Y_{1,1} + O(\epsilon))Z_{1,2} + (X_{2,1}Y_{1,2} + X_{2,2}Y_{2,2} + O(\epsilon))Z_{2,2} \end{aligned}$$

As the authors showed, this tensor T_{2112} has rank 5 and efficacy $\sqrt{6} - O(\epsilon^2)$, which makes it ideal to use. [2] The other nice property of this tensor is that the efficacy of the buckets of correlated pairs increase with ρ, resulting in a faster algorithm. As for the analysis, after taking a Kronecker power of T_{2112} (so that $q_i = q_j = q$ are larger than 2), the resulting tensor has the property that, for buckets $i, j \in \{-1, 1\}^{\log_2(q)}$ with correlation ρ, we have $\text{eff}_{i,j}\left(T_{2112}^{\otimes \log_2(q)}\right) = 2^{(1+\rho)(\log_2(q)/2)}$, whereas the median pair i, j of buckets only has $\text{eff}_{i,j}\left(T_{2112}^{\otimes \log_2(q)}\right) = 2^{\log_2(q)/2}$.

Running the algorithm with the tensor T_{2112} and with uniformly random hashing leads to the runtime $O(1.797)$ for lightbulb problem as ϵ in T_{2112} goes to infinity, as Theorem 1

2. Technically, the border rank of T_{2112} is 5: As ϵ approaches 0 in T_{2112}, this 'border rank' of the 6 of the 8 terms becomes 5. However, since border rank bounds can be used instead of rank using the technique introduced by Bini (Bini, 1980), one can just define rank of T_{2112} as 5

suggests. The algorithm gets even faster after applying a locality-sensitive hashing scheme, as stated in the following theorem.

Theorem 2 *For the tensor T_{2112}, locality-sensitive hashing method improves the bound of Theorem 1 to*

$$\omega_\ell \leq \begin{cases} \frac{2\log 5}{\log(6(1-\rho))^{\rho/2}(1+\rho)^{\rho/2}(1-\rho^2)^{1/2}\big)} & \text{when } \rho < 1/3 \\ \frac{4\log 5}{(5+\rho)\log 2} & \text{when } 1/3 \leq \rho \leq 1 \end{cases}$$

which shows that the runtime can be made better as ρ gets larger.

3 Sequential Lightbulb Problem

In this section, we study the lightbulb problem in a sequential setting. This is arguably a more natural notion of the problem, when we consider the motivation. Since this is new material, our analysis is more detailed.

Recall that, in the sequential lightbulb problem, are $[n]$ lightbulbs, i.e. sequences of Rademacher random variables $Y_i = (y_{t,i})_{t\in\mathbb{N}}$ that are independent for each t. There exists i^*, j^* such that $\mathbb{E}(y_{t,i^*}y_{t,j^*}) = \rho$ for each t. For all other indices, $y_{i,t} \sim \text{Unif}(\pm 1)$ independently. So $\mathbb{E}(y_{t,i}y_{t,j}) = 0$ for all other pairs (even where one is planted).

3.1 Naive Algorithm

The simplest approach to the sequential lightbulb problem is to keep a counter for each lightbulb, and increment (or decrement) that counter based on how many vectors it matches in that round. We expect the counters for the planted pairs to grow at a faster rate than those for the random vectors.

- Learner maintains n counters for each lightbulb, $c_1 = (c_{1,1}, ..., c_{1,n}) = \vec{0}$.
- Every day $t = 1, 2, 3, \ldots, T$:
  - Learner observes $y_{i,t} \in \{\pm 1\}$ for all $i \in [n]$.
  - For each i, update $c_{t+1,i} = c_{t,i} + \sum_{j=1,j\neq i}^{n} y_{t,i}y_{t,j}$
- Learner predicts $(\hat{i}, \hat{j})$ to be the indices with the highest absolute-value counters.

Lemma 2 *If $T = \Omega(n^2/\rho^2)$ then the algorithm succeeds with high probability.*

Proof Observe that the counter becomes a

$$\begin{aligned} c_{T,i} &= \sum_{t=1}^{T}\sum_{j=1,j\neq i}^{n} y_{t,i}y_{t,j} \\ &= \sum_{j=1,j\neq i}^{n}\sum_{t=1}^{T} y_{t,i}y_{t,j} = \sum_{j=1,j\neq i}^{n} \langle y_i, y_j\rangle \end{aligned}$$

Therefore, $\mathbb{E}[c_{T,i}] = 0$ for all non-planted light bulbs i and $\mathbb{E}[c_{T,i}] = \rho T$ for the planted light bulbs i. More succintly, $\mathbb{E}[c_{T,i}] = \rho T \cdot \mathbf{1}(i \in \{i^*, j^*\})$.

To compute the variance, one could think of c_{t_i} as the position of a simple random walker, whose each step for some j and t being $y_{t,i}y_{t,j}$, which is +1 with probability 1/2 and -1 with probability 1/2 as long as i or j is not planted. Therefore, for any non-planted lightbulb i, $c_{T,i}$ is the position of the random walker after $(N-1)T$ random symmetric steps. Denoting each independent step as X_k for $k \in [(N-1)T]$, the variance for $c_{T,i}$ for non-planted lightbulb i is given by:

$$\begin{aligned}
\mathrm{Var}[c_{T,i}] = \mathbb{E}[c_{T,i}^2] - \mathbb{E}[c_{T,i}]^2 = \mathbb{E}[c_{T,i}^2] &= \mathbb{E}\left[\left(\sum_{k=1}^{(N-1)T} X_k\right)^2\right] \\
&= \mathbb{E}\left[\sum_{k=1}^{(N-1)T}\sum_{l=1}^{(N-1)T} X_k X_l\right] \\
&= \sum_{k=1}^{(N-1)T}\sum_{l=1}^{(N-1)T} \mathbb{E}\left[X_k X_l\right] \\
&= (n-1)T + \sum_{k\neq l} \mathbb{E}\left[X_k X_l\right] = (n-1)T < nT
\end{aligned}$$

For planted c_{T,i^*} and c_{T,j^*}, at every step t, there are now $n-2$ random symmetric steps and one positive expectation (ρ) step, $y_{t,i^*}y_{t,j^*}$. Note that

$$\mathbb{E}(y_{t,i^*}y_{t,j^*}) = \Pr[y_{t,i^*}y_{t,j^*} = 1] - \Pr[y_{t,i^*}y_{t,j^*} = -1] = 2\Pr[y_{t,i^*}y_{t,j^*} = 1] - 1 = \rho$$

Therefore, $\Pr[y_{t,i^*}y_{t,j^*} = 1] = \frac{\rho+1}{2}$ for all t and recall that each $y_{t,i^*}y_{t,j^*}$ for different t are independent. Then, let $X_1 = y_{t_1,i^*}y_{t_1,j^*}$ and $X_2 = y_{t_2,i^*}y_{t_2,j^*}$ for some t_1 and t_2. Then,

$$\begin{aligned}
\mathbb{E}\left[X_1 X_2\right] &= \left(\frac{\rho+1}{2}\right)^2 + \left(\frac{1-\rho}{2}\right)^2 - 2\left(\frac{\rho+1}{2}\right)\left(\frac{1-\rho}{2}\right) \\
&= \frac{\rho^2+2\rho+1}{4} + \frac{1-2\rho+\rho^2}{4} + \frac{2\rho^2-2}{4} = \frac{4\rho^2}{4} = \rho^2
\end{aligned}$$

Therefore, using the same notation as above, the variance for c_{T,i^*} or c_{T,j^*} is:

$$\begin{aligned}
\mathrm{Var}[c_{T,i^*}] = \mathbb{E}[c_{T,i^*}^2] - \mathbb{E}[c_{T,i^*}]^2 = \mathbb{E}[c_{T,i}^2] - \rho^2T^2 &= \sum_{k=1}^{(N-1)T}\sum_{l=1}^{(N-1)T} \mathbb{E}\left[X_k X_l\right] - \rho^2T^2 \\
&= (n-1)T + \sum_{k\neq l} \mathbb{E}\left[X_k X_l\right] - \rho^2T^2 \\
&= (n-1)T + \rho^2T(T-1) - \rho^2T^2 \\
&= (n-1)T - \rho^2T < nT
\end{aligned}$$

So if $T \gg n^2/\rho^2$, then with high probability, the c_{T,i^*} or c_{T,j^*} will be largest. More explicitly, by Chebyshev, for i unplanted:

$$\mathbb{P}(|c_{T,i}| \geq \rho T/2) \leq \frac{4nT}{\rho^2T^2} = \frac{4n}{\rho^2T}$$

With a union bound, we have:

$$\mathbb{P}(\exists i \in [n], \quad |c_{T,i}| \geq \rho T/2) \leq \frac{4n^2}{\rho^2 T}$$

For $i = i^*$ planted, we have:

$$\mathbb{P}(|c_{T,i^*}| \leq \rho T/2) \leq \mathbb{P}\Big(\Big||c_{T,i^*}| - \rho T\Big| \geq \rho T/2\Big) \leq \frac{4n}{\rho^2 T}$$

■

Setting $T = \Theta\left(n^2/\rho^2\right)$, the algorithm will correctly find the correlated pair with a desired small constant probability.

3.2 Recursive Bucketing (Process of Elimination) Algorithm

In order to mitigate the number of rounds, we could bucket the vectors and then, after a certain number of rounds, narrow down the buckets of interest. This allows us to achieve sub-quadratic number of rounds and still linear space.

- Every epoch $s = 1, ..., S$:
  - Let L_s be the set of remaining light bulbs with $|L_s| = n_s$.
    Split randomly into $m_s = n_s^{1-\alpha}$ groups of size n_s^α, call them $G_1, ..., G_{m_s}$.
    Initialize m_s counters, $c_1 = (c_{1,1}, ..., c_{1,m_s}) = \vec{0}$.
  - Every day $t = 1, .., T_s$:
    - Learner observes $y_{t,i} \in \{\pm 1\}$ for $i \in [n]$.
    - For each $j \in [m_s]$, update:

$$c_{t+1,j} = c_{t,j} + \sum_{l \in G_j} \sum_{k \in [m_s] \setminus \{j\}} \sum_{r \in G_k} y_{t,l} y_{t,r}$$

  - Take c_{T_s,j_1}, c_{T_s,j_2} the largest counters.
    Let $L_{s+1} = G_{j_1} \cup G_{j_2}$ and start next epoch.

Claim 2 *After every epoch, if we start with n_s lightbulbs, we end with $2n_s^\alpha$ lightbulbs. So if we start with n lightbulbs, then $n_s = 2^{1/(1-\alpha)} n^{\alpha^s}$. After $S = O(\frac{\log\log(n)}{\log(1/\alpha)})$ rounds, we reduce to a constant number of lightbulbs, at which point a constant number of rounds of the basic algorithm reveals the planted pair (assuming the planted pair was not lost).*

Claim 3 *With each epoch, the probability of placing the lightbulbs in separate groups is $1 - (n_s^\alpha/n_s)^2 = 1 - (1/n_s^{2-2\alpha})$.*

Claim 4 *After every epoch, with $T_s \geq \Omega(n_s^{2-\alpha}/\rho^2)$ and total runtime $T_s \cdot n_s^{1-\alpha} \cdot n_s^\alpha \cdot n_s^{1-\alpha} \cdot n_s^\alpha = T_s n_s^2$ and space usage $n_s^{1-\alpha}$, we maintain the planted pair for the next epoch with constant failure probability.*

Proof Let $\tilde{y}_j = \sum_{l \in G_j} y_l$ for $j \in [m_s]$. Using similar analysis to the previous lemma:

$$\begin{aligned} c_{T,j} &= \sum_{t=1}^{T} \sum_{l \in G_j} \sum_{k \in [m_s] \setminus \{j\}} \sum_{r \in G_k} y_{t,l} y_{t,r} \\ &= \sum_{k \in [m_s] \setminus \{j\}} \sum_{l \in G_j} \sum_{r \in G_k} \langle y_l, y_r \rangle \\ &= \sum_{k \in [m_s] \setminus \{j\}} \Big\langle \sum_{l \in G_j} y_l, \sum_{r \in G_k} y_r \Big\rangle \\ &= \sum_{k \in [m_s] \setminus \{j\}} \langle \tilde{y}_j, \tilde{y}_k \rangle \end{aligned}$$

If G_j contains the planted pair, $c_{T,j}$ has expectation ρT; otherwise, it has expectation zero.

$$\mathbb{E}[c_{T,j}] = \rho T \cdot \mathbf{1}(\{i^*, j^*\} \cap G_j \neq \phi)$$

As for variance, we get $n_s T_s$ variance for each counter. We do the analysis for all random vectors (as with the earlier algorithm, having the planted pair does not make much of a difference).

$$\begin{aligned} \mathbb{E}[c_{T,j}^2] &= \sum_{k \in L_s \setminus G_j} \mathbb{E}[\langle \tilde{y}_j, \tilde{y}_k \rangle^2] \\ &= \sum_{k \in L_s \setminus G_j} \Big(\sum_{t=1}^{T_s} \sum_{l \in G_j, r \in G_k} y_{t,l} y_{t,r} \Big)^2 \\ \text{(independence)} \quad &= \sum_{k \in L_s \setminus G_j} \sum_{t=1}^{T_s} \sum_{l \in G_j} y_{t,l}^2 \\ &= n_s^{1-\alpha} \cdot T_s \cdot n_s^{\alpha} = n_s T \end{aligned}$$

From here, we use the same Chebyshev type analysis as before, except now the union bound is over $n^{1-\alpha}$. So we need $T_s = \Omega(n_s^{2-\alpha}/\rho^2)$ per epoch. ■

Combining the above claims shows us that the bucketing algorithm solves the sequential light bulb problem with:

- Space-per-round: $O(n^{1-\alpha})$
- Runtime-per-round: $O(n^2)$
- Rounds: $O\Big(\frac{\log\log(n)}{\log(1/\alpha)} \cdot n^{2-\alpha}\Big) = \tilde{O}(n^{2-\alpha})$

Taking $\alpha = 1 - \frac{C}{\log(n)}$ for a constant C gives us constant and $\tilde{O}(n)$ rounds (the number-of-epochs term remains a poly-logarithmic factor).

4 Discussion

In this work, we studied a novel sequential version of the light bulb problem, and we developed an algorithm that uses constant space and runs in a near-linear number of steps. It is worth asking how much better we can do.

One direction of growth could be to apply a tensor-embedding in an online fashion. Perhaps the learner keeps track of a random constant-sized window of elements from the past with which it generates synthetic examples. While this would enforce at least linear memory usage, it could potentially decrease the number of rounds.

Acknowledgments and Disclosure of Funding

The authors would like to thank Alex Andoni for his feedback in the writing process.

References

Josh Alman. An illuminating algorithm for the light bulb problem. *arXiv preprint arXiv:1810.06740*, 2018.

Josh Alman and Hengjie Zhang. Generalizations of matrix multiplication can solve the light bulb problem. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1471–1495. IEEE, 2023.

Dario Bini. Border rank of ap× q× 2 tensor and the optimal approximation of a pair of bilinear forms. In *International Colloquium on Automata, Languages, and Programming*, pages 98–108. Springer, 1980.

Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. New results for learning noisy parities and halfspaces. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 563–574. IEEE, 2006.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

Matti Karppa, Petteri Kaski, and Jukka Kohonen. A faster subquadratic algorithm for finding outlier correlations. *ACM Transactions on Algorithms (TALG)*, 14(3):1–26, 2018.

Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and juntas. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 11–20. IEEE, 2012.

Leslie G Valiant. Functionality in neural nets. In *Proceedings of the Seventh AAAI National Conference on Artificial Intelligence*, pages 629–634, 1988.