Provable ICA with Unknown Gaussian Noise, with Implications for Gaussian Mixtures and Autoencoders

Sanjeev Arora* **Rong Ge*** **Ankur Moitra †** **Sushant Sachdeva***

Abstract

We present a new algorithm for Independent Component Analysis (ICA) which has provable performance guarantees. In particular, suppose we are given samples of the form $y = Ax + \eta$ where A is an unknown $n \times n$ matrix and x is a random variable whose components are independent and have a fourth moment strictly less than that of a standard Gaussian random variable and η is an n-dimensional Gaussian random variable with unknown covariance Σ: We give an algorithm that provable recovers A and Σ up to an additive ϵ and whose running time and sample complexity are polynomial in n and $1/\epsilon$. To accomplish this, we introduce a novel "quasi-whitening" step that may be useful in other contexts in which the covariance of Gaussian noise is not known in advance. We also give a general framework for finding all local optima of a function (given an oracle for approximately finding just one) and this is a crucial step in our algorithm, one that has been overlooked in previous attempts, and allows us to control the accumulation of error when we find the columns of A one by one via local search.

1 Introduction

We present an algorithm (with rigorous performance guarantees) for a basic statistical problem. Suppose η is an independent n-dimensional Gaussian random variable with an unknown covariance matrix Σ and A is an unknown $n \times n$ matrix. We are given samples of the form $y = Ax + \eta$ where x is a random variable whose components are independent and have a fourth moment strictly less than that of a standard Gaussian random variable. The most natural case is when x is chosen uniformly at random from $\{+1, -1\}^n$, although our algorithms in even the more general case above. Our goal is to reconstruct an additive approximation to the matrix A and the covariance matrix Σ running in time and using a number of samples that is polynomial in n and $\frac{1}{\epsilon}$, where ϵ is the target precision (see Theorem 1.1) This problem arises in several research directions within machine learning: Independent Component Analysis (ICA), Deep Learning, Gaussian Mixture Models (GMM), etc. We describe these connections next, and known results (focusing on algorithms with provable performance guarantees, since that is our goal).

Most obviously, the above problem can be seen as an instance of *Independent Component Analysis* (ICA) with unknown Gaussian noise. ICA has an illustrious history with applications ranging from econometrics, to signal processing, to image segmentation. The goal generally involves finding a linear transformation of the data so that the coordinates are as independent as possible [1, 2, 3]. This is often accomplished by finding directions in which the projection is "non-Gaussian" [4]. Clearly, if the datapoint y is generated as Ax (i.e., with no noise η added) then applying linear transformation A^{-1} to the data results in samples $A^{-1}y$ whose coordinates are independent. This restricted case was considered by Comon [1] and Frieze, Jerrum and Kannan [5], and their goal was to recover an

*{arora, rongge, sachdeva}@cs.princeton.edu. Department of Computer Science, Princeton University, Princeton NJ 08540. Research supported by the NSF grants CCF-0832797, CCF-1117309 and Simons Investigator Grant

†moitra@ias.edu. School of Mathematics, Institute for Advanced Study, Princeton NJ 08540. Research supported in part by NSF grant No. DMS-0835373 and by an NSF Computing and Innovation Fellowship.

additive approximation to A efficiently and using a polynomial number of samples. (We will later note a gap in their reasoning, albeit fixable by our methods. See also recent papers by Anandkumar *et al.*, Hsu and Kakade[6, 7], that do not use local search and avoids this issue.) To the best of our knowledge, there are currently no known algorithms with provable guarantees for the more general case of ICA with Gaussian noise (this is especially true if the covariance matrix is unknown, as in our problem), although many empirical approaches are known. (eg. [8], the issue of "empirical" vs "rigorous" is elaborated upon after Theorem 1.1.)

The second view of our problem is as a concisely described *Gaussian Mixture Model*. Our data is generated as a mixture of 2^n identical Gaussian components (with an unknown covariance matrix) whose centers are the points $\{Ax : x \in \{-1, 1\}^n\}$, and all mixing weights are equal. Notice, this mixture of 2^n Gaussians admits a concise description using $O(n^2)$ parameters. The problem of learning Gaussian mixtures has a long history, and the popular approach in practice is to use the EM algorithm [9], though it has no worst-case guarantees (the method may take a very long time to converge, and worse, may not always converge to the correct solution). An influential paper of Dasgupta [10] initiated the program of designing algorithms with provable guarantees, which was improved in a sequence of papers [11, 12, 13, 14]. But in the current setting, it is unclear how to apply any of the above algorithms (including EM) since the trivial application would keep track of exponentially many parameters – one for each component. Thus, new ideas seem necessary to achieve polynomial running time.

The third view of our problem is as a simple form of *autoencoding* [15]. This is a central notion in Deep Learning, where the goal is to obtain a compact representation of a target distribution using a multilayered architecture, where a complicated function (the target) can be built up by composing layers of a simple function (called the autoencoder [16]). The main tenet is that there are interesting functions which can be represented concisely using many layers, but would need a very large representation if a "shallow" architecture is used instead). This is most useful for functions that are "highly varying" (i.e. cannot be compactly described by piecewise linear functions or other "simple" local representations). Formally, it is possible to represent using just (say) n^2 parameters, some distributions with 2^n "varying parts" or "interesting regions." The *Restricted Boltzmann Machine* (RBM) is an especially popular autoencoder in Deep Learning, though many others have been proposed. However, to the best of our knowledge, there has been no successful attempt to give a *rigorous* analysis of Deep Learning. Concretely, if the data is indeed generated using the distribution represented by an RBM, then do the popular algorithms for Deep Learning [17] learn the model parameters *correctly* and in *polynomial* time? Clearly, if the running time were actually found to be exponential in the number of parameters, then this would erode some of the advantages of the compact representation.

How is Deep Learning related to our problem? As noted by Freund and Haussler [18] many years ago, an RBM with real-valued visible units (the version that seems more amenable to theoretical analysis) is precisely a mixture of exponentially many standard Gaussians. It is parametrized by an $n \times m$ matrix A and a vector $\theta \in \mathbb{R}^n$. It encodes a mixture of n-dimensional standard Gaussians centered at the points $\{Ax : x \in \{-1, 1\}^m\}$, where the mixing weight of the Gaussian centered at Ax is $\exp(\|Ax\|_2^2 + \theta \cdot x)$. This is of course reminiscent of our problem. Formally, our algorithm can be seen as a nonlinear autoencoding scheme analogous to an RBM but with uniform mixing weights. Interestingly, the algorithm that we present here looks nothing like the approaches favored traditionally in Deep Learning, and may provide an interesting new perspective.

1.1 Our results and techniques

We give a provable algorithm for ICA with unknown Gaussian noise. We have not made an attempt to optimize the quoted running time of this model, but we emphasize that this is in fact the first algorithm with provable guarantees for this problem and moreover we believe that in practice our algorithm will run almost as fast as the usual ICA algorithms, which are its close relatives.

Theorem 1.1 (Main, Informally)**.** *There is an algorithm that recovers the unknown A and Σ up to additive error ϵ in each entry in time that is polynomial in n, $\|A\|_2$, $\|\Sigma\|_2$, $1/\epsilon$, $1/\lambda_{\min}(A)$ where $\|\cdot\|_2$ denotes the operator norm and $\lambda_{\min}(\cdot)$ denotes the smallest eigenvalue.*

The classical approach for ICA initiated in Comon [1] and Frieze, Jerrum and Kannan [5]) is for the noiseless case in which $y = Ax$. The first step is *whitening*, which applies a suitable linear transformation that makes the variance the same in all directions, thus reducing to the case where

A is a *rotation* matrix. Given samples $y = Rx$ where R is a rotation matrix, the rows of R can be found in principle by computing the vectors u that are local minima of $E[(u \cdot y)^4]$. Subsequently, a number of works (see e.g. [19, 20]) have focused on giving algorithms that are robust to noise. A popular approach is to use the fourth order *cumulant* (as an alternative to the fourth order moment) as a method for "denoising," or any one of a number of other functionals whose local optima reveal interesting directions. However, theoretical guarantees of these algorithms are not well understood.

The above procedures in the noise-free model can *almost* be made rigorous (i.e., provably polynomial running time and number of samples), except for one subtlety: it is unclear how to use local search to find *all* optima in polynomial time. In practice, one finds a single local optimum, projects to the subspace orthogonal to it and continues recursively on a lower-dimensional problem. However, a naive implementation of this idea is unstable since approximation errors can accumulate badly, and to the best of our knowledge no rigorous analysis has been given prior to our work. (This is not a technicality: in some similar settings the errors are known to blow up exponentially [21].) One of our contributions is a modified local search that avoids this potential instability and finds all local optima in this setting. (Section 4.2.)

Our major new contribution however is dealing with noise that is an unknown Gaussian. This is an important generalization, since many methods used in ICA are quite unstable to noise (and a wrong estimate for the covariance could lead to bad results). Here, we no longer need to assume we know even rough estimates for the covariance. Moreover, in the context of Gaussian Mixture Models this generalization corresponds to learning a mixture of many Gaussians where the covariance of the components is not known in advance.

We design new tools for denoising and especially whitening in this setting. Denoising uses the fourth order cumulant instead of the fourth moment used in [5] and whitening involves a novel use of the Hessian of the cumulant. Even then, we cannot reduce to the simple case $y = Rx$ as above, and are left with a more complicated functional form (see "quasi-whitening" in Section 2.) Nevertheless, we can reduce to an optimization problem that can be solved via local search, and which remains amenable to a rigorous analysis. The results of the local optimization step can be then used to simplify the complicated functional form and recover A as well as the noise Σ. We defer many of our proofs to the supplementary material section, due to space constraints.

In order to avoid cluttered notation, we have focused on the case in which x is chosen uniformly at random from $\{-1, +1\}^n$, although our algorithm and analysis work under the more general conditions that the coordinates of x are (i) independent and (ii) have a fourth moment that is less than three (the fourth moment of a Gaussian random variable). In this case, the functional $P(u)$ (see Lemma 2.2) will take the same form but with weights depending on the exact value of the fourth moment for each coordinate. Since we already carry through an unknown diagonal matrix D throughout our analysis, this generalization only changes the entries on the diagonal and the same algorithm and proof apply.

2 Denoising and quasi-whitening

As mentioned, our approach is based on the fourth order cumulant. The cumulants of a random variable are the coefficients of the Taylor expansion of the logarithm of the characteristic function [22]. Let $\kappa_r(X)$ be the r^{th} cumulant of a random variable X. We make use of:

Fact 2.1. *(i) If X has mean zero, then $\kappa_4(X) = \mathbf{E}[X^4] - 3\,\mathbf{E}[X^2]^2$. (ii) If X is Gaussian with mean μ and variance σ^2, then $\kappa_1(X) = \mu$, $\kappa_2(X) = \sigma^2$ and $\kappa_r(X) = 0$ for all $r > 2$. (iii) If X and Y are independent, then $\kappa_r(X + Y) = \kappa_r(X) + \kappa_r(Y)$.*

The crux of our technique is to look at the following functional, where y is the random variable $Ax + \eta$ whose samples are given to us. Let $u \in \mathbb{R}^n$ be any vector. Then $P(u) = -\kappa_4(u^T y)$. Note that for any u we can compute $P(u)$ reasonably accurately by drawing sufficient number of samples of y and taking an empirical average. Furthermore, since x and η are independent, and η is Gaussian, the next lemma is immediate. We call it "denoising" since it allows us empirical access to some information about A that is uncorrupted by the noise η.

Lemma 2.2 (Denoising Lemma)**.** $P(u) = 2\sum_{i=1}^{n}(u^T A)_i^4$.

The intuition is that $P(u) = -\kappa_4(u^T Ax)$ since the fourth cumulant does not depend on the additive Gaussian noise, and then the lemma follows from completing the square.

2.1 Quasi-whitening via the Hessian of $P(u)$

In prior works on ICA, *whitening* refers to reducing to the case where $y = Rx$ for some some rotation matrix R. Here we give a technique to reduce to the case where $y = RDx + \eta'$ where η' is some other Gaussian noise (still unknown), R is a rotation matrix and D is a diagonal matrix that depends upon A. We call this *quasi-whitening*. Quasi-whitening suffices for us since local search using the objective function $\kappa_4(u^T y)$ will give us (approximations to) the rows of RD, from which we will be able to recover A.

Quasi-whitening involves computing the Hessian of $P(u)$, which recall is the matrix of all 2nd order partial derivatives of $P(u)$. Throughout this section, we will denote the Hessian operator by $\mathcal{H}$. In matrix form, the Hessian of $P(u)$ is

$$\frac{\partial^2}{\partial u_i \partial u_j} P(u) = 24 \sum_{k=1}^{n} A_{i,k} A_{j,k} (A_k \cdot u)^2; \mathcal{H}(P(U)) = 24 \sum_{k=1}^{n} (A_k \cdot u)^2 A_k A_k^T = A D_A(u) A^T$$

where A_k is the k-th column of the matrix A (we use subscripts to denote the columns of matrices throught the paper). $D_A(u)$ is the following diagonal matrix:

Definition 2.3. Let $D_A(u)$ be a diagonal matrix in which the k^{th} entry is $24(A_k \cdot u)^2$.

Of course, the exact Hessian of $P(u)$ is unavailable and we will instead compute an empirical approximation $\widehat{P}(u)$ to $P(u)$ (given many samples from the distribution), and we will show that the Hessian of $\widehat{P}(u)$ is a good approximation to the Hessian of $P(u)$.

Definition 2.4. Given $2N$ samples $y_1, y_1', y_2, y_2' ..., y_N, y_N'$ of the random variable y, let

$$\widehat{P}(u) = \frac{-1}{N} \sum_{i=1}^{N} (u^T y_i)^4 + \frac{3}{N} \sum_{i=1}^{N} (u^T y_i)^2 (u^T y_i')^2.$$

Our first step is to show that the expectation of the Hessian of $\widehat{P}(u)$ is exactly the Hessian of $P(u)$. In fact, since the expectation of $\widehat{P}(u)$ is exactly $P(u)$ (and since $\widehat{P}(u)$ is an analytic function of the samples and of the vector u), we can interchange the Hessian operator and the expectation operator. Roughly, one can imagine the expectation operator as an integral over the possible values of the random samples, and as is well-known in analysis, one can differentiate under the integral provided that all functions are suitably smooth over the domain of integration.

Claim 2.5. $\mathbf{E}_{y,y'}[-(u^T y)^4 + 3(u^T y)^2 (u^T y')^2] = P(u)$

This claim follows immediately from the definition of $P(u)$, and since y and y' are independent.

Lemma 2.6. $\mathcal{H}(P(u)) = \mathbf{E}_{y,y'}[\mathcal{H}(-(u^T y)^4 + 3(u^T y)^2 (u^T y')^2)]$

Next, we compute the two terms inside the expectation:

Claim 2.7. $\mathcal{H}((u^T y)^4) = 12(u^T y)^2 y y^T$

Claim 2.8. $\mathcal{H}((u^T y)^2 (u^T y')^2) = 2(u^T y')^2 y y^T + 2(u^T y)^2 y'(y')^T + 4(u^T y)(u^T y')(y(y')^T + (y')y^T)$

Let $\lambda_{min}(A)$ denote the smallest eigenvalue of A. Our analysis also requires bounds on the entries of $D_A(u_0)$:

Claim 2.9. *If u_0 is chosen uniformly at random then with high probability for all i,*

$$\min_{i=1}^{n} \|A_i\|_2^2 n^{-4} \leq D_A(u_0))_{i,i} \leq \max_{i=1}^{n} \|A_i\|_2^2 \frac{\log n}{n}$$

Lemma 2.10. *If u_0 is chosen uniformly at random and furthermore we are given $2N = \mathrm{poly}(n, 1/\epsilon, 1/\lambda_{min}(A), \|A\|_2, \|\Sigma\|_2)$ samples of y, then with high probability we will have that $(1-\epsilon) A D_A(u_0) A^T \preceq \mathcal{H}(\widehat{P}(u_0)) \preceq (1+\epsilon) A D_A(u_0) A^T$.*

Lemma 2.11. *Suppose that $(1-\epsilon) A D_A(u_0) A^T \preceq \widehat{M} \preceq (1+\epsilon) A D_A(u_0) A^T$, and let $\widehat{M} = BB^T$. Then there is a rotation matrix R^* such that $\|B^{-1} A D_A(u_0)^{1/2} - R^*\|_F \leq \sqrt{n}\epsilon$.*

The intuition is: if any of the singular values of $B^{-1} A D_A(u_0)^{1/2}$ are outside the range $[1-\epsilon, 1+\epsilon]$, we can find a unit vector x where the quadratic forms $x^T A D_A(u_0) A^T x$ and $x^T \widehat{M} x$ are too far apart (which contradicts the condition of the lemma). Hence the singular values of $B^{-1} A D_A(u_0)^{1/2}$ can all be set to one without changing the Froebenius norm of $B^{-1} A D_A(u_0)^{1/2}$ too much, and this yields a rotation matrix.

3 Our algorithm (and notation)

In this section we describe our overall algorithm. It uses as a blackbox the denoising and quasi-whitening already described above, as well as a routine for computing all local maxima of some "well-behaved" functions which is described later in Section 4.

Notation: Placing a hat over a function corresponds to an empirical approximation that we obtain from random samples. This approximation introduces error, which we will keep track of.

Step 1: *Pick a random* $u_0 \in \mathbb{R}^n$ *and estimate the Hessian* $\mathcal{H}(\widehat{P}(u_0))$. *Compute* B *such that* $\mathcal{H}(\widehat{P}(u_0)) = BB^T$. *Let* $D = D_A(u_0)$ *be the diagonal matrix defined in Definition 2.3.*

Step 2: *Take* $2N$ *samples* $y_1, y_2, ..., y_N, y'_1, y'_2, ..., y'_N$, *and let* $\widehat{P}'(u) = -\frac{1}{N}\sum_{i=1}^N (u^T B^{-1} y_i)^4 + \frac{3}{N}\left(\sum_{i=1}^N (u^T B^{-1} y_i)^2 (u^T B^{-1} y'_i)^2\right)$ *which is an empirical estimation of* $P'(u)$.

Step 3: *Use the procedure* ALLOPT$(\widehat{P}'(u), \beta, \delta', \beta', \delta')$ *of Section 4 to compute all* n *local maxima of the function* $\widehat{P}'(u)$.

Step 4: *Let* R *be the matrix whose rows are the* n *local optima recovered in the previous step. Use procedure* RECOVER *of Section 5 to find* A *and* Σ.

Explanation: Step 1 uses the transformation B^{-1} computed in the previous Section to quasi-whiten the data. Namely, we consider the sequence of samples $z = B^{-1}y$, which are therefore of the form $R'Dx+\eta'$ where $\eta = B^{-1}\eta$, $D = D_A(u_0)$ and R' is close to a rotation matrix R^* (by Lemma 2.11). In Step 2 we look at $\kappa_4((u^T z))$, which effectively denoises the new samples (see Lemma 2.2), and thus is the same as $\kappa_4(R'D^{-1/2}x)$. Let $P'(u) = \kappa_4(u^T z) = \kappa_4(u^T B^{-1} y)$ which is easily seen to be $E[(u^T R' D^{-1/2} x)^4]$. Step 2 estimates this function, obtaining $\widehat{P}'(u)$. Then Step 3 tries to find local optima via local search. Ideally we would have liked access to the functional $P^*(u) = (u^T R^* x)^4$ since the procedure for local optima works only for true rotations. But since R' and R^* are close we can make it work approximately with $\widehat{P}'(u)$, and then in Step 4 use these local optima to finally recover A.

Theorem 3.1. *Suppose we are given samples of the form* $y = Ax + \eta$ *where* x *is uniform on* $\{+1, -1\}^n$, A *is an* $n \times n$ *matrix,* η *is an* n*-dimensional Gaussian random variable independent of* x *with unknown covariance matrix* Σ. *There is an algorithm that with high probability recovers* $\|\widehat{A} - A\Pi diag(k_i)\|_F \leq \epsilon$ *where* Π *is some permutation matrix and each* $k_i \in \{+1, -1\}$ *and also recovers* $\|\widehat{\Sigma} - \Sigma\|_F \leq \epsilon$. *Furthermore the running time and number of samples needed are* $\mathrm{poly}(n, 1/\epsilon, \|A\|_2, \|\Sigma\|_2, 1/\lambda_{min}(A))$

Note that here we recover A up to a permutation of the columns and sign-flips. In general, this is all we can hope for since the distribution of x is also invariant under these same operations. Also, the dependence of our algorithm on the various norms (of A and Σ) seems inherent since our goal is to recover an additive approximation, and as we scale up A and/or Σ, this goal becomes a stronger relative guarantee on the error.

4 Framework for iteratively finding all local maxima

In this section, we first describe a fairly standard procedure (based upon Newton's method) for finding a *single* local maximum of a function $f^* : \mathbb{R}^n \to \mathbb{R}$ among all unit vectors and an analysis of its rate of convergence. Such a procedure is a common tool in statistical algorithms, but here we state it rather carefully since we later give a general method to convert any local search algorithm (that meets certain criteria) into one that finds *all* local maxima (see Section 4.2).

Given that we can only ever hope for an additive approximation to a local maximum, one should be concerned about how the error accumulates when our goal is to find *all* local maxima. In fact, a naive strategy is to project onto the subspace orthogonal to the directions found so far, and continue in this subspace. However, such an approach seems to accumulate errors badly (the additive error of the last local maxima found is exponentially larger than the error of the first). Rather, the crux of our analysis is a novel method for bounding how much the error can accumulate (by refining old estimates).

Algorithm 1. LOCALOPT, **Input:** $f(u)$, u_s, β, δ **Output:** vector v

1. Set $u \leftarrow u_s$.
2. Maximize (via Lagrangian methods) $\text{Proj}_{\perp u}(\nabla f(u))^T \xi + \frac{1}{2}\xi^T \text{Proj}_{\perp u}(\mathcal{H}(f(u)))\xi - \frac{1}{2}\left(\frac{\partial}{\partial_u} f(u)\right) \cdot \|\xi\|_2^2$ Subject to $\|\xi\|_2 \leq \beta'$ and $u^T \xi = 0$
3. Let ξ be the solution, $\tilde{u} = \frac{u+\xi}{\|u+\xi\|}$
4. If $f(\tilde{u}) \geq f(u) + \delta/2$, set $u \leftarrow \tilde{u}$ and Repeat Step 2
5. Else return u

Our strategy is to first find a local maximum in the orthogonal subspace, then run the local optimization algorithm again (in the original n-dimensional space) to "refine" the local maximum we have found. The intuition is that since we are already close to a particular local maxima, the local search algorithm cannot jump to some other local maxima (since this would entail going through a valley).

4.1 Finding one local maximum

Throughout this section, we will assume that we are given oracle access to a function $f(u)$ and its gradient and Hessian. The procedure is also given a starting point u_s, a search range β, and a step size δ. For simplicity in notation we define the following projection operator.

Definition 4.1. $\text{Proj}_{\perp u}(v) = v - (u^T v)u$, $\text{Proj}_{\perp u}(M) = M - (u^T M u)uu^T$.

The basic step the algorithm is a modification of Newton's method to find a local improvement that makes progress so long as the current point u is far from a local maxima. Notice that if we add a small vector to u, we do not necessarily preserve the norm of u. In order to have control over how the norm of u changes, during local optimization step the algorithm projects the gradient ∇f and Hessian $\mathcal{H}(f)$ to the space perpendicular to u. There is also an additional correction term $-\partial/\partial_u f(u) \cdot \|\xi\|^2/2$. This correction term is necessary because the new vector we obtain is $(u + \xi)/\|(u+\xi)\|_2$ which is close to $u - \|\xi\|_2^2/2 \cdot u + \xi + O(\beta^3)$. Step 2 of the algorithm is just maximizing a quadratic function and can be solved exactly using Lagrangian Multiplier method. To increase efficiency it is also acceptable to perform an approximate maximization step by taking ξ to be either aligned with the gradient $\text{Proj}_{\perp u} \nabla f(u)$ or the largest eigenvector of $\text{Proj}_{\perp u}(\mathcal{H}(f(u)))$.

The algorithm is guaranteed to succeed in polynomial time when the function is *Locally Improvable* and *Locally Approximable*:

Definition 4.2 ((γ, β, δ)-Locally Improvable)**.** A function $f(u) : \mathbb{R}^n \to \mathbb{R}$ is (γ, β, δ)-Locally Improvable, if for any u that is at least γ far from any local maxima, there is a u' such that $\|u' - u\|_2 \leq \beta$ and $f(u') \geq f(u) + \delta$.

Definition 4.3 ((β, δ)-Locally Approximable)**.** A function $f(u)$ is locally approximable, if its third order derivatives exist and for any u and any direction v, the third order derivative of f at point u in the direction of v is bounded by $0.01\delta/\beta^3$.

The analysis of the running time of the procedure comes from local Taylor expansion. When a function is Locally Approximable it is well approximated by the gradient and Hessian within a β neighborhood. The following theorem from [5] showed that the two properties above are enough to guarantee the success of a local search algorithm even when the function is only approximated.

Theorem 4.4 ([5])**.** *If $|f(u) - f^*(u)| \leq \delta/8$, the function $f^*(u)$ is (γ, β, δ)-Locally Improvable, $f(u)$ is (β, δ) Locally Approximable, then Algorithm 1 will find a vector v that is γ close to some local maximum. The running time is at most $O((n^2 + T)\max f^*/\delta)$ where T is the time to evaluate the function f and its gradient and Hessian.*

4.2 Finding all local maxima

Now we consider how to find *all* local maxima of a given function $f^*(u)$. The crucial condition that we need is that *all local maxima are orthogonal* (which is indeed true in our problem, and is morally true when using local search more generally in ICA). Note that this condition implies that there are at most n local maxima.[1] In fact we will assume that there are exactly n local maxima. If we are given an exact oracle for f^* and can compute *exact* local maxima then we can find all local maxima

Algorithm 2. ALLOPT, **Input:** $f(u)$, β, δ, β', δ' **Output:** $v_1, v_2, ..., v_n$, $\forall i \ \|v_i - v_i^*\| \leq \gamma$.

1. Let $v_1 = \text{LOCALOPT}(f, e_1, \beta, \delta)$
2. FOR $i = 2$ TO n DO
3. Let g_i be the projection of f to the orthogonal subspace of $v_1, v_2, ..., v_{i-1}$.
4. Let $u' = \text{LOCALOPT}(g, e_1, \beta', \delta')$.
5. Let $v_i = \text{LOCALOPT}(f, u', \beta, \delta)$.
6. END FOR
7. Return $v_1, v_2, ..., v_n$

easily: find one local maximum, project the function into the orthogonal subspace, and continue to find more local maxima.

Definition 4.5. The projection of a function f to a linear subspace S is a function on that subspace with value equal to f. More explicitly, if $\{v_1, v_2, ..., v_d\}$ is an orthonormal basis of S, the projection of f to S is a function $g : \mathbb{R}^d \to \mathbb{R}$ such that $g(w) = f(\sum_{i=1}^{d} w_i v_i)$.

The following theorem gives sufficient conditions under which the above algorithm finds all local maxima, making precise the intuition given at the beginning of this section.

Theorem 4.6. *Suppose the function $f^*(u) : \mathbb{R}^n \to \mathbb{R}$ satisfies the following properties:*

1. *Orthogonal Local Maxima: The function has n local maxima v_i^*, and they are orthogonal to each other.*
2. *Locally Improvable: f^* is (γ, β, δ) Locally Improvable.*
3. *Improvable Projection: The projection of the function to any subspace spanned by a subset of local maxima is $(\gamma', \beta', \delta')$ Locally Improvable. The step size $\delta' \geq 10\delta$.*
4. *Lipschitz: If $\|u - u'\|_2 \leq 3\sqrt{n}\gamma$, then the function value $|f^*(u) - f^*(u')| \leq \delta'/20$.*
5. *Attraction Radius: Let $Rad \geq 3\sqrt{n}\gamma + \gamma'$, for any local maximum v_i^*, let T be $\min f^*(u)$ for $\|u - v_i^*\|_2 \leq Rad$, then there exist a set U containing $\|u - v_i^*\|_2 \leq 3\sqrt{n}\gamma + \gamma'$ and does not contain any other local maxima, such that for every u that is not in U but is β close to U, $f^*(u) < T$.*

If we are given function f such that $|f(u) - f^*(u)| \leq \delta/8$ and f is both (β, δ) and (β', δ') Locally Approximable, then Algorithm 2 can find all local maxima of f^* within distance γ.

To prove this theorem, we first notice the projection of the function f in Step 3 of the algorithm should be close to the projection of f^* to the remaining local maxima. This is implied by Lipschitz condition and is formally shown in the following two lemmas. First we prove a "coupling" between the orthogonal complement of two close subspaces:

Lemma 4.7. *Given v_1, v_2, ..., v_k, each γ-close respectively to local maxima $v_1^*, v_2^*, ..., v_k^*$ (this is without loss of generality because we can permute the index of local maxima), then there is an orthonormal basis $v_{k+1}, v_{k+2}, ..., v_n$ for the orthogonal space of span$\{v_1, v_2, ..., v_k\}$ such that for any unit vector $w \in \mathbb{R}^{n-k}$, $\sum_{i=1}^{n-k} w_k v_{k+i}$ is $3\sqrt{n}\gamma$ close to $\sum_{i=1}^{n-k} w_k v_{k+i}^*$.*

We prove this lemma using a modification of the Gram-Schmidt orthonormalization procedure. Using this lemma we see that the projected function is close to the projection of f^* in the span of the rest of local maxima:

Lemma 4.8. *Let g^* be the projection of f^* into the space spanned by the rest of local maxima, then $|g^*(w) - g(w)| \leq \delta/8 + \delta'/20 \leq \delta'/8$.*

5 Local search on the fourth order cumulant

Next, we prove that the fourth order cumulant $P^*(u)$ satisfies the properties above. Then the algorithm given in the previous section will find all of the local maxima, which is the missing step in our

[1]Technically, there are $2n$ local maxima since for each direction u that is a local maxima, so too is $-u$ but this is an unimportant detail for our purposes.

Algorithm 3. RECOVER, **Input:** B, $\widehat{P}'(u)$, $\widehat{R}$, ϵ **Output:** $\widehat{A}$, $\widehat{\Sigma}$

1. Let $\widehat{D}_A(u)$ be a diagonal matrix whose i^{th} entry is $\frac{1}{2}\left(\widehat{P}'(\widehat{R}_i)\right)^{-1/2}$.
2. Let $\widehat{A} = B\widehat{R}\widehat{D}_A(u)^{-1/2}$.
3. Estimate $C = \mathbf{E}[yy^T]$ by taking $O((\|A\|_2 + \|\Sigma\|_2)^4 n^2 \epsilon^{-2})$ samples and let $\widehat{C} = \frac{1}{N}\sum_{i=1}^{N} y_i y_i^T$.
4. Let $\widehat{\Sigma} = \widehat{C} - \widehat{A}\widehat{A}^T$
5. Return $\widehat{A}, \widehat{\Sigma}$

main goal: learning a noisy linear transformation $Ax + \eta$ with unknown Gaussian noise. We first use a theorem from [5] to show that properties for finding one local maxima is satisfied.

Also, for notational convenience we set $d_i = 2D_A(u_0)_{i,i}^{-2}$ and let d_{min} and d_{max} denote the minimum and maximum values (bounds on these and their ratio follow from Claim 2.9). Using this notation $P^*(u) = \sum_{i=1}^{n} d_i (u^T R_i^*)^4$.

Theorem 5.1 ([5]). *When $\beta < d_{min}/10 d_{max} n^2$, the function $P^*(u)$ is $(3\sqrt{n}\beta, \beta, P^*(u)\beta^2/100)$ Locally Improvable and $(\beta, d_{min}\beta^2/100n)$ Locally Approximable. Moreover, the local maxima of the function is exactly $\{\pm R_i^*\}$.*

We then observe that given enough samples, the empirical mean $\widehat{P}'(u)$ is close to $P^*(u)$. For concentration we require every degree four term $z_i z_j z_k z_l$ has variance at most Z.

Claim 5.2. $Z = O(d_{min}^2 \lambda_{min}(A)^8 \|\Sigma\|_2^4 + d_{min}^2)$.

Lemma 5.3. *Given $2N$ samples $y_1, y_2, ..., y_N, y'_1, y'_2, ..., y'_N$, suppose columns of $R' = B^{-1}AD_A(u_0)^{1/2}$ are ϵ close to the corresponding columns of R^*, with high probability the function $\widehat{P}'(u)$ is $O(d_{max} n^{1/2}\epsilon + n^2(N/Z\log n)^{-1/2})$ close to the true function $P^*(u)$.*

The other properties required by Theorem 4.6 are also satisfied:

Lemma 5.4. *For any $\|u - u'\|_2 \le r$, $|P^*(u) - P^*(u')| \le 5 d_{max} n^{1/2} r$. All local maxima of P^* has attraction radius $Rad \ge d_{min}/100 d_{max}$.*

Applying Theorem 4.6 we obtain the following Lemma (the parameters are chosen so that all properties required are satisfied):

Lemma 5.5. *Let $\beta' = \Theta((d_{min}/d_{max})^2)$, $\beta = \min\{\gamma n^{-1/2}, \Omega((d_{min}/d_{max})^4 n^{-3.5})\}$, then the procedure* RECOVER$(f, \beta, d_{min}\beta^2/100n, \beta', d_{min}\beta'^2/100n)$ *finds vectors $v_1, v_2, ..., v_n$, so that there is a permutation matrix Π and $k_i \in \{\pm 1\}$ and for all i: $\|v_i - (R\Pi Diag(k_i))_i^*\|_2 \le \gamma$.*

After obtaining $\widehat{R} = [v_1, v_2, ..., v_n]$ we can use Algorithm 3 to find A and Σ:

Theorem 5.6. *Given a matrix $\widehat{R}$ such that there is permutation matrix Π and $k_i \in \{\pm 1\}$ with $\|\widehat{R}_i - k_i(R^*\Pi)_i\|_2 \le \gamma$ for all i, Algorithm 3 returns matrix $\widehat{A}$ such that $\|\widehat{A} - A\Pi Diag(k_i)\|_F \le O(\gamma \|A\|_2^2 n^{3/2}/\lambda_{min}(A))$. If $\gamma \le O(\epsilon/\|A\|_2^2 n^{3/2}\lambda_{min}(A)) \times \min\{1/\|A\|_2, 1\}$, we also have $\|\widehat{\Sigma} - \Sigma\|_F \le \epsilon$.*

Recall that the diagonal matrix $D_A(u)$ is unknown (since it depends on A), but if we are given R^* (or an approximation) and since $P^*(u) = \sum_{i=1}^{n} d_i (u^T R_i^*)^4$, we can recover the matrix $D_A(u)$ approximately from computing $P^*(R_i^*)$. Then given $D_A(u)$, we can recover A and Σ and this completes the analysis of our algorithm.

Conclusions

ICA is a vast field with many successful techniques. Most rely on heuristic nonlinear optimization. An exciting question is: can we give a rigorous analysis of those techniques as well, just as we did for local search on cumulants? A rigorous analysis of deep learning —say, an algorithm that provably learns the parameters of an RBM—is another problem that is wide open, and a plausible special case involves subtle variations on the problem we considered here.

References

[1] P. Comon. Independent component analysis: a new concept? *Signal Processing*, pp. 287–314, 1994. 1, 1.1

[2] A. Hyvarinen, J. Karhunen, E. Oja. *Independent Component Analysis.* Wiley: New York, 2001. 1

[3] A. Hyvarinen, E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, pp. 411–430, 2000. 1

[4] P. J. Huber. Projection pursuit. *Annals of Statistics* pp. 435–475, 1985. 1

[5] A. Frieze, M. Jerrum, R. Kannan. Learning linear transformations. *FOCS*, pp. 359–368, 1996. 1, 1.1, 4.1, 4.4, 5, 5.1

[6] A. Anandkumar, D. Foster, D. Hsu, S. Kakade, Y. Liu. Two SVDs suffice: spectral decompositions for probabilistic topic modeling and latent Dirichlet allocation. *Arxiv:*abs/1203.0697, 2012. 1

[7] D. Hsu, S. Kakade. Learning mixtures of spherical Gaussians: moment methods and spectral decompositions. *Arxiv:*abs/1206.5766, 2012. 1

[8] L. De Lathauwer; J. Castaing; J.-F. Cardoso, Fourth-Order Cumulant-Based Blind Identification of Underdetermined Mixtures, *Signal Processing, IEEE Transactions on*, vol.55, no.6, pp.2965-2973, June 2007 1

[9] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM Algorithm. *Journal of the Royal Statistical Society Series B*, pp. 1–38, 1977. 1

[10] S. Dasgupta. Learning mixtures of Gaussians. *FOCS* pp. 634–644, 1999. 1

[11] S. Arora and R. Kannan. Learning mixtures of separated nonspherical Gaussians. *Annals of Applied Probability*, pp. 69-92, 2005. 1

[12] M. Belkin and K. Sinha. Polynomial learning of distribution families. *FOCS* pp. 103–112, 2010. 1

[13] A. T. Kalai, A. Moitra, and G. Valiant. Efficiently learning mixtures of two Gaussians. *STOC* pp. 553-562, 2010. 1

[14] A. Moitra and G. Valiant. Setting the polynomial learnability of mixtures of Gaussians. *FOCS* pp. 93–102, 2010. 1

[15] G. Hinton, R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science* pp. 504–507, 2006. 1

[16] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, pp. 1–127, 2009. 1

[17] G. E. Hinton. A Practical Guide to Training Restricted Boltzmann Machines, Version 1, UTML TR 2010-003, Department of Computer Science, University of Toronto, August 2010 1

[18] Y. Freund , D. Haussler. Unsupervised Learning of Distributions on Binary Vectors using Two Layer Networks University of California at Santa Cruz, Santa Cruz, CA, 1994 1

[19] S. Cruces, L. Castedo, A. Cichocki, Robust blind source separation algorithms using cumulants, *Neurocomputing*, Volume 49, Issues 14, pp 87-118, 2002. 1.1

[20] L., De Lathauwer; B., De Moor; J. Vandewalle. Independent component analysis based on higher-order statistics only *P*roceedings of 8th IEEE Signal Processing Workshop on Statistical Signal and Array Processing, 1996. 1.1

[21] S. Vempala, Y. Xiao. Structure from local optima: learning subspace juntas via higher order PCA. *Arxiv:*abs/1108.3329, 2011. 1.1

[22] M. Kendall, A. Stuart. *The Advanced Theory of Statistics* Charles Griffin and Company, 1958. 2