Fast Algorithms for Learning Latent Variables in Graphical Models

Mohammadreza Soltani
Iowa State University
msoltani@iastate.edu

Chinmay Hegde
Iowa State University
chinmay@iastate.edu

ABSTRACT

We study the problem of learning latent variables in Gaussian graphical models. Existing methods for this problem assume that the precision matrix of the observed variables is the superposition of a sparse and a low-rank component. In this paper, we focus on the estimation of the low-rank component, which encodes the effect of marginalization over the latent variables. We introduce fast, proper learning algorithms for this problem. In contrast with existing approaches, our algorithms are manifestly non-convex. We support their efficacy via a rigorous theoretical analysis, and show that our algorithms match the best possible in terms of sample complexity, while achieving computational speed-ups over existing methods. We complement our theory with several numerical experiments.

1 INTRODUCTION

1.1 Setup

Gaussian graphical models are a popular tool for modeling the interaction of a collection of Gaussian random variables. In Gaussian graphical models, nodes represent random variables and edges model conditional (in)dependence among the variables [WJ08]. Over the last decade, significant efforts have been directed towards algorithms for learning *sparse* graphical models. Mathematically, let Σ^* denote the positive definite covariance matrix of p Gaussian random variables, and let $\Theta^* = (\Sigma^*)^{-1}$ be the corresponding precision matrix. Then, $\Theta^*_{ij} = 0$ implies that the i^{th} and j^{th} variables are conditionally independent given all other variables and the edge (i, j) does not exist in the underlying graph. The basic modeling assumption is that Θ^* is sparse, i.e., such graphs possess only a few edges. Such models have been fruitfully used in several applications including astrophysics [PWZO16], scene recognition [SS16], and genomic analysis [YL13].

Numerous algorithms for sparse graphical model learning – both statistically as well as computationally efficient – have been proposed in the machine learning literature [FHT08, MH12, BGd08, HDRS11]. Unfortunately, sparsity is a simplistic first-order model and is not amenable to modeling more complex interactions. For instance, in certain scenarios, only some of the random variables are directly observed, and there could be relevant *latent* interactions to which we do not directly have access.

KDD MLG' 17, August 2017, Halifax, NS, Canada

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

The existence of latent variables poses a significant challenge in graphical model learning since they can confound an otherwise sparse graphical model with a dense one. This scenario is illustrated in Figure 1. Here, nodes with solid circles denote the observed variables, and solid black edges are the "true" edges in the graphical model. One can see that the "true" graph is rather sparse. However, if there is even a single unobserved (hidden) variable denoted by the node with the broken red circle, then it will induce dense, apparent interactions between nodes that are otherwise disconnected; these are denoted by the dotted black lines.

A flexible and elegant method to learn latent variables in graphical models was proposed by [CPW12]. At its core, the method imposes a superposition structure in the observed precision matrix as the sum of *sparse* and *low-rank* matrices, i.e., $\Theta^* = S^* + L^*$. Here, Θ^*, S^*, L^* are $p \times p$ matrices where p is the number of variables. The matrix S^* specifies the conditional observed precision matrix given the latent variables, while L^* encodes the effect of marginalization over the latent variables. The rank of L^*, r, is equal to the number of latent variables and we assume that r is much smaller than p.

To learn such a superposition model, [CPW12] propose a regularized maximum-likelihood estimation framework, with ℓ_1-norm and nuclear norm penalties as regularizers; these correspond to *convex* relaxations of the sparsity and rank constraints, respectively. Using this framework, they prove that such graphical models can be learned with merely $n = O(pr)$ random samples. However, this statistical guarantee comes at a steep computational price; the framework involves solving a semidefinite program (SDP) with p^2 variables and is computationally very challenging. Several subsequent works [MXZ13, HDR$^+$14] have attempted to provide faster algorithms, but all known (provable) methods involve at least *cubic* worst-case running time.

1.2 Our contributions

In this paper, we provide a new class of *fast* algorithms for learning latent variables in Gaussian graphical models. Our algorithms are (i) *provably statistically efficient*: they achieve the optimal sample complexity of latent variable learning; (ii) *provably computationally efficient*: they are linearly convergent, and their per-iteration time is close to the best possible.

We clarify the above claims using some notation. Suppose that we observe samples $X_1, X_2, \ldots, X_n \overset{i.i.d}{\sim} \mathcal{N}(0, \Sigma)$ where each $X_i \in \mathbb{R}^p$. Let $C = \frac{1}{n}\sum_{i=1}^{n} X_i X_i^T$ denote the sample covariance matrix, and $\Theta^* = (\Sigma^*)^{-1}$ denote the true precision matrix; as above, we assume that $\Theta^* = S^* + L^*$. We will exclusively function in the high-dimensional regime where $n \ll p^2$. Our sole focus is on learning the low-rank part from samples; i.e., we pre-suppose that the sparse part, S^* is a known positive definite matrix, while the low-rank

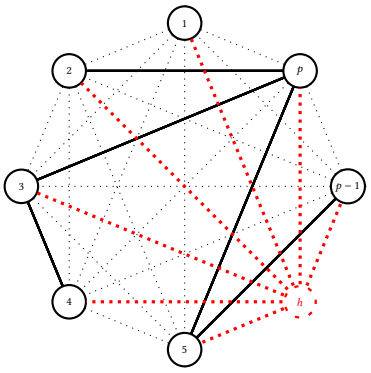

Figure 1: Illustration of effects of latent variable in graphical model learning. Solid edges represent "true" conditional dependence, while dotted edges represent apparent dependence due to the presence of the latent variable h.

part, L^* is unknown (with rank $r \ll p$). This models the situation where the"true" edges between the observed variables are known *a priori*, but there exists latent interaction among these variables that needs to be discovered.

We estimate L^* in the high-dimensional regime where $n \ll p^2$ by attempting to solve a *non-convex* optimization problem, following the formulation of [HZZ16]. We label this as *LVM*, short for *Latent Variable Gaussian Graphical Modeling*:

$$\begin{aligned} \min_{L} \quad & F(L) = -\log\,\det(S^* + L) + \langle S^* + L, C\rangle \\ \text{s.t.} \quad & \text{rank}(L) \le r,\ L \succeq 0. \end{aligned} \tag{1}$$

Above, $\langle ., . \rangle$ denotes the standard Frobenius inner product in matrix space, $\succeq 0$ denotes membership in the positive semi-definite (psd) cone, and the objective function $F(L)$ denotes the negative log likelihood of the samples. Problem (1) is highly non-convex due to the rank constraint, $\text{rank}(L) \le r$. Moreover, the log-det loss function is highly nonlinear and challenging to handle. As mentioned earlier, most known (provable) methods solve a convex relaxation of this problem, but suffer from high computational costs.

In contrast, we solve (1) without convex relaxation. Specifically, we propose two non-convex algorithms for solving (1). Our first algorithm, that we call *LVM with Exact Projections*, or EP-LVM, performs (non-convex) projected gradient descent on the objective function $F(L)$, together with the low-rank and psd constraints. This algorithm yields sample-optimal results, but its running time can be cubic, $\Omega(p^3)$, in the number of variables. Our second algorithm, that we call *LVM with Approximate Projections*, or AP-LVM, performs a variant of projected gradient descent with (deliberately) inaccurate projections onto the constraints. Interestingly, this algorithm also yields sample-optimal results, and its running time is *nearly quadratic*, $\widetilde{O}(p^2)$, in the dimension p for a fixed number of latent variables. To the best of our knowledge, this is the fastest *universal*[1] algorithm for solving (1).

Both our proposed algorithms enjoy the following benefits:

Sample efficiency. For both algorithms, the sample complexity (i.e., number of samples in order to achieve a desired estimation error ϑ) for learning a rank-r latent variable model in p variables scales as $n = O(pr)$, and this matches those of the best available methods.

Linear convergence. We provide rigorous analysis to show that both our proposed algorithms enjoy global linear convergence with no specific initialization step.

Proper learning. Our algorithms are examples of *proper* learning methods, in the sense that their output is a rank-r estimate of the true latent variable model. In contrast, methods based on convex relaxation often fail to do this and return a high-rank estimate, thus potentially having a negative effect on interpretability of the discovered latent variables.

1.3 Techniques

Our first algorithm is a variant of the *singular value projection* approach of [JMD10], with an extra psd projection step. Our second algorithm is a variant of approximate subspace-IHT [HIL16] and uses a careful combination of approximate singular value decomposition techniques. While our proposed methods are structurally similar to these previously proposed methods, their analysis is considerably different; we elaborate on this below.

Our technique for establishing linear convergence of our first algorithm (EP-LVM) is based on bounding the restricted strong convexity/smoothness (RSC/RSS) constants [NYWR11] of the objective function $F(L)$ in (1). The key observation is that $F(L)$ is globally strongly convex, and when restricted to any compact psd cone, it also satisfies strong smoothness. The above analysis technique is fairly standard [JTK14]. However, unlike in previously considered scenarios, the RSS/RSC constants of $F(L)$ are harder to bound. These may vary across iterations, and depend on several properties of the true precision matrix Θ^*. Therefore, additional effort is required to establish global linear convergence.

As a byproduct of this analysis, we show that with $n = O(pr)$ independent samples, EP-LVM returns an estimate up to constant error. Moreover, we show that EP-LVM provides the best empirical performance (in terms of estimation error) among all considered methods.

However, since EP-LVM performs an exact eigenvalue decomposition (EVD) *per iteration*, its overall running time can be slow since it incurs cubic per-iteration running time. Our second algorithm (AP-LVM) resolves this issue. The basic idea is to replace exact EVDs with *approximate* low-rank projections in each iteration. However, it is known [HIS15] that a straightforward replacement of all EVDs with approximate low-rank projections within non-convex projected gradient descent is not a successful strategy.

In order to guarantee convergence, we use a careful combination of *tail* and *head* approximate low-rank projections in each iteration [HIL16]. This enables us to improve the overall running time to $\widetilde{O}(p^2 r)$. Moreover, the statistical accuracy matches that of EP-LVM in theory (up to constants) as well as in practice (negligible loss in performance). We show that both EP-LVM and AP-LVM provide better empirical performance than convex methods.

[1]The running time of some other existing algorithms achieve similar scaling in p, but also depend adversely on matrix properties such as condition number and/or the minimum singular value.

Table 1: Summary of our contributions, and comparison with existing methods. Here, $\gamma = \sqrt{\frac{\sigma_r}{\sigma_{r+1}} - 1}$ represents the spectral gap parameter in intermediate iterations. The overall running time of the ADMM approach is marked as poly(p) since the precise rate of convergence is unknown.

Algorithm	Reference	Running Time	Spectral dependency	Output rank
SDP	[CPW12]	poly(p)	Yes	$\gg r$
ADMM	[MXZ13]	poly(p)	Yes	$\gg r$
QUIC & DIRTY	[YR13]	$\widetilde{O}(p^3)$	Yes	$\gg r$
SVP	[JTK14]	$\widetilde{O}(p^3)$	No	$\gg r$
Factorized	[BKS16]	$\widetilde{O}(p^2 r/\gamma)$	Yes	r
EP-LVM	**This paper**	$\widetilde{O}\left(p^3\right)$	**No**	**r**
AP-LVM	**This paper**	$\widetilde{O}(p^2 r)$	**No**	**r**

Table 1 provides a summary of the theoretical properties of our methods, and contrasts them with other existing methods for latent variable modeling.

2 RELATION TO PRIOR WORK

Learning graphical models in high dimensional settings have been of special interest, and most existing works assume some sort of low-dimensional structure on the covariance or precision matrix [CRZ16]. Among all the structured models, sparse graphical model learning has received the most attention. The typical approach for learning sparse graphical models is to obtain a regularized maximum likelihood (ML) estimate given the observations. The work of [RWRY11] establishes the statistical efficiency of the regularized ML estimate. Parallel to such statistical analysis is the development of efficient computational techniques for solving the regularized ML estimate [FHT08, MH12, RRG$^+$12].

To capture latent interactions between variables, more complex structures (beyond mere sparsity) are necessary. [CPW12] propose a superposition of sparse and low-rank structure in the precision matrix. To solve this latent variable problem, they introduce an extra nuclear norm term to the regularized ML objective function as a convex surrogate of the rank. However, they propose using a generic semi-definite programming (SDP) solver, which is very cumbersome for even moderate size problem. Subsequently, the authors in [MXZ13] have proposed Alternating Direction Method of Multipliers (ADMM) which scales to relatively large size problems. The work of [YR13, HDR$^+$14] also consider general superposition structures in precision matrix estimation. However, the running time of these methods is still cubic in the number of variables (p^2).

Problem (1) is an instance of the more general problem of *low-rank matrix recovery*. Broadly, three classes of approaches for low-rank matrix recovery exist. The first (and most popular) class is based on convex relaxation [CR12, MXZ13, HDR$^+$14]; while their statistical properties are well-established, such methods often suffer from high computational costs.

Methods in the second class of approaches are fundamentally non-convex, and are based on the approach of [BM03]. In these algorithms, the psd rank-r matrix L is factorized as $L = UU^T$, where $U \in \mathbb{R}^{p\times r}$. Using this idea removes the difficulties caused by the non-convex rank constraint; however, the objective function is not convex anymore and proving convergence remains tricky. Nevertheless, under certain conditions, such methods succeed and have recently gained in popularity in the machine learning literature [TBS$^+$16, BKS16, PKCS16, CW15, ZL15]. In general, these methods need a careful spectral initialization that usually involves a full singular value decomposition, and their convergence depends heavily on the condition number of the input as well as other spectral properties of the true low-rank component.

The third class of methods are also non-convex. Unlike the second class, they do not factorize the optimization variable, L, but instead use low-rank projections within the classical gradient descent framework. This approach was introduced by [JMD10] for matrix recovery from linear measurements, and was later modified for general M-estimation problems with well-behaved objective functions [JTK14]. In principle, the approach of [JTK14] can be used to solve (1) (using a similar analysis of the RSS/RSC constants as in this paper.) However, it is an improper learning algorithm, and the rank of the estimate of L^* is several times larger than the target rank. Moreover, each iteration is computationally expensive, since it involves computing an SVD in each iteration.

Our contributions in this paper fall under the third category. We first propose an iterative PSD projection algorithm similar to that of [JMD10] but for the specific problem stated in (1). We then accelerate this algorithm (with no loss in statistical performance) using the approximate low-rank projections method of [HIL16].

3 PRELIMINARIES

Throughout this paper, the minimum and maximum eigenvalues of the sparse matrix S^* will be denoted by S_p and S_1 respectively. We use $\|A\|_2$ and $\|A\|_F$ for spectral norm and Frobenius norm of a matrix A, respectively. We denote A_r as the best rank-r approximation (in Frobenius norm) of a given matrix A. In addition, $\lambda_1(A), \lambda_p(A)$ denote the maximum and minimum eigenvalues of $A \in \mathbb{R}^{p\times p}$ respectively. For any subspace $U \subset \mathbb{R}^{p\times p}$, we denote $\mathcal{P}_U$ as the orthogonal projection operator onto U.

Our analysis will rely upon on the following definition [NYWR11, JTK14, YLZ14].

ALGORITHM 1: EP-LVM

Input: Matrices S^* and C, rank r, step size η.
Output: Estimates $\widehat{L}$, $\widehat{\Theta} = S^* + \widehat{L}$.
Initialization: $L^0 \leftarrow 0$, $t \leftarrow 0$;
repeat
  $L^{t+1} = \mathcal{P}_r^+ \left(L^t - \eta \nabla F(L^t)\right)$;
  $t \leftarrow t + 1$;
until $t \leq T$;

Definition 3.1. A function f satisfies the Restricted Strong Convexity (RSC) and Restricted Strong Smoothness (RSS) conditions if for all $L_1, L_2 \in \mathbb{R}^{p\times p}$ such that $\text{rank}(L_1) \leq r, \text{rank}(L_2) \leq r$:

$$\frac{m_r}{2}\|L_2 - L_1\|_F^2 \leq f(L_2) - f(L_1) - \langle \nabla f(L_1), L_2 - L_1\rangle \leq \frac{M_r}{2}\|L_2 - L_1\|_F^2, \tag{2}$$

where m_r and M_r are called the RSC and RSS constants respectively.

We denote $\mathbb{U}_r$ as the set of all rank-r matrix subspaces, i.e., subspaces of $\mathbb{R}^{p\times p}$ that are spanned by any r atoms of the form uv^T where $u, v \in \mathbb{R}^p$ are unit-norm vectors.

We will also employ the idea of *head* and *tail* projection introduced by [HIS15], and instantiated in the context of low-rank approximation by [HIL16].

Definition 3.2 (Approximate tail projection). Let $c_{\mathcal{T}} > 1$ be a constant. Then $\mathcal{T} : \mathbb{R}^{p\times p} \to \mathbb{U}_r$ is a $c_{\mathcal{T}}$-approximate tail projection algorithm if for all $L \in^{p\times p}$, $\mathcal{T}$ returns a subspace $W = \mathcal{T}(L)$ that satisfies: $\|L - \mathcal{P}_W L\|_F \leq c_{\mathcal{T}}\|L - L_r\|_F$.

Definition 3.3 (Approximate head projection). Let $0 < c_{\mathcal{H}} < 1$ be a constant. Then $\mathcal{H} : \mathbb{R}^{p\times p} \to \mathbb{U}_r$ is a $c_{\mathcal{H}}$-approximate head projection if for all $L \in^{p\times p}$, the returned subspace $V = \mathcal{H}(L)$ satisfies: $\|\mathcal{P}_V L\|_F \geq c_{\mathcal{H}}\|L_r\|_F$.

4 ALGORITHMS AND ANALYSIS

First, we present our projected gradient-descent algorithm to solve 1. This algorithm provides the best sample complexity (both theoretical and empirical) among all existing approaches. Our algorithm, that we call *LVM with exact projections* (EP-LVM), is described in pseudocode form as Alg 1.

In Alg (1), the exact projection step, $\mathcal{P}_r^+(\cdot)$ denotes projection onto the space of rank-r psd matrices. This is implemented through performing an exact eigenvalue decomposition (EVD) of the argument and selecting the nonnegative eigenvalues and corresponding eigenvectors [HM12]. The gradient of the objective function $F(L)$ in (1) can be calculated as:

$$\nabla F(L) = -(S^* + L)^{-1} + C = -\Theta^{-1} + C. \tag{3}$$

Since L is a low-rank matrix with rank r, it can be factorized as $L = UU^T$ for some $U \in \mathbb{R}^{p\times r}$. Hence, to calculate efficiently the inverse in (3), we utilize the low-rank structure of L by applying the Woodbury matrix identity:

$$(S^* + L)^{-1} = S^{*-1} - S^{*-1}U\left(I + U^T S^{*-1} U\right)^{-1} U^T S^{*-1}.$$

We now provide our first main theoretical result, supporting the statistical and computational efficiency of EP-LVM. In particular, we derive an upper bound on the estimation error of the low-rank matrix at each iteration (Please see [SH17] for all the proofs).

Theorem 4.1 (Linear convergence of EP-LVM). *Assume that the objective function $F(L)$ satisfies the RSC/RSS conditions with constant M_{3r} and m_{3r}. Let J_t denote the subspace formed by the span of the column spaces of the matrices L^t, L^{t+1}, and L^*. In addition, assume that $1 \leq \frac{M_{3r}}{m_{3r}} \leq \frac{2}{\sqrt{3}}$. Choose step size as $\frac{0.5}{M_{3r}} \leq \eta \leq \frac{1.5}{m_{3r}}$. Then, EP-LVM outputs a sequence of estimates L^t such that:*

$$\|L^{t+1} - L^*\|_F \leq \rho\|L^t - L^*\|_F + 2\eta\|\mathcal{P}_{J_t}\nabla F(L^*)\|_F, \tag{4}$$

where $\rho = 2\sqrt{1 + M_{3r}^2\eta^2 - 2m_{3r}\eta} < 1$.

We will show below that the second term on the right hand side of this inequality is upper-bounded by an arbitrarily small constant with sufficient number of samples. Also, the first term decreases exponentially with iteration count. Overall, after $T = O\left(\log_{1/\rho}\left(\frac{\|L^*\|_F}{\vartheta}\right)\right)$ iterations, we obtain an upper-bound of $O(\vartheta)$ on the total estimation error, indicating linear convergence.

Next, we provide bounds on the RSS/RSC constants of $F(L)$, justifying the assumptions made in Theorem 4.1.

Theorem 4.2 (Bounding RSC/RSS constants.). *Let the number of samples scaled as $n = O\left(\frac{1}{\delta^2}\left(\frac{\eta}{1-\rho}\right)^2 rp\right)$ for some small constant $\delta > 0$ and ρ defined above. Also, assume that*

$$S_p \leq S_1 \leq \sqrt{\frac{2}{\sqrt{3}}}S_p - \left(1 + \sqrt{r}\right)\|L^*\|_2 - \delta.$$

Then, the loss function $F(L)$ satisfies RSC/RSS conditions with constants m_{3r} and M_{3r} that satisfy the assumptions of Theorem 4 in each iteration.

The above theorem states that convergence of our method is guaranteed when the eigenvalues of S^* are roughly of the same magnitude, and large when compared to the spectral norm of L^*. We believe that this is merely a sufficient condition arising from our proof technique, and our numerical evidence shows that the algorithm succeeds for more general S^* and L^*.

Time complexity. Each iteration of EP-LVM needs a full EVD, which requires cubic running time. Since the total number of iterations is logarithmic, the overall running time scales as $\widetilde{O}(p^3)$.

For large p, the cubic running time of EP-LVM can be very challenging. To alleviate this issue, one can instead attempt to replace the full EVD in each iteration with an *ε-approximate* low-rank psd projection; it is known that such projections can computed in $O(p^2 \log p)$ time [CW17]. However, a naïve replacement of the EVD with an ε-approximate low-rank projection method does not lead to algorithms with rigorous convergence guarantees.[2]

Instead, we use a combination of approximate *tail* and *head* projections, as suggested in [HIL16]. The high level idea is that the use of *two* inaccurate low-rank projections instead of one, if done carefully, will balance out the errors and will result in provable convergence. The full algorithm, that we call *LVM with approximate projections* (AP-LVM), is described in pseudocode form as Alg. 2.

[2]Indeed, algorithms with only "tail-approximate" projections can be shown to get stuck at a solution arbitrarily far from the true estimate, even with a large number of samples; see Section 2 of [HIS15].

ALGORITHM 2: AP-LVM

Input: Matrices S^* and C, rank r, step size η.
Output: Estimates $\widehat{L}, \widehat{\Theta} = S^* + \widehat{L}$.
Initialization: $L^0 \leftarrow 0$, $t \leftarrow 0$;
repeat
  $L^{t+1} = \mathcal{T}\left(L^t - \eta \mathcal{H}\left(\nabla F(L^t)\right)\right)$;
  $t \leftarrow t + 1$;
until $t \leq T$;

Note that we do not impose a psd projection within every iteration. If an application requires a psd matrix as the output (i.e., if proper learning is desired), then we can simply post-process the final estimate L^T by retaining the nonnegative eigenvalues (and corresponding eigenvectors) through an exact EVD. We note that this EVD can be done only once, and is applied to the final output of Alg 2. This is itself a rank-r matrix, therefore leaving the overall asymptotic running time unchanged.

The choice of approximate low-rank projections is flexible, as long as the approximate tail and head projection guarantees are satisfied. We note that tail-approximate low-rank projection algorithms are widespread in the literature [CW13, MD09, RST09]; however, head-approximate projection algorithms (or at least, algorithms with head guarantees) are less common.

We focus on the randomized Block Krylov SVD method (BK-SVD) method of [MM15]. BK-SVD generates a rank-r subspace approximating the top right r singular vectors of a given input matrix. Moreover, for constant approximation factors, its running time is $\widetilde{O}(p^2 r)$, *independent* of any spectral properties of the input matrix. Formally, let $A \in \mathbb{R}^{n\times n}$ be a given matrix and let A_r denote its best rank-r approximation. Then BK-SVD generates a matrix $B = ZZ^T A$ which is the projection of A onto the column space of matrix Z with orthonormal vectors $z_1, z_2, \ldots, z_r$. Moreover, with probability 99/100, we have:

$$\|A - B\|_F \leq c_{\mathcal{T}} \|A - A_r\|_F. \tag{5}$$

where $c_{\mathcal{T}} > 1$ is the tail projection constant. Equation (5) is equivalent to the tail approximation guarantee according to Definition 3.2.

In addition to (5), [MM15] also provide the so-called *per vector* approximation guarantee for BK-SVD with probability 99/100:

$$|u_i^T A A^T u_i - z_i A A^T z_i| \leq (1 - c_{\mathcal{H}})\sigma_{r+1}^2,$$

where u_i are the right eigenvectors of A and $c_{\mathcal{H}} < 1$ is the head projection constant. [HIL16] show that the above property implies the head approximation guarantee:

$$\|B\|_F \geq c_{\mathcal{H}} \|A_r\|_F.$$

Therefore, in AP-LVM we invoke the BK-SVD method for both head and tail projections[3]. Using BK-SVD as the approximate low-rank projection method of choice, we now provide our second main theoretical result supporting the statistical and computational efficiency of AP-LVM.

[3]We note that since the BK-SVD algorithm is randomized while our definitions of tail and head guarantees are deterministic. Fortunately, the running time of BK-SVD depends only logarithmically on the failure probability of the algorithm, and therefore a union bound argument over the iterations of AP-LVM is required to precisely prove algorithmic correctness.

Theorem 4.3 (Linear convergence). *Assume that the objective function $F(L)$ satisfies the RSC/RSS conditions with constants M_{2r} and m_{2r}. In addition, assume that $1 \leq \frac{M_{2r}^2}{m_{2r}^2} \leq \frac{1}{1-\rho_0^2}$ where $\rho_0 = \frac{1}{1+c_{\mathcal{T}}} - \sqrt{1-\eta_0^2}$ and $\eta_0 = \left(c_{\mathcal{H}} m - \sqrt{1 + M_{2r}^2 - 2m_{2r}}\right)$. Choose step size as $\frac{1-\rho_0^2}{M_{2r}} \leq \eta \leq \frac{1+\rho_0^2}{m_{2r}}$. Let V_t be the subspace returned by the head approximate projection $\mathcal{H}(\cdot)$ applied to the gradient. Then, for any $t > 0$, AP-LVM outputs a sequence of estimates L^t that satisfy:*

$$\|L^{t+1} - L^*\|_F \leq \rho_1 \|L^t - L^*\|_F + \rho_2 \|\mathcal{P}_{V_t} \nabla F(L^*)\|_F, \tag{6}$$

where $\rho_1 = \left(\sqrt{1 + M_{2r}^2\eta^2 - 2m_{2r}\eta} + \sqrt{1-\eta_0^2}\right)(1 + c_{\mathcal{T}})$ and $\rho_2 = \left(\frac{\eta_0}{\sqrt{1-\eta_0^2}} + 1\right)(1 + c_{\mathcal{T}})$.

A similar calculation as before shows that AP-LVM converges after $T = O\left(\log\left(\frac{\|L^*\|_F}{\vartheta}\right)\right)$ iterations. AP-LVM is structurally similar to the approximate subspace-IHT algorithm of [HIL16]. However, their proofs are specific to least-squares loss functions. On the other hand, the loss function $F(L)$ for recovering latent variables is complicated[4] and in general the RSS/RSC constants can vary across iterations. Therefore, considerable effort is needed to prove algorithm convergence. First, we provide conditions under which the assumption of RSC/RSS in Theorem 4.3 are satisfied.

Theorem 4.4 (Bounding RSC/RSS constants). *Let n scaled as $n = O\left(\frac{1}{\delta'^2}\left(\frac{\rho_2}{1-\rho_1}\right)^2 rp\right)$ for some small constant $\delta' > 0$, with ρ_1 and ρ_2 as defined in theorem 4.3. Also, assume that,*

$$\|L^*\|_2 \leq \frac{1}{1+\sqrt{r}}\left(\frac{S_p}{1+\sqrt{1-\rho_0^2}} - \frac{S_1\sqrt{1-\rho_0^2}}{1+\sqrt{1-\rho_0^2}} - \frac{c_3\rho_2}{1-\rho_1}\sqrt{\frac{rp}{n}}\right). \tag{7}$$

Finally, assume that:

$$S_p \leq S_1 \leq \frac{1}{\sqrt{1-\rho_0^2}}(S_p - a') - \left(1+\sqrt{r}\right)\|L^*\|_2 - \delta'$$

where $0 < a' \leq \left(1+\sqrt{r}\right)\|L^*\|_2 + \delta'$ for some $\delta' > 0$. Then, the loss function $F(L)$ satisfies RSC/RSS conditions with constants m_{2r} and M_{2r} that satisfy the assumptions of Theorem 4.3 in each iteration.

Theorem 4.4 specifies a family of true precision matrices $\Theta^* = S^* + L^*$ that can be provably estimated using our approach with an optimal number of samples. Note that since we do not perform psd projection within AP-LVM, it is possible that some of the eigenvalues of L^t are negative. Next, we show that with high probability, the absolute value of the minimum eigenvalue of L^t is small.

Theorem 4.5. *Under the assumptions in Theorem 4.4 on L^*, if we use AP-LVM to generate a rank r matrix L^t for all $t = 1, \ldots, T$, then with high probability the minimum eigenvalue of L^t satisfies: $\lambda_p(L^t) \geq -a'$ where $0 < a' \leq \left(1+\sqrt{r}\right)\|L^*\|_2 + \frac{c_3\rho_2}{1-\rho_1}\sqrt{\frac{rp}{n}}$.*

[4]Indeed, $F(L)$ is not well-defined everywhere, e.g. at matrices L that have large negative eigenvalues.

Time complexity. Each iteration of AP-LVM needs a head and a tail projection on the rank $2r$ and rank r matrices respectively. According to [MM15], these operations takes $k' = O\left(\frac{p^2 r \log p}{\sqrt{\varepsilon}}\right)$ for error ε. Since the total number of iterations is once again logarithmic, the overall running time scales as $\widetilde{O}(p^2 r)$.

The above analysis shows that our proposed algorithms for discovering latent variables are linearly convergent. We now show that they converge to the true underlying low-rank matrix. The quality of the estimates in Theorems 4.1 and 4.3 is upper-bounded by the gradient terms $\|\mathcal{P}_{J_t} \nabla F(L^*)\|_F$ and $\|\mathcal{P}_{V_t} \nabla F(L^*)\|_F$ in (4) and (6), respectively, within each iteration. The following theorem bounds these gradient terms in terms of the number of observed samples.

Theorem 4.6. *Under the assumptions of Theorem 4.1, for any fixed t we have:*

$$\|\mathcal{P}_{J_t} \nabla F(L^*)\|_F \leq c_2 \sqrt{\frac{rp}{n}}, \tag{8}$$

Similarly, under the assumptions of Theorem 4.3,

$$\|\mathcal{P}_{V_t} \nabla F(L^*)\|_F \leq c_3 \sqrt{\frac{rp}{n}}, \tag{9}$$

Both hold with probability at least $1 - 2\exp(-p)$ where $c_2, c_3 > 0$ are absolute constants.

Sample complexity. Plugging in the upper bounds in (8) and (9) into Theorems 4.2 and 4.4, the sample complexity of both algorithms scales as $n = O(pr)$ to achieve constant estimation error. This matches the number of degrees of freedom of a $p \times p$ matrix with rank r.

5 EXPERIMENTS

We provide a range of numerical experiments supporting our proposed algorithms and comparing with existing convex approaches. Our comparisons is with the regularized maximum likelihood approach of [CPW12], which we solve using CVX [GBY08]. The second algorithm that we have used is a modification of the ADMM-type method proposed by [MXZ13]. We assume that our algorithms are provided with the rank parameter r, and have manually tuned step-sizes/regularization parameters of all algorithms to achieve best possible performance.

Synthetic data. we use a diagonal matrix with positive values for the (known) sparse part, S^*. For a given number of observed variables p, we set $r = 5\%$ as the number of latent variables. We then follow the method proposed in [MXZ13] for generating the sparse and low-rank components S^* and L^*. For simplicity, we impose the sparse component to be psd by forcing it to be diagonal. All reported results on synthetic data are the average of 5 independent Monte-Carlo trials. Our observations comprise n samples, $x_1, x_2, \ldots, x_n \overset{i.i.d}{\sim} \mathcal{N}(0, (S^* + L^*)^{-1})$. In our experiments, we used a full SVD as projection step in EP-LVM. (Due to numerical stability, we use SVD rather than EVD.) For AP-LVM, we compare two versions: *AP-LVM(1)* denotes the use of BK-SVD for the approximate tail and head projections, while *AP-LVM(2)* denotes the use of the more well-known (but spectrum-dependent) Lanczos method for these projections.

In the first experiment, we set $p = 100$, $n = 400p$, and $r = 5$. Table 2 lists several metrics that we use for algorithm comparison. An algorithm terminates if it satisfies one of two conditions: the evaluated objective function in the estimated L in each iteration falls below the true negative log likelihood (NLL) (i.e., $F(L^*)$), or the total number of iterations exceeds 600. From Table 2, we see that both EP-LVM, AP-LVM(1) and AP-LVM(2) produce better estimates of L compared to ADMM and CVX, with AP-LVM(1) and AP-LVM(2) having the edge in running time and EP-LVM having the edge in accuracy. Note that the convex methods strictly produce an estimate of rank larger than 5 (indicating that they are *improper* learning methods). As anticipated, the total running time with CVX is much larger than other algorithms. Finally, the estimated objective function for our proposed algorithms is very close to the optimal (true) objective function compared to ADMM and CVX.

We increase the dimension to $p = 1000$ and reported the same metrics in Table 3 similar to Table 2. Since CVX cannot solve the problem with size $p = 1000$, we did not report its results. Again, we get the same conclusions as Table 2; however, the improvement obtained by AP-LVM in terms of running time is considerably magnified.

In addition, Tables 4 and 5 shows the same experiment discussed in Tables 2 and 3 but for small number of samples, $n = 50p$.

In Figures 2 (a) and (b), we graphically compare four algorithms in terms of the relative error of the estimated L in Frobenius norm versus the "oversampling" ratio n/p. In this experiment, we fixed $p = 100$ in (a) and $p = 1000$ in (b) and vary n. We observe that EP-LVM, AP-LVM(1), and AP-LVM(2) estimate the low-rank matrix even for the regime where n is very small, whereas both ADMM and CVX does not produce very meaningful results.

Real data. We evaluate our methods through the *Rosetta* gene expression data set [HMJ+00]. This data set includes 301 samples with 6316 variables. We run the ADMM algorithm by [MXZ13] with $p = 1000$ variables and obtained an estimate of the sparse component S^*. Then we used S^* as the input for EP-LVM, AP-LVM(1) and AP-LVM(2). The target rank for all three algorithms is set to be the same as that returned by ADMM. In Figure 2 plot (c), we illustrate the NLL for these three algorithms versus wall-clock time (in seconds) over 50 iterations. We observe that all three algorithms demonstrate linear convergence, as predicted in the theory. Among the three algorithms, AP-LVM(1) obtains the quickest rate of decrease of the objective function.

ACKNOWLEDGEMENTS

This work was supported in part by grants from the National Science Foundation and NVIDIA.

REFERENCES

[BGd08] O. Banerjee, L. Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *J. Machine Learning Research*, 9(Mar):485–516, 2008.

[BKS16] S. Bhojanapalli, A. Kyrillidis, and S. Sanghavi. Dropping convexity for faster semi-definite optimization. In *29th Annual Conference on Learning Theory*, pages 530–582, 2016.

[BM03] S. Burer and R. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95(2):329–357, 2003.

[BV04] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[CPW12] V. Chandrasekaran, P. Parrilo, and A. Willsky. Latent variable graphical model selection via convex optimization. *The Annals of Statistics*, 40(4):1935–1967, 2012.

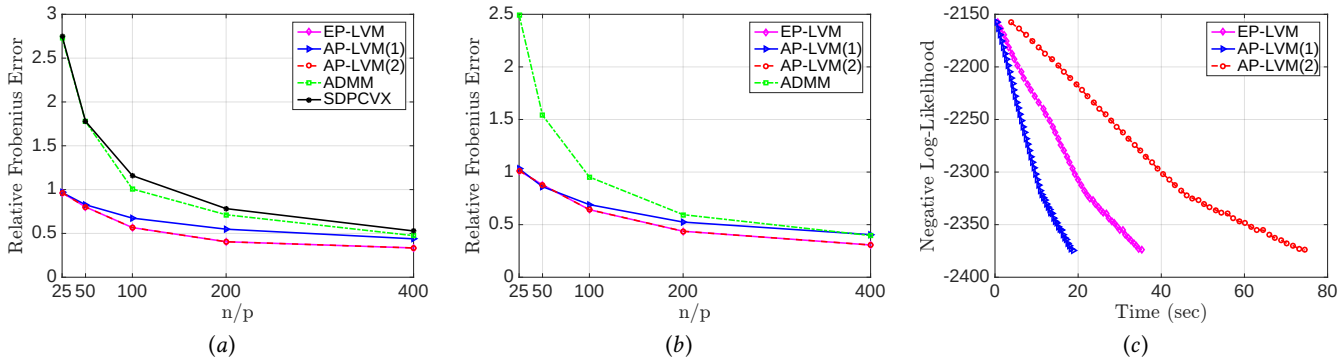

Figure 2: Comparison of algorithms both in synthetic and real data. (a) relative error of L in Frobenius norm with $p = 100$. (b) relative error of L in Frobenius norm with $p = 1000$. (c) NLL versus time in Rosetta data set with $p = 1000$.

Table 2: Comparison of different algorithms for $p = 100$ and $n = 400p$. NLL stands for negative log-likelihood.

Alg	Estimated NLL	Estimated NLL w/reg.	True NLL	True NLL w/reg.
EP-LVM	$-8.884646e+01$	–	$-8.884365e+01$	–
AP-LVM(1)	$-8.883135e+01$	–	$-8.884365e+01$	–
AP-LVM(2)	$-8.884646e+01$	–	$-8.884365e+01$	–
ADMM	$-9.374270e+01$	$-9.372003e+01$	–	$-8.639705e+01$
CVX	$-8.891208e+01$	$-8.889070e+01$	–	$-8.883386e+01$
Alg	Relative error	Per-iteration time	Total time	Output rank
EP-LVM	$3.341525e-01$	$4.982088e-03$	$2.386314e+00$	5
AP-LVM(1)	$4.381772e-01$	$5.626742e-03$	$1.886186e+00$	5
AP-LVM(2)	$3.341525e-01$	$1.017454e-02$	$4.710893e+00$	5
ADMM	$4.746382e-01$	$9.093788e-03$	$5.069543e+00$	49
CVX	$5.281450e-01$	-	$8.505811e+02$	100

Table 3: Comparison of different algorithms for $p = 1000$ and $n = 400p$.

Alg	Estimated NLL	Estimated NLL w/reg.	True NLL	True NLL w/reg.
EP-LVM	$-2.640322e+03$	–	$-2.640204e+03$	–
AP-LVM(1)	$-2.640186e+03$	–	$-2.640204e+03$	–
AP-LVM(2)	$-2.640322e+03$	–	$-2.640204e+03$	–
ADMM	$-2.640565e+03$	$-2.640000e+03$	–	$-2.522379e+03$
Alg	Relative error	Per-iteration time	Total time	Output rank
EP-LVM	$3.065917e-01$	$2.557906e-01$	$1.534744e+02$	50
AP-LVM(1)	$4.048012e-01$	$9.880854e-02$	$5.928513e+01$	50
AP-LVM(2)	$3.065917e-01$	$3.759073e-01$	$2.255444e+02$	50
ADMM	$3.962763e-01$	$5.990084e-01$	$3.397271e+02$	350

[CR12] Emmanuel Candes and Benjamin Recht. Exact matrix completion via convex optimization. *Communications of the ACM*, 55(6):111–119, 2012.

[CRZ16] T. Cai, Z. Ren, and H. Zhou. Estimating structured high-dimensional covariance and precision matrices: Optimal rates and adaptive estimation. *Electronic Journal of Statistics*, 10(1):1–59, 2016.

[CSPW09] V. Chandrasekaran, S. Sanghavi, P. Parrilo, and A. S. Willsky. Sparse and low-rank matrix decompositions. In *Proc. Allerton Conf. on Comm., Contr., and Comp.*, pages 962–967, 2009.

[CW13] Kenneth L Clarkson and David P Woodruff. Low rank approximation and regression in input sparsity time. In *Proc. ACM Symp. Theory of Comput.*, pages 81–90. ACM, 2013.

[CW15] Y. Chen and M. Wainwright. Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees. *arXiv preprint arXiv:1509.03025*, 2015.

[CW17] K. Clarkson and D. Woodruff. Low-rank psd approximation in input-sparsity time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2061–2072. SIAM, 2017.

[FHT08] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[GBY08] M. Grant, S. Boyd, and Y. Ye. Cvx: Matlab software for disciplined convex programming, 2008.

[HDR+14] C. Hsieh, I. Dhillon, P. Ravikumar, S. Becker, and P. Olsen. Quic & dirty: A quadratic approximation approach for dirty statistical models. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 2006–2014, 2014.

[HDRS11] C. Hsieh, I. Dhillon, P. Ravikumar, and M. Sustik. Sparse inverse covariance matrix estimation using quadratic approximation. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 2330–2338, 2011.

[HIL16] C. Hegde, I. Indyk, and S. Ludwig. Fast recovery from a union of subspaces. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, 2016.

Table 4: Comparison of different algorithms for $p = 100$ and $n = 50p$. NLL stands for negative log-likelihood.

Alg	Estimated NLL	Estimated NLL w/reg.	True NLL	True NLL w/reg.
EP-LVM	$-8.889947e+01$	–	$-8.887721e+01$	–
AP-LVM(1)	$-8.884089e+01$	–	$-8.887721e+01$	–
AP-LVM(2)	$-8.889947e+01$	–	$-8.887721e+01$	–
ADMM	$-8.732370e+01$	$-8.727177e+01$	–	$-8.643062e+01$
CVX	$-8.946498e+01$	$-8.941170e+01$	–	$-8.886743e+01$
Alg	Relative error	Per-iteration time	Total time	Output rank
EP-LVM	$8.020263e-01$	$4.652518e-03$	$2.791511e+00$	5
AP-LVM(1)	$8.269273e-01$	$6.010683e-03$	$3.387369e+00$	5
AP-LVM(2)	$8.020263e-01$	$9.625240e-03$	$5.775144e+00$	3
ADMM	$1.776615e+00$	$1.281858e-02$	$7.691148e+00$	52
CVX	$1.779234e+00$	–	$8.416155e+02$	100

Table 5: Comparison of different algorithms for $p = 1000$ and $n = 50p$.

Alg	Estimated NLL	Estimated NLL w/reg.	True NLL	True NLL w/reg.
EP-LVM	$-2.640797e+03$	–	$-2.640199e+03$	–
AP-LVM(1)	$-2.640143e+03$	–	$-2.640199e+03$	–
AP-LVM(2)	$-2.640797e+03$	–	$-2.640199e+03$	–
ADMM	$-2.645466e+03$	$-2.643407e+03$	–	$-2.522374e+03$
Alg	Relative error	Per-iteration time	Total time	Output rank
EP-LVGGM	$8.789615e-01$	$2.510983e-01$	$1.506590e+02$	50
AP-LVGGM(1)	$8.609531e-01$	$9.864577e-02$	$5.918746e+01$	50
AP-LVGGM(2)	$8.789615e-01$	$3.756105e-01$	$2.253663e+02$	50
ADMM	$1.540313e+00$	$6.669523e-01$	$3.793097e+02$	462

[HIS15] C. Hegde, P. Indyk, and L. Schmidt. Approximation algorithms for model-based compressive sensing. *IEEE Trans. Inform. Theory*, 61(9):5129–5147, 2015.

[HM12] D. Henrion and J. Malick. Projection methods in conic optimization. In *Handbook on Semidefinite, Conic and Polynomial Optimization*, pages 565–600. Springer, 2012.

[HMJ+00] T. Hughes, M. Marton, A. Jones, C. Roberts, R. Stoughton, C. Armour, H. Bennett, E. Coffey, H. Dai, Y. He, et al. Functional discovery via a compendium of expression profiles. *Cell*, 102(1):109–126, 2000.

[HZZ16] L. Han, Y. Zhang, and T. Zhang. Fast component pursuit for large-scale inverse covariance estimation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1585–1594. ACM, 2016.

[JMD10] P. Jain, R. Meka, and I. Dhillon. Guaranteed rank minimization via singular value projection. In *Adv. Neural Inf. Proc. Sys. (NIPS*, pages 937–945, 2010.

[JTK14] P. Jain, A. Tewari, and P. Kar. On iterative hard thresholding methods for high-dimensional m-estimation. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 685–693, 2014.

[MD09] Michael W Mahoney and Petros Drineas. Cur matrix decompositions for improved data analysis. *Proc. Natl. Acad. Sci.*, 106(3):697–702, 2009.

[MH12] R. Mazumder and T. Hastie. The graphical lasso: New insights and alternatives. *Electronic journal of statistics*, 6:2125, 2012.

[MM15] C. Musco and C. Musco. Randomized block krylov methods for stronger and faster approximate singular value decomposition. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 1396–1404, 2015.

[MXZ13] S. Ma, L. Xue, and H. Zou. Alternating direction methods for latent variable gaussian graphical model selection. *Neural computation*, 25(8):2172–2198, 2013.

[NYWR11] S. Negahban, B. Yu, M. Wainwright, and P. Ravikumar. A unified framework for high-dimensional analysis of m-estimators with decomposable regularizers. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, 2011.

[PKCS16] D. Park, A. Kyrillidis, C. Caramanis, and S. Sanghavi. Non-square matrix sensing without spurious local minima via the burer-monteiro approach. *stat*, 1050:12, 2016.

[PWZO16] N. Padmanabhan, M. White, H. Zhou, and R. O'Connell. Estimating sparse precision matrices. *Monthly Notices of the Royal Astronomical Society*, 460(2):1567–1576, 2016.

[RRG+12] B. Rolfs, B. Rajaratnam, D. Guillot, I. Wong, and A. Maleki. Iterative thresholding algorithm for sparse inverse covariance estimation. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 1574–1582, 2012.

[RST09] Vladimir Rokhlin, Arthur Szlam, and Mark Tygert. A randomized algorithm for principal component analysis. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1100–1124, 2009.

[RWRY11] P. Ravikumar, M. Wainwright, G. Raskutti, and B. Yu. High-dimensional covariance estimation by minimizing ℓ_1-penalized log-determinant divergence. *Electronic Journal of Statistics*, 5:935–980, 2011.

[SH17] M. Soltani and C. Hegde. Fast algorithms for learning latent variables in graphical models. *arXiv preprint arXiv:1706.08936*, 2017.

[SS16] N. Souly and M. Shah. Scene labeling using sparse precision matrix. In *IEEE Conf. Comp. Vision and Pattern Recog*, pages 3650–3658, 2016.

[TBS+16] S. Tu, R. Boczar, M. Simchowitz, M. Soltanolkotabi, and B. Recht. Low-rank solutions of linear matrix equations via procrustes flow. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 964–973, 2016.

[WJ08] M. Wainwright and M. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

[YL13] J. Yin and H.e Li. Adjusting for high-dimensional covariates in sparse precision matrix estimation by âĎŞ1-penalization. *Journal of multivariate analysis*, 116:365–381, 2013.

[YLZ14] X. Yuan, P. Li, and T. Zhang. Gradient hard thresholding pursuit for sparsity-constrained optimization. In *Proc. Int. Conf. Machine Learning*, pages 127–135, 2014.

[YR13] E. Yang and P. Ravikumar. Dirty statistical models. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 611–619, 2013.

[ZL15] Q. Zheng and J. Lafferty. A convergent gradient descent algorithm for rank minimization and semidefinite programming from random linear measurements. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 109–117, 2015.