We present a novel view on principal component analysis (PCA) as a competitive game in which each approximate eigenvector is controlled by a player whose goal is to maximize their own utility function. We analyze the properties of this PCA game and the behavior of its gradient based updates. The resulting algorithm—which combines elements from Oja’s rule with a generalized Gram-Schmidt orthogonalization—is naturally decentralized and hence parallelizable through message passing. We demonstrate the scalability of the algorithm with experiments on large image datasets and neural network activations. We discuss how this new view of PCA as a differentiable game can lead to further algorithmic developments and insights.

1 Introduction

The principal components of data are the vectors that align with the directions of maximum variance. These have two main purposes: a) as interpretable features and b) for data compression. Recent methods for principal component analysis (PCA) focus on the latter, explicitly stating objectives to find the \(k\)-dimensional subspace that captures maximum variance (e.g., [59]), and leaving the problem of rotating within this subspace to, for example, a more efficient downstream singular value decomposition (SVD) step [13]. This point is subtle, yet critical. For example, any pair of two-dimensional, orthogonal vectors spans all of \(\mathbb{R}^2\) and, therefore, captures maximum variance of any two-dimensional dataset. However, for these vectors to be principal components, they must, in addition, align with the directions of maximum variance which depends on the covariance of the data. By learning the optimal subspace, rather than the principal components themselves, objectives focused on subspace error ignore the first purpose of PCA. In contrast, modern nonlinear representation learning techniques focus on learning features that are both disentangled (uncorrelated) and low dimensional [13, 31, 41, 43, 54].

It is well known that the PCA solution of the \(d\)-dimensional dataset \(X \in \mathbb{R}^{n \times d}\) is given by the eigenvectors of \(X^\top X\) or equivalently, the right singular vectors of \(X\). Impractically, the cost of computing the full SVD scales with \(O(\min\{nd^2, n^2d\})\)-time and \(O(nd)\)-space [55, 59]. For moderately sized data, randomized methods can be used [26]. Beyond this, stochastic—or online—methods based on Oja’s rule [47] or power iterations [52] are common. Another option is to use streaming k-PCA algorithms such as Frequent Directions (FD) [21] or Oja’s algorithm [2] with storage complexity \(O(kd)\). Sampling or sketching methods also scale well, but again, focus on the top-\(k\) subspace [14, 19, 55].

In contrast to these approaches, we view each principal component (equivalently, eigenvector) as a player in a game whose objective is to maximize their own local utility function in competition with

---

1After learning the top-\(k\) subspace \(V \in \mathbb{R}^{d \times k}\), the rotation can be recovered via an SVD of \(XV\).
2FD approximates the top-\(k\) subspace; Oja’s algorithm approximates the top-\(k\) eigenvectors.
other vectors. The proposed utility gradients are interpretable as a combination of Oja’s rule and a
generalized Gram-Schmidt process. We make the following contributions:

- A novel formulation of solving for the principal components as finding the Nash equilibrium of
  a suitable game,
- A sequential, globally convergent algorithm for approximating the Nash in the batch, non-
  streaming setting,
- A decentralized version of the algorithm with an accompanying empirical analysis demonstrat-
  ing the proposed approach as competitive with modern streaming $k$-PCA algorithms on
  synthetic and real data,
- In demonstration of the scaling of the approach, we compute the top-32 principal components
  of the matrix of RESNET-200 activations on the IMAGEnET dataset ($n \approx 10^6$, $d \approx 20 \cdot 10^6$).

Each of these contributions is important. Novel formulations often lead to deeper understanding of
problems, thereby, opening doors to improved techniques. In particular, $k$-player games are in general
complex and hard to analyze. In contrast, PCA has been well-studied. By combining the two fields
we hope to develop useful analytical tools. Our specific formulation is important because it obviates
the need for any centralized orthonormalization step and lends itself naturally to decentralization.
And lastly, theory and experiments support the viability of this approach for continued research.

2 Related work

PCA is a century-old problem and a massive literature exists and we only scratch the surface here
(see e.g. [34] for a comprehensive overview from a statistical perspective and [24] for a numerical
linear algebra perspective). When the dataset size allows it, the preferable solution is the SVD. For
moderately sized datasets, SVD combined with randomized algorithms can be used to recover the
top-$k$ components [26].

Consider searching for the first eigenvalue of a symmetric matrix $M$. In neuroscience, Hebb’s rule [28]
refers to a connectionist rule that solves for the top components with additive updates of a vector
$v$ as $v \leftarrow v + \eta M v$. Likewise, Oja’s rule [47] refers to a similar update $v \leftarrow v + \eta (I - vv^\top) M v$.
In machine learning, using a normalization step of $v \leftarrow v/||v||$ with Hebb’s rule is somewhat
confusingly referred to as Oja’s algorithm [56], the reason being that the subtractive term in Oja’s rule
can be viewed as a regularization term for implicitly enforcing the normalization. If a normalization
step is added to Oja’s rule, this is referred to as Krasulina’s algorithm [36]. In the language of
Riemannian manifolds, $v/||v||$ can be recognized as a retraction and $(I - vv^\top)$ as projecting the
gradient $M v$ onto the tangent space of the sphere [1].

An extension of Krasulina’s algorithm to the top-$k$ setting, termed Matrix Krasulina [59], was recently
proposed in the machine learning literature. This algorithm can be recognized as projecting the
gradient onto the Stiefel manifold (the space of orthonormal matrices) followed by a QR step\footnote{Most
generalizations to the top-$k$ involve adding an orthonormalization step.} (plus
some minor sign accounting), which is a well known retraction.

Of these, Oja’s algorithm has arguably been the most extensively studied. Recent approaches have
augmented Oja’s algorithm with variance reduction to improve convergence rate [58]. It is now
known that the top-$k$ subspace can be approximated to $\rho$ accuracy in $O(1/\rho)$ iterations with constants
depending in particular on the spectral gap. Gap-free bounds also exist and require additional space,
see [2] for a detailed discussion. For the top component ($k = 1$), there exist sharp convergence rates [27, 33, 57], later extended to $k > 1$ [2, 60] with global convergence guarantees.

Maintaining orthonormality of the components via QR is computationally expensive. Amid and
Warmuth [3] propose an alternative Krasulina method which does not require re-orthonormalization
but instead requires inverting a $k \times k$ matrix; in a streaming setting restricted to minibatches of size 1
($X_t \in \mathbb{R}^d$), Sherman-Morrison [24] can be used to efficiently replace the inversion step. Raja and
Bajwa [50] develop a data-parallel distributed algorithm for the top eigenvector. In contrast, other
methods extract the top components in sequence by solving for the $i$th component using an algorithm
such as power iteration or Oja’s, and then enforcing orthogonality by removing the learned subspace
from the matrix, a process known as deflation. Alternatively, the deflation process may be intertwined.
with the learning of the top components. The generalized Hebbian algorithm [53] (GHA) works this way as do Lagrangian inspired formulations [22] as well as our own approach. We make the connection between GHA and our algorithm concrete in Prop. [G.1]. Note, however, that the GHA update is not the gradient of any utility (Prop. [G.2] and therefore, lacks a clear game interpretation.

Our approach estimates the top-\(k\) principal components without reorthonormalization by means of a generalized Gram-Schmidt process. In Section [5] we prove theoretical properties of our proposed game and convergence of a sequential version of our algorithm to the actual principal components, not just the top-\(k\) subspace. This algorithm is sound but not decentralized. We propose a natural decentralized version in Algorithm [2] that allows us to achieve data and model parallelism and support this choice by demonstrating competitive performance on large scale experiments.

For context, Oja’s algorithm converges to the actual principal components [2] and Matrix Krasulina’s [59] converges to the top-\(k\) subspace. However, neither can be obviously decentralized. GHA [53] converges to the actual principal components asymptotically and can be decentralized. Each of these methods is applicable in the streaming \(k\)-PCA setting.

## 3 PCA as an Eigen-Game

We adhere to the following notation. Vectors and matrices meant to approximate principal components (equivalently eigenvectors) are designated with hats, \(\hat{v}\) and \(\hat{V}\) respectively, whereas true principal components are \(v\) and \(V\). Subscripts indicate which eigenvalue a vector is associated with. For example, \(v_i\) is the \(i\)th largest eigenvector. By an abuse of notation, \(v_{j<i}\) refers to the set of vectors \(\{v_j | j \in \{1, \ldots, i - 1\}\}\) and are also referred to as the parents of \(v_i\) (\(v_j\) is their child). Sums over indices should be clear from context, e.g., \(\sum_{j<i} = \sum_{j=1}^{i-1}\). The inner product is written \(\langle u, v \rangle = u^\top v\).

We denote the unit sphere by \(S^{d-1}\) and simplex by \(\Delta^{d-1}\) in \(d\)-dimensional ambient space.

### Outline of derivation

As argued in the introduction, the PCA problem is often mis-interpreted as learning a projection of the data into a subspace that captures maximum variance (equiv. maximizing the trace of a suitable matrix \(R\) introduced below). This is in contrast to the original goal of learning the principal components. We first develop the intuition for deriving our utility functions by (i) showing that only maximizing the trace of \(R\) is not sufficient for recovering all principal components (equiv. eigenvectors), and (ii) showing that minimizing off-diagonal terms in \(R\) is a complementary objective to maximizing the trace and can recover all components. We then consider learning only the top-\(k\) and construct utilities that are consistent with findings in (i) and (ii), equal the true eigenvalues at the Nash of the game we construct, and result in a game that is amenable to analysis.

### Derivation of player utilities

The eigenvalue problem for a symmetric matrix \(X^\top X = M \in \mathbb{R}^{d \times d}\) is to find a matrix of \(d\) orthonormal column vectors \(V\) (implies \(V\) is full-rank) such that \(MV = V\Lambda\) with \(\Lambda\) diagonal. Given a solution to this problem, the columns of \(V\) are known as eigenvectors and corresponding entries in \(\Lambda\) are eigenvalues. By left-multiplying by \(V^\top\) and recalling \(V^\top V = VV^\top = I\) by orthonormality (i.e., \(V\) is unitary), we can rewrite the equality as

\[
V^\top MV = V^\top V\Lambda = \Lambda. \tag{1}
\]

Let \(\hat{V}\) denote a guess or estimate of the true eigenvectors \(V\) and define \(R(\hat{V}) \equiv \hat{V}^\top M\hat{V}\). The PCA problem is often posed as maximizing the trace of \(R\) (equivalent to minimizing reconstruction error):

\[
\max_{\hat{V}^\top \hat{V} = I} \left\{ \sum_i R_{ii} = \text{Tr}(R) = \text{Tr}(\hat{V}^\top M\hat{V}) = \text{Tr}(\hat{V}\hat{V}^\top M) = \text{Tr}(M) \right\}. \tag{2}
\]

Surprisingly, the objective in (2) is independent of \(\hat{V}\), so it cannot be used to recover all (i.e., \(k = d\)) the eigenvectors of \(M\)—(i). Alternatively, Equation (1) implies the eigenvalue problem can be phrased as ensuring all off-diagonal terms of \(R\) are zero, thereby ensuring \(R\) is diagonal—(ii):

\[
\min_{\hat{V}^\top \hat{V} = I} \sum_{i \neq j} R_{ij}^2. \tag{3}
\]
It is worth further examining the entries of $R$ in detail. Diagonal entries $R_{ii} = \langle \hat{v}_i, M\hat{v}_i \rangle$ are recognized as Rayleigh quotients because $||\hat{v}_i|| = 1$ by the constraints. Off-diagonal entries $R_{ij} = \langle \hat{v}_i, M\hat{v}_j \rangle$ measure alignment between $\hat{v}_i$ and $\hat{v}_j$ under a generalized inner product $\langle \cdot, \cdot \rangle_M$.

So far, we have considered learning all the eigenvectors. If we repeat the logic for the top-$k$ eigenvectors, maximizing the trace encourages learning an $M$-subspace, which is a (strict-)Nash equilibrium if and only if for all $i$,$ 1 \leq i \leq k$, for each $\hat{v}_i$, $\hat{v}_i$ is a (strict-)Nash equilibrium if and only if for all $j$, $\hat{v}_i\langle \hat{v}_i, \hat{v}_j \rangle > \hat{v}_i\langle \hat{v}_j, \hat{v}_j \rangle$ for all $i,j \in \{1, \ldots, k\}$.

If we only desire the top-$k$ eigenvectors, maximizing the $\ell_2$-norm encourages learning a $k$-subspace spanned by the top-$k$ eigenvectors, but does not recover the eigenvectors themselves. On the other hand, Equation (3) places no preference on recovering large over small eigenvectors, but does enforce $\mathbf{V}^\top \mathbf{V} = \mathbf{I}$, and assuming $\mathbf{V}$ is orthonormal as before, we have $\mathbf{V}\mathbf{V}^\top = \mathbf{P}$ is a projection matrix.

Left-multiplying Equation (1) by $\mathbf{V}$ now reads $\langle \mathbf{P}\mathbf{M}\mathbf{V} = \mathbf{V}\Lambda \mathbf{V}$ so we are solving an eigenvalue problem for a subspace of $\mathbf{M}$.

If we only desire the top-$k$ eigenvectors, maximizing the trace encourages learning a subspace spanned by the top-$k$ eigenvectors, but does not recover the eigenvectors themselves. On the other hand, Equation (2) places no preference on recovering large over small eigenvectors, but does enforce the columns of $\mathbf{V}$ to actually be eigenvectors. The preceding exercise is intended to introduce minimizing the off-diagonal terms of $\mathbf{R}$ as a possible complementary objective for solving top-$k$ PCA. Next, we will use these two objectives to construct utility functions for each eigenvector $\hat{v}_i$.

We want to combine the objectives to take advantage of both their strengths. A valid proposal is

$$
\max_{\mathbf{V}^\top \mathbf{V} = \mathbf{I}} \sum_i R_{ii} - \sum_{i \neq j} R_{ij}^2.
$$

(4)

However, this objective ignores the natural hierarchy of the top-$k$ eigenvectors. For example, $\hat{v}_1$ is penalized for aligning with $\hat{v}_k$ and vice versa, but $\hat{v}_1$, being the estimate of the largest eigenvector, should be free to search for the direction that captures the most variance independent of the locations of the other vectors. Instead, first consider solving for the top-$1$ eigenvector, $\hat{v}_1$, in which case, $\mathbf{R} = [\langle \hat{v}_1, M\hat{v}_1 \rangle]$ is a $1 \times 1$ matrix. In this setting, Equation (3) is not applicable because there are no off-diagonal elements, so $\max_{\hat{v}_1^\top \hat{v}_1 = 1} \langle \hat{v}_1, M\hat{v}_1 \rangle$ is a sensible utility function for $\hat{v}_1$.

If considering the top-$2$ eigenvectors, $\hat{v}_1$’s utility remains as before, and we introduce a utility for $\hat{v}_2$. Equation (4) is now applicable, so $\hat{v}_2$’s utility is

$$
\max_{\hat{v}_2^\top \hat{v}_2 = 1, \hat{v}_1^\top \hat{v}_2 = 0} \langle \hat{v}_2, M\hat{v}_2 \rangle - \frac{\langle \hat{v}_2, M\hat{v}_1 \rangle^2}{\langle \hat{v}_2, M\hat{v}_2 \rangle}
$$

(5)

where we have divided the off-diagonal penalty by $\langle v_2, Mv_2 \rangle$ so a) the two terms in Equation (5) are on a similar scale and b) for reasons that ease analysis. Additionally note that the constraint $\hat{v}_1^\top \hat{v}_2 = 0$ may be redundant at the optimum ($\hat{v}_1^\top = v_1$, $\hat{v}_2^\top = v_2$) because the second term, $\langle \hat{v}_2, M\hat{v}_1 \rangle^2 = \langle v_2, Mv_1 \rangle^2 = \Lambda_1^2 \langle v_2, v_1 \rangle^2$, already penalizes such deviations ($\Lambda_{ii}$ is the $i$th largest eigenvalue).

These reasons motivate the following set of objectives (utilities), one for each vector $i \in \{1, \ldots, k\}$:

$$
\max_{\hat{v}_i^\top \hat{v}_i = 1} \left\{ u_i(\hat{v}_i | \hat{v}_j < i) = \hat{v}_i^\top M\hat{v}_i - \sum_{j < i} \frac{\langle \hat{v}_i, M\hat{v}_j \rangle^2}{\langle \hat{v}_j, M\hat{v}_j \rangle} = ||X\hat{v}_i||^2 - \sum_{j < i} \frac{\langle X\hat{v}_i, X\hat{v}_j \rangle^2}{\langle X\hat{v}_j, X\hat{v}_j \rangle} \right\}
$$

(6)

where the notation $u_i(a_i | b)$ emphasizes that player $i$ adjusts $a_i$ to maximize a utility conditioned on $b$.

It is interesting to note that by incorporating knowledge of the natural hierarchy, we are immediately led to constructing asymmetric utilities, and thereby, inspired to formulate the PCA problem as a game, rather than a direct optimization problem as in Equation (4).

**Differentiable games.** The player utility functions are all differentiable. A differentiable game consists of $k$ players, each with a differentiable optimization problem that depends on possibly all $k$ players. The study of differentiable games has recently found application in machine learning for GANs [5, 20], multi-agent reinforcement learning [38], and draws on a rich foundation in dynamical systems, variational inequalities, and game theory [45, 46]. Our specific formulation has connections to resource congestion games [51]—here, resources are directions on the hypersphere and congestion is penalized through a generalized cosine distance.

A key concept in (differentiable) games is a Nash equilibrium. A Nash equilibrium specifies a variable for each player from which no player can unilaterally deviate and improve their outcome. In this case, $\hat{V}$ is a (strict)-Nash equilibrium if and only if for all $i$, $u_i(\hat{v}_i | \hat{v}_j < i) > u_i(\hat{v}_i | \hat{v}_j < i)$ for all $\hat{z}_i \in S^{d-1}$.

**Theorem 3.1 (PCA Solution is the Unique strict-Nash Equilibrium).** Assume that the top-$k$ eigenvalues of $X^\top X$ are distinct. Then the top-$k$ eigenvectors form the unique strict-Nash equilibrium of the proposed game in Equation (5). The proof is deferred to Appendix C.
Algorithm 1

EigenGame

The resulting gradient with normalized penalty term has an intuitive meaning. It consists of a single vector \( \hat{v}_i \) to be problematic in practice. To this end, we propose running Algorithm 2 on each respective optima, they become quasi-stationary, so we do not expect maximizing utilities in parallel as ascent with gradients given by Equation (7). Recall that each eigenvector can be learned by maximizing its utility. The vectors are learned concurrently, then \( \hat{v}_i \) is maximizing a non-stationary objective. Proving convergence in the non-stationary setting is very difficult. Instead, for completeness, we prove convergence assuming each \( \hat{v}_i \) is learned in sequence. Algorithm 1 learns \( \hat{v}_i \) given fixed parents \( \hat{v}_{j<i} \); we present the convergence guarantee in Section 6 and details on setting \( \rho_i \) and \( \alpha \) in Appendix K.

Algorithm 1 EigenGame\( ^R \)-Sequential

Given: matrix \( X \in \mathbb{R}^{n \times d} \), initial vector \( \hat{v}^0_i \in S^{d-1} \), learned approximate parents \( \hat{v}_{j<i} \), step size \( \alpha \), and maximum error tolerance \( \rho_i \).

\[
\hat{v}_i \leftarrow \hat{v}^0_i, \quad t_i = \left\lfloor \frac{2}{\alpha} \min\left( \frac{1}{\sqrt{\|\nabla v_i u_i\|^2}}, 2, \rho_i \right)^{-2} \right\rfloor
\]

for \( t = 1 \) to \( t_i \) do

\[
\nabla \hat{v}_i \leftarrow 2X^\top \left[ X\hat{v}_i \right] - \sum_{j<i} \frac{(X\hat{v}_i, X\hat{v}_j)}{(X\hat{v}_j, X\hat{v}_j)} X\hat{v}_j
\]

\[
\nabla^R \hat{v}_i \leftarrow \nabla \hat{v}_i - \langle \nabla \hat{v}_i, \hat{v}_i \rangle \hat{v}_i
\]

\[
\hat{v}^t_i \leftarrow \hat{v}_i + \alpha \nabla^R \hat{v}_i
\]

\[
\hat{v}_i \leftarrow \hat{v}^t_i / \|\hat{v}^t_i\|
\]

end for

return \( \hat{v}_i \)

4Unique up to a sign change; this is expected as both \( v_i \) and \( -v_i \) represent the same principal component.

Figure 1: Each player \( i \)'s utility function depends on its parents represented here by a directed acyclic graph. Each parent must broadcast its vector, “location”, down the hierarchy. Solving for the Nash of a game is difficult in general (specifically, it belongs to the class of PPAD-complete problems [15, 23]). However, because the game is hierarchical and each player’s utility only depends on its parents, it is possible to construct a sequential algorithm that is convergent by solving each player’s optimization problem in sequence. We elaborate in the next two sections.

4 Method

Utility gradient. In Section 3 we mentioned that normalizing the penalty term from Equation (5) had a motivation beyond scaling. Dividing by \( \langle \hat{v}_j, M\hat{v}_j \rangle \) results in the following gradient for player \( i \):

\[
\nabla \hat{v}_i u_i(\hat{v}_i | \hat{v}_{j<i}) = 2M \left[ \hat{v}_i - \sum_{j<i} \frac{\hat{v}_j^\top M \hat{v}_j}{\hat{v}_j^\top M \hat{v}_j} \hat{v}_j \right] = 2X^\top \left[ X\hat{v}_i - \sum_{j<i} \frac{\langle X\hat{v}_i, X\hat{v}_j \rangle}{\langle X\hat{v}_j, X\hat{v}_j \rangle} X\hat{v}_j \right]. \tag{7}
\]

The resulting gradient with normalized penalty term has an intuitive meaning. It consists of a single generalized Gram-Schmidt step followed by the standard matrix product found in power iteration and Oja’s rule. Also, notice that applying the gradient as a fixed point operator in sequence \( (\hat{v}_i \leftarrow \frac{1}{2} \nabla \hat{v}_i u_i(\hat{v}_i | \hat{v}_{j<i})) \) on \( M = I \) recovers the standard Gram-Schmidt procedure for orthogonalization.

A sequential algorithm. Each eigenvector can be learned by maximizing its utility. The vectors are constrained to the unit sphere, a non-convex Riemannian manifold, so we use Riemannian gradient ascent with gradients given by Equation (7). Recall that each \( u_i \) depends on \( \hat{v}_{j<i} \). If any of \( \hat{v}_{j<i} \) are being learned concurrently, then \( \hat{v}_i \) is maximizing a non-stationary objective. Proving convergence in the non-stationary setting is very difficult. Instead, for completeness, we prove convergence assuming each \( \hat{v}_i \) is learned in sequence. Algorithm 1 learns \( \hat{v}_i \) given fixed parents \( \hat{v}_{j<i} \); we present the convergence guarantee in Section 6 and details on setting \( \rho_i \) and \( \alpha \) in Appendix K.

A decentralized algorithm. While Algorithm 1 enjoys a convergence guarantee, learning every parent \( \hat{v}_{j<i} \) before learning \( \hat{v}_i \) may be unnecessarily restrictive. Intuitively, as parents approach their respective optima, they become quasi-stationary, so we do not expect maximizing utilities in parallel to be problematic in practice. To this end, we propose running Algorithm 2 on each \( \hat{v}_i \) in parallel visualized in Figure 2.
Algorithm 2 EigenGame$^R$ (EigenGame—update with $\nabla \hat{v}_i$ instead of $\nabla^R \hat{v}_i$)

Given: data stream, $X_t \in \mathbb{R}^{m \times d}$, total iterations $T$, initial vector $\hat{v}_i^0 \in S^{d-1}$, and step size $\alpha$.

For $t = 1 : T$ do

1. $\nabla \hat{v}_i \leftarrow 2X_t^T[X_t \hat{v}_i - \sum_{j < t} \langle X_t \hat{v}_i, X_t \hat{v}_j \rangle X_t \hat{v}_j]$
2. $\nabla^R \hat{v}_i \leftarrow \nabla \hat{v}_i - \langle \nabla \hat{v}_i, \hat{v}_i \rangle \hat{v}_i$
3. $\hat{v}_i' \leftarrow \hat{v}_i + \alpha \nabla^R \hat{v}_i$
4. $\hat{v}_i \leftarrow \frac{\hat{v}_i'}{||\hat{v}_i'||} \text{broadcast}(\hat{v}_i)$

end for

return $\hat{v}_i$

In practice we can assign each eigenvector update to its own device (e.g. a GPU or TPU). Systems with fast interconnects may facilitate tens, hundreds or thousands of accelerators to be used. In such settings, the overhead of broadcast($\hat{v}_i$) is minimal. We can also specify that the data stream is co-located with the update so $\hat{v}_i$ updates with respect to its own $X_t$. This is a standard paradigm for e.g. data-parallel distributed neural network training. We provide further details in Section 6.

Message Passing on a DAG. Our proposed utilities enforce a strict hierarchy on the eigenvectors. This is a simplification that both eases analysis (see Appendix I) and improves convergence\footnote{Systems such as Google Cloud offer a large number of accelerator cores \url{https://cloud.google.com/tpu/docs/system-architecture}}; however, it is not necessarily optimal. We assume vectors are initialized randomly on the sphere and, for instance, $\hat{v}_k$ may be initialized closer to $\hat{v}_1$ than even $\hat{v}_1$ and vice versa. The hierarchy shown in Figure 1 enforces a strict graph structure for broadcasting information of parents to the children’s utilities.

Variations. We considered several variants of Equation (6). To our knowledge, our formulation is novel and uniquely extensible—notice the utility is composed entirely of inner products on $X \hat{v}_i$, which can be replaced by more general function approximators, $f_i(X)$, e.g., neural networks. The inner products themselves can be replaced by kernels. Other variants may solve PCA but may not be gradients of any function. One disadvantage of our formulation is that a naive estimation of gradients in the stochastic setting is biased. This is mitigated with large batch sizes (see experiments in Section 6 and further discussion in Appendix E). We leave reducing bias to future work.

5 Convergence of EigenGame

Here, we first show that Equation (6) has a simple form such that any local minimum of $u_i$ is also a global minimum. Player $i$’s utility depends on its parents, so we next explain how error in the parents propagates to children through mis-specification of the player $i$’s utility. Using the first result and accounting for this error, we are then able to give global, finite-sample convergence guarantees in the deterministic setting by leveraging recent non-convex Riemannian optimization theory.

The utility landscape and parent-to-child error propagation. Equation (6) is abstruse, but we prove that the shape of player $i$’s utility is simply sinusoidal in the angular deviation of $\hat{v}_i$ from the optimum. The amplitude of the sinusoid varies with the direction of the angular deviation along the sphere and is dependent on the accuracy of players $j < i$. In the special case where players $j < i$
have learned the top-$(i - 1)$ eigenvectors exactly, player $i$’s utility simplifies (see Lemma I.1) to
\[
    u_i(\hat{v}_i, v_{j<i}) = \Lambda_{ii} - \sin^2(\theta_i) \left( \Lambda_{ii} - \sum_{l > i} z_l \Lambda_{il} \right)
\]
(8)
where $\theta_i$ is the angular deviation and $z \in \Delta^{d-1}$ parameterizes the deviation direction. Note that $\sin^2$ has period $\pi$ instead of $2\pi$, which simply reflects the fact that $v_i$ and $-v_i$ are both eigenvectors.

An error propagation analysis reveals that it is critical to learn the parents to a given degree of accuracy. The angular distance between $v_i$ and the maximizer of player $i$’s utility with approximate parents has $\tan^{-1}$ dependence (i.e., a soft step-function; see Lemma I.5 and Figure 10 in Appendix I).

**Theorem 5.1 (Global convergence).** Algorithm 7 achieves finite sample convergence to within $\theta_{tol}$ angular error of the top-$k$ principal components, independent of initialization. Furthermore, if each $\hat{v}_i$ is initialized to within $\xi$ of $v_i$, Algorithm 7 returns the components with angular error less than $\theta_{tol}$ in $T = \mathcal{O}\left( k \left( \frac{(k-1)!}{\theta_{tol}} \prod_{i=1}^k \left( \frac{16\Lambda_{ii}}{g_i} \right) \right)^2 \right)$ iterations. Proofs are deferred to Appendices K.4 and K.5.

Angular error is defined as the angle between $\hat{v}_i$ and $v_i$: $\theta_i = \sin^{-1}(\sqrt{1 - (\langle v_i, \hat{v}_i \rangle)^2})$. The first $k$ in the formula for $T$ appears from a naive summing of worst case bounds on the number of iterations required to learn each $\hat{v}_{j<k}$ individually. The constant 16 arises from the error propagation analysis; parent vectors, $\hat{v}_{j<i}$, must be learned to under 1/16th of a canonical error threshold for the child $\hat{v}_i$, $g_i = \Lambda_{ii} - \Lambda_{ii}$. The Riemannian optimization theory we leverage dictates that $\frac{1}{g_i}$ iterations are required to meet a $\mathcal{O}(\rho)$ error threshold. This is why the squared inverse of the error threshold appears here. Breaking down the error threshold itself, the ratio $\Lambda_{ii}/g_i$ says that more iterations are required to distinguish eigenvectors when the difference between them (summarized by the gap $g_i$) is small relative to the scale of the spectrum, $\Lambda_{ii}$. The $(k-1)!$ term appears because learning smaller eigenvectors requires learning a much more accurate $\hat{v}_i$ higher up the DAG.

Lastly, the utility function for each $\hat{v}_i$ is sinusoidal, and it is possible that we initialize $\hat{v}_i$ with initial utility arbitrarily close to the trough (bottom) of the function where gradients are arbitrarily small. This is why the global convergence rate depends on the initialization in general. Note that Algorithm 1 effectively detects the trough by measuring the norm of the initial gradient ($\nabla \hat{v}_i u_i$) and scales the number of required iterations appropriately. A complete theorem that considers the probability of initializing $\hat{v}_i$ within $\xi$ of $v_i$ is in Appendix I, but this possibility shrinks to zero in high dimensions.

We would also like to highlight that these theoretical findings are actually strong relative to other claims. For example, the exponential convergence guarantee for Matrix Krasulina requires the initial guess at the eigenvectors capture the top-$(k-1)$ subspace [59], unlikely when $d \gg k$. A similar condition is required in [58]. These guarantees are given for the mini-batch setting while ours is for the full-batch, however, we provide global convergence without restrictions on initialization. Improving our convergence rate to exponential is outside the scope of this work, but we characterize
the shape of the utilities as sinusoidal in Equation (8). If the eigenvectors are guessed within $\frac{\pi}{4}$, they are within a region of their utility that is strongly-concave. An exponential convergence rate in the full-batch setting can be obtained by using recent Riemannian acceleration techniques [40].

6 Experiments

We compare our approach against GHA, Matrix Krasulina, and Oja’s algorithm[7]. We present both EigenGame and EigenGame$^R$ which projects the gradient onto the tangent space of the sphere each step. We measure performance of methods in terms of principal component accuracy and subspace distance. We measure principal component accuracy by the number of consecutive components that are estimated within an angle of $\frac{\pi}{8}$ from ground truth. For example, if the angular errors of the $\hat{v}_i$’s returned by a method are, in order, $[\theta_1, \theta_2, \theta_3, \ldots] = [\frac{\pi}{16}, \frac{\pi}{4}, \frac{\pi}{10}, \ldots]$, then the method is credited with a streak of only 1 regardless of the errors $\theta_i > 2$. For Matrix Krasulina, we first compute the optimal matching from $\hat{v}_i$ to ground truth before measuring angular error. We measure normalized subspace distance using $1 - \frac{1}{k} \cdot \text{Tr}(U^*P) \in [0, 1]$ where $U^* = VV^\dagger$ and $P = \hat{V}\hat{V}^\dagger$ [59].

**Synthetic data.** Experiments on synthetic data demonstrate the viability of our approach (Figure 3a). Oja’s algorithm performs best on synthetic experiments because strictly enforcing orthogonalization with an expensive QR step greatly helps when solving for all eigenvectors. EigenGame is able to effectively parallelize this over $k$ machines and the advantage of QR diminishes in Figure 3b. The remaining algorithms perform similarly on a linearly decaying spectrum, however, EigenGame performs better on an exponentially decaying spectrum due possibly to instability of Riemannian gradients near the equilibrium (see Appendix F for details). GHA and EigenGame$^R$ are equivalent under specific conditions (see Proposition G.1).

**MNIST handwritten digits.** We compare EigenGame against GHA, Matrix Krasulina, and Oja’s algorithm on the MNIST dataset (Figure 3b). We flatten each image in the training set to obtain a 60,000 $\times$ 784 dimensional matrix. EigenGame is competitive with Oja’s in a high batch size regime (1024 samples per minibatch). The performance gap between EigenGame and the other methods shrinks as the minibatch size is reduced (see Appendix E), expectedly due to biased gradients.

**The principal components of RESNET-200 activations on IMAGE NET are edge filters.** A primary goal of PCA is to obtain interpretable low-dimensional representations. To this end we present an example of using EigenGame to compute the top-32 principal components of the activations

---

A detailed discussion of Frequent Directions [21] can be found in Appendix [D].
of a pretrained ResNet-200 on the ImageNet dataset. We concatenate the flattened activations from the output of each residual block resulting in a $d = 20M$ dimensional vector representation for each of the 1.2M input images. It is not possible to store the entire 195TB matrix in memory, nor incrementally compute the Gram/covariance matrix.

We implemented a data-and-model parallel version of EigenGame in JAX [12] where each $\tilde{v}_i$ is assigned to its own TPU [35]. Each device keeps a local copy of the ResNet parameters and the ImageNet datastream. Sampling a minibatch (of size 128), computing the network activations and updating $\tilde{v}_i$ are all performed locally. The broadcast($\tilde{v}_i$) step is handled by the pmap and lax.all_gather functions. Computing the top-32 principal components takes approximately nine hours on 32 TPUv3s.

Figure 4 shows the top principal components of the activations of the trained network organized by network topology (consisting of five residual blocks). Note that EigenGame is not applied block-wise, but on all $20M$ dimensions. We do not assume independence between blocks and the eigenvector has unit norm across all blocks. We observe that Block 1 (closest to input) of PC 1 has very small magnitude activations relative to the other PCs. This is because PC 1 should capture the variance which discriminates most between the classes in the dataset. Since Block 1 is mainly concerned with learning low-level image filters, it stands to reason that although these are important for good performance, they do not necessarily extract abstract representations which are useful for classification. Conversely, we see that PC 1 has larger relative activations in the later blocks.

We visualize the average principal activation in Block 19 in Figure 5. The higher PCs learn distinct filters (Gabor filters, Laplacian-of-Gaussian filters, edge filters in different orientations c.f. [8]). Figure 6 shows a scree plot of the Rayleigh quotients recovered by EigenGame and the respective utility achieved by each player. The two curves almost perfectly overlap. The mean relative magnitude of the penalty terms to the respective Rayleigh quotient in the utility is 0.025 indicating that the solutions of each player are close to orthogonal with respect to the generalized inner product (Equation (6)). This implies that the solutions are indeed eigenvectors. The scree plot has two distinct elbows at PC2 and PC6, corresponding to the differences in filters observed in Figure 5.

---

9The activations in Block 1 result from convolving 64 filters over the layer’s input. We take the mean over the 64 channels and plot the resulting $112 \times 112$ image.
7 Conclusion

It seems easier to train a bi-directional LSTM with attention than to compute the SVD of a large matrix. –Chris Re
NeurIPS 2017 Test-of-Time Award, Rahimi and Recht [49].

In this work we have motivated PCA from the perspective of a multi-player game. Based on this we developed a decentralized algorithm which enables truly large-scale principal components estimation. To demonstrate this we used EigenGame to analyze the behavior of a large neural network through the lens of PCA. To our knowledge this is the first analysis of its type and scale (for example [59] compute only 6 components of the \( d = 2300 \) output layer of VGG) and would be otherwise impractical with previous PCA approaches. Beyond this, EigenGame opens a variety of promising research directions.

**Deep Variants.** Player utilities are composed of \( X \hat{v}_i \) projections that can be replaced with end-to-end trainable deep networks, \( f_i(X|\text{weights}) \). It is known that shallow autoencoders recover the top-\( k \) subspace [4] [11]. However, as we have shown, that is not equivalent to recovering the principal components, and, 30 years later, disentanglement is still a live research topic. What does our approach mean for (variational) autoencoders where disentanglement is still a challenge [41, 43, 54]?

**Scale.** In experiments, we broadcast across all edges in Figure 1 every iteration. Introducing lag or considering other random graph network structures may improve efficiency. Can we further reduce our memory footprint by storing only scalars of the losses (bandit feedback) and avoiding congestion through online bandit or reinforcement learning techniques? Our decentralized algorithm may have implications for federated and privacy preserving learning as well [9, 29, 30].

**Games.** EigenGame has a unique Nash equilibrium due to the fixed DAG structure, but vectors are initialized randomly so \( \hat{v}_k \) may start closer to \( v_1 \) than \( \hat{v}_1 \) does. Adapting the DAG could make sense, but might also introduce spurious fixed points or suboptimal Nash. Interesting connections exist between EigenGame and the algorithm [20] for solving LQ-GAN [18]. Can these be leveraged to improve upon both? Might replacing vectors with populations accelerate extraction of the top PCs?

**Core ML.** This work generalizes to any symmetric positive definite matrix. Can it be extended to the asymmetric matrices [6] [7] that arise in game theoretic analyses? EigenGame could be useful as a diagnostic or for accelerating training [16] [37]; similarly, spectral normalization has shown to be a valuable tool for stabilizing GAN training [44]. Also, eigenvectors of the graph Laplacian provide features for RL problems [42]. Spectral PCA also shares deep connections with clustering [17].

Lastly, GANs [25] recently reformulated learning a generative model as a two-player zero-sum game. Here, we show how another fundamental unsupervised learning task can be formulated as a \( k \)-player game. While two-player, zero-sum games are well understood, research on \( k \)-player, general-sum games lies at the forefront in machine learning. We hope that marrying a fundamental, well-understood task in PCA with the relatively less understood domain of many player games will help advance techniques on both ends.

Acknowledgements

We are grateful to Trevor Cai for his help scaling the JAX implementation of EigenGame to handle the large IMAGENET experiment and to Daniele Calandriello for sharing his expert knowledge of related work and advice on revising parts of the manuscript.

References

[1] P-A. Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2009.

[2] Zeyuan Allen-Zhu and Yuanzhi Li. First efficient convergence for streaming k-PCA: a global, gap-free, and near-optimal rate. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 487–492. IEEE, 2017.

[3] Ehsan Amid and Manfred K Warmuth. An implicit form of Krasulina’s k-PCA update without the orthonormality constraint. *arXiv preprint arXiv:1909.04803*, 2019.

[4] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: learning from examples without local minima. *Neural Networks*, 2(1):53–58, 1989.
[5] David Balduzzi, Sebastien Racaniere, James Martens, Jakob Foerster, Karl Tuyls, and Thore Graepel. The mechanics of n-player differentiable games. *arXiv preprint arXiv:1802.05642*, 2018.

[6] David Balduzzi, Karl Tuyls, Julien Perolat, and Thore Graepel. Re-evaluating evaluation. In *Advances in Neural Information Processing Systems*, pages 3268–3279, 2018.

[7] David Balduzzi, Marta Garnelo, Yoram Bachrach, Wojciech M Czarnecki, Julien Perolat, Max Jaderberg, and Thore Graepel. Open-ended learning in symmetric zero-sum games. *arXiv preprint arXiv:1901.08106*, 2019.

[8] Anthony J Bell and Terrence J Sejnowski. The “independent components” of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997.

[9] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecny, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: system design. *arXiv preprint arXiv:1902.01046*, 2019.

[10] Nicolas Boumal, Pierre-Antoine Absil, and Coralia Cartis. Global rates of convergence for nonconvex optimization on manifolds. *IMA Journal of Numerical Analysis*, 39(1):1–33, 2019.

[11] Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59(4-5):291–294, 1988.

[12] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, and Skye Wanderman-Milne. JAX: composable transformations of Python+NumPy programs. 2018. URL https://github.com/google/jax.

[13] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[14] Michael B Cohen, Cameron Musco, and Christopher Musco. Input sparsity time low-rank approximation via ridge leverage score sampling. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1758–1777. SIAM, 2017.

[15] Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009.

[16] Guillaume Desjardins, Karen Simonyan, Razvan Pascanu, et al. Natural neural networks. In *Advances in Neural Information Processing Systems*, pages 2071–2079, 2015.

[17] Chris Ding and Xiaofeng He. K-means clustering via principal component analysis. In *Proceedings of the International Conference on Machine Learning*, page 29, 2004.

[18] Soheil Feizi, Farzan Farnia, Tony Ginart, and David Tse. Understanding GANs: the LQG setting. *arXiv preprint arXiv:1710.10793*, 2017.

[19] Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, PCA, and projective clustering. *SIAM Journal on Computing*, 49(3):601–657, 2020.

[20] Ian Gemp and Sridhar Mahadevan. Global convergence to the equilibrium of GANs using variational inequalities. *arXiv preprint arXiv:1808.01531*, 2018.

[21] Mina Ghashami, Edo Liberty, Jeff M Phillips, and David P Woodruff. Frequent directions: simple and deterministic matrix sketching. *SIAM Review*, 53(2):217–288, 2011.
[27] Moritz Hardt and Eric Price. The noisy power method: a meta algorithm with applications. In Advances in Neural Information Processing Systems, pages 2861–2869, 2014.

[28] Donald Olding Hebb. The Organization of Behavior: A Neuropsychological Theory. Psychology Press, 2005.

[29] Christina Heinze, Brian McWilliams, and Nicolai Meinshausen. Dual-loco: distributing statistical estimation using random projections. In Artificial Intelligence and Statistics, pages 875–883, 2016.

[30] Christina Heinze-Deml, Brian McWilliams, and Nicolai Meinshausen. Preserving privacy between features in distributed estimation. Stat, 7(1):e189, 2018.

[31] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. Beta-VAE: learning basic visual concepts with a constrained variational framework. International Conference on Learning Representations, 2(5):6, 2017.

[32] Roger A Horn and Charles R Johnson. Matrix Analysis. Cambridge University Press, 2012.

[33] Prateek Jain, Chi Jin, Sham M Kakade, Praneeth Netrapalli, and Aaron Sidford. Streaming PCA: matching matrix Bernstein and near-optimal finite sample guarantees for Oja’s algorithm. In Conference on Learning Theory, pages 1147–1164, 2016.

[34] Ian T Jolliffe. Principal components in regression analysis. In Principal Component Analysis. Springer, 2002.

[35] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In Proceedings of the 44th Annual International Symposium on Computer Architecture, pages 1–12, 2017.

[36] TP Krasulina. The method of stochastic approximation for the determination of the least eigenvalue of a symmetrical matrix. USSR Computational Mathematics and Mathematical Physics, 9(6):189–195, 1969.

[37] Gabriel Krummenacher, Brian McWilliams, Yannic Kilcher, Joachim M Buhmann, and Nicolai Meinshausen. Scalable adaptive stochastic optimization using random projections. In Advances in Neural Information Processing Systems, pages 1750–1758, 2016.

[38] Alistair Letcher, Jakob Foerster, David Balduzzi, Tim Rocktäschel, and Shimon Whiteson. Stable opponent shaping in differentiable games. arXiv preprint arXiv:1811.08469, 2018.

[39] Shengqiao Li. Concise formulas for the area and volume of a hyperspherical cap. Asian Journal of Mathematics and Statistics, 4(1):66–70, 2011.

[40] Yuanyuan Liu, Fanhua Shang, James Cheng, Hong Cheng, and Licheng Jiao. Accelerated first-order methods for geodesically convex optimization on Riemannian manifolds. In Advances in Neural Information Processing Systems, pages 4868–4877, 2017.

[41] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In Proceedings of the International Conference on Machine Learning, pages 4114–4124, 2019.

[42] Sridhar Mahadevan. Proto-value functions: developmental reinforcement learning. In Proceedings of the International Conference on Machine learning, pages 553–560, 2005.

[43] Emile Mathieu, Tom Rainforth, N Siddharth, and Yee Whye Teh. Disentangling disentanglement in variational autoencoders. arXiv preprint arXiv:1812.02833, 2018.

[44] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957, 2018.

[45] Anna Nagurney and Ding Zhang. Projected Dynamical Systems and Variational Inequalities with Applications, volume 2. Springer Science & Business Media, 2012.

[46] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. Algorithmic Game Theory. Cambridge University Press, 2007.

[47] Erkki Oja. Simplified neuron model as a principal component analyzer. Journal of Mathematical Biology, 15(3):267–273, 1982.
[48] Elad Plaut. From principal subspaces to principal components with linear autoencoders. *arXiv preprint arXiv:1804.10253*, 2018.

[49] Ali Rahimi and Benjamin Recht. Reflections on random kitchen sinks, 2017. URL http://www.argmin.net/2017/12/05/kitchen-sinks/.

[50] Haroon Raja and Waheed U Bajwa. Distributed stochastic algorithms for high-rate streaming principal component analysis. *arXiv preprint arXiv:2001.01017*, 2020.

[51] Robert W Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2(1):65–67, 1973.

[52] H. Rutishauser. Simultaneous iteration method for symmetric matrices. In *Handbook for Automatic Computation*, pages 284–302. Springer, 1971.

[53] Terence D Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2(6):459–473, 1989.

[54] Mhd Hasan Sarhan, Abouzar Eslami, Nassir Navab, and Shadi Albarqouni. Learning interpretable disentangled representations using adversarial VAEs. In *Domain Adaptation and Representation Transfer and Medical Image Learning with Less Labels and Imperfect Data*, pages 37–44. Springer, 2019.

[55] Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS’06)*, pages 143–152. IEEE, 2006.

[56] Ohad Shamir. A stochastic PCA and SVD algorithm with an exponential convergence rate. In *Proceedings of the International Conference on Machine Learning*, pages 144–152, 2015.

[57] Ohad Shamir. Convergence of stochastic gradient descent for PCA. In *Proceedings of the International Conference on Machine Learning*, pages 257–265, 2016.

[58] Ohad Shamir. Fast stochastic algorithms for SVD and PCA: Convergence properties and convexity. In *Proceedings of the International Conference on Machine Learning*, pages 248–256, 2016.

[59] Cheng Tang. Exponentially convergent stochastic k-PCA without variance reduction. In *Advances in Neural Information Processing Systems*, pages 12393–12404, 2019.

[60] Nilesh Tripuraneni, Nicolas Flammarion, Francis Bach, and Michael I Jordan. Averaging stochastic gradient descent on Riemannian manifolds. *arXiv preprint arXiv:1802.09128*, 2018.

[61] Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Advances in Neural Information Processing Systems*, pages 3835–3844, 2018.
A Experiment Details

In the synthetic experiments, \( \hat{V} \) is initialized randomly so \( M \in \mathbb{R}^{50 \times 50} \) is constructed as a diagonal matrix without loss of generality. The linear spectrum ranges from 1 to 1000 with equal spacing. The exponential spectrum ranges from \( 10^3 \) to \( 10^0 \) with equal spacing on the exponents.

A.1 Clarification of Oja Variants

As discussed in Section 2, it is easy to confuse the various Oja methods. In our experiments, Oja’s algorithm refers to applying Hebb’s rule \( v_i \leftarrow v_i + \eta M v_i \) followed by an orthonormalization step computed with QR as in Algorithm 3:

**Algorithm 3 Oja’s Algorithm**

Given: data stream, \( X_t \in \mathbb{R}^{m \times d} \), \( T \), \( \hat{V}^0 \in S^{d-1} \times \ldots \times S^{d-1} \), step size \( \eta \)

\[
\hat{V} \leftarrow \hat{V}^0 \\
\text{mask} \leftarrow \text{LT}(2I_k - 1_k) \\
\text{for} \ t = 1 : T \text{ do} \\
\hat{V} \leftarrow \hat{V} + \eta X^T_i X_i \hat{V} \\
Q, R \leftarrow \text{QR}(\hat{V}) \\
S = \text{sign} (\text{sign}(\text{diag}(R)) + 0.5) \\
\hat{V} = QS \\
\text{end for} \\
\text{return } \hat{V}
\]

where \( 1_k \) is a \( k \times k \) matrix of all ones, LT returns the lower-triangular part of a matrix (includes the diagonal), and \( \text{sign} = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases} \). Oja’s algorithm is the standard nomenclature for this variant in the machine learning literature [2].

In the scaled-down ResNet experiments (see Section D.3), we use Hebb’s rule with deflation, also sometimes referred to as Oja’s. Deflation is accomplished by directly subtracting out the parent vectors from the dataset. In detail, each batch of data samples, \( X_t \in \mathbb{R}^{m \times d} \), is preprocessed as \( X_t(i) \leftarrow X_t(i) (I - \sum_{j<i} \hat{v}_j \hat{v}_j^T) \). Then to learn each \( \hat{v}_i \), we repeatedly apply Hebb’s rule with \( M_t = X^T(i) X(i) \), and then \( \hat{v}_i \leftarrow \frac{v_i}{\|v_i\|} \) to project \( \hat{v}_i \) back to the unit-shere. After several iterations \( t \) and once \( \hat{v}_i \)’s Rayleigh quotient appears to have stabilized, we move on to \( \hat{v}_{i+1} \).

B EigenGame Vectorized for CPU

Algorithm 4 presents Algorithm 2 in a vectorized form for implementation on a CPU. LT returns the lower-triangular part of a matrix (includes the diagonal). \( \text{sum}(A, \text{dim} = 0) \) sums over the rows of \( A \). \( \text{norm}(A, \text{dim} = 0) \) returns an array with the \( L_2 \)-norm of each column of \( A \). \( \odot \) denotes elementwise multiplication. \( 1_k \) is a square \( k \times k \) matrix of all ones. \( I_k \) is the \( k \times k \) identity matrix. When dividing a matrix by a vector (\( A/v \)), we assume broadcasting. Specifically, \( v \) is interpreted as a row-vector and stacked vertically to match the dimensions of \( A \); the two matrices are then divided element wise.

C Smallest Eigenvectors

EigenGame can be used to recover the \( k \) smallest eigenvectors as well. Simply use EigenGame to estimate the top eigenvector with eigenvalue \( \Lambda_{11} \). Then run EigenGame on the matrix \( M' = \Lambda_{11}I - M \). The top-\( k \) eigenvectors of \( M' \) are the bottom-\( k \) eigenvectors of \( M \). For example, the \( d \)th eigenvector of \( M, v_d \), is the largest eigenvector of \( M' \): \( M'v_d = \Lambda_{11}v_d - Mv_d = (\Lambda_{11} - \Lambda_{dd})v_d \).
Algorithm 4 EigenGame & EigenGame — Vectorized

Given: data stream, \( X_t \in \mathbb{R}^{m \times d}, T, \hat{V}^0 \in S^{d-1} \times \ldots \times S^{d-1} \), step size \( \alpha \)

\[
V \leftarrow \hat{V}^0 \\
\text{mask} \leftarrow \text{LT}(2I_k - 1_k) \\
\text{for } t = 1 : T \text{ do} \\
\quad R \leftarrow (X_t \hat{V})^\top (X_t \hat{V}) \\
\quad R_{\text{norm}} \leftarrow R / \text{diag}(R) \\
\quad G_s \leftarrow \hat{V} (R_{\text{norm}} \odot \text{mask})^\top \\
\quad \nabla \hat{V} \leftarrow X_t^\top (X_t G_s) \\
\quad \nabla R \hat{V}^- = \hat{V} \sum (\nabla \hat{V} \odot \hat{V}, \text{dim} = 0) \\
\quad \hat{V} \leftarrow \hat{V} + \alpha \nabla R \hat{V} \\
\quad \hat{V} \leftarrow \hat{V} / \text{norm}(\hat{V}, \text{dim} = 0) \\
\text{end for} \\
\text{return } \hat{V}
\]

D Frequent Directions

A reviewer from a previous submission of this work requested a comparison and discussion with Frequent Directions [21], another decentralized subspace-error minimizing \( k \)-PCA algorithm. Frequent Directions (FD) is a streaming algorithm that maintains an overcomplete sketch matrix with the goal of capturing the subspace of maximal variance within the span of its vectors. Each step of FD operates by first replacing a row of the sketch matrix with a single data sample. It then runs SVD on the sketch matrix and uses the resulting decomposition to construct a new sketch. Note that FD relies on SVD as a core inner step. In theory, EigenGame could replace SVD, however, we do not explore that direction here.

D.1 Recovering Principal Components from Principal Subspace

FD returns a sketch \( B = \hat{V}^\top \) of size \( \mathbb{R}^{2l \times d} \) where \( l \geq k \). The rows of FD are not principal components, but they should approximate the top-\( k \) subspace of the dataset. To recover approximate principal components, the optimal rotation of the vectors can be computed with \( Q \leftarrow \text{SV}D(X \hat{V})^\top \). This can be shown by inspecting \( R \) (as defined in Section 3) with rotated vectors:

\[
(\hat{V} Q)^\top M (\hat{V} Q) = Q^\top \hat{V}^\top M \hat{V} Q = Q^\top (X \hat{V})^\top (X \hat{V}) Q = Q^\top M' Q. \tag{9}
\]

By inspection, the problem of computing the optimal \( Q \) reduces to computing the eigenvectors of \( M' \in \mathbb{R}^{k \times k} \). This requires projecting the dataset into the principal subspace, \((X \hat{V})\), to compute \( M' \); however, this is typically a desired step anyways when performing PCA.

D.2 Complexity Analysis

We base our analysis on Section 3.1 of [21] which discusses parallelizing FD. Let \( b \) be number of shards to split the original dataset \( X \in \mathbb{R}^{n \times d} \) into, each shard being in \( \mathbb{R}^{\frac{n}{b} \times d} \). Let \( k \) be the number of principal components sought. Finally, let \( l = [k + \frac{1}{\epsilon}] \) be the sketch size where \( \epsilon \ll 1 \) is a desired tolerance on the Frobenius norm of the subspace approximation error.

The runtime of FD is \( O(nld) \); call this \( Anld \) for some \( A \). To decentralize FD, [21] instructs to

1. Split \( X \) into \( b \) shards and run FD on each individually in parallel.
   - total runtime: \( A(\frac{n}{b})ld = Anld(\frac{l}{b}) \)
   - output: \( b \) sketches \( (B_i \in \mathbb{R}^{2l \times d}) \)
2. Merge sketches and run FD on the merged sketch to produce sketch \( B \).
   - total runtime: \( A(2bl)ld = Anld(\frac{2ld}{n}) \)
• output: 1 sketch \( (B \in \mathbb{R}^{2l \times d}) \)

Finally, normalize the rows of \( B \), project the dataset \( Y \leftarrow XB^\top \), compute the right-singular vectors of the projected dataset, \( Q \in \mathbb{R}^{2l \times 2l} \leftarrow \text{SVD}(Y) \), compute \( \hat{V} \leftarrow B^\top Q \), and compute the corresponding Rayleigh quotients \( \hat{V}^\top M \hat{V} = (YQ)^\top (YQ) \) to determine the top-\( k \) eigenvectors with error within the desired tolerance. We assume this final step takes negligible runtime because we assume \( 2l \ll d \), however, for datasets with many samples (large \( n \)), this step could be nonnegligible without further approximation.

Using the runtimes listed above, we can determine the potential runtime multiplier from decentralization is \( (b + \frac{2b}{n}) \) which is convex in \( b \). If we minimize this w.r.t. \( b \) for the optimal number of shards, we find \( b^* = \sqrt{\frac{2n}{2l}} \). Plugging this back in gives an optimal runtime multiplier of \( 2\sqrt{2} \sqrt{\frac{l}{n}} \).

The analysis above only considers one recursive step. Step 1) can be decentralized as well. For simplicity, we assume the computation is dominated by Step 2), the merge step. Note these relaxations result in a lower bound on FD runtime, i.e., they favor FD in a comparison with EigenGame.

D.3 Small ImageNet Experiments

Consider running on a scaled down RESNET-50 experiment which has approximately 1.2M images \((n = 1.2 \times 10^6, 24\text{TB})\) and searching for the top-25 eigenvectors \((k = 25)\). Using a modest \( \epsilon = \frac{9 \times 25}{1} \) implies \( l = 5k = 125 \) with optimal batch size \( b^* \approx 70 \). Therefore, running FD on \( \frac{n}{b} \) samples with a sketch size of 125 should give a rough lower bound on the runtime for an optimally decentralized FD implementation. The runtime obtained was 9 hours for FD vs 2 hours for EigenGame which actually processes the full dataset 3 times.

![Figure 7: Comparison of mean activation maps between Oja’s with deflation, EigenGame, and FD for a section of the top principal components of RESNET-50 on IMAGENET.](image)

The reason we run FD on a scaled down RESNET-50 experiment as opposed to the RESNET-200 is that the algorithm requires a final SVD step to recover the actual eigenvectors and we were not able to run SVD on a sketch of size \( k \times d \) where \( d = 20 \times 10^6 \) for the full scale experiment. That is to say FD is not applicable in this extremely large data regime. In contrast, EigenGame handles this setting without modification.

To obtain an approximate “ground truth” solution for the principal components we run Oja’s algorithm with a low learning rate with a batch size of 128 for 3 epochs to extract the first eigenvector. We find successive eigenvectors using deflation. By running each step for many iterations and monitoring the convergence of the Rayleigh quotient (eigenvalue) \( v_i^\top M v_i \), we can control the quality of the recovered eigenvectors. This is the simplest and most reliable approach to creating ground truth on a problem where no solution already exists. See Section A.1 for further details.

E Gradient Bias

As expected, Figure 8 shows the performance of EigenGame degrades in the low batch size regime. This is expected because we use the same minibatch for all inner products in the gradient which contains products and ratios of random variables. GHA, on the other hand, is linear in the matrix \( M \) and as such is naturally unbiased. However, GHA does not appear to readily extend to more
general function approximators, whereas EigenGame should. Instead we look to reduce the bias of EigenGame gradients using larger batch sizes (current hardware easily supports batches of 1024 for MNIST and 128 for ImageNet). Further reducing bias is left to future work.

![Figure 8](image1.png)

(a) Longest Streak  
(b) Subspace Distance

Figure 8: (a) The longest streak of consecutive vectors with angular error less than $\frac{\pi}{8}$ radians is plotted vs algorithm iterations on MNIST for minibatch sizes of 256 and 512. Shaded regions highlight ± standard error of the mean for the best performing learning rates. Average runtimes are reported in seconds next to the method names. (b) Subspace distance on MNIST. (a,b) Learning rates were chosen from $\{10^{-3}, \ldots, 10^{-6}\}$ on 10 held out runs. All plots show means over 10 trials.

F To project or not to project?

Projecting the update direction onto the unit-sphere, as suggested by Riemannian optimization theory, can result in much larger update steps. This effect is due to the composition of the retraction ($z' \leftarrow \hat{z}/||\hat{z}||$) and update step ($\hat{z} \leftarrow z + \Delta z$). Omitting the projection can actually mimic modulating the learning rate, decaying it near an equilibrium and improving stability.

![Figure 9](image2.png)

(a) Gradient Instability Near Optimum  
(b) Riemannian Terminology

Figure 9: (a) When the $\hat{v}_i$ is near the optimum of its utility and its gradient is nearly orthogonal to the sphere, pointing directly away from the center (@ 90°), the combination of updating using the projected gradient ($\nabla^R$) and the retraction can result in a large update, possibly moving $\hat{v}_i$ away from the optimum. (b) Diagram presenting Riemannian optimization terminology. The retraction is not a projection in general although our specific choice appears that way for the sphere. A retraction applied at $\hat{v}_i$ takes as input a scaled projected gradient and returns a vector on the manifold: $\hat{v}_i' \leftarrow R_{\hat{v}_i}(\alpha \nabla^R)$.

G Theoretical comparison with GHA

**Proposition G.1.** When the first $i - 1$ eigenvectors have been learned exactly, GHA on $\hat{v}_i$ is equivalent to projecting the first term in $\nabla \hat{v}_i u_i$ onto the sphere, but omitting to project the second set of penalty terms.

**Proof.** The GHA update is

$$
\Delta \hat{v}_i = 2 \left[ M \hat{v}_i - (\hat{v}_i^T M \hat{v}_i) \hat{v}_i - \sum_{j<i} (\hat{v}_j^T M \hat{v}_j) \hat{v}_j \right].
$$

(10)
Plugging $v_{j<i}$ for $v_{j<i}$ into the GHA update, we find

$$\Delta_i = 2 \left[ M \hat{v}_i - (\hat{v}_i^T M \hat{v}_i) \hat{v}_i - \sum_{j<i} (\hat{v}_i^T M v_j) v_j \right] \quad (11)$$

$$= 2 \left[ M \hat{v}_i - (\hat{v}_i^T M \hat{v}_i) \hat{v}_i - \sum_{j<i} A_{jj}(\hat{v}_i^T v_j) v_j \right]. \quad (12)$$

Likewise for the gradient with the first term projected onto the tangent space of sphere:

$$2 \left[ (I - \hat{v}_i \hat{v}_i^T) M \hat{v}_i - M \sum_{j<i} \frac{\hat{v}_i^T M v_j}{v_j^T M v_j} v_j \right] = 2 \left[ (I - \hat{v}_i \hat{v}_i^T) M \hat{v}_i - M \sum_{j<i} (\hat{v}_i^T v_j) v_j \right]$$

$$= 2 \left[ M \hat{v}_i - (\hat{v}_i^T M \hat{v}_i) \hat{v}_i - \sum_{j<i} A_{jj}(\hat{v}_i^T v_j) v_j \right]. \quad (14)$$

**Proposition G.2.** The GHA update for $\hat{v}_i$ is not the gradient of any function.

**Proof.** The Jacobian of $\Delta \hat{v}_i$ w.r.t. $\hat{v}_i$ is

$$Jac(\Delta \hat{v}_i) = 2 \left[ M - (\hat{v}_i^T M \hat{v}_i) I - 2 \hat{v}_i \hat{v}_i^T M - \sum_{j<i} \hat{v}_j \hat{v}_j^T M \right]. \quad (15)$$

The sum of the $\hat{v} \hat{v}^T M$ terms are not, in general, symmetric, therefore, the Jacobian is not symmetric. The Jacobian of a gradient is the Hessian and the Hessian of a function is necessarily symmetric, therefore, the GHA update is not the gradient of any function. \qed

### G.1 Design Decisions

We made a number of algorithmic design decisions that led us to the proposed algorithm. The first to note is that a naive utility that simply subtracts off $\sum_{j<i} \langle \hat{v}_i, \hat{v}_j \rangle$ will not solve PCA. This is because large $\langle \hat{v}_i, M \hat{v}_i \rangle$ (read eigenvalues) can drown out these penalties. The intuition is that including $M$ in the inner product gives the right boost to create a natural balance among terms. Next, it is possible to formulate the utilities without normalizing the terms as we did, however, this is harder to analyze and is akin to minimizing $(err)^4$ instead of $(err)^2$ which generally has better convergence properties near optima. Also, while updates formed using the standard Euclidean Gram-Schmidt procedure will solve the PCA problem, they are not the gradients of any utility function. Lastly, our formulation consists entirely of generalized inner products: $\langle \hat{v}_i, M \hat{v}_j \rangle = \langle X \hat{v}_i, X \hat{v}_j \rangle$. Each $X \hat{v}_i$ can be thought of as a shallow function approximator with weights $\hat{v}_i$. This means that our formulation is readily extended to more general function approximation, i.e., $X \hat{v}_i \rightarrow f_i(X)$. Note that any formulation that operates on $\langle \hat{v}_i, \hat{v}_j \rangle$ instead is not easily generalized.

### H Nash Proof

Let $\hat{V}$ be a matrix of arbitrary unit-length column vectors $\langle \hat{v}_j \rangle$ and let $M$ (symmetric) be diagonalized as $U A U^T$ with $U$ a unitary matrix. Then,

$$R \triangleq \hat{V}^T M \hat{V} = \hat{V}^T U A U^T \hat{V} = (U^T \hat{V})^T A (U^T \hat{V}) = Z^T A Z$$

where $Z$ is also a matrix of unit-length column vectors because unitary matrices preserve inner products $(\langle U^T \hat{v}_i, U^T \hat{v}_j \rangle = \hat{v}_i^T U U^T \hat{v}_j = \hat{v}_i^T \hat{v}_j = 1)$. Therefore, rather than considering the action of an arbitrary matrix $\hat{V}$ on $M$, we can consider the action of an arbitrary matrix $Z$ on $A$. This simplifies the analysis.

In light of this reduction, Equation 22 of Theorem H.1 can be rewritten as

$$u_i(\hat{v}_i | v_{j<i}) = w^T A_{jj \geq i} w \quad (17)$$

$$\hat{v}_i^T A_{jj \geq i} \hat{v}_i \quad (18)$$

\[10\text{Empirically, replacing } ||\hat{v}_i|| = 1 \text{ with } ||\hat{v}_i|| \leq 1 \text{ does not harm performance while the latter is easier to enforce on neural networks for example.} \]
because \( V \) is identity w.l.o.g. Therefore, player \( i \)'s problem is simply to find the maximum eigenvector of a transformed matrix \( \Lambda_{jj} \geq 0 \), i.e., \( \Lambda \) with the first \( i - 1 \) eigenvalues removed.

**Theorem H.1** (PCA Solution is the Unique strict-Nash Equilibrium). Assume that the top-\( k \) eigenvalues of \( X^T X \) are distinct. Then the top-\( k \) eigenvectors form the unique strict-Nash equilibrium of the proposed game in Equation (6).

**Proof.** In what follows, let \( p, q = \{1, \ldots, d\} \) and \( i \in \{1, \ldots, k\} \). We will prove optimality of \( v_i \) by induction. Clearly, \( v_1 \) is the optimum of \( u_1 \) because \( u_1 = (v_1, Mv_1) = (\Lambda_{ii}, v_i) = \Lambda_{11} \) is the Rayleigh quotient which is known to be maximized for the maximal eigenvalue \( \Lambda_{11} \). Now, Consider

\[
\hat{v}_i = \sum_{p=1}^d w_p v_p \quad \text{as a linear combination of the true eigenvectors.}
\]

To ensure \( ||v_1|| = 1 \), we require \( ||w|| = 1 \). Then,

\[
u_i(\hat{v}_i | v_j < i) = \hat{v}_i^T M \hat{v}_i - \sum_{j < i} (\hat{v}_i^T M v_j)^2 / \Lambda_{jj}
\]

where \( z_p = w_p^2 \), and \( z^* = \arg \max(A_{pp \geq i}) = e_i \) is unique. Assume each player \( i \) plays \( e_i \). Any player \( j \) that unilaterally deviates from \( e_i \) strictly decreases their utility, therefore, the Nash is unique up to a sign change due to \( z^* = e_i = w_i^2 \). This is expected as both \( v_i \) and \( -v_i \) are principal components. \( \square \)

## I Without the Hierarchy

In Section 3 we defined utilities to respect the natural hierarchy of eigenvectors sorted by eigenvalue and mentioned that this eased analysis. Here, we provide further detail as to the difficulty of analyzing the game without the hierarchy. Consider the following alternative definition of the utilities:

\[
u_i(\hat{v}_i | v_{-i}) = \hat{v}_i^T M \hat{v}_i - \sum_{j \neq i} (\hat{v}_i^T M \hat{v}_j)^2 / \hat{v}_j^T M \hat{v}_j
\]

where the sum is now over all \( j \neq i \) instead of \( j < i \) as in Equation (6). With this form, the game is now symmetric across all players \( i \). Despite the symmetry of the game, we can easily rule out the existence of a symmetric Nash.

**Proposition I.1.** The EigenGame defined using symmetric utilities in Equation (23) does not contain a symmetric Nash equilibrium (assuming \( k \geq 2 \) and \( \text{rank}(M) \geq 2 \)).

**Proof by Contradiction.** Assume a symmetric Nash exists, i.e., \( \hat{v}_i = \hat{v}_j \) for all \( i, j \). The utility of a symmetric Nash using equation Equation (23) is

\[
u_i(\hat{v}_i | v_{-i}) = (1 - (n - 1))(\hat{v}_i^T M \hat{v}_i) = (2 - n)(\hat{v}_i^T M \hat{v}_i) \leq 0.
\]

Consider a unilateral deviation of \( \hat{v}_i \) to a direction orthogonal to \( \hat{v}_i \), i.e., \( \hat{v}_{\perp} \perp \hat{v}_i \) such that

\[
u_i(\hat{v}_i, \hat{v}_{\perp}) = (\hat{v}_{\perp}^T M \hat{v}_{\perp}) > 0.
\]

This utility is positive because \( \text{rank}(M) \geq 2 \) and therefore, always greater than the supposed Nash. Therefore, there is no symmetric Nash. \( \square \)

We can also prove that the true PCA solution is a Nash of this version of EigenGame.

**Proposition I.2.** The the top-k eigenvectors of \( M \) form a strict-Nash equilibrium of the EigenGame defined using symmetric utilities in Equation (23) (assuming \( \text{rank}(M) \geq k \)).
Proof. Let $\hat{v}_i = v_i$. We will assume this standard ordering, however, the proof follows through for any permutation of the eigenvectors. Clearly, the largest eigenvector is a best response to the spectrum because the penalty term (2nd term in Equation (23)) cannot be decreased below zero and the Rayleigh term (first term) is maximal, i.e., $v_1 = \arg \max_{v_1} u_1(\hat{v}_1, v_{-1})$. So assume $v_i$ is another eigenvector and consider representing $\hat{v}_i$ as $\hat{v}_i = \sum_{p=1}^{d} w_p v_p$ as before in Section H. Repeating those same steps, we find

$$u_i(\hat{v}_i, v_{-i}) = \sum_{q} w_q^2 \Lambda_{qq} - \sum_{j \neq i} \Lambda_{jj} w_j^2 = \Lambda_{ii} z_i$$

(26)

where $z_k = w_k^2, z \in \Delta^{n-1}$. Assuming $\Lambda_{ii} > 0$, this objective is uniquely maximized for $z_i = 1$ and $z_k = 0$ for all $k \neq i$. Therefore, $v_i = \arg \max_{v_i} u_i(\hat{v}_i, v_{-i})$.

However, we were unable to prove that it is the only Nash. It is possible that other Nash equilibria exist. Instead of focusing on determining whether a second Nash equilibrium exists (which is NP-hard [15, 23]), we learned through experiments that the EigenGame variant that incorporates knowledge of the hierarchy is much more performant. We leave determining uniqueness of the PCA solution for the less performant variant as an academic exercise.
J Error Propagation

J.1 Generalities

Notation. We can parameterize a vector on the sphere using the Riemannian exponential map, $\text{Exp}$, applied to a vector deviation from an anchor point. Formally, let $\hat{v}_j = \text{Exp}_{v_j}(\theta_j \Delta_j) = \cos(\theta_j)v_j + \sin(\theta_j)\Delta_j$ where $v_j$ is the $j$th largest eigenvector and $\Delta_j \in S^{d-1}$ is such that $\langle \Delta_j, v_j \rangle = 0$. Therefore, $\theta_j$ measures how far $\hat{v}_j$ deviates from $v_j$ in radians and $\Delta_j$ denotes the direction of deviation.

Let $\Lambda_{ii}$ denote the $i$th largest eigenvalue and $v_i$ the associated eigenvector. Also define the eigenvalue gap $g_i = \Lambda_{ii} - \Lambda_{i+1,i+1}$. Finally, let $\kappa_i = \frac{\Lambda_{ii}}{\Lambda_{i+1,i+1}}$ denote the $i$th condition number.

The following Lemma decomposes the utility of a player when the parents have learnt the preceding eigenvectors perfectly.

**Lemma J.1.** Let $\hat{v}_i = \cos(\theta_i)v_i + \sin(\theta_i)\Delta_i$ without loss of generality. Then

$$u_i(\hat{v}_i, v_{j<i}) = u_i(v_i, v_{j<i}) - \sin^2(\theta_i)\left(\Lambda_{ii} - \sum_{l>i} z_l \Lambda_{ll}\right).$$

(27)

**Proof.** Note that $\Delta_i$ can also be decomposed as $\Delta_i = \sum_{l=1}^d w_l v_l \parallel w \parallel = 1$ without loss of generality and that by Theorem [H.1], this implies $u_i(\langle \Delta_i, v_{j<i} \rangle) = \sum_{l>i} z_l \Lambda_{ll}$. This can be simplified further because $\langle \Delta_i, v_{j<i} \rangle = 0$ by its definition, which implies that $z_l = 0$. Therefore, more precisely, $u_i(\langle \Delta_i, v_{j<i} \rangle) = \sum_{l>i} z_l \Lambda_{ll}$. Continuing we find

$$u_i(\hat{v}_i, v_{j<i}) = \langle \hat{v}_i, \Lambda \hat{v}_i \rangle - \sum_{j<i} \langle \hat{v}_i, \Lambda v_j \rangle^2.$$  

(28)

$$= \langle \hat{v}_i, \Lambda \hat{v}_i \rangle - \sum_{j<i} \Lambda_{jj} \langle \hat{v}_i, v_j \rangle^2.$$  

(29)

$$= (\cos^2(\theta_i)\Lambda_{ii} + \sin^2(\theta_i)\langle \Delta_i, \Lambda \Delta_i \rangle) - \sum_{j<i} \Lambda_{jj} \langle \cos(\theta_i)v_i + \sin(\theta_i)\Delta_i, v_j \rangle^2.$$  

(30)

$$= (\cos^2(\theta_i)\Lambda_{ii} + \sin^2(\theta_i)\langle \Delta_i, \Lambda \Delta_i \rangle) - \sum_{j<i} \Lambda_{jj} \sin^2(\theta_i) \langle \Delta_i, v_j \rangle^2.$$  

(31)

$$= \Lambda_{ii} - \sin^2(\theta_i)\Lambda_{ii} + \sin^2(\theta_i)\left[\langle \Delta_i, \Lambda \Delta_i \rangle - \sum_{j<i} \Lambda_{jj} \langle \Delta_i, v_j \rangle^2 \right]$$  

(32)

$$= u_i(v_i, v_{j<i}) - \sin^2(\theta_i)\left(\Lambda_{ii} - u_i(\langle \Delta_i, v_{j<i} \rangle)\right).$$  

(33)

$$= u_i(v_i, v_{j<i}) - \sin^2(\theta_i)\left(\Lambda_{ii} - \sum_{l>i} z_l \Lambda_{ll}\right).$$  

(34)

$\square$

J.2 Summary of Error Propagation Results

Player $i$’s utility is sinusoidal in the angular deviation of $\theta_i$ from the optimum. The amplitude of the sinusoid varies with the direction of the angular deviation along the sphere and is dependent on the accuracy of players $j < i$. In the special case where players $j < i$ have learned the top-$(i-1)$ eigenvectors exactly, player $i$’s utility simplifies (see Lemma [J.1]) to

$$u_i(\hat{v}_i, v_{j<i}) = \Lambda_{ii} - \sin^2(\theta_i)\left(\Lambda_{ii} - \sum_{l>i} z_l \Lambda_{ll}\right).$$  

(35)

Note that $\sin^2$ has period $\pi$ as opposed to $2\pi$, which simply reflects the fact that $v_i$ and $-v_i$ are both eigenvectors.

The angular distance between $v_i$ and the maximizer of player $i$’s utility with approximate parents has $\tan^{-1}$ dependence (i.e., a soft step-function; see Lemma [J.5]). Figure [10] plots the dependence for a synthetic problem. This dependence reveals that there is an error threshold players $j < i$ must fall below in order for player $i$ to accurately learn the $i$-th eigenvector.
J.3 Theorem and Proofs

In Theorem J.2, we prove that given parents close enough to their corresponding true eigenvectors, the angular deviation of a local maximizer of a child’s utility from the child’s true eigenvector is below a derived threshold. In other words, given accurate parents, a child can successfully proceed to approximate its corresponding eigenvector (its utility is well posed). We prove this theorem in several steps.

First we show in Lemma J.3 that the child’s utility function can be written as a composition of sinusoids with dependence on the angular deviation from the child’s true eigenvector. The amplitude of the sinusoid depends on the directions in which the child and parents have deviated from their true eigenvectors along their spheres. We then simplify the composition of sinusoids to a single sinusoid in Lemma J.4. Any local max of a sinusoid is also a global max. Therefore, to upper bound the angular deviation of the child’s local maximizer from its true corresponding eigenvector, we consider the worst case direction for the maximizer to deviate from the true eigenvector.

In Lemma J.5, we give a closed form solution for the angular deviation of a maximizer of a child’s utility given any parents and deviation directions. This dependence is given by the arctan function which resembles a soft step function with a linear regime for small angular deviations, followed by a step, and then another linear regime for large angular deviations. The argument of the arctan function is a ratio of terms, each with dependence on the parents’ angular deviations and directions of deviation.

We establish two minor lemmas, Lemma J.6 and Lemma J.7, to help bound the denominator in Theorem J.2. Assume it is given that $\theta_j \approx \frac{\theta_i}{|z|}$ fall below a threshold (e.g., $\approx 18^\circ$ here) in order for the maximizer of player $i$’s utility to lie near the true $i$th eigenvector ($y$-axis). The matrix $M$ for this example has a condition number $\kappa_i = \frac{\Lambda_{ii}}{\Lambda_{ii}} = 10$.

Proof. By Lemma J.11, $A < 0$ for $c_i \leq \frac{1}{8}$. Therefore, $|\theta_j^*| = \frac{1}{2} \tan^{-1} \left( \frac{|B|}{|A|} \right)$ by Lemma J.5. Also, note that for $z \leq \frac{1}{2}$, $\tan^{-1}(|z|) \leq |z|$. Setting $c_i \leq \frac{1}{16}$ to ensure $z = \frac{|z|}{1} \leq \frac{1}{2}$. Then,

$$|\theta_j^*| = \frac{1}{2} \tan^{-1} \left( \frac{|B|}{|A|} \right) \leq \frac{1}{2} \frac{|B|}{|A|} \leq \frac{1}{2} \frac{|B|}{1-8c_i} \leq 8c_i.$$

Another way of writing this is $|\theta_j^*| = \frac{1}{2} \tan^{-1} \left( \frac{|B|}{|A|} \right) \leq \frac{1}{2} \frac{|B|}{1-8c_i}$.

Lemma J.3. Let $\hat{v}_j = \cos(\theta_j)v_j + \sin(\theta_j)\Delta_j$ for all $j \leq i$ without loss of generality. Then

$$u_i(\hat{v}_i, \hat{v}_{j<i}) = A(\theta_j, \Delta_j, \Delta_i) \sin^2(\theta_i) - B(\theta_j, \Delta_j, \Delta_i) \frac{\sin(2\theta_i)}{2} + C(\theta_j, \Delta_j, \Delta_i)$$

(38)
We abbreviate the above to $A$, $B$, $C$ to avoid clutter in all upcoming statements and proofs. These functions are dependent on all variables except $\theta_i$.

\[
A(\theta_j, \Delta_j, \Delta_i) = \|\Delta_i\|_{\Lambda^{-1}} - \Lambda_{ii}
\]
\[
- \sum_{j<i} \frac{\Lambda_{jj}^2 \cos^2(\theta_j) \langle \Delta_i, v_j \rangle^2 - \Lambda_{ii}^2 \sin^2(\theta_j) \langle \Delta_j, v_i \rangle^2 + \sin^2(\theta_j) \langle \Delta_i, \Lambda \Delta_j \rangle^2}{\Lambda_{jj} \cos(\theta_j)^2 + \|\Delta_j\|_{\Lambda^{-1}} \sin^2(\theta_j)}
\]
\[
- \sum_{j<i} \frac{\Lambda_{jj} \sin(2\theta_j) \langle \Delta_i, v_j \rangle \langle \Delta_i, \Lambda \Delta_j \rangle}{\Lambda_{jj} \cos(\theta_j)^2 + \|\Delta_j\|_{\Lambda^{-1}} \sin^2(\theta_j)}
\]
\[
B(\theta_j, \Delta_j, \Delta_i) = \sum_{j<i} \frac{\Lambda_{ii} \Lambda_{jj} \sin(2\theta_j) \langle \Delta_j, v_i \rangle \langle \Delta_i, \Lambda \Delta_j \rangle + 2 \Lambda_{ii} \sin^2(\theta_j) \langle \Delta_j, v_i \rangle \langle \Delta_i, \Lambda \Delta_j \rangle}{\Lambda_{jj} \cos(\theta_j)^2 + \|\Delta_j\|_{\Lambda^{-1}} \sin^2(\theta_j)}
\]
\[
C(\theta_j, \Delta_j, \Delta_i) = \Lambda_{ii} - \sum_{j<i} \frac{\Lambda_{jj}^2 \sin^2(\theta_j) \langle \Delta_j, v_i \rangle^2}{\Lambda_{jj} \cos(\theta_j)^2 + \|\Delta_j\|_{\Lambda^{-1}} \sin^2(\theta_j)}
\]

**Proof.** Note that the true eigenvectors are orthogonal, so in what follows, any $\langle v_i, v_j \rangle = 0$ where $j \neq i$. Also, recall that $2 \sin(\varphi) \cos(\varphi) = \sin(2\varphi)$. We highlight some but not all such simplifications. Finally, we recognize $\langle \Delta_i, \Lambda \Delta_i \rangle = \|\Delta_i\|_{\Lambda^{-1}}$ as the generalized norm of $\Delta_i$ or the Mahalanobis distance from the origin.

\[
u_i(\hat{v}_i, \hat{v}_{j<i})
\]
\[
= \langle \hat{v}_i, \Lambda \hat{v}_i \rangle - \sum_{j<i} \frac{\langle \hat{v}_i, \Lambda \hat{v}_j \rangle^2}{\langle \hat{v}_j, \Lambda \hat{v}_j \rangle}
\]
\[
= \langle \cos(\theta_i) v_i + \sin(\theta_i) \Delta_i, \Lambda (\cos(\theta_i) v_i + \sin(\theta_i) \Delta_i) \rangle
\]
\[
- \sum_{j<i} \langle \cos(\theta_j) v_j + \sin(\theta_j) \Delta_j, \Lambda (\cos(\theta_j) v_j + \sin(\theta_j) \Delta_j) \rangle^2
\]
\[
= \Lambda_{ii} \cos(\theta_i)^2 + (\Delta_i, \Lambda \Delta_i) \sin^2(\theta_i)
\]
\[
- \sum_{j<i} \langle \cos(\theta_j) v_j + \sin(\theta_j) \Delta_j, \Lambda (\cos(\theta_j) v_j + \sin(\theta_j) \Delta_j) \rangle^2
\]
\[
= \Lambda_{ii} \cos(\theta_i)^2 + \|\Delta_i\|_{\Lambda^{-1}} \sin^2(\theta_i)
\]
\[
- \sum_{j<i} \frac{\langle \Lambda_{jj} \sin(\theta_i) \cos(\theta_j) \langle \Delta_i, v_j \rangle + \Lambda_{ii} \sin(\theta_i) \cos(\theta_j) \langle \Delta_j, v_i \rangle + \sin(\theta_i) \sin(\theta_j) \langle \Delta_i, \Lambda \Delta_j \rangle^2}{\Lambda_{jj} \cos(\theta_j)^2 + \|\Delta_j\|_{\Lambda^{-1}} \sin^2(\theta_j)}
\]
Developing the numerator of the fraction, we obtain terms in \( \sin \) and in \( \sin^2 \) that we later regroup to obtain the result:

\[
\begin{align*}
= \Lambda_{ii} - \Lambda_{ii} \sin(\theta_i)^2 + ||\Delta_i||^2_{\Lambda^{-1}} \sin^2(\theta_i) \\
- \sum_{j<i} \frac{\Lambda_{jj}^2 \sin^2(\theta_j) \cos^2(\theta_j) (\Delta_j, v_j)^2 + \Lambda_{jj}^2 \sin^2(\theta_j) (\Delta_j, \Lambda \Delta_j)^2}{\Lambda_{jj} \cos(\theta_j)^2 + ||\Delta_j||^2_{\Lambda^{-1}} \sin^2(\theta_j)} \\
- 2 \sum_{j<i} \frac{\Lambda_{ii} \Lambda_{jj} \sin(\theta_i) \sin(\theta_j) \cos(\theta_j) \cos(\theta_j) (\Delta_i, v_i) (\Delta_i, v_j)}{\Lambda_{jj} \cos(\theta_j)^2 + ||\Delta_j||^2_{\Lambda^{-1}} \sin^2(\theta_j)} \\
- 2 \sum_{j<i} \frac{\Lambda_{jj} \sin^2(\theta_i) \sin(\theta_j) \cos(\theta_j) (\Delta_i, v_j) (\Delta_i, \Lambda \Delta_j)}{\Lambda_{jj} \cos(\theta_j)^2 + ||\Delta_j||^2_{\Lambda^{-1}} \sin^2(\theta_j)} \\
- 2 \sum_{j<i} \frac{\Lambda_{ii} \sin(\theta_i) \cos(\theta_j) \cos(\theta_j) (\Delta_j, v_i) (\Delta_i, \Lambda \Delta_j)}{\Lambda_{jj} \cos(\theta_j)^2 + ||\Delta_j||^2_{\Lambda^{-1}} \sin^2(\theta_j)} \\
= \Lambda_{ii} - \Lambda_{ii} \sin^2(\theta_i) + ||\Delta_i||^2_{\Lambda^{-1}} \sin^2(\theta_i) \\
- \sum_{j<i} \frac{\Lambda_{jj}^2 \sin^2(\theta_j) \cos^2(\theta_j) (\Delta_j, v_j)^2 + \Lambda_{jj}^2 \sin^2(\theta_j) (\Delta_j, \Lambda \Delta_j)^2}{\Lambda_{jj} \cos(\theta_j)^2 + ||\Delta_j||^2_{\Lambda^{-1}} \sin^2(\theta_j)} \\
- \frac{1}{2} \sum_{j<i} \frac{\Lambda_{ii} \Lambda_{jj} \sin(2\theta_j) \sin(2\theta_j) (\Delta_i, v_i) (\Delta_i, v_j)}{\Lambda_{jj} \cos(\theta_j)^2 + ||\Delta_j||^2_{\Lambda^{-1}} \sin^2(\theta_j)} \\
- \sum_{j<i} \frac{\Lambda_{jj} \sin^2(\theta_i) \sin(2\theta_j) (\Delta_i, \Lambda \Delta_j)}{\Lambda_{jj} \cos(\theta_j)^2 + ||\Delta_j||^2_{\Lambda^{-1}} \sin^2(\theta_j)} \\
- \sum_{j<i} \frac{\Lambda_{ii} \sin(2\theta_i) \sin(2\theta_j) (\Delta_j, v_i) (\Delta_i, \Lambda \Delta_j)}{\Lambda_{jj} \cos(\theta_j)^2 + ||\Delta_j||^2_{\Lambda^{-1}} \sin^2(\theta_j)} \\
= \frac{1}{2} \sum_{j<i} \frac{\Lambda_{ii} \Lambda_{jj} \sin(2\theta_i) (\Delta_j, v_i) (\Delta_i, v_j) + 2 \Lambda_{ii} \sin(2\theta_i) (\Delta_j, v_i) (\Delta_i, \Lambda \Delta_j)}{\Lambda_{jj} \cos(\theta_j)^2 + ||\Delta_j||^2_{\Lambda^{-1}} \sin^2(\theta_j)} \\
+ \Lambda_{ii} - \frac{\sum_{j<i} \Lambda_{jj}^2 \sin^2(\theta_j) (\Delta_j, v_j)^2}{\Lambda_{jj} \cos(\theta_j)^2 + ||\Delta_j||^2_{\Lambda^{-1}} \sin^2(\theta_j)} \\
\overset{\text{def}}{=} \Lambda \sin^2(\theta_i) - B \frac{\sin(2\theta_i)}{2} + C.
\end{align*}
\]

Collecting terms, we find

\[
\begin{align*}
\frac{u_i(\dot{v}_i, \dot{v}_j < \iota)}{u_i(\dot{v}_i, \Delta_i, \dot{v}_j < \iota)} &= \sin^2(\theta_i) \left( ||\Delta_i||^2_{\Lambda^{-1}} - \Lambda_{ii} \right) \\
- \sum_{j<i} \Lambda_{jj} \cos^2(\theta_j) (\Delta_j, v_j)^2 - \Lambda_{ii} \sin^2(\theta_j) (\Delta_j, v_j)^2 + \sin^2(\theta_j) (\Delta_i, \Lambda \Delta_j)^2 \\
- \sum_{j<i} \Lambda_{jj} \sin^2(\theta_i) \sin(2\theta_j) (\Delta_i, \Lambda \Delta_j) \\
- \frac{\sin(2\theta_i)}{2} \left[ \sum_{j<i} \Lambda_{ii} \Lambda_{jj} \sin(2\theta_j) (\Delta_j, v_j) (\Delta_i, v_j) + 2 \Lambda_{ii} \sin(2\theta_i) (\Delta_j, v_i) (\Delta_i, \Lambda \Delta_j) \right] \\
&+ \Lambda_{ii} - \sum_{j<i} \Lambda_{jj} \sin^2(\theta_j) (\Delta_j, v_j)^2 \\
&\overset{\text{def}}{=} \Lambda \sin^2(\theta_i) - B \frac{\sin(2\theta_i)}{2} + C.
\end{align*}
\]

\[\square\]

**Lemma J.4.** The utility function along \( \Delta_i, \theta :\rightarrow u_i(\dot{v}_i(\theta_i, \Delta_i), \dot{v}_j < \iota) \), is sinusoidal with period \( \pi \):

\[
\begin{align*}
\frac{u_i(\dot{v}_i(\theta_i, \Delta_i), \dot{v}_j < \iota)}{(\dot{v}_i(\theta_i, \Delta_i), \dot{v}_j < \iota))} &= \frac{1}{2} \left( \sqrt{A^2 + B^2 \cos(\theta_i + \phi)} + A + 2C \right)
\end{align*}
\]

where \( \phi = \tan^{-1} \left( \frac{B}{A} \right) \).
Proof. Starting from Lemma J.3, we find

\[ u_i(\hat{v}_i(\theta_i, \Delta_i), \hat{v}_{j<i}) = A \sin^2(\theta_i) - B \frac{\sin(2\theta_i)}{2} + C \]  \hspace{1cm} (65)

\[ = A \frac{1 - \cos(2\theta_i)}{2} - B \frac{\sin(2\theta_i)}{2} + C \]  \hspace{1cm} (66)

\[ = \frac{1}{2} \left[ -A \cos(2\theta_i) - B \sin(2\theta_i) + A + 2C \right] \]  \hspace{1cm} (67)

\[ = \frac{1}{2} \left[ \sqrt{A^2 + B^2 \cos(2\theta_i + \phi)} + A + 2C \right] \]  \hspace{1cm} (68)

where \( \phi = \tan^{-1} \left( \frac{B}{A} \right) \).

Lemma J.5. The angular deviation, \( \theta_i \), of the vector that maximizes the mis-specified objective, \( \arg \max_{\theta_i} u_i(\hat{v}_i(\theta_i, \Delta_i), \hat{v}_{j<i}) \), is given by

\[ |\theta_i^*| = \begin{cases} \frac{1}{2} \tan^{-1} \left( \frac{B}{A} \right) & \text{if } A < 0 \\ \frac{\pi}{2} & \text{if } A = 0 \\ \frac{1}{2} \left[ \pi - \tan^{-1} \left( \frac{B}{A} \right) \right] & \text{if } A > 0 \end{cases} \]  \hspace{1cm} (69)

where \( A \) and \( B \) are given by Lemma J.3.

Proof. First, we identify the critical points:

\[ \frac{\partial}{\partial \theta_i} u_i(\hat{v}_i, \hat{v}_{j<i}) = 2A \sin(\theta_i) \cos(\theta_i) - B \cos(2\theta_i) = 0 \]  \hspace{1cm} (70)

\[ = A \sin(2\theta_i) - B \cos(2\theta_i) = 0 \]  \hspace{1cm} (71)

\[ = \frac{1}{\cos(2\theta_i)} \left[ \tan(2\theta_i)A - B \right] = 0 \]  \hspace{1cm} (72)

\[ \tan(2\theta_i) = \frac{B}{A}. \]  \hspace{1cm} (73)

Then we determine maxima vs minima:

\[ \frac{\partial^2}{\partial \theta_i^2} u_i(\hat{v}_i, \hat{v}_{j<i}) = \frac{2}{\cos(2\theta_i)} \left[ B \tan(2\theta_i) + A \right] = \frac{2}{\cos(2\theta_i)} \left[ \frac{B^2}{A} + A \right], \]  \hspace{1cm} (74)

therefore, \( \text{sign}(\frac{\partial^2}{\partial \theta_i^2} u_i) = \text{sign}(\cos(2\theta_i)) \text{sign}(A) < 0 \) for \( \theta_i \) to be a local maximum. If \( A < 0 \), then \( \theta_i^* \) must lie within \( [-\frac{\pi}{4}, \frac{\pi}{4}] \). If \( A > 0 \), then \( \theta_i^* \) must lie within \( [-\frac{\pi}{2}, -\frac{\pi}{4}] \) or \( [\frac{\pi}{2}, \frac{\pi}{4}] \). By inspection, if \( A = 0 \), then \( u_i \) is maximized at \( \theta_i = -\frac{\pi}{4} \text{sign}(B) \). In general, we are interested in the magnitude of \( \theta_i \), not its sign. \( \Box \)

Lemma J.6. The following relationship is useful for proving Lemma J.8

\[ \frac{b}{a+c} = \frac{b}{a} \left[ 1 - \frac{c}{a+c} \right] \]  \hspace{1cm} (75)

Proof.

\[ \frac{b}{a+c} = \frac{b}{a} + x \]  \hspace{1cm} (76)

\[ \Rightarrow x = \frac{b}{a+c} - \frac{b}{a} = \frac{1}{a+c} - \frac{1}{a} \]  \hspace{1cm} (77)

\[ = \frac{a - (a+c)}{a(a+c)} = -\frac{b}{a} \left[ \frac{a}{a+c} \right], \]  \hspace{1cm} (78)

Lemma J.7. If \( \langle \Delta_i, v_i \rangle = 0 \), then \( u_i(\Delta_i, v_{j<i}) \leq \Lambda_{i+1,i+1} \).
Proof. Recall the Nash proof in Appendix II:

\[ u_i(\Delta_i, v_{j<i}) = \sum_{p \geq i} \Lambda_{pp} z_p \]  

(79)

where \( z_p = w_p^2, \Delta_i = \sum_{p=1}^d w_p v_p \), and \( z \in \Delta^{d-1} \). The fact that \( \langle \Delta_i, v_i \rangle = 0 \) implies that \( z_i = 0 \). Therefore, the utility simplifies to

\[ u_i(\Delta_i, v_{j<i}) = \sum_{p \geq i+1} \Lambda_{pp} z_p \]  

(80)

which is upper bounded by \( \Lambda_{i+1,i+1} \).

\[ \square \]

Lemma J.8. Assume \( |\theta_j| \leq \epsilon \) for all \( j < i \) (implies \( \sin^2(\theta_j) \leq \epsilon^2 \)). Then

\[ A \leq -g_i + (i - 1)(\Lambda_{11} + \Lambda_{ii}) \frac{\epsilon^2}{1 - \epsilon^2} + 2(i - 1)\Lambda_{11} \frac{\epsilon}{\sqrt{1 - \epsilon^2}}. \]  

(81)

Proof.

\[ A(\theta_{j<i}) = ||\Delta_i||^2_{\lambda^{-1}} - \Lambda_{ii} \]

\[ = \sum_{j<i} \frac{\Lambda^2_{jj} \cos^2(\theta_j)\langle \Delta_i, v_j \rangle^2 - \Lambda^2_{ii} \sin^2(\theta_j)\langle \Delta_i, v_j \rangle^2 + \sin^2(\theta_j)\langle \Delta_i, \Delta_j \rangle^2}{\Lambda_{jj} \cos(\theta_j) + \Lambda_{jj} \cos^2(\theta_j) + ||\Delta_i||^2_{\lambda^{-1}} \sin^2(\theta_j)} - \Lambda_{ii} \]  

(82)

\[ \leq \sum_{j<i} \frac{-\Lambda^2_{jj} \sin^2(\theta_j)\langle \Delta_i, v_j \rangle^2 + \sin^2(\theta_j)\langle \Delta_i, \Delta_j \rangle^2 + \Lambda_{jj} \sin(2\theta_j)\langle \Delta_i, v_j \rangle\langle \Delta_i, \Delta_j \rangle}{\Lambda_{jj} \cos(\theta_j) + \Lambda_{jj} \cos^2(\theta_j) + ||\Delta_i||^2_{\lambda^{-1}} \sin^2(\theta_j)} \]  

(83)

\[ \leq \sum_{j<i} \frac{-\Lambda^2_{jj} \sin^2(\theta_j)\langle \Delta_i, v_j \rangle^2 + \sin^2(\theta_j)\langle \Delta_i, \Delta_j \rangle^2 + \Lambda_{jj} \sin(2\theta_j)\langle \Delta_i, v_j \rangle\langle \Delta_i, \Delta_j \rangle}{\Lambda_{jj} \cos(\theta_j) + \Lambda_{jj} \cos^2(\theta_j) + ||\Delta_i||^2_{\lambda^{-1}} \sin^2(\theta_j)} \]  

(84)

\[ \leq \sum_{j<i} \frac{-\Lambda^2_{jj} \sin^2(\theta_j)\langle \Delta_i, v_j \rangle^2 + \sin^2(\theta_j)\langle \Delta_i, \Delta_j \rangle^2 + \Lambda_{jj} \sin(2\theta_j)\langle \Delta_i, v_j \rangle\langle \Delta_i, \Delta_j \rangle}{\Lambda_{jj} \cos(\theta_j) + \Lambda_{jj} \cos^2(\theta_j) + ||\Delta_i||^2_{\lambda^{-1}} \sin^2(\theta_j)} \]  

(85)

\[ = u_i(\Delta_i, v_{j<i}) + \sum_{j<i} \frac{\sin^2(\theta_j)\langle \Delta_i, v_j \rangle^2}{\Lambda_{jj} \cos^2(\theta_j)} - \Lambda_{ii} \]  

(86)

\[ + \sum_{j<i} \frac{\Lambda^2_{ii} \sin^2(\theta_j)\langle \Delta_i, v_j \rangle^2 + 2\Lambda_{jj} \sqrt{\sin^2(\theta_j) \cos^2(\theta_j)}\langle \Delta_i, v_j \rangle\langle \Delta_i, \Delta_j \rangle}{\Lambda_{jj} \cos^2(\theta_j)} \]  

(87)
Assume $\epsilon$. Proof.

Assume Lemma J.10.

Lemma J.11. Let $\Lambda_i \leq \Lambda$ for all $i < j$.

Note $\frac{\Lambda_i^2}{\Lambda_{jj}} < \Lambda_{ii}$ because $\Lambda_{ii} < \Lambda_{jj}$ for all $j < i$.

Lemma J.9. Assume $\epsilon^2 \leq \frac{1}{2}$. Then

$$A \leq -g_i + 8(i - 1)\Lambda_{ii}\epsilon. \quad (91)$$

Assume $\epsilon^2 \leq \frac{1}{2}$ so $\frac{\epsilon}{\sqrt{1 - \epsilon^2}} \leq 1$. Then

$$A \leq \Lambda_{ii} + (i - 1)(\Lambda_{ii} + \Lambda_{ij}) \frac{\epsilon^2}{1 - \epsilon^2} + 2(i - 1)\Lambda_{ii} \frac{\epsilon}{\sqrt{1 - \epsilon^2}} \quad (92)$$

$$\leq -g_i + (i - 1) \left[ \frac{\epsilon}{\sqrt{1 - \epsilon^2}} \right] [3\Lambda_{ii} + \Lambda_{ij}] \quad (93)$$

$$\leq -g_i + 4(i - 1)\Lambda_{ii} \frac{\epsilon}{\sqrt{1 - \epsilon^2}} \quad (94)$$

$$\leq -g_i + 8(i - 1)\Lambda_{ii}\epsilon. \quad (95)$$

Lemma J.10. Assume $\epsilon^2 \leq \frac{1}{2}$. Then

$$|B| \leq 8(i - 1)\Lambda_{ii}\kappa_{i-1}\epsilon. \quad (96)$$

Proof.

$$|B| = \sum_{j < i} |\Lambda_{ii}\Lambda_{jj}\sin(2\theta_j)(\Delta_{ii}, v_j)(\Delta_{ii}, v_j) + 2\Lambda_{ii}\sin^2(\theta_j)(\Delta_{ii}, v_j)(\Delta_{ii}, \Lambda_{jj})| \quad (97)$$

$$\leq \sum_{j < i} \Lambda_{ii}\Lambda_{jj}\frac{\sin^2(2\theta_j)}{\Lambda_{jj}\cos(\theta_j)^2} + 2\Lambda_{ii}\sin^2(\theta_j)\Lambda_{jj} \quad (98)$$

$$\leq \sum_{j < i} \Lambda_{ii}\Lambda_{jj}\frac{4\sin^2(\theta_j)\cos^2(\theta_j) + 2\Lambda_{ii}\sin^2(\theta_j)\Lambda_{jj}}{\Lambda_{jj}\cos(\theta_j)^2} \quad (99)$$

$$\leq 2 \sum_{j < i} \frac{\Lambda_{ii}\Lambda_{jj}\epsilon + \Lambda_{ii}\epsilon^2\Lambda_{jj}}{\Lambda_{jj}(1 - \epsilon^2)} \quad (100)$$

$$= 2\Lambda_{ii} \frac{\epsilon}{1 - \epsilon^2} \left( i - 1 + \epsilon \sum_{j < i} \kappa_j \right) \quad (101)$$

$$\leq 4\Lambda_{ii}\epsilon \left( i - 1 + \epsilon(i - 1)\kappa_{i-1} \right) \quad (102)$$

$$= 4(i - 1)\Lambda_{ii}\epsilon \left( 1 + \epsilon\kappa_{i-1} \right) \quad (103)$$

$$\leq 4(i - 1)\Lambda_{ii}\epsilon \left( 1 + \frac{1}{\sqrt{2}}\kappa_{i-1} \right) \quad (104)$$

$$\leq 8(i - 1)\Lambda_{ii}\kappa_{i-1}\epsilon. \quad (105)$$

Lemma J.11. Let $\epsilon_i = \frac{c_i g_i}{(i - 1)\Lambda_{ii}}$ with $c_i < \frac{1}{8}$. Then
\[ (i) \quad A \leq 0. \]
\[ (ii) \quad \left| \frac{B}{A} \right| \leq \frac{8c_i}{1-8c_i}. \]

**Proof.** Plugging in Lemma J.9 and \( \epsilon_i \), we find
\[ A \leq -g_i + 8c_i \frac{(i-1)A_{11}g_i}{(i-1)A_{11}} = -g_i + 8c_i g_i = (8c_i - 1)g_i. \] (106)

Since we assumed \( c_i < 1/8 \), this proves \((i)\). Plugging in Lemma J.10 and \( \epsilon_i \) solves \((ii)\):
\[ \text{Equation (106)} \implies |A| \geq (1 - 8c_i)g_i \quad \text{(107)} \]
\[ |B| \leq 8c_i \frac{(i-1)A_{ii}A_{i-1,i-1}g_i}{(i-1)A_{11}} = 8c_i g_i \frac{A_{ii}}{A_{i-1,i-1}} \leq 8c_i g_i \quad \text{(108)} \]
\[ \implies \left| \frac{B}{A} \right| \leq \frac{8c_i}{1 - 8c_i}. \quad \text{(109)} \]

**Example 1.** We construct the following example in order to concretely demonstrate the arctan dependence of a child \((v_i)\) on a parent \((v_1)\) in this case.

Let \( \Delta_1 = v_i \), \( \Delta_i = v_i \), \( \Delta_1 < j < i \), \( \Delta_1 < j < i \), and constrain all parents to have error \( \sin(\theta_j) = \epsilon \) for all \( j < i \). Then the child’s optimum has an angular deviation from the true eigenvector direction of

\[ |\theta_i^*| = \begin{cases} \frac{1}{2} \tan^{-1} \left( \frac{|B|}{A} \right) & \text{if } A < 0 \\ \frac{\pi}{4} & \text{if } A = 0 \\ \frac{1}{2} \left[ \pi - \tan^{-1} \left( \frac{|B|}{A} \right) \right] & \text{if } A > 0 \end{cases} \] (110)

where \( \frac{|B|}{A} = \frac{2c_i - 1 - c_i^2}{1 - r^2(\cos \theta_j + \sin \theta_j^*)} \).

**Proof.** Note that \( \langle \Delta_i, v_1 < i \rangle, \langle \Delta_1 < j < i, v_j \rangle, \) and \( \langle \Delta_i, \Delta_1 \rangle \) all equal 0 by design; and \( \langle \Delta_i, v_i \rangle = \langle \Delta_1, v_i \rangle = 1 \). Plugging into Lemma J.3 all elements of the sum disappear for \( j \geq 1 \) and only the blue terms survive for \( j \geq 1 \). We find
\[ A = ||\Delta_i||_{\Lambda^{-1}} - A_{ii} \quad \text{(111)} \]
\[ - \sum_{j < i} \Lambda_{jj}^2 \cos^2(\theta_j) \langle \Delta_i, v_j \rangle^2 - \Lambda_{ii}^2 \sin^2(\theta_j) \langle \Delta_i, v_i \rangle^2 + \sin^2(\theta_j) \langle \Delta_i, \Delta_1 \rangle^2 \]
\[ - \sum_{j < i} \Lambda_{jj} \sin(\theta_j) \langle \Delta_i, v_j \rangle \langle \Delta_i, \Delta_1 \rangle + ||\Delta_i||_{\Lambda^{-1}} \sin^2(\theta_j) \]
\[ = \Lambda_{11} - A_{ii} - \Lambda_{11}^2 (1 - \epsilon^2) - \Lambda_{ii}^2 \epsilon^2 \]
\[ = \Lambda_{11} - A_{ii} - \Lambda_{11}^2 (1 - \epsilon^2) - \Lambda_{ii}^2 \epsilon^2 \]
\[ = -A_{ii} + \epsilon^2 (\Lambda_{11} + \frac{A_{ii}}{\kappa_i}) \]

28
\[ B = \sum_{j<i} \Lambda_{ii} \Lambda_{jj} \sin(2\theta_j) \langle \Delta_j, v_i \rangle \langle \Delta_i, v_j \rangle + 2\Lambda_{ii} \sin^2(\theta_j) \langle \Delta_j, v_i \rangle \langle \Delta_i, \Lambda \Delta_j \rangle \]
\[ \Lambda_{jj} \cos(\theta_j)^2 + ||\Delta_j||^2_{\Lambda^{-1}} \sin^2(\theta_j) \]
\[ \Lambda_{ii} \Lambda_{11} \sin(2\theta_1) \]
\[ \Lambda_{11} \cos(\theta_1)^2 + ||\Delta_1||^2_{\Lambda^{-1}} \sin^2(\theta_1) \]
\[ = 2\Lambda_{ii} \Lambda_{11} \sqrt{\epsilon^2(1 - \epsilon^2)} \]
\[ \Lambda_{11}(1 - \epsilon^2) + \Lambda_{ii}^2 \epsilon^2 \]
\[ = 2\Lambda_{ii} \epsilon \sqrt{1 - \epsilon^2} \]

Then
\[ \frac{|B|}{A} = \frac{2\Lambda_{ii} \epsilon \sqrt{1 - \epsilon^2}}{|\Lambda_{ii} - \epsilon^2(\Lambda_{11} + \frac{\Delta_{ii}}{\Lambda_{ii}})} = \frac{2\epsilon \sqrt{1 - \epsilon^2}}{1 - \epsilon^2(\kappa_i + \frac{1}{\kappa_i})}. \]
K Convergence Proof

K.1 Non-Convex Riemannian Optimization Theory

We repeat the non-convex Riemannian optimization rates here from [10] for convenience.

**Lemma K.1.** Under Assumptions K.2 and K.3, generic Riemannian descent (Algorithm 5) returns $x \in \mathcal{M}$ satisfying $f(x) \leq f(x_0)$ and $\|\nabla R f(x)\| \leq \rho$ in at most

$$\left[ \frac{f(x_0) - f^*}{\xi} \cdot \frac{1}{\rho^2} \right]$$

iterations, provided $\rho \leq \frac{\xi}{\xi}$. If $\rho > \frac{\xi}{\xi}$, at most $\left[ \frac{f(x_0) - f^*}{\xi} \cdot \frac{1}{\rho} \right]$ iterations are required.

**Proof.** See Theorem 2.5 in [10].

**Assumption K.2.** There exists $f^* > -\infty$ such that $f(x) \geq f^*$ for all $x \in \mathcal{M}$. See Assumption 2.3 in [10].

**Assumption K.3.** There exist $\xi, \xi' > 0$ such that, for all $k \geq 0$, $f(x_k) - f(x_{k+1}) \geq \min(\xi\|\nabla R f(x_k)\|, \xi')\|\nabla R f(x_k)\|$. See Assumption 2.4 in [10].

**Algorithm 5** Generic Riemannian descent algorithm

Given: $f : \mathcal{M} \to \mathbb{R}$ differentiable, a retraction $\text{Retr}$ on $\mathcal{M}$, $x_0 \in \mathcal{M}$, $\rho > 0$

Init: $k \leftarrow 0$

while $\|\nabla R f(x_k)\| > \rho$ do

Pick $\eta_k \in T_{x_k} \mathcal{M}$

end while

return $x_k$

K.2 Convergence of EigenGame

Theorem K.4 provides an asymptotic convergence guarantee for Algorithm 1 (below) to recover the top-$k$ principal components. Assuming $\hat{v}_i$ is initialized within $\frac{\pi}{2}$ of $v_i$ for all $i \leq k$, Theorem K.5 provides a finite sample convergence rate. In particular, it specifies the total number of iterations required to learn parents such that $\hat{v}_k$ can be learned within a desired tolerance.

The proof of Theorem K.4 proceeds in several steps. First, recall that player $i$’s utility is sinusoidal in its angular deviation from $v_i$ and therefore, technically, non-concave although it is simple in the sense that every local maximum is a global maximum (w.r.t. angular deviation). Also, note that our ascent is not performed on the natural parameters of the sphere ($\theta_i$ and $\Delta_i$), but rather on $v_i$ directly with $\hat{v}_i \in S^{d-1}$, a Riemannian manifold.

We therefore leverage recent results in non-convex optimization, specifically minimization, for Riemannian manifolds [10], repeated here for convenience (see Theorem K.1). Note, we are maximizing a utility so we simply flip the sign of our utility to apply this theory. The convergence rate guaranteed by this theory is for generic Riemannian descent with a constant step size, Algorithm 5 and makes two assumptions. One is a bound on the utility (Lemma K.2) and the other is a smoothness or Lipschitz condition (Lemma K.3). The convergence rate itself states the number of iterations required for the norm of the Riemannian gradient to fall below a given threshold. The theory also guarantees descent in that the solution returned by the algorithm will have lower loss (higher utility) than the vector passed to the algorithm.

The probability of sampling a vector $\hat{v}^0_i$ at angular deviation within $\phi$ of the maximizer is given by

$$P[|\theta^0_i - \theta^*_i| \leq \phi] = I_{\sin^2(\phi)}(\frac{d-1}{2}, \frac{1}{2}) = \frac{\text{Beta}(\sin^2(\phi), \frac{d-1}{2}, \frac{1}{2})}{\text{Beta}(1, \frac{d-1}{2}, \frac{1}{2})}$$

where $\text{Beta}$ is the incomplete beta function, and $f$ is the normalized incomplete beta function [39]. This probability quickly approaches zero for $\phi < \frac{\pi}{2}$ as the dimension $d$ increases. Therefore, for large $d$, it becomes highly probable that $\hat{v}_i$ will be initialized near an angle $\frac{\pi}{2}$ from the true eigenvector—in
other words, all points are far from each other in high dimensions. In this case, \( \hat{v}_i \) lies near a trough of the sinusoidal utility where gradients are small. Without a bound on the minimum possible gradient norm, a finite sample rate cannot be constructed (how many iterations are required to escape the trough?). Therefore, we can only guarantee asymptotic convergence in this setting. Next, we consider the fortuitous case where all \( \hat{v}_i \) have been initialized within \( \frac{\pi}{2} \). This is both to obtain a convergence rate for this setting, but also to highlight the Big-O dependencies. Note that the utility is symmetric across \( \frac{\pi}{2} \) and the number of iterations required to escape a trough and reach the \( \frac{\pi}{2} \) mark is equal to the number of iterations required to ascend from \( \frac{\pi}{2} \) to the same distance from the peak.

In order to ensure this theory can provide meaningful bounds for EigenGame, we first show, assuming a child is within \( \frac{\pi}{2} \) of its maximizer, that the norm of the Riemannian gradient bounds the angular deviation of a child from this maximizer.

To begin the proof, we relate the error in the parents to a bound on the ambient gradient in Lemma K.8. This bound is then tightened assuming parents with error below a certain threshold in Lemma K.9. Using the fact that \( u_i = \hat{v}_i^T \nabla_{\hat{v}_i} u_i \), this bound directly translates to a bound on the utility in Corollary K.9.1 thereby satisfying Assumption K.2. Again, given accurate parents, Lemma K.10 proves Assumption K.3 on smoothness is satisfied and derives some of the constants for the ultimate convergence rate.

Recall that we have so far been proving convergence to a local maximizer of a child’s utility, which, assuming inaccurate parents, is not the same as the true eigenvector. Lemma K.11 upper bounds the angular deviation of an approximate maximizer from the true eigenvector using the angular deviation of a maximizer plus the approximate maximizer’s approximation error. Lemma K.12 then provides the convergence rate for the child to approach the true eigenvector given accurate enough parents. Finally, Theorem K.4 compiles the chain of convergence rates leading up the DAG towards \( \hat{v}_1 \) and derives a convergence rate for child \( k \) given all previous parents have been learned to a high enough degree of accuracy. The number of iterations required for each parent in the chain is provided.

**Theorem K.4.** Assume all spectral gaps are positive, i.e. for \( i = 1...k, g_i > 0 \). Let \( \theta_k \) denote the angular distance (in radians) of \( \hat{v}_k \) from the true eigenvector \( v_k \). Let the maximum desired error for \( \theta_k = \theta_{tol} \leq 1 \) radian. Then set \( c_{1} = \frac{\theta_{tol}}{16}, \rho_{1} = \frac{g_{1}}{2\pi} \theta_{tol} \), and

\[
\rho_{i} = \left[ \frac{g_{i}g_{i+1}}{2\pi\Lambda_{11}} \right] c_{i+1}
\]

for \( i < k \) where the \( c_i \)’s are dictated by each \( \hat{v}_i \) to its parents and represent fractions of a canonical error threshold; for example, if \( \hat{v}_k \) sets \( c_k = \frac{1}{16} \), then this threshold gets communicated up the DAG to each parent, each time strengthening.

Consider learning \( \hat{v}_i \) by applying Algorithm 2 successively, i.e., learn \( \hat{v}_1 \), stop ascent, learn \( \hat{v}_2 \), and so on, each with step size \( \frac{1}{4\pi} \) and corresponding \( \rho_i \) where \( L = 4 \left( \Lambda_{11}k + (1 + \kappa_{k-1}) \frac{1}{25} \right) \). Then the top-\( k \) principal components will be returned, each within tolerance \( \theta_{tol} \), in the limit.

**Proof.** In order to learn \( \hat{v}_k \), we need \( |\theta_j| \leq \frac{c_{j}g_{j}A_{11}}{(k-j)A_{11}} \) with \( c_{k} \leq \frac{1}{16} \) for all \( j < k \). If this requirement is met, then by Lemma K.11, the angular error in \( \hat{v}_k \) after running Riemannian gradient ascent is bounded as

\[
|\theta_k| \leq \tilde{\epsilon} + 8c_k
\]

where \( \tilde{\epsilon} \) denotes the convergence error and the error propagated by the parents is \( 8c_k \). The quantity, \( \frac{g_{i}}{(k-j)A_{11}} \), in the parents bound is \( \ll 8 \), so the parents must be very accurate to reduce the error propagated to the child. Each parent must then convey this information up the chain, strengthening the requirement each hop.

Let half the error in \( |\theta_k| \) come from mis-specifying the utility with imperfect parents, \( \hat{v}_{j<k} \), and the other half from convergence error. The error after learning \( \hat{v}_{k-1} \) via Riemannian gradient ascent must be less than the threshold required for learning the \( k \)th eigenvector. Assuming \( \hat{v}_{k-1} \)'s parents have been learned accurately enough, \( |\theta_{j<k}| \leq \frac{\tilde{\epsilon}_{k-1}g_{k-1}}{(k-2)A_{11}} \), and that \( \hat{v}_{j<k} \) were initialized within \( \frac{\pi}{2} \) of
their maximizers, we require:

\[ |\theta_{k-1}| \leq \frac{\pi}{g_{k-1}} \rho_{k-1} + 8c_{k-1} \leq \frac{c_k g_k}{(k-1)\Lambda_{11}}. \]  

(128)

More generally, the error after learning \( \hat{v}_{i-1} \) must be less than the threshold for learning any of its successors:

\[ |\theta_{i-1}| \leq \frac{\pi}{g_{i-1}} \rho_{i-1} + 8c_{i-1} \leq \min_{i-1 < l \leq k} \left( \frac{c_l g_l}{(l-1)\Lambda_{11}} \right). \]  

(129)

Assume for now that the arg min of the expression is \( i \), the immediate child. First we bound the error from \( \hat{v}_{i-1} \)'s parents:

\[ 8c_{i-1} \leq \frac{c_i g_i}{2(i-1)\Lambda_{11}} \]  

(130)

\[ \Rightarrow c_{i-1} \leq \frac{c_i g_i}{16(i-1)\Lambda_{11}}. \]  

(131)

Note the \( 2 \) in the denominator of Equation (130) which appears because we desired half the error to come from the parents (half is an arbitrary choice in the analysis). Continuing this process recursively implies

\[ c_{i-2} \leq \frac{c_{i-1}g_{i-1}}{16(i-2)\Lambda_{11}} \leq \frac{c_{i-1}g_{i-1}}{16^2(i-2)(i-1)\Lambda_{11}^2} \]  

(132)

\[ \Rightarrow c_{i-2} \leq \left[ \frac{(i-n-1)!}{(16\Lambda_{11})^n(i-1)!} \right] c_i. \]  

(133)

One can see that \( c_{j<i} \) is strictly smaller than \( c_i \) because each additional term added to the product is strictly less than \( 1 \)—the assumption of the arg min above is therefore correct. In particular, this requires the first eigenvector to be learned to very high accuracy to enable learning the \( k \)-th:

\[ c_1 \leq \left[ \frac{\prod_{j=2}^{k} g_j}{(16\Lambda_{11})^{k-1}(k-1)!} \right] c_k. \]  

(134)

More generally

\[ c_i \leq \frac{(i-1)!\prod_{j=i+1}^{k} g_j}{(16\Lambda_{11})^{k-i}(k-1)!} c_k \]  

(135)

This completes the requirement for mitigating error in the parents.

The convergence error from gradient ascent must also be bounded as (again, note the \( 2 \))

\[ \frac{\pi}{g_i} \rho_i \leq \frac{c_{i+1}g_{i+1}}{2i\Lambda_{11}} \]  

(136)

\[ \Rightarrow \rho_i \leq \left[ \frac{g_i g_{i+1}}{2\pi i\Lambda_{11}} \right] c_{i+1} \]  

(137)

which requires at most

\[ t_i = \left[ \frac{5}{\sqrt{g_i g_{i+1}}} \right]^2 \left[ \frac{1}{c_{i+1}} \right] \]  

(138)

iterations. Given \( \hat{v}_i \) is initialized within \( \pi g_i \) of its maximizer, it follows that learning each \( \hat{v}_{j<k} \) consecutively via Riemannian gradient ascent for at most \( \sum_{i=1}^{k-1} t_i \) iterations is sufficient for learning the \( k \)-th eigenvector. Riemannian gradient ascent on \( \hat{v}_k \) then returns (Lemma K.12)

\[ |\theta_k| \leq \frac{\pi}{g_k} \rho_k + 8c_k \leq \frac{\pi}{g_k} \rho_k + \frac{\theta_{\text{tol}}}{2} \]  

(139)

after at most

\[ t_k = \left[ \frac{5}{4} \frac{1}{\rho_k^2} \right] = \left[ \frac{5\pi^2}{(\theta_{\text{tol}} g_k)^2} \right] \]  

(140)
We can relax the assumption that \( \hat{v}_i \) is initialized within \( \frac{\pi}{4} \) of its maximizer and obtain global convergence. Assume that \( \frac{\pi}{2} - |\theta^0_i| \leq \frac{\pi}{4} \) and let \( ||\nabla_i \theta^0|| \) be the initial norm of the Riemannian gradient. The utility function \( u_i(\hat{v}_i, \hat{v}_{j<i}) \) is symmetric across \( \frac{\pi}{4} \). Therefore, the number of iterations required to ascend to within \( \frac{\pi}{4} \) is given by Lemma K.12:

\[
t^+_i = \left[ \frac{5}{4} \left( \frac{\pi}{4g_i} \right)^2 \frac{1}{2} - |\theta^0_i|^2 \right].
\]

(141)

Alternatively, simply set the desired gradient norm to be less than the initial. This necessarily requires iterates to ascend to past \( \frac{\pi}{4} \). As long as \( \hat{v}_i \) is not initialized to exactly \( \frac{\pi}{4} \) from the maximum (an event with Lebesgue measure 0), the ascent process will converge to the maximizer. \( \square \)

**Theorem K.5.** Apply the algorithm outlined in Theorem K.4 with the same assumptions. Then with probability

\[
P[|\theta^0_i - \theta^*_i| \leq \frac{\pi}{4}] = I_{\frac{1}{2}} \left( \frac{d-1}{2}, \frac{1}{2} \right)
\]

(142)

where \( I \) is the normalized incomplete beta function, the max total number of iterations required for learning all vectors to adequate accuracy is

\[
T_k = \left[ \mathcal{O} \left( k \left( \frac{16\Lambda_{11}}{g_i} \right)^{(k-1)!} \frac{1}{\theta_{\text{tol}}} \right)^{\frac{1}{g_i}} \right].
\]

(143)

**Discussion.** In other words, assuming all \( \hat{v}_i \) are fortuitously initialized within \( \frac{\pi}{4} \) of their maximizers, then we can state a finite sample convergence rate. The first \( k \) in the Big-O formula for total iterations appears simply from a naive summing of worst case bounds on the number of iterations required to learn each \( \hat{v}_{j<k} \) individually. The constant 16 is a loose bound that arises from the error propagation analysis. Essentially, parent vectors, \( \hat{v}_{j<i} \), must be learned to under \( \frac{1}{10} \) a canonical error threshold for the child \( \hat{v}_i, \frac{g_i}{10\Lambda_{11}} \). The Riemannian optimization theory we leverage dictates that \( \frac{1}{\rho_i} \) iterations are required to meet a \( \mathcal{O}(\rho_i) \) error threshold. This is why the squared inverse of the error threshold appears here. Breaking down the error threshold itself, the ratio \( \frac{\Lambda_{11}}{g_i} \) says that more iterations are required to distinguish eigenvectors when the difference between them (summarized by the gap \( g_i \)) is small relative to the scale of the spectrum, \( \Lambda_{11} \). The \( (k-1)! \) term appears because learning smaller eigenvectors requires learning a much more accurate \( \hat{v}_1 \) higher up the chain.

**Proof.** Assume \( \hat{v}_i \) is sampled uniformly in \( S^{d-1} \). Note this can be accomplished by normalizing a sample from a multivariate Gaussian. We will prove

(i) the probability of the event that \( \hat{v}_i^0 \) is within \( \frac{\pi}{4} \) of the maximizer of \( u_i(\hat{v}_i, \hat{v}_{j<i}) \),

(ii) an upper bound on the number of iterations required to return all \( \hat{v}_i \) with angular error less than \( \theta_{\text{tol}} \).

The probability of sampling a vector \( \hat{v}_i^0 \) at angular deviation within \( \frac{\pi}{4} \) of the maximizer is given by twice the probability of sampling from one of the spherical caps around \( v_i \) or \( -v_i \). This probability is

\[
P[|\theta^0_i - \theta^*_i| \leq \phi] = I_{\sin^2(\phi)} \left( \frac{d-1}{2}, \frac{1}{2} \right) = \frac{\text{Beta}(\sin^2(\phi), \frac{d-1}{2}, \frac{1}{2})}{\text{Beta}(1, \frac{d-1}{2}, \frac{1}{2})}
\]

(144)

where \( \text{Beta} \) is the incomplete beta function, and \( I \) is the normalized incomplete beta function. This probability quickly approaches zero for \( \phi < \frac{\pi}{4} \) as the dimension \( d \) increases. This proves (i).

Plugging the bound on \( c_i \)

\[
c_i \leq \frac{(i-1)! \prod_{j=i+1}^k g_j}{(16\Lambda_{11})^{k-1}(k-1)!} c_k
\]

(145)
We lower bound the norm of the Riemannian gradient as follows:

\[
t_i = \left[ 5 \left( \frac{\pi c_{i+1}}{g_i g_{i+1}} \right)^2 \frac{1}{c_i^2} \right]
\]

we find

\[
t_i = \left[ 5 \left( \frac{\pi c_{i+1}}{g_i g_{i+1}} \right)^2 \frac{16 \Lambda_{i1}^{(k-i-1)} ((k-1)!)^2}{(i!)^2 \prod_{j=i+2}^{k} g_j^2} \frac{1}{c_i^2} \right]
\]

\[
= \left[ 5 \pi^2 \frac{16 \Lambda_{i1}^{(k-i-1)} ((k-1)!)^2}{(i!)^2 \prod_{j=i+1}^{k} g_j^2} \frac{1}{16 c_k} \right]
\]

\[
\leq \left[ 5 \pi^2 \left( \frac{16 \Lambda_{i1}^{(k-1)} (k-1)!}{\prod_{j=1}^{k} g_j} \right)^2 \left( \frac{1}{16 c_k} \right) \right]
\]

\[
= \left[ \mathcal{O} \left( \left( \frac{16 \Lambda_{i1}^{(k-1)} (k-1)!}{\prod_{j=1}^{k} g_j} \right)^2 \right) \right]
\]

which is now in a form independent of \(i\) (worst case). It can be shown that \(t_k \leq t_1\) by taking their log and applying Jensen’s inequality. The total iterations required for learning \(\hat{v}_{j < k}\) is at most \(k - 1\) times this. Therefore,

\[
T_k = \left[ \mathcal{O} \left( k \left[ \frac{(16 \Lambda_{i1}^{(k-1)})(k-1)!}{\prod_{j=1}^{k} g_j} \right]^2 \right) \right].
\]

**Lemma K.6.** Assume \(\hat{v}_i\) is within \(\frac{\pi}{4}\) of its maximizer, i.e., \(|\theta_i - \theta^*_i| \leq \frac{\pi}{4}\). Also, assume that \(|\theta_{j < i}| \leq \frac{\pi}{4} \sqrt{\frac{(k-1)!}{\Lambda_{i1}}} \leq \sqrt{\frac{\pi}{4}} \) with \(0 \leq c_i \leq \frac{\pi}{4}\). Then the norm of the Riemannian gradient of \(u_i\) upper bounds this angular deviation:

\[
|\theta_i - \theta^*_i| \leq \frac{\pi}{g_i} ||\nabla^R_{\hat{v}_i} u_i (\hat{v}_i, \hat{v}_{j < i})||.
\]

**Proof.** The Riemannian gradient measures how the utility \(u_i\) changes while moving along the manifold. In contrast, the ambient gradient measures how \(u_i\) changes while moving in ambient space, possibly off the manifold. Rather than bounding the angular deviation using the projection of the ambient gradient onto the tangent space of the manifold, \((I - \hat{v}_i \hat{v}_i^\top) \nabla_{\hat{v}_i} u_i\), we instead reparameterize \(\hat{v}_i\) to ensure it lies on the manifold, \(\hat{v}_i = \cos(\theta_i) v_i + \sin(\theta_i) \Delta_i\), where \(\Delta_i\) is a unit vector and \((v_i, \Delta_i) = 0\). Computing gradients with respect to the new unconstrained arguments allows recovering a bound on the Riemannian gradient via a simple chain rule calculation.

We lower bound the norm of the Riemannian gradient as follows:

\[
\frac{\partial u_i}{\partial \theta_i} = \nabla^R_{\hat{v}_i} u_i (\hat{v}_i, \hat{v}_{j < i})^\top \frac{\partial v}{\partial \theta_i}
\]

\[
\Rightarrow ||\frac{\partial u_i}{\partial \theta_i}|| \leq ||\nabla^R_{\hat{v}_i} u_i (\hat{v}_i, \hat{v}_{j < i})|| ||\frac{\partial \hat{v}_i}{\partial \theta_i}||
\]

\[
\Rightarrow ||\nabla^R_{\hat{v}_i} u_i (\hat{v}_i, \hat{v}_{j < i})|| \geq ||\frac{\partial u_i}{\partial \theta_i}|| ||\frac{\partial \hat{v}_i}{\partial \theta_i}||.
\]

Note that \(||\frac{\partial \hat{v}_i}{\partial \theta_i}|| = 1\) by design. And the numerator can be bounded using Lemma J.4 as

\[
||\frac{\partial u_i}{\partial \theta_i}|| = \sqrt{A^2 + B^2} ||\sin(2(\theta_i - \theta^*_i))||
\]

where \(\theta^*_i = -\phi + \phi = \tan^{-1} \left( \frac{B}{A} \right)\). Furthermore, assume \(|\theta_i - \theta^*_i| \leq \frac{\pi}{4}\). Then

\[
||\sin(2(\theta_i - \theta^*_i))|| \geq \left| \frac{2}{\pi} (\theta_i - \theta^*_i) \right|.
\]
Combining the results gives

\[
\| \nabla_{\hat{v}, i} u_i(\hat{v}_i, \hat{v}_j) \| \geq \frac{\| \partial u_i / \partial \theta_i \|}{\| \partial u_i / \partial \theta_i \|} \geq \frac{2}{\pi} \sqrt{A^2 + B^2} |\theta_i - \theta_i^*| \geq \frac{2}{\pi} A |\theta_i - \theta_i^*| \geq \frac{2}{\pi} (1 - 8c) g_i |\theta_i - \theta_i^*| \geq \frac{g_i}{\pi} |\theta_i - \theta_i^*| \tag{163}
\]

completing the proof.

\[\square\]

**Lemma K.7.** Let \(|\theta_j| \leq \epsilon < 1\) for all \(j < i\). Then the ratio of generalized inner products is bounded as

\[
\frac{\langle \hat{v}_i, \Lambda \hat{v}_j \rangle}{\langle \hat{v}_j, \Lambda \hat{v}_j \rangle} \leq \frac{1 + (1 + \kappa_j) \epsilon}{\sqrt{1 - \epsilon^2}}. \tag{164}
\]

**Proof.** We write \(\hat{v}_j \leq i = \cos(\theta_j) v_j + \sin(\theta_j) \Delta_j\) where \(\langle \Delta_j, v_j \rangle = 0\) without loss of generality. Note that \(|\theta_j| \leq \epsilon\) implies \(|\sin(\theta_j)| \leq \epsilon\). Then

\[
\frac{\langle \hat{v}_i, \Lambda \hat{v}_j \rangle}{\langle \hat{v}_j, \Lambda \hat{v}_j \rangle} = \frac{\langle \cos(\theta_i) v_i + \sin(\theta_j) \Delta_i, \Lambda \left( \cos(\theta_j) v_j + \sin(\theta_j) \Delta_j \right) \rangle}{\langle \cos(\theta_j) v_j + \sin(\theta_j) \Delta_j, \Lambda \left( \cos(\theta_j) v_j + \sin(\theta_j) \Delta_j \right) \rangle} \tag{166}
\]

\[
= \frac{\Lambda_{ij} \cos(\theta_j)^2 + \langle \Delta_j, \Lambda \Delta_j \rangle \sin^2(\theta_j)}{\Lambda_{jj} \cos(\theta_j)^2 + \langle \Delta_j, \Lambda \Delta_j \rangle \sin^2(\theta_j)} \tag{167}
\]

\[
\leq \frac{\Lambda_{ij} \sin(\theta_i)}{\Lambda_{jj} (1 - \epsilon^2)} \frac{|\Delta_i| \sqrt{1 - \epsilon^2} + \Lambda_{ii} \epsilon |\cos(\theta_i)| + |\sin(\theta_i)| \epsilon \Lambda_{11}}{\Lambda_{ij} (1 - \epsilon^2)} \tag{169}
\]

\[
\leq \frac{\Lambda_{ij} \sqrt{1 - \epsilon^2} + \Lambda_{ii} \epsilon + \epsilon \Lambda_{11}}{\Lambda_{jj} (1 - \epsilon^2)} \tag{170}
\]

\[
= \frac{1}{\sqrt{1 - \epsilon^2}} + \left( \frac{\Lambda_{ii}}{\Lambda_{jj}} + \kappa_j \right) \frac{\epsilon}{\sqrt{1 - \epsilon^2}} \tag{171}
\]

\[
\leq 1 + (1 + \kappa_j) \epsilon \tag{172}
\]

\[\square\]

**Lemma K.8 (Lipschitz Bound).** Let \(|\theta_j| \leq \epsilon < 1\) for all \(j < i\). Then the norm of the ambient gradient of \(u_i\) is bounded as

\[
\| \nabla_{\hat{v}, i} u_i(\hat{v}_i, \hat{v}_j) \| \leq 2 \Lambda_{11} \left[ 1 + (i - 1) \frac{1 + (1 + \kappa_{i-1}) \epsilon}{\sqrt{1 - \epsilon^2}} \right]. \tag{173}
\]
Proof. Starting with the gradient (Equation 7), we find
\[
\|\nabla \hat{v}_i u_i(\hat{v}_i, \hat{v}_{j<i})\| = \|2M \left[ \hat{v}_i - \sum_{j<i} \frac{\hat{v}_j^\top M \hat{v}_j}{\hat{v}_j^\top M \hat{v}_j} \right]\| \leq 2\|M \hat{v}_i\| + 2 \sum_{j<i} \|\frac{\hat{v}_j^\top M \hat{v}_j}{\hat{v}_j^\top M \hat{v}_j} M \hat{v}_j\| \leq 2\|M \hat{v}_i\| + 2 \sum_{j<i} \|\frac{\hat{v}_j^\top M \hat{v}_j}{\hat{v}_j^\top M \hat{v}_j}\|||M \hat{v}_j|| \leq 2\|M \hat{v}_i\| + 2 \sum_{j<i} \frac{1 + (1 + \kappa_j)\epsilon}{\sqrt{1 - \epsilon}} \Lambda_{11} \leq 2\Lambda_{11} + 2 \sum_{j<i} \frac{1 + (1 + \kappa_{i-1})\epsilon}{\sqrt{1 - \epsilon}} \Lambda_{11} = 2\Lambda_{11} \left[1 + (i - 1) \frac{1 + (1 + \kappa_{i-1})\epsilon}{\sqrt{1 - \epsilon}}\right].
\]

Lemma K.9 (Lipschitz Bound with Accurate Parents). Assume \(|\theta_j| \leq \epsilon \leq \frac{c_i g_i}{(i-1)\Lambda_{11}} \leq \sqrt{\frac{1}{2}} for all j < i with 0 \leq c_i \leq \frac{1}{16}\). Then the norm of the ambient gradient of \(u_i\) is bounded as
\[
\|\nabla \hat{v}_i u_i(\hat{v}_i, \hat{v}_{j<i})\| \leq 4 \left[\Lambda_{11} i + (1 + \kappa_{i-1})c_i g_i\right] = L_i. \tag{179}
\]

Proof. Starting with Lemma K.8, we find
\[
\|\nabla \hat{v}_i u_i(\hat{v}_i, \hat{v}_{j<i})\| \leq 2\Lambda_{11} \left[1 + (i - 1) \frac{1 + (1 + \kappa_{i-1})\epsilon}{\sqrt{1 - \epsilon}}\right] \leq 2\Lambda_{11} \left[1 + 2(i - 1) + 2 \frac{(1 + \kappa_{i-1})c_i g_i}{\Lambda_{11}}\right] \leq 4 \left[\Lambda_{11} (1 + (i - 1)\epsilon) + (1 + \kappa_{i-1})c_i g_i\right] = 4 \left[\Lambda_{11} i + (1 + \kappa_{i-1})c_i g_i\right]. \tag{184}
\]

Corollary K.9.1 (Bound on Utility). Assume \(|\theta_j| \leq \epsilon \leq \frac{c_i g_i}{(i-1)\Lambda_{11}} \leq \sqrt{\frac{1}{2}} for all j < i with 0 \leq c_i \leq \frac{1}{16}\). Then the absolute value of the utility is bounded as follows
\[
|u_i(\hat{v}_i, \hat{v}_{j<i})| = |\hat{v}_i^\top \nabla \hat{v}_i| = \|\nabla \hat{v}_i\| \|\nabla \hat{v}_i\| = \|\nabla \hat{v}_i\| \leq L_i, \tag{185}
\]
thereby satisfying Assumption K.2.

Lemma K.10. Assume \(|\theta_j| \leq \frac{c_i g_i}{(i-1)\Lambda_{11}} \leq \sqrt{\frac{1}{2}} for all j < i with 0 \leq c_i \leq \frac{1}{16}\). Then Assumption K.3 is satisfied with \(\xi = \xi' = \frac{8}{5} L_i\).
Proof. Let $\eta = \alpha \nabla_{\hat{v}_i} u_i = \alpha (I - \hat{v}_i \hat{v}_i^\top) \nabla_{\hat{v}_i} u_i$, $\alpha > 0$, and $\hat{\eta} = \frac{\eta}{||\eta||}$. Let $\hat{v}_i' = \frac{\hat{v}_i + \hat{\eta}}{\gamma}$ where $\gamma = ||\hat{v}_i + \hat{\eta}||$. Then

$$u_i(\hat{v}_i') = \frac{1}{\gamma^2} \left[ (\hat{v}_i + \hat{\eta})^\top \Lambda (\hat{v}_i + \hat{\eta}) - \sum_{\hat{v}_j \neq \hat{v}_i} \left( \frac{(\hat{v}_i + \hat{\eta})^\top \Lambda \hat{v}_j}{\hat{v}_j^\top \Lambda \hat{v}_j} \right)^2 \right]$$

(186)

$$= \frac{1}{\gamma^2} \left[ \hat{v}_i^\top \Lambda \hat{v}_i - \sum_{\hat{v}_j \neq \hat{v}_i} \left( \frac{\hat{v}_i^\top \Lambda \hat{v}_j}{\hat{v}_j^\top \Lambda \hat{v}_j} \right)^2 + \eta^\top \Lambda \eta - \sum_{\hat{v}_j \neq \hat{v}_i} \left( \frac{\eta^\top \Lambda \hat{v}_j}{\hat{v}_j^\top \Lambda \hat{v}_j} \right)^2 + 2\eta^\top \Lambda \hat{v}_i - 2 \sum_{\hat{v}_j \neq \hat{v}_i} \left( \frac{\hat{v}_i^\top \Lambda \hat{v}_j}{\hat{v}_j^\top \Lambda \hat{v}_j} \right) \left( \frac{\eta^\top \Lambda \hat{v}_j}{\hat{v}_j^\top \Lambda \hat{v}_j} \right) \right]$$

(187)

$$= \frac{1}{\gamma^2} \left[ u_i(\hat{v}_i) + u_i(\hat{\eta}) + 2\eta^\top \nabla_{\hat{v}_i} u_i(\hat{v}_i) \right]$$

(188)

$$= \frac{1}{\gamma^2} \left[ u_i(\hat{v}_i) + ||\eta||^2 u_i(\hat{\eta}) + 2\eta^\top \nabla_{\hat{v}_i} u_i(\hat{v}_i) \right]$$

(189)

The vectors $\hat{v}_i$ and $\nabla_{\hat{v}_i} u_i(\hat{v}_i)$ define a 2-d plane in which $\hat{v}_i'$ lies independent of the step size $\alpha$. Therefore, we can consider gradients confined to a 2-d plane without loss of generality. Specifically, let $\hat{v}_i = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and $\nabla = \nabla_{\hat{v}_i} u_i(\hat{v}_i) = \beta \begin{bmatrix} \cos(\psi) \\ \sin(\psi) \end{bmatrix}$. Then $\nabla^R = \nabla_{\hat{v}_i}^R u_i(\hat{v}_i) = \beta \begin{bmatrix} \cos(\psi) \\ 0 \end{bmatrix}$ and $\gamma = \sqrt{1 + ||\eta||^2} = \sqrt{1 + \alpha^2 \beta^2 \cos^2(\psi)}$. Also, let $z = \beta \cos(\psi)$ and $\alpha < \frac{1}{\hat{L}_i}$ (see Equation (179) for definition) which implies $\alpha^2 ||\nabla^R||^2 < 1$. Then

$$u_i(\hat{v}_i') - u_i(\hat{v}_i)$$

(190)

$$\leq \left( \frac{1}{\gamma^2} - 1 \right) u_i(\hat{v}_i) + \frac{1}{\gamma^2} (||\eta||^2 u_i(\hat{\eta}) + 2\eta^\top \nabla_{\hat{v}_i} u_i(\hat{v}_i))$$

(191)

$$\geq \left( \frac{1}{\gamma^2} - 1 \right) L_i + \frac{1}{\gamma^2} (||\eta||^2 u_i(\hat{\eta}) + 2\alpha \nabla^\top \nabla^R)$$

(192)

$$\geq \left( \frac{1}{\gamma^2} - 1 \right) L_i + \frac{1}{\gamma^2} (2\alpha \nabla^\top \nabla^R + \alpha^2 ||\nabla^R||^2 (-L_i))$$

(193)

$$= \left( \frac{1}{1 + \alpha^2 \beta^2 \cos^2(\psi)} - 1 \right) L_i + \frac{\alpha}{1 + \alpha^2 \beta^2 \cos^2(\psi)} (2 - \alpha L_i) \beta^2 \cos^2(\psi)$$

(194)

$$= \left( \frac{1}{1 + \alpha^2 \beta^2 \cos^2(\psi)} - 1 \right) L_i + \frac{\alpha (2 - \alpha L_i)}{1 + \alpha^2 \beta^2 \cos^2(\psi)} z^2$$

(195)

$$= \frac{1}{1 + \alpha^2 \beta^2 \cos^2(\psi)} (L_i - L_i \alpha^2 \beta^2 - L_i (2 - \alpha L_i) z^2)$$

(196)

$$= \frac{1}{1 + \alpha^2 \beta^2 \cos^2(\psi)} (-2L_i \alpha^2 \beta^2 + 2 \alpha z^2)$$

(197)

$$= \frac{2\alpha z^2}{1 + \alpha^2 \beta^2 \cos^2(\psi)} (1 - \alpha L_i) > 0$$

(198)

where the assumption that $|\theta_j| \leq \frac{\xi_j \alpha}{(v - 1) \Lambda_1}$ was used to leverage Corollary K.9.1. Let $\alpha = \frac{1}{2\hat{L}_i}$. Then $||\eta||^2 = \alpha^2 z^2 \leq \frac{1}{\xi}$ and

$$u_i(\hat{v}_i') - u_i(\hat{v}_i) \geq \frac{2\alpha z^2}{1 + \alpha^2 \beta^2 \cos^2(\psi)} (1 - \alpha L_i)$$

(199)

$$= \frac{2\alpha z^2}{1 + \alpha^2 \beta^2 \cos^2(\psi)} \frac{1 - \alpha L_i}{\alpha}$$

(200)

$$= \frac{2\alpha z^2}{1 + \alpha^2 \beta^2 \cos^2(\psi)} \frac{2L_i \alpha^2 \beta^2}{1 + \alpha^2 \beta^2 \cos^2(\psi)}$$

(201)

$$= \frac{2L_i ||\eta||^2}{1 + \alpha^2 \beta^2 \cos^2(\psi)}$$

(202)

$$\geq \min(\xi ||\eta||^2, \xi' ||\eta||)$$

(203)

with $\xi = \xi' = \frac{8}{5} L_i$. \qed
Lemma K.11 (Approximate Optimization is Reasonable Given Accurate Parents). Assume \( \| \theta_j \| \leq \frac{c_i g_i}{(i-1)\Lambda_{11}} \leq \sqrt{\frac{1}{2}} \) for all \( j < i \) with \( 0 \leq c \leq \frac{1}{16} \), i.e., the parents have been learned accurately. Then for any approximate local maximizer \( (\tilde{\theta}_i, \Delta_i) \) of \( u_i(\hat{v}_i, \Delta_i, \hat{v}_{j<i}) \), if the angular deviation \( |\tilde{\theta}_i - \theta^*_i| \leq \bar{\epsilon} \) where \( \theta^*_i \) forms the global max,

\[
|\tilde{\theta}_i| \leq \bar{\epsilon} + 8c_i
\]

where \( \tilde{\theta}_i \) denotes the angular distance of the approximate local maximizer to the true eigenvector \( v_i \).

Proof. Note that the true eigenvector occurs at \( \tilde{\theta}_i = 0 \). The result follows directly from Theorem J.2:

\[
|\tilde{\theta}_i| = |\tilde{\theta}_i - 0| \leq |\tilde{\theta}_i - \theta^*_i| + |\theta^*_i - 0| \leq \bar{\epsilon} + 8c_i.
\]

Lemma K.12. Assume \( \hat{v}_i \) is initialized within \( \frac{\pi}{4} \) of its maximizer and its parents are accurate enough, i.e., that \( |\theta_{j<i}| \leq \frac{c_i g_i}{(i-1)\Lambda_{11}} \leq \sqrt{\frac{1}{2}} \) with \( 0 \leq c_i \leq \frac{1}{16} \). Let \( \rho_i \) be the maximum tolerated error desired for \( \hat{v}_i \). Then Riemannian gradient ascent returns

\[
|\theta_i| \leq \frac{\pi}{g_i} \rho_i + 8c_i
\]

after at most

\[
\left\lceil \frac{5}{4} \cdot \frac{1}{\rho_i^2} \right\rceil
\]

iterations.

Proof. Note that the assumptions of Lemma K.1 are met by Corollary K.9.1 and Lemma K.10 with \( \xi = \xi' = \frac{\pi}{4} \) and Riemannian gradient ascent. Plugging into Lemma K.1 ensures that Riemannian gradient ascent returns unit vector \( \hat{v}_i \) satisfying \( u(\hat{v}_i) \geq u(\hat{v}_0) \) and \( ||\nabla^A|| \leq \rho_i \) in at most

\[
\left\lceil \frac{5}{4} \cdot \frac{1}{\rho_i^2} \right\rceil
\]

iterations (where \( \hat{v}_i \) is initialized to \( \hat{v}_0 \)). Additionally, note that for any \( \hat{v}_i \), \( u_i(\hat{v}_i^*) - u_i(\hat{v}_i) \leq 2L_i \) where \( L_i \) bounds the absolute value of the utility \( u_i \) (see Corollary K.9.1) and \( \hat{v}_i^* = \arg \max u_i(\hat{v}_i) \). Combining this with Lemma K.6 gives

\[
|\theta_i - \theta^*_i| \leq \frac{\pi}{g_i} \rho_i
\]

after at most

\[
\left\lceil \frac{5}{4} \cdot \frac{1}{\rho_i^2} \right\rceil
\]

iterations. Lastly, translating \( |\theta_i - \theta^*_i| \) to \( |\theta_i| \) using Lemma K.11 gives the desired result.