⚀ EIGENGAME UNLOADED ⚁
WHEN PLAYING GAMES IS BETTER THAN OPTIMIZING

Anonymous authors
Paper under double-blind review

ABSTRACT

We build on the recently proposed EigenGame that views eigendecomposition as a competitive game. EigenGame's updates are biased if computed using minibatches of data, which hinders convergence and more sophisticated parallelism in the stochastic setting. In this work, we propose an unbiased stochastic update that is asymptotically equivalent to EigenGame, enjoys greater parallelism allowing computation on datasets of larger sample sizes, and outperforms EigenGame in experiments. We present applications to finding the principal components of massive datasets and performing spectral clustering of graphs. We analyze and discuss our proposed update in the context of EigenGame and the shift in perspective from optimization to games.

1 INTRODUCTION

Large, high-dimensional datasets containing billions of samples are commonplace. Dimensionality reduction to extract the most informative features is an important step in the data processing pipeline which enables faster learning of classifiers and regressors (Dhillon *et al.*, 2013), clustering (Kannan and Vempala, 2009), and interpretable visualizations. Many dimensionality reduction and clustering techniques rely on eigendecomposition at their core including principal component analysis (Jolliffe, 2002), locally linear embedding (Roweis and Saul, 2000), multidimensional scaling (Mead, 1992), Isomap (Tenenbaum *et al.*, 2000), and graph spectral clustering (Von Luxburg, 2007).

Numerical solutions to the eigenvalue problem have been approached from a variety of angles for centuries: Jacobi's method, Rayleigh quotient, power (von Mises) iteration (Golub and Van der Vorst, 2000). For large datasets that do not fit in memory, approaches that access only subsets—or *minibatches*—of the data at a time have been proposed.

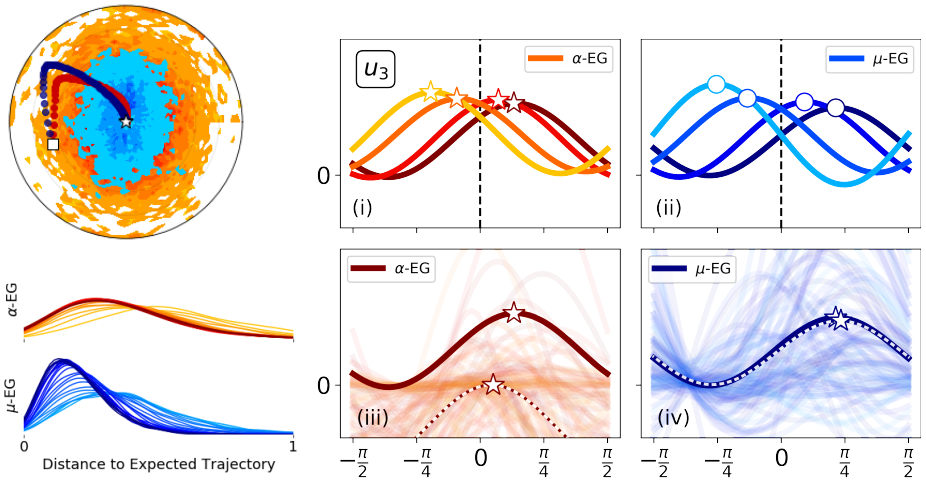
Recently, EigenGame (Gemp *et al.*, 2021) was introduced with the novel perspective of viewing the set of eigenvectors as the Nash strategy of a suitably defined game. While this work demonstrated an algorithm that was empirically competitive given access to only subsets of the data, its performance degraded with smaller minibatch sizes, which are required to fit high dimensional data onto devices.

One path towards circumventing EigenGame's need for large minibatch sizes is parallelization. In a data parallel approach, updates are computed in parallel on partitions of the data and then combined such that the aggregate update is equivalent to a single large-batch update. The technical obstacle preventing such an approach for EigenGame lies in the bias of its updates, i.e., the divide-and-conquer EigenGame update is not equivalent to the large-batch update. Biased updates are not just a theoretical nuisance; they can slow and even prevent convergence to the solution (made obvious in Figure 4).

In this work we introduce a formulation of EigenGame which admits unbiased updates which we term μ-EigenGame. We will refer to the original formulation of EigenGame as α-EigenGame.[3]

μ-EigenGame and α-EigenGame are contrasted in Figure 1. Unbiased updates allow us to increase the effective batch size using data parallelism. Lower variance updates mean that μ-EigenGame should converge faster and to more accurate solutions than α-EigenGame regardless of batch size. In Figure 1a (top), the density of the shaded region shows the distribution of steps taken by the stochastic variant of each algorithm after 100 burn-in steps. Although the expected path of α-EG is

[3] μ signifies unbiased or *un*loaded and α denotes original.



(a) (b)

Figure 1: **(a)** Comparing α-EigenGame (Gemp *et al.*, 2021) and μ-EigenGame (this work) over 1000 trials with a batch size of 1. **(top)** The expected trajectory[1] of each algorithm from initialization (□) to the true value of the third eigenvector ($\star$). **(bottom)** The distribution of distances between stochastic update trajectories and the expected trajectory of each algorithm as a function of iteration count (**bolder** lines are later iterations and modes further left are more desirable).

(b) Empirical support for Lemma 2. In the top row, player 3's utility is given for parents mis-specified by an angular distance along the sphere of $\angle(\hat{v}_{j<i}, v_{j<i}) \in [-20°, -10°, 10°, 20°]$ moving from light to dark. Player 3's mis-specification, $\angle(\hat{v}_i, v_i)$, is given by the x-axis (optimum is at 0 radians). α-EigenGame (i) exhibits slightly lower sensitivity than μ-EigenGame (ii) to mis-specified parents (see equation (8)). However, when the utilities are estimated using samples $X_t \sim p(X)$ (faint lines), μ-EigenGame remains accurate (iv), while α-EigenGame (iii) returns a utility (dotted line) with an optimum that is shifted to the left and down. The downward shift occurs because of the random variable in the denominator of the penalty terms (see equation (3)).[2]

[1]The trajectory when updating with $\mathbb{E}[X_t^\top X_t]$.

[2]Overestimation is expected by Jensen's: $\mathbb{E}[\frac{1}{X}] \geq \frac{1}{\mathbb{E}[X]}$.

slightly more direct, its stochastic variant has much larger variance. Figure 1a (bottom) shows that with increasing iterations, the μ-EG trajectory approaches its expected value whereas α-EG exhibits larger bias. Figure 1b further supports μ-EigenGame's reduced bias with details in Sections 3 and 4.

Our contributions: In the rest of the paper, we present our new formulation of EigenGame, analyze its bias and propose a **novel unbiased parallel variant, μ-EigenGame** with *stochastic* convergence guarantees. μ-EigenGame's utilities are distinct from α-EigenGame and offer an alternative perspective. We demonstrate its performance with extensive experiments including dimensionality reduction of massive data sets and clustering a large social network graph. We conclude with discussions of the algorithm's design and context within optimization, game theory, and neuroscience.

2 Preliminaries and Related Work

In this work, we aim to compute the top-k right singular vectors of data X, which is either represented as a matrix, $X \in \mathbb{R}^{n \times d}$, of n d-dimensional samples, or as a d-dimensional random variable. In either case, we assume we can repeatedly sample a minibatch X_t from the data of size $n' < n$, $X_t \in \mathbb{R}^{n' \times d}$. The top-$k$ right singular vectors of the dataset are then given by the top-k eigenvectors of the (sample) covariance matrix, $C = \mathbb{E}[\frac{1}{n'} X_t^\top X_t] = \mathbb{E}[C_t]$.

For small datasets, SVD is appropriate. However, the time, $\mathcal{O}(\min\{nd^2, n^2 d\})$, and space, $\mathcal{O}(nd)$, complexity of SVD prohibit its use for larger datasets (Shamir, 2015) including when X is a random variable. For larger datasets, stochastic, randomized, or sketching algorithms are better suited. Stochastic algorithms such as Oja's algorithm (Oja, 1982; Allen-Zhu and Li, 2017) perform power iteration (Rutishauser, 1971) to iteratively improve an approximation, maintaining orthogonality

of the eigenvectors typically through repeated QR decompositions. Alternatively, randomized algorithms (Halko *et al.*, 2011; Sarlos, 2006; Cohen *et al.*, 2017) first compute a random projection of the data onto a $(k+p)$-subspace approximately containing the top-k subspace. This is done using techniques similar to Krylov subspace iteration methods (Musco and Musco, 2015). After projecting, a call to SVD is then made on this reduced-dimensionality data matrix. Sketching algorithms (Feldman *et al.*, 2020) such as Frequent Directions (Ghashami *et al.*, 2016) also target learning the top-k subspace by maintaining an overcomplete sketch matrix of size $(k+p) \times d$ and maintaining a span of the top subspace with repeated calls to SVD. In both the randomized and sketching approaches, a final SVD of the $n \times (k+p)$ dataset is required to recover the desired singular vectors. Although the SVD scales linearly in n, some datasets are too large to fit in memory; in this case, an out-of-memory SVD may suffice (Haidar *et al.*, 2017). For this reason, the direct approach of stochastic algorithms, which avoid an SVD call altogether, is appealing when processing very large datasets.

A large literature on *distributed* approaches to PCA exists (Liang *et al.*, 2014; Garber *et al.*, 2017; Fan *et al.*, 2019). These typically follow the pattern of computing solutions locally and then aggregating them in a single round (or minimal rounds) of communication. The modern distributed machine learning setting which has evolved to meet the needs of deep learning is fundamentally different. Many accelerators joined with fast interconnects means the cost of communication is low compared to the cost of a single update step, however existing approaches to distributed PCA cannot take full advantage of this.

Algorithm 1 μ-EigenGame^R

Given: data stream $X_t \in \mathbb{R}^{n' \times d}$, vectors $\hat{v}_i^0 \in \mathcal{S}^{d-1}$, step sequence η_t, and iterations T.
$\hat{v}_i \leftarrow \hat{v}_i^0$ for all i
for $t = 1 : T$ **do**
  parfor $i = 1 : k$ **do**
    $\texttt{rewards} \leftarrow \frac{1}{n'} X_t^\top X_t \hat{v}_i$
    $\texttt{penalties} \leftarrow \frac{1}{n'} \sum_{j<i} \langle X_t \hat{v}_i, X_t \hat{v}_j \rangle \hat{v}_j$
    $\tilde{\nabla}_i^\mu \leftarrow \texttt{rewards} - \texttt{penalties}$
    $\tilde{\nabla}_i^{\mu,R} \leftarrow \tilde{\nabla}_i^\mu - \langle \tilde{\nabla}_i^\mu, \hat{v}_i \rangle \hat{v}_i$
    $\hat{v}_i' \leftarrow \hat{v}_i + \eta_t \tilde{\nabla}_i^{\mu,R}$
    $\hat{v}_i \leftarrow \frac{\hat{v}_i'}{||\hat{v}_i'||}$
  end parfor
end for
return all $\hat{v}_i$

Notation: We follow the same notation as Gemp *et al.* (2021). Variables returned by an approximation algorithm are distinguished from the true solutions with hats, e.g., the column-wise matrix of eigenvectors $\hat{V}$ approximates V. We order the columns of V such that the ith column, v_i, is the eigenvector with the ith largest eigenvalue λ_i. The set of all eigenvectors $\{v_j\}$ with λ_j larger than λ_i, namely v_i's *parents*, will be denoted by $v_{j<i}$. Similarly, sums over subsets of indices may be abbreviated as $\sum_{j<i} = \sum_{j=1}^{i-1}$. The set of all parents *and* children of v_i are denoted by v_{-i}. Let the ith eigengap $g_i = \lambda_i - \lambda_{i+1}$. We assume the standard Euclidean inner product $\langle u, v \rangle = u^\top v$ and denote the unit-sphere and simplex in ambient space $\mathbb{R}^d$ with $\mathcal{S}^{d-1}$ and Δ^{d-1} respectively.

α-EigenGame. We build on the algorithm introduced by Gemp *et al.* (2021), which we refer to here as α-EigenGame. This algorithm is derived by formulating the eigendecomposition of a symmetric positive definite matrix as the Nash equilibrium of a game among k players, each player i owning the approximate eigenvector $\hat{v}_i \in \mathcal{S}^{d-1}$. Each player is also assigned a utility function, $u_i^\alpha(\hat{v}_i | \hat{v}_{j<i})$, that they must maximize:

$$u_i^\alpha(\hat{v}_i|\hat{v}_{j<i}) = \overbrace{\hat{v}_i^\top C \hat{v}_i}^{\texttt{Var}} - \sum_{j<i} \overbrace{\frac{\langle \hat{v}_i, C\hat{v}_j \rangle^2}{\langle \hat{v}_j, C\hat{v}_j \rangle}}^{\texttt{Align}\text{-penalty}} . \tag{1}$$

These utilities balance two terms, one that rewards a $\hat{v}_i$ that captures more variance in the data and a second term that penalizes $\hat{v}_i$ for failing to be orthogonal to each of its parents $\hat{v}_{j<i}$ (these terms are indicated with Var and Align-penalty in equation (1)). In α-EigenGame, each player simultaneously updates $\hat{v}_i$ with gradient ascent, and it is shown that this process converges to the Nash equilibrium. We are interested in extending this approach to the data parallel setting where each player i may distribute its update computation over multiple devices.

3 A SCALABLE UNBIASED ALGORITHM

We present our novel modification to α-EigenGame called μ-EigenGame along with intuition, theory, and empirical support for critical lemmas. We begin with identifying and systematically removing the bias that exists in the α-EigenGame updates. We then explain how removing bias allows us to exploit modern compute architectures culminating in the development of a highly parallelizable algorithm.

3.1 α-EIGENGAME'S BIASED UPDATES

Consider partitioning the sample covariance matrix C_t into a sum of m matrices as $C_t = \frac{1}{n'} X_t^\top X_t = \frac{1}{m}\sum_m \frac{m}{n'} X_{tm}^\top X_{tm} = \frac{1}{m}\sum_m C_{tm}$. For sake of exposition, we drop the additional subscript t on C in what follows. We would like α-EigenGame to parallelize over these partitions. However, the gradient of u_i^α with respect to $\hat{v}_i$ does not decompose cleanly over the data partitions:

$$\nabla_i^\alpha \propto \overbrace{C\hat{v}_i}^{\texttt{Var}} - \sum_{j<i} \overbrace{\frac{\hat{v}_i^\top C \hat{v}_j}{v_j^\top C \hat{v}_j} C\hat{v}_j}^{\texttt{Align-penalty}} = \frac{1}{m}\sum_m \Big[C_m \hat{v}_i - \sum_{j<i} \boxed{\frac{\hat{v}_i^\top C \hat{v}_j}{\hat{v}_j^\top C \hat{v}_j}} C_m \hat{v}_j \Big]. \tag{2}$$

We include the superscript α on the EigenGame gradient to differentiate it from the μ-EigenGame direction later. The nonlinear appearance of C in the penalty terms makes obtaining an unbiased gradient difficult. The quadratic term in the numerator of equation (2) could be made unbiased by using two sample estimates of C, one for each term. But the appearance of the term in the denominator does not have an easy solution. C_m is likely singular for small n' ($n' < d$) which increases the likelihood of a small denominator, i.e., a large penalty coefficient (boxed), if we were to estimate the denominator with samples. The result is an update that emphasizes penalizing orthogonality over capturing data variance. Techniques exist to reduce the bias of samples of ratios of random variables, but to our knowledge, techniques to obtain unbiased estimates are not available. This was conjectured by Gemp *et al.* (2021) as the reason for why α-EigenGame performed worse with small minibatches.

3.2 REMOVING α-EIGENGAME'S BIAS

It is helpful to rearrange equation (2) to shift perspective from estimating a penalty coefficient (in red) to estimating a penalty direction (in blue):

$$\nabla_i^\alpha \propto \frac{1}{m}\sum_m \Big[C_m \hat{v}_i - \sum_{j<i} \hat{v}_i^\top C_m \hat{v}_j \frac{C\hat{v}_j}{\hat{v}_j^\top C \hat{v}_j} \Big]. \tag{3}$$

The penalty direction in equation (3) is still difficult to estimate. However, consider the case where $\hat{v}_j$ is any eigenvector of C with associated (unknown) eigenvalue λ'. In this case, $C\hat{v}_j = \lambda' \hat{v}_j$ and the penalty direction (in blue) simplifies to $\hat{v}_j$ because $||\hat{v}_j|| = 1$. While this assumption is certainly not met at initialization, α-EigenGame leads each $\hat{v}_j$ towards v_j, so we can expect this assumption to be met asymptotically.

This intuition motivates the following μ-EigenGame update direction for $\hat{v}_i$ with inexact parents $\hat{v}_j$ (compare orange in equation (4) to blue in equation (3)):

$$\Delta_i^\mu = C\hat{v}_i - \sum_{j<i} (\hat{v}_i^\top C \hat{v}_j)\hat{v}_j = \frac{1}{m}\sum_m \Big[C_m \hat{v}_i - \sum_{j<i} (\hat{v}_i^\top C_m \hat{v}_j)\hat{v}_j \Big]. \tag{4}$$

We use Δ instead of ∇ because the direction is not a gradient (discussed later). Notice how the strictly linear appearance of C in μ-EigenGame allows the update to easily decompose over the data partitions in equation (4). The μ-EigenGame update satisfies two important properties.

Lemma 1 (Asymptotic equivalence)**.** *The μ-EigenGame direction, Δ_i^μ, with exact parents ($\hat{v}_j = v_j \ \forall\ j < i$) is equivalent to α-EigenGame.*

Proof. We start with α-EigenGame and add a superscript e to its gradient to emphasize this is the gradient computed with exact parents ($\hat{v}_j = v_j$). Then simplifying, we find

$$\nabla_i^{\alpha,e} \propto C\hat{v}_i - \sum_{j<i} \frac{\hat{v}_i^\top C v_j}{v_j^\top C v_j} C v_j = C\hat{v}_i - \sum_{j<i} \frac{\hat{v}_i^\top C v_j}{v_j^\top \cancel{\lambda_j} v_j} \cancel{\lambda_j} v_j = C\hat{v}_i - \sum_{j<i} (\hat{v}_i^\top C v_j) v_j = \Delta_i^\mu. \tag{5}$$

Therefore, once the first $(i-1)$ eigenvectors are learned, learning the ith eigenvector with μ-EigenGame is equivalent to learning with α-EigenGame. □

Lemma 2 (Zero bias)**.** *Unbiased estimates of Δ_i^μ can be obtained with samples from $p(X)$.*

Proof. Let $X \sim p(X)$ where $X \in \mathbb{R}^d$ and $p(X)$ is the uniform distribution over the dataset. Then

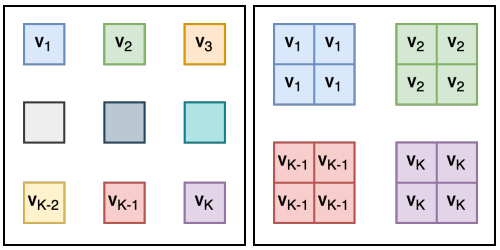
$$\mathbb{E}[\Delta_i^\mu] = \mathbb{E}[XX^\top]\hat{v}_i - \sum_{j<i}(\hat{v}_i^\top \mathbb{E}[XX^\top]\hat{v}_j)\hat{v}_j = C\hat{v}_i - \sum_{j<i}(\hat{v}_i^\top C\hat{v}_j)\hat{v}_j. \tag{6}$$

where all expectations are with respect to $p(X)$. □

These two lemmas provide the foundation for a performant algorithm. The first enables convergence to the desired solution, while the second facilitates scaling to larger datasets. Algorithm 1 presents pseudocode for μ-EigenGame where computation is parallelized over the k players.

3.3 Model and data parallelism

In our setting we have a number of connected devices. Specifically we consider the parallel framework specified by TPUv3 available in Google Cloud, however our setup is applicable to any multi-host, multi-device system. The α-EigenGame formulation (Gemp *et al.*, 2021) considers an extreme form of model parallelism (Figure 2a) where each device has its own unique set of eigenvectors.



(a) (b)

Figure 2: (a) Extreme model parallelism as proposed in α-EigenGame. (b) Model and data parallelism enabled by μ-EigenGame. Squares are separate devices (here, $M = 4$). Copies of estimates are color-coded. Updates are averaged across copies for a larger effective batch size.

In this work we further consider a different form of model and data parallelism which is directly enabled by having unbiased updates (Figure 2b). This enables μ-EigenGame to deal with both high-dimensional problems as well as massive sample sizes. Here each set of eigenvectors is copied on M devices. Update directions are computed on each device individually using a different data stream and then combined by summing or averaging. Updates are applied to a single copy and this is duplicated across the $M-1$ remaining devices. In this way, updates are computed using an $M\times$ larger effective batch size while still allowing device-wise model parallelism. This setting is particularly useful when the number of samples is very large. This form of parallelism is not possible using the original EigenGame formulation since it relies on combining *unbiased* updates. In this sense, the parallelism discussed in this work generalizes that introduced by Gemp *et al.* (2021).

Note that we also allow for within-device parallelism. That is, each v_i in Figure 2 is a contiguous collection of eigenvectors which are updated independently, in parallel, on a given device (for example using `vmap` in Jax). We provide pseudocode in Algorithm 2 in the appendix which simply augments Algorithm 1 with an additional parallelized for-loop and aggregation step over available devices. We also provide detailed Jax pseudo-code for parallel μ-EigenGame in Appendix F. We compare the empirical scaling performance of μ-EigenGame against α-EigenGame on a 14 billion sample dataset in section 5.

4 SVD as the solution to a new EigenGame

We theoretically examine the μ-EigenGame algorithm and 1) prove that, using only minibatches of data, μ-EigenGame converges globally to the true eigenvectors, which 2) comprise the Nash equilibrium of a novel game formulation we recover through deriving *pseudo*-utility functions from update rules. Beyond proving specific theoretical properties of μ-EigenGame, we believe these proof techniques may be of wider interest to the community.

4.1 CONVERGENCE TO SVD

The asymptotic equivalence of μ-EigenGame to α-EigenGame ensures μ-EigenGame is globally, asymptotically convergent and its unbiased updates ensure it is scalable. Proof in appendix C.

Theorem 1 (Global convergence)**.** *Given a positive definite covariance matrix C with the top-k eigengaps positive and a square-summable, not summable step size sequence η_t (e.g., $1/t$), Algorithm 1 converges to the top-k eigenvectors asymptotically ($\lim_{T\to\infty}$) with probability* 1.

This stochastic asymptotic convergence result is complimentary to the deterministic (full-batch) finite-sample result in Gemp *et al.* (2021) where each $\hat{v}_i$ is learned in sequence. In contrast, the proof above applies when learning all $\hat{v}_i$ in parallel. We leave finite-sample convergence to future work (Durmus *et al.*, 2020).

4.2 SVD IS NASH OF μ-EIGENGAME

We arrived at μ-EigenGame by analyzing and improving properties of the α-EigenGame update. However, the μ-EigenGame update direction is linear in each $\hat{v}_i$. This suggests we may be able to design a *pseudo*-utility function for it. Rearranging the update direction from equation (4) as

$$\Delta_i^\mu = C\hat{v}_i - \sum_{j<i} \hat{v}_j(\hat{v}_j^\top C\hat{v}_i) = \Big[I - \sum_{j<i} \hat{v}_j\hat{v}_j^\top\Big]C\hat{v}_i = \tilde{\nabla}_i^\mu \tag{7}$$

reveals that we can reverse-engineer the following utility function

$$u_i^\mu = \hat{v}_i^\top \Big[\overbrace{I - \sum_{j<i} \hat{v}_j\hat{v}_j^\top}^{\text{deflation}}\Big] C \bullet[\hat{v}_i] \tag{8}$$

where $\bullet$ is the stop gradient operator commonly used in deep learning packages. As the name implies, $\bullet$ stops gradients from flowing through its argument so that equation (8) appears linear in $\hat{v}_i$ instead of quadratic when differentiating the expression. In light of this, we have renamed Δ_i^μ to $\tilde{\nabla}_i^\mu$ to emphasize that it is a *pseudo*-gradient of u_i^μ. Note that without the stop gradient, the true gradient of u_i^μ would be $[A + A^\top]\hat{v}_i$ rather than $A\hat{v}_i$ where $A = [I - \sum_{j<i} \hat{v}_j^\top \hat{v}_j]C$. We analyze this alternative in Appendix H.1 and find it, interestingly, to perform worse than μ-EigenGame empirically.

The utility function u_i^μ has an intuitive meaning. It is the Rayleigh quotient for the matrix $C_i = [I - \sum_{j<i} \hat{v}^\top \hat{v}_j]C$, which gives the covariance after the subspace spanned by $\hat{v}_{j<i}$ has been removed. In other words, player i is directed to find the largest eigenvalue in the orthogonal complement of the approximate top-$(i-1)$ subspace. This approach is known as "deflating" the matrix C. Figure 1b illustrates μ-EigenGame's reduced bias when estimating the new utility function (and resulting optimum) from an average over minibatches.

Definition 1 (μ-Eigen**Game**)**.** *Let μ-EigenGame be the game with players $i \in \{1, \dots, k\}$, their respective strategy spaces $\hat{v}_i \in \mathcal{S}^{d-1}$, and their corresponding utilities u_i^μ as defined in equation (8).*

Theorem 2. *Top-k SVD is the unique Nash of μ-EigenGame given symmetric C with the top-k eigengaps positive.*

Proof. We will show by induction that each v_i is the unique best response to v_{-i}, which implies they constitute the unique Nash equilibrium. First, consider player 1's utility. It is the Rayleigh quotient of C because $\hat{v}_1$ is constrained to the unit-sphere, i.e., $u_1^\mu = \hat{v}_1^\top C\hat{v}_1 = \frac{\hat{v}_1^\top C\hat{v}_1}{\hat{v}_1^\top \hat{v}_1}$. Therefore, we know v_1 maximizes u_1^μ and the maximizer is unique because its eigengap $g_1 > 0$. In game theory parlance, v_1 is a *best response* to v_{-1}. The proof continues by induction. The utility of player i is $u_i^\mu = \hat{v}_i^\top[I - \sum_{j<i} v_j v_j^\top]C\hat{v}_i$, which is the Rayleigh quotient with the subspace spanned by the top $(i-1)$ eigenvectors removed. Therefore, the maximizer of u_i^μ is the largest eigenvector in the remaining subspace, i.e., v_i. As before, $g_i > 0$, so this maximizer is unique. This shows that each v_i is the unique best response to v_{-i}, therefore, the set of v_i forms the unique Nash. □

Notice how the induction proof of Theorem 2 relies on a) the hierarchy of vectors (v_1 does not depend on v_{-1}) and b) the fact that u_i^μ need only be a sensible utility when all player i's parents

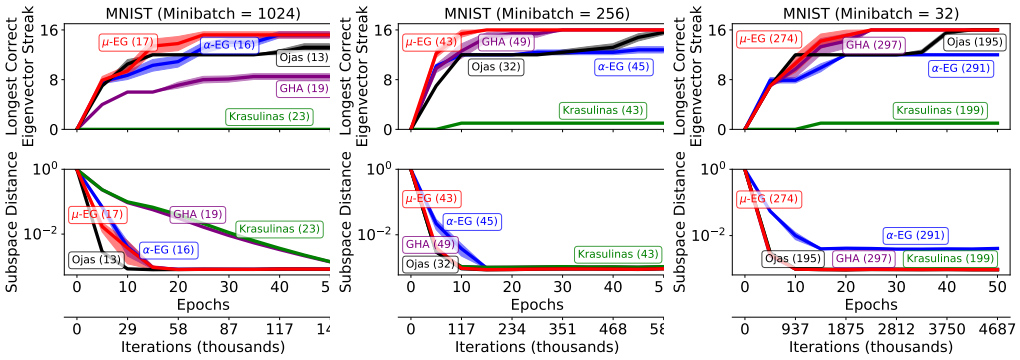


Figure 3: MNIST Experiment. Runtime (seconds) in legend on CPU ($m = 1$). Each column evaluates a different minibatch size $\in \{1024, 256, 32\}$. Shading indicates $\pm$ standard error of the mean. Learning rates were chosen from $\{10^{-3}, \ldots, 10^{-6}\}$ on 10 held out runs. Solid lines denote results with the best performing learning rate. All plots show means over 10 trials (randomness arising from minibatches and initialization). Shaded regions highlight $\pm$ standard error of the mean.

are eigenvectors. We revisit this in conjunction with Figure 5b later in discussion section 6.1 to aid researchers in the design of future approaches.

The Nash property is important because it enables the use of any black-box procedure for computing best responses. Like prior work, we develop a gradient method for optimizing each utility, however, that is not a requirement. Any approach suffices if it can efficiently compute a best response.

5 Experiments

As in EigenGame, we omit the projection of gradients onto the tangent space of the sphere; specifically, we omit line 8 in Algorithm 1. As discussed in Gemp *et al.* (2021), this has the effect of intelligently adapting the step size to use smaller learning rates near the fixed point. To ease comparison with previous work, we count the *longest correct eigenvector streak* as introduced by Gemp *et al.* (2021), which measures the number of eigenvectors that have been learned, in order, to within an angular threshold (e.g., $\pi/8$) of the true eigenvectors. We also measure how well the set of $\hat{v}_i$ captures the top-k subspace with a normalized subspace distance: $1 - 1/k \cdot \mathrm{Tr}(U^* P) \in [0, 1]$ where $U^* = VV^\dagger$ and $P = \hat{V}\hat{V}^\dagger$ (Tang, 2019). We provide additional experiments in Appendix A.

MNIST. We compare μ-EigenGame against α-EigenGame, GHA (Sanger, 1989), Matrix Krasulina (Tang, 2019), and Oja's algorithm (Allen-Zhu and Li, 2017) on the MNIST dataset. We flatten each image in the training set to obtain a $60,000 \times 784$ dimensional matrix X. Figure 3 demonstrates μ-EigenGame's robustness to minibatch size. It performs best in the longest streak metric and better than α-EigenGame in subspace distance. We attribute this improvement to its unbiased updates and additional acceleration effects which we discuss in detail in section H.2.

Meena conversational model. This dataset consists a subset of the 40 billion words used to train the transformer-based Meena language model (Adiwardana *et al.*, 2020). The subset was preprocessed to remove duplicates and then embedded using the trained model.

The dataset consists of $n \approx 14$ billion embeddings each with dimensionality $d = 2560$; its total size is 131TB. Due to its moderate dimensionality we can exactly compute the ground truth solution by iteratively accumulating the covariance matrix of the data and computing its eigendecomposition. On a single machine this takes 1.5 days (but is embarrassingly parallelizable with MapReduce).

We use minibatches of size 4,096 in each TPU. We do model parallelism across 4 TPUs so we see 16,384 samples per iteration. We test two additional degrees of data parallelism with $4\times$ (16 TPUs, 65,536 samples) and $8\times$ (32 TPUs, 131,072 samples) the amount of data per iteration respectively. We compute and apply updates using SGD with a learning rate of 5×10^{-5} and Nesterov momentum with a factor of 0.9.

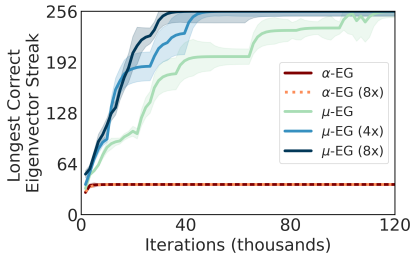


Figure 4: Comparison between μ-EigenGame and α-EigenGame with different degrees of data parallelism (in parentheses) on the Meena dataset.

Figure 4 compares the mean performance of μ-EigenGame against α-EigenGame as a function of the degree of parallelism in computing the top $k = 256$ eigenvectors (standard errors computed over 5 random seeds). Each TPU is tasked with learning 32 contiguous eigenvectors. We see that increasing the degree of parallelism has no effect on the performance of α-EigenGame. As expected, it is unable to take advantage of the higher data throughput since its updates are biased and cannot be meaningfully linearly combined across copies. In contrast, the performance of μ-EigenGame scales with the effective batch size achieved through parallelism. μ-EigenGame ($8\times$) is able to recover 256 eigenvectors in less than 40,000 iterations in 2 hours 45 minutes (approximately 0.5 epochs).

Spectral clustering on graphs. We conducted an experiment on learning the eigenvectors of the graph Laplacian of a social network graph (Leskovec and McAuley, 2012) for the purpose of spectral clustering. The eigenvalues of the graph Laplacian reveal several interesting properties as well such as the number of connected components, an approximation to the sparsest cut, and the diameter of a connected graph (Chung *et al.*, 1994).

Given a graph with a set of nodes $\mathcal{V}$ and set of edges $\mathcal{E}$, the graph Laplacian can be written as $\mathcal{L} = X^\top X$ where each row of the incidence matrix $X \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{V}|}$ represents a distinct edge; $X_{e=(i,j)\in\mathcal{E}}$ is a vector containing only 2 nonzero entries, a 1 at index i and a -1 at index j (Horaud, 2009). In this setting, the eigenvectors of primary interest are the bottom-k ($\lambda_{|\mathcal{V}|}, \lambda_{|\mathcal{V}|-1}, \ldots$) rather than the top-$k$ ($\lambda_1, \lambda_2, \ldots$), however, a simple algebraic manipulation allows us to reuse a top-k solver. By defining the matrix $\mathcal{L}^- = \lambda^* I - \mathcal{L}$ with $\lambda^* > \lambda_1$, we ensure $\mathcal{L}^- \succ 0$ and the top-k eigenvectors of $\mathcal{L}^-$ are the bottom-k of $\mathcal{L}$. The update in equation (4) is transformed into $\tilde{\nabla}_i^\mu = (\lambda^* I - \mathcal{L})\hat{v}_i - \sum_{j<i}\left(\hat{v}_i^\top(\lambda^* I - \mathcal{L})\hat{v}_j\right)\hat{v}_j$. We provide efficient pseudo-code in Appendix G.

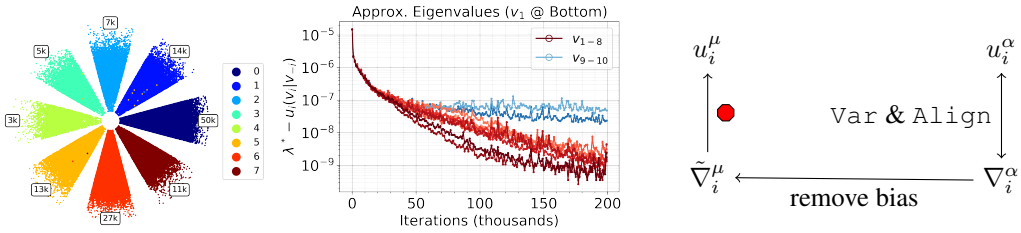
The Facebook graph consists of $134,833$ nodes, $1,380,293$ edges, and 8 connected components, each formed by a set of Facebook pages belonging to a distinct category, e.g., Government, TV shows, etc. (Leskovec and Krevl, 2014; Rozemberczki *et al.*, 2019). We add a single edge between every pair of components to create a connected graph. By projecting this graph onto the bottom 8 eigenvectors of the graph Laplacian using μ-EG ($M = 1, n' = \eta_t = \frac{|\mathcal{E}|}{1000}$) and then running k-means clustering (Pedregosa *et al.*, 2011), we are able to recover the ground truth clusters (see Figure 5a) with 99.92% accuracy. The experiment was run on a single CPU.

6 Discussion

6.1 Utilities to updates and back

Figure 5b summarizes the relationships advising the designs of the various EigenGame algorithms. Starting from the α-EigenGame utility, its update is arrived at by simply following the standard gradient ascent paradigm. In noticing that stochastic estimates of the gradient are biased, we arrive at the μ-EigenGame update by considering how to remove this bias in a principled manner.

Sacrificing the exact steepest decent direction for a direction that allows unbiased estimates is a tradeoff that in this case has benefits. Also, while $\tilde{\nabla}_i^\mu$ is not a gradient (except with exact parents), the new penalties have properties (above) that make them intuitively more desirable than the originals; they are adaptive to the state of the system (discussed further in section H.2).



(a) (b)

Figure 5: (5a) Facebook Page Networks. (Left) Petals differentiate ground truth clusters; colors differentiate learned clusters. Petals are ideally colored according to the color bar starting with the rightmost petal and proceeding counterclockwise. Numbers indicate ground truth cluster size. Clusters are extracted by running k-means clustering on the learned eigenvectors $\hat{V} \in \mathbb{R}^{|\mathcal{V}| \times k}$ (samples on rows). (Right) Rayleigh quotient plot reveals a gap between the 8th and 9th eigenvalues indicating ≈ 8 clusters exist. (5b) Relationships between utilities and updates. An arrow indicates the endpoint is reasonably derived from the origin; the lack of an arrow indicates the direction is unlikely.

We derive pseudo-utilities with desired theoretical properties by *integrating* the new updates with help from the stop gradient operator. However, it is unlikely that this utility would be developed independently of these steps to solve the problem at hand (see Appendix H for more details). This suggests an alternative approach to algorithm design complementary to the optimization perspective: directly designing updates themselves which converge to the desired solution, reminiscent of previous paradigms that drove neuro-inspired learning rules.

6.2 Bridging Hebbian and Optimization Approaches

The Generalized Hebbian Algorithm (GHA) (Sanger, 1989; Gang *et al.*, 2019; Chen *et al.*, 2019) update direction for $\hat{v}_i$ with inexact parents $\hat{v}_j$ is similar to μ-EigenGame:

$$\Delta_i^{gha} = C\hat{v}_i - \sum_{j \leq i} (\hat{v}_i^\top C \hat{v}_j)\hat{v}_j. \tag{9}$$

C appears linearly in this update so GHA can also be parallelized. In contrast to μ-EigenGame, GHA additionally penalizes the alignment of $\hat{v}_i$ to itself and removes the unit norm constraint on $\hat{v}_i$ (not shown). Without any constraints, GHA overflows in experiments. We take the approach of Gemp *et al.* (2021) and constrain $\hat{v}_i$ to the unit-ball ($||\hat{v}_i|| \leq 1$) rather than the unit-sphere ($||\hat{v}_i|| = 1$).

The connection between GHA and μ-EigenGame is interesting because unlike μ-EigenGame, GHA is a Hebbian learning algorithm inspired by neuroscience and its update rule is not motivated from the perspective of maximizing of a utility function. Game formulations of classical machine learning problems may provide a bridge between statistical and biologically inspired viewpoints.

7 Conclusion

We introduced μ-EigenGame, an unbiased, globally convergent, parallelizable algorithm that recovers the top-k eigenvectors of a symmetric positive definite matrix. We demonstrated the performance of μ-EigenGame on large scale dimension reduction and clustering problems. We discussed technical details of μ-EigenGame within the context of game theory, machine learning and neuroscience.

Like its predecessor, μ-EigenGame is a k-player, general-sum game allowing model parallelism over players; our unbiased reformulation allows even greater parallelism over data. Furthermore, the hierarchy and Nash property enable the exploration of more sophisticated best responses.

μ-EigenGame's improved robustness to smaller minibatches makes it more amenable to being used as part of deep learning, optimization (Krummenacher *et al.*, 2016), and regularization (Miyato *et al.*, 2018) techniques which leverage spectral information of gradient covariances or Hessians. Graph spectral methods have also recently shown to be related to state-of-the-art representation learning algorithms (HaoChen *et al.*, 2021) further cementing the importance of efficient SVD algorithms in modern machine learning.

REFERENCES

P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2009.

D. Adiwardana, M.-T. Luong, D. R. So, J. Hall, N. Fiedel, R. Thoppilan, Z. Yang, A. Kulshreshtha, G. Nemade, Y. Lu, et al. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977*, 2020.

Z. Allen-Zhu and Y. 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.

R. W. Brockett. Dynamical systems that sort lists, diagonalize matrices, and solve linear programming problems. *Linear Algebra and its applications*, 146:79–91, 1991.

Z. Chen, X. Li, L. Yang, J. Haupt, and T. Zhao. On constrained nonconvex stochastic optimization: A case study for generalized eigenvalue decomposition. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 916–925. PMLR, 2019.

F. R. Chung, V. Faber, and T. A. Manteuffel. An upper bound on the diameter of a graph from eigenvalues associated with its Laplacian. *SIAM Journal on Discrete Mathematics*, 7(3):443–457, 1994.

M. B. Cohen, C. Musco, and C. 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.

P. S. Dhillon, D. P. Foster, S. M. Kakade, and L. H. Ungar. A risk comparison of ordinary least squares vs ridge regression. *The Journal of Machine Learning Research*, 14(1):1505–1511, 2013.

A. Durmus, P. Jiménez, É. Moulines, S. Said, and H.-T. Wai. Convergence analysis of Riemannian stochastic approximation schemes. *arXiv preprint arXiv:2005.13284*, 2020.

J. Fan, D. Wang, K. Wang, and Z. Zhu. Distributed estimation of principal eigenspaces. *Annals of statistics*, 47(6):3009, 2019.

D. Feldman, M. Schmidt, and C. 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.

A. Gang, H. Raja, and W. U. Bajwa. Fast and communication-efficient distributed PCA. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7450–7454. IEEE, 2019.

D. Garber, O. Shamir, and N. Srebro. Communication-efficient algorithms for distributed stochastic principal component analysis. In *International Conference on Machine Learning*, pages 1203–1212. PMLR, 2017.

I. Gemp, B. McWilliams, C. Vernade, and T. Graepel. Eigengame: PCA as a Nash equilibrium. In *International Conference for Learning Representations*, 2021.

M. Ghashami, E. Liberty, J. M. Phillips, and D. P. Woodruff. Frequent directions: simple and deterministic matrix sketching. *SIAM Journal on Computing*, 45(5):1762–1792, 2016.

G. H. Golub and H. A. Van der Vorst. Eigenvalue computation in the 20th century. *Journal of Computational and Applied Mathematics*, 123(1-2):35–65, 2000.

A. Haidar, K. Kabir, D. Fayad, S. Tomov, and J. Dongarra. Out of memory SVD solver for big data. In *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–7. IEEE, 2017.

N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

J. Z. HaoChen, C. Wei, A. Gaidon, and T. Ma. Provable guarantees for self-supervised deep learning with spectral contrastive loss. *arXiv preprint arXiv:2106.04156*, 2021.

M. Hessel, D. Budden, F. Viola, M. Rosca, E. Sezener, and T. Hennigan. Optax: composable gradient transformation and optimisation, in JAX!, 2020.

R. Horaud. A short tutorial on graph Laplacians, Laplacian embedding, and spectral clustering, 2009.

I. T. Jolliffe. Principal components in regression analysis. In *Principal Component Analysis*. Springer, 2002.

R. Kannan and S. Vempala. *Spectral algorithms*. Now Publishers Inc, 2009.

G. Krummenacher, B. McWilliams, Y. Kilcher, J. M. Buhmann, and N. Meinshausen. Scalable adaptive stochastic optimization using random projections. In *Advances in Neural Information Processing Systems*, pages 1750–1758, 2016.

J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

J. Leskovec and J. McAuley. Learning to discover social circles in ego networks. *Advances in Neural Information Processing Systems*, 25:539–547, 2012.

Y. Liang, M.-F. Balcan, V. Kanchanapally, and D. P. Woodruff. Improved distributed principal component analysis. In *NIPS*, 2014.

A. Mead. Review of the development of multidimensional scaling methods. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 41(1):27–39, 1992.

T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

C. Musco and C. Musco. Randomized block Krylov methods for stronger and faster approximate singular value decomposition. In *Advances in Neural Information Processing Systems*, 2015.

E. Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15(3):267–273, 1982.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

B. Rozemberczki, R. Davies, R. Sarkar, and C. Sutton. Gemsec: Graph embedding with self clustering. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2019*, pages 65–72. ACM, 2019.

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

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

T. 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.

S. M. Shah. Stochastic approximation on Riemannian manifolds. *Applied Mathematics & Optimization*, pages 1–29, 2019.

O. 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.

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

J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

U. Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

Y. Wang, N. Xiu, and J. Han. On cone of nonsymmetric positive semidefinite matrices. *Linear Algebra and its Applications*, 433(4):718–736, 2010.