Eigengame: PCA as a Nash equilibrium

Master M2A - Sorbonne Universite - Statistical Learning

Nicolas Olivain

Abstract: **Dimensionality Reduction** is one of the key challenges of modern data science. With the advent of large datasets with thousands of feature, an estimator can take advantage of this abundance of data. However, this increase in the number of features raises other issues such as relevance or interpretability. Principal Component Analysis or **PCA**, is well known tool used for this purpose. PCA help identifying the *principal component* of a dataset and thus removing redundant or irrelevant features as well as improving interpretability. While PCA is usually derived as a projection problem on a subspace, Ian Gemp, Brian McWilliams, Claire Vernade Thore Graepel introduced at ICLR 2021 a novel interpretation of the problem based on **Game Theory**. According to the authors [4], the correct way to picture PCA would be as a **Nash Equilibrium** in an **Eigengame**. This comes with a new problem setup and new opportunities to identify *Principal Component*. We present here a comprehensive summary of their main results. This paper was written as part of the course *Statistical Learning* directed by Professor G. Biau at Sorbonne Universerte.

Key-words: Dimensionality reduction, Principal Component Analysis, Nash Equilibrium

1. Introduction

1.1. Notations

Given a dataset of samples $X \in \mathbb{R}^{n\times d}$, n being the number of samples and d the number of features per samples, we consider the usual Pearson Correlation of each feature $M = X^T X$. This conveniently makes $M \in \mathbb{R}^{d\times d}$ a symmetric positively defined matrix, ensuring the existence of positive eigenvalues by the spectral theorem.
We denote matrices as capital letters, and vectors as lower case letters. We use $\hat{}$to represent estimated values. Eigenvectors are considered order and usually denoted as v. For instance, V would be the matrix of real eigenvectors, $\hat{V}$ its approximation, v_i the eigenvector with the i-th largest eigenvalue and $\hat{v}_i$ its estimate. We denote as $v_{j<i}$ the set of eigenvectors $(v_1, ..., v_{i-1})$.

1.2. Principal Component Analysis

Principal Component Analysis, often called PCA, is an algorithm which addresses the problem of dimensionality reduction. When estimating a variable, one can have many different variable to consider. For instance, when estimating the GDP of a country, thousands of different indicators can be of relative use. The natural question following this scenario is then how to identify the most relevant one and go from thousands to tens without sacrificing precision too much ?
PCA proposes to identify these variables, the principal, component using Pearson's correlation. The usual interpretation would be to retain the variable with the highest variance and discarding those with redundant information or smaller variance. Thus the goal of the PCA dimensionality reduction is to identify a projected space of smaller dimension which retains most of the variance of data. .

Definition 1.1. *The PCA algorithm goal is to maximize the variance of the projected space. Considering a projection space of dimension 1, we want to estimate a vector* $\hat{v}_1$ *such that it maximizes* $r(v_1) = v_1^T M v_1$ *In a projection space of dimension* $k \leq d$*, we can rewrite the previous definition as:* $R(V) = V^T M V$*. So with* $\hat{V} = \arg\max_V R(V)$*, we get the projection in* k *dimensions* $P = X\hat{V}$ *[3], [8]*

Theorem 1.1. *The vector* $\hat{v}_1$ *which maximizes* $r(\hat{v}_1)$ *is the eigenvector of* M *with the highest associated eigenvalue. The matrix* $\hat{v}$ *which maximizes* $R(\hat{v})$ *is a base of eigenvectors with the top* k *highest associated eigenvalues.*

Proof. We want to maximize $r(\hat{v}_1) = \hat{v}_1^T M \hat{v}_1$ under the constraint $||\hat{v}_1|| = 1$, as we are looking for directions.

We derive the associated Lagrangian with its multiplier α_1 as

$$L = \hat{v}_1^T M \hat{v}_1 - \alpha_1 (1 - \hat{v}_1^T \hat{v}_1)$$

Looking for the critical points of L give us the following set of equations:

$$\begin{cases} \dfrac{\partial L}{\partial \hat{v}_1} = M\hat{v}_1 - \alpha_1 \hat{v}_1 & \text{(1a)} \\ \dfrac{\partial L}{\partial \alpha_1} = 1 - \hat{v}_1^T \hat{v}_1 & \text{(1b)} \end{cases}$$

Thus we have $M\hat{v}_1 = \alpha_1 \hat{v}_1$, so $\hat{v}_1$ is an eigenvector of M. As we want to maximize the objective r, we get that α_1 must be the biggest available eigenvalue, as $r = \alpha_1 \hat{v}_1^T \hat{v}_1 = \alpha_1$. We can prove similarly that $\hat{v}_i$ is the i^{th} biggest eigenvalue of M by adding the new constraints: $\hat{v}_j^T \hat{v}_i = 0$ for $j < i$. Thus we get the result in dimension k that, for $1 \leq i \leq k$ M$_i = \alpha_i \hat{v}_i$
$\hat{V} = (\hat{v}_1, ..., \hat{v}_k)$
As a consequence of Theorem 1.1, finding the PCA decomposition of X simply consists in identifying the top k eigenvectors v_i and their associated eigenvalues α_i.

1.3. Game Theory

Game Theory is a field of mathematics studying interactions between agents (*ie:* players). In Game Theory, each player i has a set of strategies S_i and a utility function u_i. Assuming players are rational, a player is always trying to maximize his utility, picking the best strategy $s_i \in S_i$ with regards u_i. However, the best strategy for one player usually depends on the strategies picked by other players [1].

Definition 1.2. *A utility function u_i for player i is a function of a strategy profile (the strategies of all the players) $s = \{s_1, ..., s_i, ..., s_n\}$ representing the reward for player i with this strategy profile s.*

Denoting the strategies of all players *except* i as s_{-i}, we have that u_i is a function of s_i and s_{-i}, that is to say, of player's i strategy and of the other players strategy. Using this notation, a strategy profile s can be written as (s_i, s_{-i}), which is the preferred notations for u_i's arguments.
As each player is supposed aware of the actions of its counterparts, thus the rational player will try to tune his own strategy s_i in order to maximize its utility u_i given a strategy profile s. However, other players are likely to react to this change in s_i modifying s_{-i}, leading to further changes in s_i and so on. An equilibrium is reached when no player has an incentive to change his action and will choose to stick to its given strategy. This concept is formalized in the Nash Equilibrium.

Definition 1.3. *A Nash equilibrium is a stable state where no player has an incentive to deviate its strategy with regards to u_i. This mean s^* is an equilibrium if for any player i, $u_i(s_i^*, s_{-i}^*) \geq u_i(s_k, s_{-i}^*)$, for $s_k \in S_i$.*

Definition 1.4. *A strict Nash equilibrium is a set of strategies s^* where $u_i(s_i^*, s_{-i}^*) > u_i(s_k, s_{-i}^*)$, for $s_k \in S_i$.*

A consequence of this definition is that an equilibrium s^* is a critic point of u_i. Finally, we introduce a specific case of Nash Equilibrium.

Definition 1.5. *A symmetric Nash equilibrium is a Nash equilibrium where all player follow the same strategy, ie: $s^* = (s_0, s_0, ..., s_0)$, for $s_0 \in S_i$*

2. PCA as an Eigengame

Instead of interpreting **PCA** as a projection on a sub-space of maximum variance, authors in [4] suggest to re-focus on learning the *principal components*, meaning finding vectors that align with the maximum variance. For this purpose we construct a competitive game where each player has control of one eigenvector and tries to maximize its utility.

Definition 2.1. *We define as **Eigengame** a competitive game where player i has to find the unitary eigenvector associated to the i-th largest eigenvalue of a matrix M*

2.1. Utility Functions

In order to define the Eigengame, we need to define utility functions for each player. Consider we are trying to learn the top $k < d$ principal components, we have $V \in \mathbb{R}^{d \times k}$. As seen in 1.1, we need to maximize $R(\hat{V}) = \hat{V}^T M \hat{V}$. However, as $\hat{V}$ is no longer square, we do not have the identity $\hat{V}^T \hat{V} = I$, but rather

$\hat{V}^T\hat{V} = P$, with P a projection matrix as we can construct $\hat{V}$ as orthonormal. Thus, with Λ a diagonal matrix of eigenvalues we get:

$$(PM)V = V \tag{3}$$

This identity means that we are solving an eigenvalue problem on a subspace of M
The PCA problem is often posed as minimizing the reconstruction error, that is to say maximizing the trace

$$\max_{\hat{V}^T\hat{V}=1} \mathrm{Tr}(R(\hat{V})) = \max_{\hat{V}^T\hat{V}=1} \sum_i R_{ii} \tag{4}$$

While this helps learning a subspace maximizing the variance, this does not enforce any constraints on the eigenvectors themselves: it is not granted that maximizing this term will lead to learning the actual eigenvectors of M. A complementary idea would be penalize on the non-diagonal terms:

$$\min_{\hat{V}^T\hat{V}=1} \sum_{i\neq j} R_{ij}^2 \tag{5}$$

This objective however has no incentive to learn the eigenvectors associated to the largest eigenvalues. The natural idea would then be to combine equation 4 and 5 in:

$$\max_{\hat{V}^T\hat{V}=1} \sum_i R_{ii} - \sum_{i\neq j} R_{ij}^2 \tag{6}$$

While this proposal is valid, it does not take into account the hierarchy of the eigenvectors. That means that we do not ensure that $\hat{\lambda}_1$ is the eigenvector with the highest eigenvalue (recall that we defined the λs with the usual order $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_d$). Deriving from this expression, we can write a utility for each player, sequentially restraining their degree of freedom. The first eigenvector is free to choose any direction maximizing variable, second one will be penalized if it aligns with the first and so on: the eigenvector i will be penalized if it aligns itself with any vector up to $i-1$. This way, each vector will maximize its variance in directions that are still non penalized for them. With this logic, we find $\hat{v}_1$ with

$$\hat{v}_1 = \arg\max_{\hat{v}_1^T\hat{v}_1=1} \hat{v}_1^T M \hat{v}_1 \tag{7}$$

Now that $\hat{v}_1$ is fixed, $\hat{v}_2$ is restrained in its available directions $\langle \hat{v}_2, M\hat{v}_1\rangle$, and we need a new constraint to ensure orthonormality of V: $\langle \hat{v}_1, \hat{v}_2\rangle = 0$. These two imperatives are redundant, as by construction $M\hat{v}_1 = \alpha_1\hat{v}_1$. We can now express $\hat{v}_2$ in equation 8, note that we add a rescaling term for the diagonal penalization in the form of $\langle \hat{v}_1, M\hat{v}_1\rangle$.

$$\hat{v}_2 = \arg\max_{\hat{v}_2^T\hat{v}_2=1} \hat{v}_2^T M \hat{v}_2 - \frac{\langle \hat{v}_2, M\hat{v}_1\rangle^2}{\langle \hat{v}_1, M\hat{v}_1\rangle} \tag{8}$$

We can now generalize equation 8 to compute $\hat{v}_i$ the i-th eigenvector. If we now place ourselves in the Game Theory framework, this function to optimize on $\hat{v}_i$ is the utility function u_i (see equation 9) for player i, as its objective is to find the i-th eigenvector, based on what the $i-1$ previous player picked. As a result of the preceding construction, we can derive:

Theorem 2.1. *Assume that the top k eigenvalues of $M = X^TX$ are positive and distinct. Under the constraint $\langle \hat{v}_i, \hat{v}_i\rangle = 1$, the function u_i defined below is a suitable* ***utility function*** *for the i-th player of the* ***Eigengame:***

$$u_i(\hat{v}_i|\hat{v}_{j<i}) = \hat{v}_i^T M \hat{v}_i - \sum_{j<i} \frac{\langle \hat{v}_i, M\hat{v}_j\rangle^2}{\langle \hat{v}_j, M\hat{v}_j\rangle} = ||X\hat{v}_i||^2 - \sum_{j<i} \frac{\langle \hat{v}_i, M\hat{v}_j\rangle^2}{\langle \hat{v}_j, M\hat{v}_j\rangle} \tag{9}$$

Proof. We want to construct utility functions such that $\arg\max_{\hat{v}_i} u_i(\hat{v}_i|v_{j<i}) = v_i$ so that player i tries to reach the i-th eigenvector. As $M = X^TX$, we can find an orthonormal matrix U and a diagonal matrix Λ such that we have the identity $M = U^T\Lambda U$. For $i = 1$, we have $\langle \hat{v}_1, \hat{v}_1\rangle = 1$ and $u_1(\hat{v}_1) = \hat{v}_1^T M \hat{v}_1 = (U\hat{v}_1)^T\Lambda(U\hat{v}_1)$. Thus, as $||U\hat{v}_1|| = 1$, we have $v_1 = \arg\max_{\hat{v}_1} u_1(\hat{v}_1)$. Assuming that for $j < i$ we have $\hat{v}_j = v_j$, let us derive u_i.
First we write $\hat{v}_i = \sum_{p=1}^d w_p v_p$ as combination of the true eigenvectors, which is always possible as M is full-rank. In order to ensure that $||\hat{v}_i|| = 1$, we choose w such that $||w|| = 1$. Now we have:

$$
\begin{aligned}
u_i(\hat{v}_i|v_{j<i}) &= \hat{v}_i^T M \hat{v}_i - \sum_{j<i} \frac{\langle \hat{v}_i, M v_j \rangle^2}{\langle \hat{v}_j, M v_j \rangle} \\
&= (\sum_p w_p v_p^T) M (\sum_q w_q v_q) - \sum_{j<i} \frac{\langle \hat{v}_i, \Lambda_{jj} v_j \rangle^2}{\Lambda_{jj}} \\
&= \sum_{p,q} w_p w_q v_p^T M v_q - \sum_{j<i} \frac{(\sum_p w_p v_p^T M v_j)^2}{\Lambda_{jj}} \\
&= \sum_{p,q} w_p w_q \Lambda_{qq} v_p^T v_q - \sum_{j<i} \frac{(\sum_p w_p \Lambda_{jj} v_p^T v_j)^2}{\Lambda_{jj}}
\end{aligned} \tag{10}
$$

Now, recall that as $(v_i)_{1 \leq i \leq d}$ forms an orthonormal basis of $\mathbb{R}^d$, we have that $v_i^T v_j = \mathbb{1}_{i=j}$. Thus we now have:

$$
\begin{aligned}
u_i(\hat{v}_i|v_{j<i}) &= \sum_p w_p^2 \Lambda_{pp} - \sum_{j<i} \frac{(w_j \Lambda_{jj})^2}{\Lambda_{jj}} \\
&= \sum_p w_p^2 \Lambda_{pp} - \sum_{j<i} w_j^2 \Lambda_{jj} \\
&= \sum_{p \geq i} w_p^2 \Lambda_{pp}
\end{aligned} \tag{11}
$$

As $||w|| = 1$, we have $\sum_i w_i^2 = \sum_i z_i = 1$. This means that we need to find a vector z on the simplex maximizing $\sum_{p>i} z_p \Lambda_{pp}$. Now recall that by construction the eigenvalues of M, *ie:* Λ's diagonal terms, are arrange in a decreasing manner. Thus it is clear that the best possible z for $u_i(\hat{v}_i|v_{j<i})$ is e_i, the i-th vector from the canonical basis of $\mathbb{R}^d$. As a follow up, we get that $w = e_i$ and that $\hat{v}_i = v_i$, which proves our point.

2.2. Nash Equilibrium

Now that the game is settled and we have a utility function representing each player's objective, we try to solve it by finding the equilibrium and its link to the original PCA problem.

Theorem 2.2. *Assume that the top k eigenvalues of $M = X^T X$ are positive and distinct. Then the top k eigenvectors form the unique strict Nash equilibrium of the proposed game from definition 2.1 and theorem 2.1.*

Proof. Recall that from the proof of 2.1, we get that $\hat{v}_i = v_i$ when player i plays $z_i^* = e_i = (0, ..., 0, w_i^2, 0, ..., 0)$. As a matter of fact, with $\Lambda_{ii} > \Lambda_{pp}$ for $p > i$, any player j deviating unilaterally from z_i^* is guaranteed to decrease its utility. This means that only v_i and $-v_i$ maximize the player utility, meaning that the eigenvectors are the only strict Nash Equilibrium up to a sign change. This was to be expected as both v_i and $-v_i$ are principal components.

2.3. What happens without the hierarchy?

This formulation of the problem detailed in 2.1 makes each utility u_i dependent on the previous one $u_{i<j}$. We could represent these dependencies in a Directed Acyclic Graph (DAG) as in Figure 1.

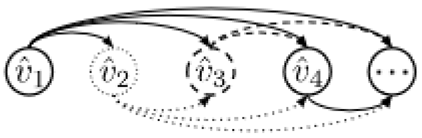


Figure 1: a DAG representation of the dependencies between utility functions. Each node representing player i depends on all the previous nodes, meaning all nodes from 1 to $i-1$.

However to reach this results and derive the utility function in 2.1, we assumed a hierarchical order between the eigenvalues. While this makes the analysis easier and seems to help with the convergence in practice, this would not be optimal in terms of iterations. Suppose we initialize all eigenvectors $\hat{v}_i$ on the unit sphere, then $\hat{v}_k$ can be initialized closer to v_1 than v_k, which could be seen as an optimization opportunity: each player could individually converge to the closest distinct eigenvector. In this case we could define the alternate utility function 12

$$u_i(\hat{v}_i, \hat{v}_{-i}) = \hat{v}_i^T M \hat{v}_i - \sum_{j \neq i} \frac{\langle \hat{v}_i, M\hat{v}_j \rangle^2}{\langle \hat{v}_j, M\hat{v}_j \rangle} \tag{12}$$

Yet, this new utility function is symmetric with regards to the player i. Therefore, one could think that all player could be tempted to converge towards the same eigenvector, meaning we reach a symmetric Nash equilibrium. This is would obviously be detrimental as we would identify only one principal component. Hopefully, we can refute situation with the following theorem.

Theorem 2.3. *Assuming $k \geq 2$ and $\text{rank}(M) \geq 2$, the Eigengame defined using equation 12 does not contain any symmetric Nash equilibrium.*

Proof. We will prove this theorem by contradiction. Suppose we have for all i and j $\hat{v}_i = \hat{v}_j$, then from 12:

$$u_i(\hat{v}_i, \hat{v}_{-i}) = (1 - (n-1))\hat{v}_i^T M \hat{v}_i = (2-n)\hat{v}_i^T M \hat{v}_i \leq 0$$

Now consider a unilateral orthogonal deviation from player i $v_\perp$ such that $v_\perp \perp \hat{v}_i$, meaning that $v_\perp^T \hat{v}_j = 0$ for all j as $\hat{v}_j$ are all identical. We now get from equation 12 as $\text{rank}(M) \geq 2$

$$u_i(v_\perp, \hat{v}_{-i}) = v_\perp^T M v_\perp > 0$$

Thus $v_\perp$ is a relevant deviation for player i, which shows that a symmetric Nash equilibrium is impossible.
We can also prove that the PCA solutions are a Nash equilibrium of this game, however we loose the unicity achieved with ordered case in theorem 2.2.

Theorem 2.4. *Assuming $\text{rank}(M) \geq 2$, the top k eigenvectors of M form a strict Nash equilibrium of the Eigengame defined in equation 12.*

Proof. As in theorem 2.1, clearly $v_1 = \arg\max_v u_1(v, v_{-1})$. Now let us consider the other vectors v_i. Given a permutation σ, let us assume that player i plays $\hat{v}_{\sigma(i)}$. As in the proof of 2.1, we write $\hat{v}_i$ as a combination of the real eigenvectors with $||w|| = 1$: $\hat{v}_i = \sum_{p=1}^d w_p v_p$. By the same simplification, reach the following identity

$$\begin{aligned} u_i(\hat{v}_{\sigma(i)}, v_{-\sigma(i)}) &= \sum_q w_q^2 \Lambda_{qq} - \sum_{j \neq \sigma(i)} w_j^2 \Lambda_{jj} \\ &= \Lambda_{\sigma(i)\sigma(i)} w_{\sigma(i)}^2 \end{aligned} \tag{13}$$

Which is maximized for $w_{\sigma(i)}^2 = 1$, assuming $\Lambda_{\sigma(i)\sigma(i)} > 0$. Thus we get that $w_{\sigma(i)} = \pm 1$ and $w_{j \neq \sigma(i)} = 0$, meaning $\hat{v}_{\sigma(i)} = \pm v_{\sigma(i)} = \arg\max_v u_i(\hat{v}, v_{-\sigma(i)})$. Consequently the eigenvectors family $(v_1, v_\sigma 2, ..., v_{\sigma(d)})$ forms a strict Nash equilibrium for the Eigengame defined in equation 12.
Thus removing the hierarchy hypothesis on the eigenvalues does not harm the equilibrium of the Eigengame, however, authors were unable to prove that this Nash Equilibrium was the only one present in this version of the game.

3. Finding the equilibrium

3.1. Utility Gradient

Proposition 3.1. *The utility function derived in 2.1 has a gradient of:*

$$\nabla_{\hat{v}_i} u(\hat{v}_i, \hat{v}_{j<i}) = 2M \left[\hat{v}_i - \sum_{j<i} \frac{\hat{v}_i^T M \hat{v}_j}{\hat{v}_j^T M \hat{v}_j} \hat{v}_j \right] = 2X^T \left[X\hat{v}_i - \sum_{j<i} \frac{\langle X\hat{v}_i, X\hat{v}_j \rangle}{\langle X\hat{v}_j, X\hat{v}_j \rangle} X\hat{v}_j \right]$$

This gradient with its rescaling penalization term that we introduce in the previous section can be interpreted as a generalized Gram-Schmidt step. We project $\hat{v}_i$ on $M\hat{v}_j$ for $j < i$. It is interesting to note that when $M = I$, applying $\hat{v} = \frac{1}{2}\nabla_{\hat{v}_i} u_i$ gives us back the standard Gram-Schmidt basis orthnormalisation step.

3.2. Sequential Algorithm

Each eigenvector can be learnt by maximizing its utility. We have shown that the utility function u_i depends on $\hat{v}_i$ and on $\hat{v}_{j<i}$, meaning that assuming we previously computed $\hat{v}_{j<i}$, we can sequentially derive $\hat{v}_i$. If $\hat{v}_{j<i}$ were to be learnt concurrently to $\hat{v}_i$, we would be maximizing a non stationary objective, making the convergence difficult. Also, remember that a constraint for all the eigenvectors is that they all belong to the unit sphere as they are supposed to be unitary. This calls for the *Riemanian Gradient Ascent* method. This methods adds an extra step to the gradient descent where the gradient $\nabla_{\hat{v}_i} u_i$ is projected on the tangent space of the unit sphere, meaning that we get the following Riemanian gradient.

$$\nabla^R_{\hat{v}_i} u_i = \nabla_{\hat{v}_i} u_i - \langle \nabla_{\hat{v}_i} u_i, \hat{v}_i \rangle \hat{v}_i$$

Algorithm 1 EigenGame^R-Sequential

```
Require: $X \in \mathbb{R}^{n \times d}$, maximum error tolerance $\rho_i$, step size $\alpha$, unit vector $\hat{v}_i^0$ and previous parents $\hat{v}_{j<i}$
  $\hat{v}_i \longleftarrow \hat{v}_i^0$
  $t_i = \frac{5}{4} \min(\frac{1}{2} ||\nabla_{\hat{v}_i^0} u_i||, \rho_i)^{-2}$
  for $1 \leq t \leq t_i$ do
    rewards $\longleftarrow X\hat{v}_i$
    penalties $\longleftarrow \sum_{j<i} \frac{\langle X\hat{v}_i, X\hat{v}_j \rangle}{\langle X\hat{v}_j, X\hat{v}_j \rangle} X\hat{v}_j$
    $\nabla_{\hat{v}_i} = 2X^T$[rewards - penalties ]
    $\nabla^R_{\hat{v}_i} = \nabla_{\hat{v}_i} u_i - \langle \nabla_{\hat{v}_i} u_i, \hat{v}_i \rangle \hat{v}_i$
    $\hat{v'}_i \longleftarrow \hat{v}_i + \alpha \nabla^R_{\hat{v}_i}$
    $\hat{v}_i = \frac{\hat{v'}_i}{||\hat{v'}_i||}$
  end for
  return $\hat{v}_i$
```

3.3. Decentralized Algorithm

The algorithm 1 is very interesting as it provides converge guarantees as detailed in its proof. However it comes with some serious limitation. Indeed, it requires all the previous $\hat{v}_{j<i}$ to be computed beforehand, this algorithm is purely sequential and cannot be parallelized. On the other hand, when getting closer to the optimum, the values of $\hat{v}_{j<i}$ should not change by much, meaning they are quasi-stationary so they should allow to start the training for the next $\hat{v}_i$. Authors addressed this issue by proposing a second algorithm which is decentralized and allow each player to learn independently on its own hardware. This second algorithm is extremely similar to the first one, except that it does not ask for $\hat{v}_{j<i}$ as a prerequisite and each player i broadcasts the estimate $\hat{v}_i$ for the other player's next iteration.

Algorithm 2 EigenGame^R

```
Require: $X \in \mathbb{R}^{n \times d}$, maximum error tolerance $\rho_i$,step size $\alpha$, unit vector $\hat{v}_i^0$
  $\hat{v}_i \longleftarrow \hat{v}_i^0$
  for $1 \leq t \leq T$ do
    rewards $\longleftarrow X\hat{v}_i$
    penalties $\longleftarrow \sum_{j<i} \frac{\langle X\hat{v}_i, X\hat{v}_j \rangle}{\langle X\hat{v}_j, X\hat{v}_j \rangle} X\hat{v}_j$
    $\nabla_{\hat{v}_i} = 2X^T$[rewards - penalties ]
    $\nabla^R_{\hat{v}_i} = \nabla_{\hat{v}_i} u_i - \langle \nabla_{\hat{v}_i} u_i, \hat{v}_i \rangle \hat{v}_i$
    $\hat{v'}_i \longleftarrow \hat{v}_i + \alpha \nabla^R_{\hat{v}_i}$
    $\hat{v}_i = \frac{\hat{v'}_i}{||\hat{v'}_i||}$
    broadcast($\hat{v}_i$)
  end for
  return $\hat{v}_i$
```

4. Error and Convergence of the Eigengame

4.1. Generalities

We introduced in section 3 two algorithms to reach the equilibrium of the Eigengame defined in section 2. In this section, we discuss the theoretical results on convergence and error propagation from $\hat{v}_{j<j}$ to $\hat{v}_i$. First we introduce some important concepts.
As our research space and all our vectors $\hat{v}_i$ are located on the unit sphere, they can be easily defined with a set of angle. Thus, we can estimate our precision with an angular error between $\hat{v}_i$ and the true eigenvectorv_i.

Definition 4.1. *We define the angular error between two unit vectors as* $\theta(u, v) = \sin^{-1}(\sqrt{1 - \langle u, v\rangle^2})$

The unit sphere can be parameterized with the Riemannian exponential mapping applied to a vector deviation from an anchor point, here the true eigenvector v_i. Let us derive $\hat{v}_i = \cos(\theta_i)v_i + \sin(\theta_i)\Delta_i$, with Δ_i a vector of the sphere orthogonal to v_i. While, θ_i denotes the angular deviation between $\hat{v}_i$ and v_i in radian, while Δ_i can be interpreted as the direction of deviation.

Definition 4.2. *We define Δ_i as the vector such that $||\Delta_i|| = 1$, $\langle \Delta_i, v_i\rangle = 0$ and that $\hat{v}_i = \cos(\theta_i)v_i + \sin(\theta_i)\Delta_i$, where θ_i is the angular deviation $\hat{v}_i$ and v_i.*

Definition 4.3. *We introduce the eigenvalue gap $g_i = \Lambda_{ii} - \Lambda_{i+1,i+1}$ for $0 \le i \le d-1$. With the usual ordering, we have that $g_i \ge 0$.*

4.2. Parent to child error propagation

First let us consider the error when the previous parent eigenvectors are perfectly learnt, *ie:* with $j < i$ we have $\hat{v}_j = v_j$. This mean that we try to estimate $|u_i(\hat{v}_i, v_{j<i}) - \Lambda_{ii}|$, the gap between the utility of our estimation $\hat{v}_i$ and the actual eigenvalue reached when player i plays v_i.

Lemma 4.1. *Let $\hat{v}_i = \cos(\theta_i)v_i + \sin(\theta_i)\Delta_i$. Then with $||z|| = 1$ we get that*

$$u_i(\Delta_i, v_{j<i}) = \sum_{l>i} z_l \Lambda_{ll} \tag{14}$$

Proof. Recall from equation 11 that $u_i(v, v_{j<i}) = \sum_{l\ge i} z_l\Lambda_{ll}$ with $v = \sum_i w_i v_i$ and $z_i = w_i^2$. As $\langle \Delta_i, v_i\rangle = 0$, this enforces that $w_i = 0$ and consequently that $z_i = 0$. Thus $u_i(\Delta_i, v_{j<i}) = \sum_{l>i} z_l\Lambda_{ll}$

Theorem 4.1. *Using the Riemannian decomposition $\langle \Delta_i, v_i\rangle = 0$ and that $\hat{v}_i = \cos(\theta_i)v_i + \sin(\theta_i)\Delta_i$, we have*

$$u_i(\hat{v}_i, v_{j<i}) = \Lambda_{ii} - \sin^2(\theta_i)\left(\Lambda_{ii} - \sum_{l>i} w_l^2 \Lambda_{ll}\right) \tag{15}$$

Proof. Continuing the simplification from the proof of theorem 2.1, we get that:

$$\begin{aligned}
u_i(\hat{v}_i, v_{j<i}) &= \langle \hat{v}_i, \Lambda\hat{v}_i\rangle - \sum_{j<i} \frac{\langle \hat{v}_i, \Lambda v_j\rangle^2}{\langle \hat{v}_j, \Lambda v_j\rangle} \\
&= \langle \hat{v}_i, \Lambda\hat{v}_i\rangle - \sum_{j<i} \Lambda_{jj}\langle \hat{v}_i, v_j\rangle^2 \\
&= \cos^2(\theta_i)\Lambda_{ii} + \sin^2(\theta_i)\langle \Delta_i, \Lambda\Delta_i\rangle - \sum_{j<i} \Lambda_{jj}\langle \cos(\theta_i)v_i + \sin(\theta_i)\Delta_i, v_j\rangle^2 \\
&= \cos^2(\theta_i)\Lambda_{ii} + \sin^2(\theta_i)\langle \Delta_i, \Lambda\Delta_i\rangle - \sum_{j<i} \Lambda_{jj}\sin^2(\theta_i)\langle \Delta_i, v_j\rangle^2 \\
&= \Lambda_i i(1 - \sin^2(\theta_i)) + \sin^2(\theta_i)\left[\langle \Delta_i, \Lambda\Delta_i\rangle - \sum_{j<i} \Lambda_{jj}\langle \Delta_i, v_j\rangle^2\right] \\
&= \Lambda_{ii} - \sin^2(\theta_i)\Lambda_{ii} - \sin^2(\theta_i)u_i(\Delta_i, v_{j<i})
\end{aligned} \tag{16}$$

Applying the lemma 4.1 to the last equation, we finally get that $u_i(\hat{v}_i, v_{j<i}) = \Lambda_{ii} - \sin^2(\theta_i)\left(\Lambda_{ii} - \sum_{l>i} w_l^2\Lambda_{ll}\right)$

This result allows us to characterize the error in utility given the angular error between $\hat{v}_i$ and v_i. It is important to note that $\sin^2$ is π-periodic, showing that both v_i and $-v_i$ are acceptable solution as explained in the equilibrium derivation. While this result is very useful for the sequential application, it only help us to grasp the utility error in the sequential algorithm, as we assume the parents to be perfectly computed. However if we want to be able to parallelize the algorithm, we need to estimate at which point the error on the parents will be good enough to compute the descendant. That is to say, we now need to compute the error with an estimated objective, *ie:* $u_i(\hat{v}_i, \hat{v}_{j<i})$. For this purpose we start with a lemma which will not be proved nor further developed.

Lemma 4.2. *Let $\hat{v}_i = \cos(\theta_i)v_i + \sin(\theta_i)\Delta_i$ for all $j < i$ without loss of generality. Then we can find A, B and C such that:*

$$u_i(\hat{v}_i, \hat{v}_{j<i}) = A(\theta_j, \Delta_j, \Delta_i)\sin^2(\theta_i) - B(\theta_j, \Delta_j, \Delta_i)\frac{\sin(2\theta_i)}{2} + C(\theta_j, \Delta_j, \Delta_i) \tag{17}$$

With the use of this Lemma we can now tackle the challenge of the error with mis-specified objective. Yet, we are now more interested in the relationship between θ_i and $\hat{v}_{j<i}$, because the objective is no longer the real eigenvector v_i. As such, we now have a bias in the target and no longer aiming toward Λ_{ii}. Thus, we want to characterize the angular error θ_i^* between the optimal estimate given the imperfect $\hat{v}_{j<i}$ and the real eigenvector v_i. This can tell us at which point the $\hat{v}_{j<i}$ are precise enough for the target to be close enough to v_i so maximizing it makes sense.

Theorem 4.2. *The angular deviation θ_i^* of the vector $\hat{v}_i^*$ 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:*

$$|\theta_i^*| = \begin{cases} \frac{1}{2}\tan^{-1}(|\frac{A}{B}|) & \text{if } A < 0 \\ \frac{\pi}{4} & \text{if } A = 0 \\ \frac{1}{2}\left[\pi - \tan^{-1}(|\frac{B}{A}|)\right] & \text{if } A > 0 \end{cases} \tag{18}$$

With A and B from lemma 4.2

Proof. We know from 4.1 that $u_i(\hat{v}_i, \hat{v}_{j<i}) = A(\theta_j, \Delta_j, \Delta_i)\sin^2(\theta_i) - B(\theta_j, \Delta_j, \Delta_i)\frac{\sin(2\theta_i)}{2} + C(\theta_j, \Delta_j, \Delta_i)$. We first identify the critical points.

$$\begin{aligned} \frac{\partial}{\partial\theta_i}u_i(\hat{v}_i, \hat{v}_{j<i}) &= 2A\sin(\theta_i)\cos(\theta_i) - B\cos(2\theta_i) \\ &= A\sin(2\theta_i) + B\cos(2\theta_i) \\ &= \frac{1}{\cos(2\theta_i)}[A\tan(2\theta_i) - B] \end{aligned} \tag{19}$$

This we reach the following equation to characterize critical points:

$$\frac{\partial}{\partial\theta_i}u_i(\hat{v}_i, \hat{v}_{j<i}) = 0 \iff \tan(2\theta_i) = \frac{B}{A} \tag{20}$$

Let us now discuss whether these are maxima or minima:

$$\begin{aligned} \frac{\partial^2}{\partial\theta_i^2}u_i(\hat{v}_i, \hat{v}_{j<i}) &= \frac{-2\sin(2\theta_i)}{\cos^2(2\theta_i)}(A\tan(2\theta_i) - B) + \frac{2}{\cos(2\theta_i)}(A\tan^2(2\theta_i) - A) \\ &= \frac{2}{\cos(2\theta_i)}\left[-A\tan^2(2\theta_i) + B\tan(2\theta_i) + A\tan^2(2\theta_i) + A\right] \\ &= \frac{2}{\cos(2\theta_i)}[B\tan(2\theta_i) + A] \\ &= \frac{2}{\cos(2\theta_i)}\left[\frac{B^2}{A} + A\right] \\ &= \frac{2}{\cos(2\theta_i)}\frac{B^2 + A^2}{A} \end{aligned} \tag{21}$$

Thus we need $\text{sign}(\frac{\partial^2}{\partial\theta_i^2}u_i(\hat{v}_i, \hat{v}_{j<i})) = \text{sign}(\cos(2\theta_i))\,\text{sign}(A) < 0$ for θ_i to be a local minimum. If $A < 0$, then we need θ_i^* within $[\frac{-\pi}{4}; \frac{\pi}{4}]$. If $A > 0$, then $\theta_i \in [\frac{-\pi}{2}; \frac{-\pi}{4}] \cup [\frac{\pi}{4}; \frac{-\pi}{2}]$. Finally if $A = 0$, then u_i is maximized for $\theta_i = \frac{\pi}{4}\,\text{sign}(B)$. We reach the different cases stated in the theorem by solving equation 20 for θ_i within the corresponding interval depending on $\text{sign}(A)$.

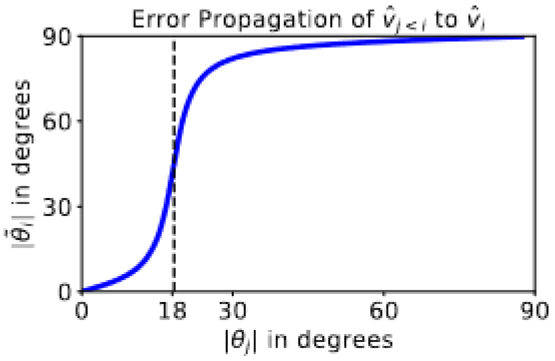


Figure 2: Angular error on the learnt parent $i < j$ (x-axis) vs angular error on the i-th player utility. In this case we see that there is a clear threshold around 18°.

As an application of this result, we display here a graph from an experiment conducted by the author of the original paper. We clearly see in figure 2 that we experience a threshold at some point when the angular error on the parent $i < j$, the angular error on the mis-specified target i significantly drops, meaning that we could use this threshold as an indicator of when the parents are *close enough* from the real eigenvector so we can start learning the next one.

4.3. Convergence of the Eigengame

In this section, we derive the general idea of the proof of convergence of the Eigengame. In order to keep the demonstration concise, we will not prove every result but rather focus on giving the key ideas of the convergence proof. First, recall that from definition 4.3 we denote the eigenvalue gap as $g_i = \Lambda_{ii} - \Lambda_{i+1,i+1}$. We first introduce an important result that is a consequence of the previous section's theorem's to then give a more general result on convergence.

Theorem 4.3. *Assume that $|\theta_j| \leq \frac{c_i g_i}{(i-1)\Lambda_{11}} \leq \sqrt{\frac{1}{2}}$ for all $j < i$ with $0 \leq c_i \leq \frac{1}{16}$ (ie: the parents of i have been learnt accurately to a certain degree). Then:*

$$|\theta_i^*| = |\arg\max_{\theta_i} u_i(\hat{v}_i(\theta_i, \Delta_i), \hat{v}_{i_j})| \leq 8c_i \tag{22}$$

Proof. To prove this result we will assume without explicitly proving it the result **N.11** from the original paper. It gives us in this setup that $A < 0$ for $c_i < \frac{1}{8}$ and that $|\frac{B}{A}| \leq \frac{8c_i}{1-8c_i}$. Recall from theorem 4.2 that as $A < 0$, we get $|\theta_i^*| = \frac{1}{2}\tan^{-1}(|\frac{B}{A}|)$. Also as $\tan(|z|) \leq |z|$ for $|z| \leq 1$, we can derive that:

$$|\theta_i^*| = \frac{1}{2}\tan^{-1}\left(|\frac{B}{A}|\right) \leq \frac{1}{2}|\frac{B}{A}| \leq \frac{1}{2}\frac{c_i}{1-8c_i} \leq 8c_i \tag{23}$$

Theorem 4.4. *Assuming all the eigenvalue gaps are positive $g_i > 0$, let θ_k denote the angular distance of $\hat{v}_k$ from the true eigenvector v_k. Let the maximum desired error for $\theta_k = \theta_{tol} < 1$ radian. Then set $c_k = \frac{\theta_{tol}}{16}$; $\rho_k = \frac{g_k}{2\pi}\theta_{tol}$ and for $i < k$:*

$$\rho_i = \left[\frac{g_i g_{i+1}}{2\pi i \Lambda_{11}}\right] c_{i+1} \tag{24}$$

$$c_i \leq \frac{(i-1)! \prod_{j=i+1}^{k} g_j}{(16\Lambda_{11})^{k-1}(k-1)!} c_k \tag{25}$$

where c_i's are fixed by each $\hat{v}_i$ to its parents and represent a fraction of an error threshold θ_{tol}. For instance, if $\hat{v}_k$ sets $c_k = \frac{1}{16}$, this threshold is communicated up the Graph (see fig 1) to each parents, strengthening the imperatives of precision.
Now, consider learning $\hat{v}_i$ by using algorithm 1 with a step size of $\frac{1}{2L}$ with $L = 4\left[\Lambda_{11}k + (1 + \frac{\Lambda_{11}}{\Lambda_{k-1,k-1}})\frac{g_k}{16}\right]$. Then the top-k principal component will be returned, within the tolerance in angular error θ_{tol}.

Consequences of this theorem are two fold. First it ensures that algorithm 1 will indeed learn the principal component of our problem and second, it gives us parent precision boundaries in order to achieve a requested precision on the child. This clearly is fundamental to estimate the behaviour of the algorithm and to understand the expected precision of player's i when combined with the theorem 4.3. With this two theorems, we can ensure of the convergence of the algorithm and easily bound the expected angular error on the i-th principal component. While this result is a consequence of the previous section, we will not conduct its proof as it would require to introduce several more intermediary lemmas and to carry on some tedious calculations. The core idea of this theorem that proves convergence of algorithm 1 can use for the decentralized algorithm 2 too. We can now easily derive with the theorems 4.2 and 4.3 what *"a good enough approximation of the parents"* could mean depending on our precision requirements on i.
Now that we are assured of the convergence of the algorithm 1, the next convergence related question is naturally *How do we get the guaranteed convergence in t_i iterations ?* This question is answer by the following theorem 4.5 which links angular error and the number of iteration in algorithm 1. We first need to introduce an intermediary result:

Lemma 4.3. *Assume that $\hat{v}_i$ is initialized within $\frac{\pi}{4}$ of its maximizer and its parents are accurate enough at according to theorem 4.3. Let ρ_i be the maximal tolerated error desired for $\hat{v}_i$. Then the Riemannian gradient ascent will return $|\theta_i| \leq \frac{\pi}{g_i}\rho_i + 8c_i$ after at most $\lceil \frac{5}{4}\frac{1}{\rho_i^2} \rceil$ iterations.*

Proof. Although we will not formally prove this result, we can use a common result form Riemannian optimization theory give a core idea of the proof. In this framework, the theory dictates that $\frac{1}{\rho_i^2}$ iterations are required to meet a $\mathcal{O}(\rho_\rangle)$ error threshold. This explains the presence of the $\frac{1}{\rho_i^2}$ factor. Recall that ρ_i was given an explicit value depending on θ_{tol} in theorem 4.4. We here add the interpretation of ρ_i as the maximum desired error for $\hat{v}_i$.

Theorem 4.5. *Applying the theorem 4.4 with the same assumptions, we get with a probability:*

$$\mathbb{P}\left(|\theta_i^0 - \theta_i^*| \leq \frac{\pi}{4}\right) = I_{\frac{1}{2}}\left(\frac{d-1}{2}, \frac{1}{2}\right) \tag{26}$$

with I being the normalized incomplete beta function, the max total number of iteration required for learning all the vectors to adequate accuracy is

$$T_k = \left\lceil \mathcal{O}\left(k\left[\frac{16\Lambda_{11}^k(k-1)!}{\prod_{j=1}^k g_j}\frac{1}{\theta_{tol}}\right]^2\right)\right\rceil \tag{27}$$

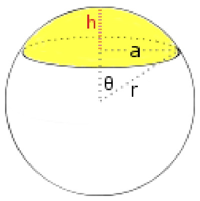
Proof. Assume that $\hat{v}_i^0$ is sampled uniformly in the unit sphere. We will first compute the probability for $\hat{v}_i$ to be within $\frac{\pi}{4}$ of the maximizer of $u_i(\hat{v}_i, \hat{v}_{i<j})$, *ie:* $P(|\theta_i^0 - \theta_i^*|)$. Then we will derive an upper bound n the number of iterations required to return all $\hat{v}_i$ within an angular error margin of θ_{tol}.

The probability of sampling $\hat{v}_i^0$ at an angular deviation within $\frac{\pi}{4}$ of the maximizer (*ie:* v_i if the target is correctly specified) is given by twice the probability of sampling a unit vector from the spherical cap with summit v_i or $-v_i$. This probability is given by:

$$\mathbb{P}(|\theta_i^0 - \theta_i^*| \leq \phi) = \frac{\beta(\sin^2(\phi), \frac{d-1}{2}, \frac{1}{2})}{\beta(1, \frac{d-1}{2}, \frac{1}{2})} = I_{\sin^2(\phi)}(\frac{d-1}{2}, \frac{1}{2}) \tag{28}$$

Using the case of $\phi = \frac{\pi}{4}$, this proves the first point as $\sin^2(\phi) = \frac{1}{2}$.

Figure 3: An illustration of a sphere cap in dimension 3. The summit would be here v_i and the angle $|\theta| \leq \frac{\pi}{4}$. We want to draw a vector on the yellow surface or its symmetrical cap with $-v_i$ as summit.



Using the case of $\phi = \frac{\pi}{4}$, this proves the first point as $\sin^2(\phi) = \frac{1}{2}$. We now prove the upper bounder T_k. Plugging the upper bound on c_i from theorem 4.4 into the bound on iterations from lemma 4.3, and unpacking ρ_i we get:

$$\begin{aligned}
t_i &= \lceil 5(\frac{\pi i \Lambda_{11}}{g_i g_{i+1}})^2 \frac{1}{c_{i+1}^2} \rceil \\
&= \left\lceil 5(\frac{\pi i \Lambda_{11}}{g_i g_{i+1}})^2 \frac{(16\Lambda_{11})^{2(k-i-1)}((k-1)!)^2}{(i!)^2 \prod_{j=i+2}^k g_j^2} \frac{1}{c_k^2} \right\rceil \\
&= \left\lceil 5\pi^2 \frac{16^{2(k-i)}\Lambda_{11}^{2(k-i)}((k-1)!)^2}{((i-1)!)^2 \prod_{j=i}^k g_j^2} \frac{1}{(16c_k)^2} \right\rceil \\
&\leq \left\lceil 5\pi^2 \left[\frac{(16\Lambda_{11})^{k-1}(k-1)!}{\prod_{j=1}^k g_j} \frac{1}{16c_k} \right]^2 \right\rceil \text{ as we have for all } i\text{: } \Lambda_{11} \geq g_i \\
&= \left\lceil \mathcal{O}\left(\left[\frac{(16\Lambda_{11})^{k-1}(k-1)!}{\prod_{j=1}^k g_j} \frac{1}{16c_k} \right]^2 \right) \right\rceil
\end{aligned} \tag{29}$$

We end up with a form of t_i independent of i. Applying the Jensen's inequality to the log of t_k and t_i, it can be shown that $t_k \leq t_1$. The total iterations required for learning $\hat{v}_{i<k}$ is at most $k-1$ times this bound. This gives finally gives us:

$$T_k = \lceil \mathcal{O}(k \left[\frac{16\Lambda_{11}^k (k-1)!}{\prod_{j=1}^k g_j} \frac{1}{\theta_{tol}} \right]^2)\rceil \tag{30}$$

This theorem gives us a very powerful result. Assuming that all $\hat{v}_i$ are by chance initialized within $\frac{\pi}{4}$ of their objective, we are assure to reach the maximum within a finite number of steps. From the error propagation analysis, we get that essentially, each parent vector must be learnt under a $\frac{1}{16}$ canonical error threshold for the child. Moreover, we can interpret the ratio $\frac{\Lambda_{11}}{g_i}$ to say that the closer the eigenvectors are (relatively to the scale of the spectrum), the more iteration we need to find them. Finally, the $(k-1)!$ term would mean that learning a small $\hat{v}_i$ requires more accurate parents in the DAG.

5. Experiments

We now present some experiment we conducted to test the performances of the Eigengame. After implementing algorithm 1 and several other comparable algorithms (see section 6), we conducted two batch of experiments. First a series of tests using the MNIST dataset to test the game under real conditions. For this purpose, we use the principal component calculated through *Numpy*'s SVD as a reference for the true eigenvectors. The Eigengame performances are compared to other PCA algorithms such as Oja, GHA and Krasulina briefly detailed in section 6. The second testing campaign involve synthetic data we known eigenvalues and eigenvectors so we can experiment on the influence of the angular error in the initialization.

5.1. MNIST Dataset

The MNIST Dataset is a well known dataset in data science, it consists of 70000 28× 28 black and white images from handwritten digits ranging from 0 to 9. We trained our models on 60000 flattened images, *ie:* vectors of size 784 and tested on the remaining 10000. Firstly we conducted a simple experiement, learning the top 3 principal component using different vector and projecting the test set in a the usual 3D-space and coloring them according the label in order to visualize the remaining variance, see if some clusters appear qualitatively. We see on figures 4 and 5 that the Eigengame decomposition is actually very similar to the *original* one. Despite going from 784 to 3 dimension, we can still distinguish several clusters depending on the label of the image. Even if PCA is not the best algorithm for such kind of clusterization, it still show that the learnt principal components retain some variance.
Secondly, we re-run the different algorithm to identify this time the top 50 principal components. After ordering accordingly the Eigenvector for the methods which do not enforce the ordering, we compare the cumulative squared error and angular error, still using the decomposition from *Numpy* as a reference. We conducted the experiment twice first with a mini batch size of 1000, and then 10000. Figures 7 to 12 present those results.

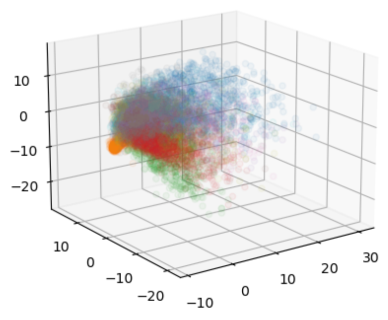


Figure 4: Eigengame

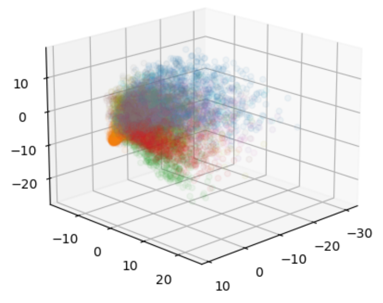


Figure 5: Reference Eigenvectors

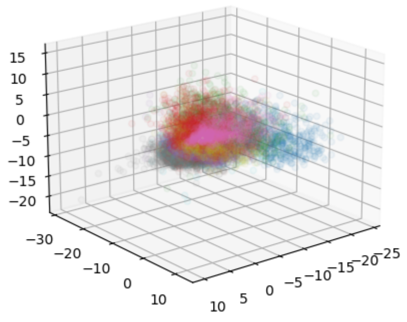


Figure 6: Krasulina

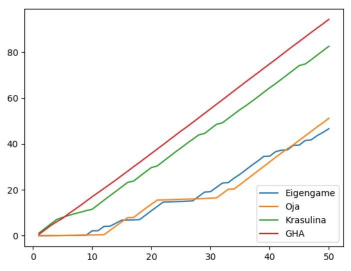


Figure 7: Cumulative Squared Error Loss, batch size of 1000

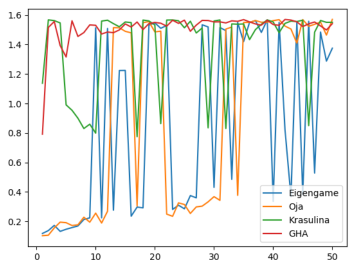


Figure 8: Angular Error, batch size of 1000

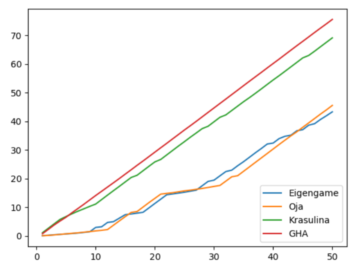


Figure 9: Cumulative Angular Error, batch size of 1000

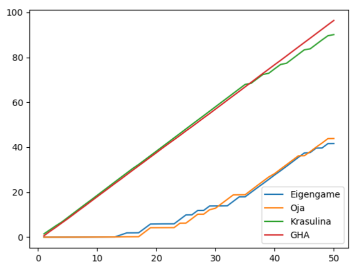


Figure 10: Cumulative Squared Error Loss, batch size of 10000

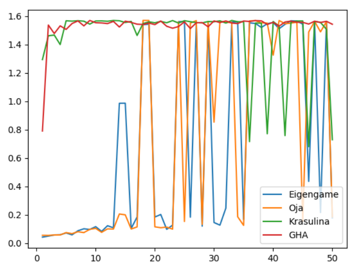


Figure 11: Angular Error, batch size of 10000

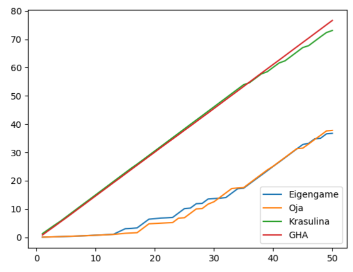


Figure 12: Cumulative Angular Error, batch size of 10000

Interestingly, we see that all the algorithm have mirror performances in terms of square error and angular error. This could be interpreted as all the coordinates are learnt the same, there is no privileged direction.

5.2. Synthetic data

For this part of the experiment, we generate our own dataset from given eigenvectors. This will allow us to study the impact of initialization, as we know in advance the exact real eigenvectors. We generated our data using the following procedure. First we sample a random matrix U of shape $d \times d$, using the Gram Schmidt ortho-normalization process, we get a unit matrix with perpendicular vector which we denote $U_\perp$. This matrix will be our eigenvectors. We then sample and order a list of positive eigenvalues $\Lambda = NN$, thus we can generate a covariance matrix $\Sigma = U_\perp^T \Lambda U_\perp$. We now need to generate samples following exactly this covariance matrix Σ. We first sample Z of shape $n \times d$ with the standard Gaussian and re-center it with $Z^0 = Z - \bar{Z}$, then we write $C = (Z^0)^T Z^0 = LL^H$ with C the covariance matrix of generated Z^* and L the Cholesky decomposition of C. We can then compute $Z^* = Z^0 L$, which now has a covariance matrix of exactly I_d. Finally we can generate our data D with $D = Z^* U_\perp N$. Thus our generated data D will have a predetermined characteristics Σ, $U_\perp$, Λ. Figures 13 to 15 present the cumulative square loss, angular loss and cumulative angular loss depending on the feature number, just like on the previous section with MNIST. It seems that the impact of the initialisation is minimal, even if being within $\frac{\pi}{4}$ of the real eigenvectors is a benefical.

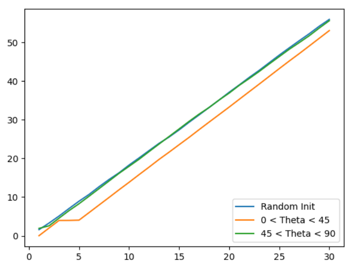


Figure 13: Cumulative Squared Error Loss, batch size of 2000

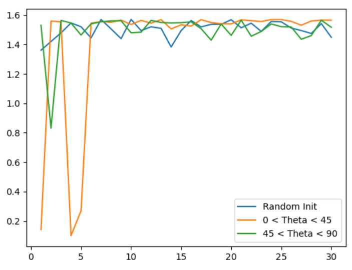


Figure 14: Angular Error, batch size of 2000

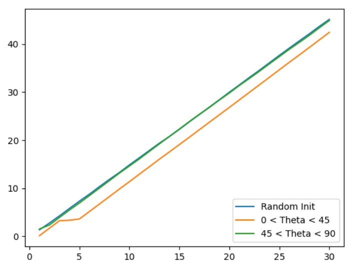


Figure 15: Cumulative Angular Error, batch size of 2000

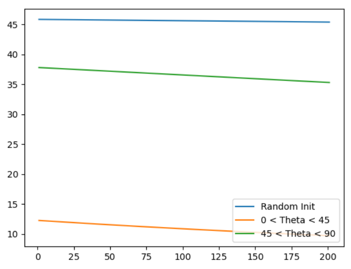


Figure 16: Average angular error depending on the iteration number

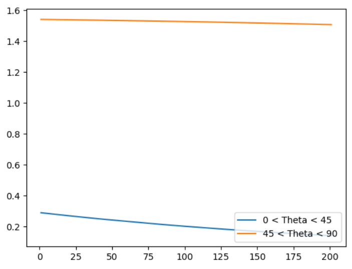


Figure 17: Angular Error for feature 8 depending on the iteration number

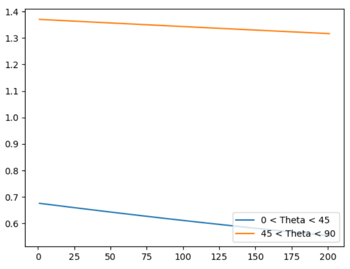


Figure 18: Angular Error for feature 17 depending on the iteration number

Figure 16 present the average angular error of all eigenvectors learnt depending on the iteration for different initialisation. As expected, the initialisation within $\frac{\pi}{4}$ of the real eigenvectors is much lower. However interestingly, while all the curves seem linear, the initialisation within $\frac{\pi}{4}$ has a better slope than than the initialisation between $\frac{\pi}{4}$ and $\frac{\pi}{2}$. Figures 17 and 18 display the same curve than 16 but for specific vectors, without the random initialisation which is irrelevant for one specific vector.

6. Related Work

Oja's algorithm is a very simple estimation using a unique output neuron, performing a linear combination of the input with a weight vector w [9]. In Oja's algorithm, the update method for the weights at iteration n is derived as follow: $\Delta w = w_{n+1} - w_n = \eta y_n(x_n - y_n w_n)$. With y_n being the output, x_n the input and η the learning rate. For a PCA application on a matrix M, the update of an estimated eigenvector v would be: $v_{n+1} - v_n = \eta(I - v^T v)Mv$. Optimality of the Oja's method is discussed in [7]

Generalized Hebbian Algorithm (GHA) is a variant of Oja's algorithm. Very similar in the idea, when applied to an eigenvector search use case for a given matrix M, a candidate v would be update as follows: $v_{n+1} - v_n = \eta M v$. This method was first introduced in 2005 in [6].

Krasulina is another variant of Oja's algorithm, this time by adding a normalization step [2]. This time the transformation $\frac{v}{||v||}$ is used to *retract* the vector v on the sphere. By comparison, the transformation $(I - v^T v)$ project the vector on the tangent space of the sphere.

EigenGame as first introduced in [4] is now called the α-EigenGame. Indeed the authors released a second paper in 2021 detailing a new variant called μ-Eigengame [5]. In this new version, the authors come up with some significant improvement such as an unbiased utility gradient estimation, better parallelism and better performances in the experiments.

Conclusion

In this work, authors proposed a change of perspective for principal component analysis, moving from a centralized variance maximization problem to a multiplayer game. This allows for a decentralized and as such parallel learning of the principal components of dataset. This also leads to an improve scale ability of the algorithm to larger datasets. In the original paper, the authors were able to conduct an analysis on ImageNet with $d = 2300$, which demonstrates the opportunities for larger scale analysis. Moreover, the author forecast that this reformulation of the PCA could open other research direction based on this work, such as federated learning, online bandit, reinforcement learning or GANs for instance. Finally as mentioned in the previous section, authors have recently released a new version of the EigenGame with several improvements in performances.
In this paper, we summarized the main results of the original EigenGame paper from 2020 and tried to organise them in a clear and didactic way. We also conducted some original experiences based on our implementation of the algorithms. While we could not test several claims such a parallel learning or ImageNet/VGG results due to a lack of resources, we hope that the modest experiment conducted will help to grasp the extent of the results from [4]. Please note that this work holds no claim of novelty and is purely academic, all credit goes to the original authors.

References

[1] Essentials of Game Theory: A Concise, Multidisciplinary Introduction, Dec 2021.

[2] T. P. Krasulina, "A method of stochastic approximation for the determination of the least eigenvalue of a symmetric matrix", Zh. Vychisl. Mat. Mat. Fiz., 9:6 (1969), 1383–1387; U.S.S.R. Comput. Math. Math. Phys., 9:6 (1969), 189–195, Dec 2021.

[3] Contributors to Wikimedia projects. Principal component analysis - Wikipedia, Dec 2021.

[4] Ian Gemp, Brian McWilliams, Claire Vernade, and Thore Graepel. EigenGame: PCA as a Nash Equilibrium. *arXiv*, Oct 2020.

[5] Ian Gemp, Brian McWilliams, Claire Vernade, and Thore Graepel. EigenGame Unloaded: When playing games is better than optimizing. *arXiv*, Feb 2021.

[6] D. O. Hebb. *The Organization of Behavior: A Neuropsychological Theory.* Psychology Press, London, England, UK, Apr 2005.

[7] Xin Liang. On the Optimality of the Oja's Algorithm for Online PCA. *arXiv*, Mar 2021.

[8] Sidharth Mishra, Uttam Sarkar, Subhash Taraphder, Sanjoy Datta, and Menalsh Laishram. Principal Component Analysis. *International Journal of Livestock Research*, page 1, Jan 2017.

[9] Erkki Oja. Simplified neuron model as a principal component analyzer. *J. Math. Biol.*, 15(3):267–273, Nov 1982.