Deep Learning Project on the paper "EigenGame : PCA as a Nash Equilibrium"

Juraj Rosinsky

1 Introduction

The goal of this project is to reimplement the methods used in the paper "EigenGame : PCA as a Nash Equilibrium". In this work, I will explain in details the methods of the paper in more details.

2 PCA

What is the goal of PCA ? Some datasets are just too large to be analyzed. PCA is a method that extracts the maximum of data in a much smaller dataset. This is done by finding the directions of the dataset with maximum variance, these directions will have the most information. Non-orthogonal axis have redundant information, so to prevent that, we will create a base of orthogonal vectors. It is not necessary to find all Principal Components of a dataset, the first k-Principal are sufficient to keep most of the non-redundant data.

In practice, we have a dataset $X \in \mathbb{R}^{n\times d}$ (n vectors of dimension d) and we create the covariance matrix $M := X^T X$ and the principal components of X are the eigenvectors of M. Finding these eigenvectors becomes tricky when the d is large.

3 Game Theory and Nash Equilibrium

In this section $\hat{v}$ and $\hat{V}$ are approximations of the true eigenvectors v and V.

How does Game Theory help ? Today's algorithms that where used to solve PCA (find the Principal Components) where searching them all at once (all k at once). The authors of the paper found a new approach. They defined a multiplayer game where the players are vectors and their goal is to find the eigenvectors of the matrix $M = X^T X$, so the matrix of eigenvectors V such that $V^T V = I$ and $V^T M V = \Lambda$, where Λ is the diagonal matrix containing all the eigenvalues.

For a guess $\hat{V}$ of the matrix V, we define $R(\hat{V}) = \hat{V}^T M \hat{V}$. Our goal is that $R(\hat{V})$ is a diagonal matrix. We want to maximize the trace of R (equivalent to minimizing reconstruction error) :

$$\underset{\hat{V}^T\hat{V}=I}{max} \left\{ \sum_i R_{ii} = Tr(R) = TR\left(\hat{V}^T M \hat{V}\right) = Tr\left(\hat{V}\hat{V}^T M\right) = Tr(R) \right\} \tag{1}$$

Only maximize the trace of R is independent of $\hat{V}$, so this function doesn't help to recover the eigenvectors. We need to look again at the equation $V^T M V = \Lambda$ and we can see that to recover the eigenvectors of M, Λ needs to be diagonal, and so does R. To ensure this, we need to minimize $\sum\limits_{i\neq j} R_{ij}^2$.

We can combine these maximizing and minimizing parts of R to create the utility function that we will use :

$$u_i(\hat{v_i} \,|\, \hat{v_{j<i}}) = \sum_i R_{ii} - \sum_{i \neq j} R_{ij}^2 \tag{2}$$

Here is where the Game Implementation comes into place : players are playing one be one. So the first player finds the eigenvector with the biggest eigenvalue, so the biggest Principal Component. After this, the second player will find the second biggest eigenvector knowing the first one. So it has to be orthogonal to the first one, it is the second biggest Principal Component. The second eigenvector knows that it has be orthogonal to the first two and it gives us the third biggest principal component, etc.

So each player will have a different utility function because they will different amount of information to take into account. The utility function of the first player is : $\underset{\hat{v_1}^T\hat{v_1}=1}{max} \langle \hat{v_1}, M\hat{v_1} \rangle$. But for the second player, it's utility function will be $\underset{\hat{v_2}^T\hat{v_2}=1, \hat{v_2}^T\hat{v_1}=0}{max} \langle \hat{v_2}, M\hat{v_2} \rangle - \frac{\langle \hat{v_2}, M\hat{v_1} \rangle^2}{\langle \hat{v_1}, M\hat{v_1} \rangle}$ and in general, the utility function of the player i is :

$$\underset{\hat{v_i}^T\hat{v_i}=1}{max} \left\{ \langle \hat{v_i}, M\hat{v_i} \rangle - \sum_{j<i} \frac{\langle \hat{v_i}, M\hat{v_j} \rangle^2}{\langle \hat{v_j}, M\hat{v_j} \rangle} \right\} \tag{3}$$

We can see that we divide $\langle \hat{v_i}, M\hat{v_j} \rangle$ by $\langle \hat{v_j}, M\hat{v_j} \rangle$. It puts the positive term and the negative term on the same scale.

The first algorithm is a sequential algorithm that computes the eigenvectors v_i in order. It uses the gradient of the utility function to find the direction that maximizes the utility function :

$$\nabla_{\hat{v_i}}(\hat{v_i} \,|\, \hat{v_{j<i}}) = 2M \underbrace{\left[\hat{v_i} - \sum_{j<i} \frac{\hat{v_i}^T M\hat{v_j}}{\hat{v_j}^T M\hat{v_j}} \hat{v_j} \right.}_{\text{Generalized Gram-Schmidt product}} = 2X^T \left[X\hat{v_i} - \sum_{j<i} \frac{\langle X\hat{v_i}, X\hat{v_j} \rangle}{\langle X\hat{v_j}, X\hat{v_j}} X\hat{v_j} \right] \tag{4}$$

The scaling that we had in equation

4 Implementation

5 Results