PCA as Gradient Descent

Andrew Draganov

February 8, 2022

1 PCA objective function and gradient

Given a high-dimensional dataset $X \in \mathbb{R}^{N \times D}$, we aim to identify a gradient descent objective to find points $Y \in \mathbb{R}^{N \times d}$ with $d << D$ such that Y is the result of performing PCA on X.

We start with formalizing PCA as minimizing the function

$$f_{PCA}(X, Y) = ||L(E^X - E^Y)L||_F^2 \tag{1}$$

where $E^X \in \mathbb{R}^{N \times N}$ and $E^Y \in \mathbb{R}^{N \times N}$ are the pairwise squared distance matrix for X and Y and L is the $N \times N$ centering matrix. Optimizing this function inherently amounts to minimizing the Frobenius norm between the two matrices. We show two separate ways to calculate the gradient of f_{PCA} with respect to the low-dimensional points Y.

1.1 Element-wise gradient calculations

To identify the effect of gradient descent on Y, we can deconstruct the matrices into a set of element-wise operations. We can re-arrange the term in the Frobenius norm:

$$\begin{aligned} L(E^X - E^Y)L &= LE^X L - LE^Y L \\ &= E^X - \bar{X}^{\rightarrow} - \bar{X}^{\downarrow} - E^Y + \bar{Y}^{\rightarrow} + \bar{Y}^{\downarrow} \end{aligned}$$

where $\bar{X}^{\rightarrow}$ is the $N \times N$ matrix of row-means that the centering matrix on the right-hand-side of X subtracts. The other $\bar{X}$ and $\bar{Y}$ variables are defined accordingly. We can now rearrange these to obtain:

$$L(E^X - E^Y)L = E^X - E^Y - \Lambda$$

for $\Lambda = -\bar{X}^{\rightarrow} - \bar{X}^{\downarrow} + \bar{Y}^{\rightarrow} + \bar{Y}^{\downarrow} \in \mathbb{R}^{N \times N}$

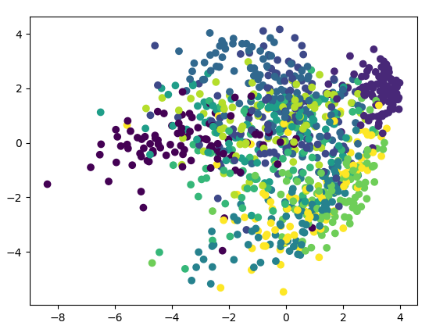

Figure 1: The result of performing gradient descent PCA on the MNIST dataset

Then the PCA objective function can equivalently be written as

$$f_{PCA}(X,Y) = \sum_{i,j} \left(||x_i - x_j||^2 - ||y_i - y_j||^2 - \lambda_{ij}\right)^2 \tag{2}$$

To perform gradient descent, we take the partial derivative with respect to y_i to obtain

$$\nabla_{y_i} f(X,Y) = -4 \sum_j (||x_i - x_j||^2 - ||y_i - y_j||^2 - \lambda_{ij})(\frac{\partial \vec{\lambda}_{ij}}{\partial y_i} + \vec{y}_i - \vec{y}_j)$$

This gives us the following gradient descent algorithm with learning rate ν:

$$\boxed{\begin{aligned} \nabla_{y_i} f(X,Y) &= -4 \sum_j (||x_i - x_j||^2 - ||y_i - y_j||^2 - \lambda_{ij}) \left(\frac{\partial \vec{\lambda}_{ij}}{\partial y_i} + \vec{y}_i - \vec{y}_j\right) \\ Y_{t+1} &= Y_t + \nu \nabla_Y f(X,Y) \end{aligned}}$$

This gives us the following algorithm:

Algorithm 1 PCA by gradient descent on the points

Require: Input: $X \in \mathbb{R}^{N \times D}$, n_epochs, ν
  $D^X \leftarrow$ pairwise_dists$(X) \in \mathbb{R}^{N \times N}$
  $D^X \leftarrow D^X -$ row_means(D^X)
  $D^X \leftarrow D^X -$ col_means(D^X)
  $Y \leftarrow \mathcal{N}_{(0,1)} \in \mathbb{R}^{N \times d}$
  while $e <$ n_epochs **do**
    $V^Y \leftarrow$ pairwise_vectors$(Y) \in \mathbb{R}^{N \times N \times 2}$

    $D^Y \leftarrow$ pairwise_dists$(Y) \in \mathbb{R}^{N \times N}$
    $D^Y \leftarrow D^Y -$ row_means(D^Y)
    $D^Y \leftarrow D^Y -$ col_means(D^Y)

    $\nabla_Y \leftarrow -4 \cdot \text{SUM}\left((D^X - D^Y)V^Y, \text{ axis} = 1\right)$
    $Y \leftarrow Y + \nu \nabla_Y$
    $e{+}{+}$
  end while
  return Y

Lastly, we note that we can separate the PCA gradient descent algorithm into a pair of alternating forces. For $\Lambda^X = \bar{X}^{\rightarrow} + \bar{X}^{\downarrow}$ and $\Lambda^Y = -\bar{Y}^{\rightarrow} - \bar{Y}^{\downarrow}$ Namely, we have

$$\begin{aligned} \nabla_{y_i} f(X,Y) &= -4 \sum_{i,j} (||x_i - x_j||^2 - ||y_i - y_j||^2 - \lambda_{ij})(\vec{y}_i - \vec{y}_j) \\ &= 4 \sum_{i,j} (||x_i - x_j||^2 - \lambda_{ij}^X)(\vec{y}_i - \vec{y}_j) - 4 \sum_{i,j} (||y_i - y_j||^2 - \lambda_{ij}^Y)(\vec{y}_i - \vec{y}_j) \\ &= 4\left(F_X - F_Y\right) \end{aligned}$$

Each force is dominated by the large and small distances. However, we can deconstruct these into clear attractive and repulsive forces. If points x_i and x_j are one another's nearest neighbors in the high-dimensional space, then $||x_i - x_j||^2$ will always be less than its respective centering λ_{ij}^X. Thus, nearest neighbors in the high-dimensional space *only* exert attractions on the embedding. Similarly, farthest neighbors in the high-dimensional space *only* create repulsive forces. Inversely, nearest neighbors in the embedding exert repulsions while farthest neighbors exert attractions. PCA is the single convergence of these individual systems. Therefore, as one progresses from x_i's nearest to its farthest neighbor, the force must this neighbor exerts on x_i must change monotonically.

1.2 Vectorized PCA gradient

A different approach would be to rely on matrix differentiation using vectorizations and Kronecker products. We first establish the following preliminaries:

1. Given Gram matrix $G_X = X^T X$, we can write E^X as $E^X = \text{diag}(G_X)\mathbb{1} + \mathbb{1}\text{diag}(G_X)^T - 2G$ as the matrix of pairwise squared distances
   - We simplify notation by writing $\text{diag}(G_X) = g_x$, giving us
   $$E^X = g_X \mathbb{1} + \mathbb{1} g_X^T - 2G_X$$
   - $\mathbb{1}$ is the vector consisting of all 1's
2. The product of three matrices ABC can be *vectorized* (transformed into a column vector by linear transformation) using the operation $vec(ABC) = (C^T \otimes A)vec(B)$, where $\otimes$ represents the Kronecker product.
3. Vectorized matrices are transposed by appropriately sized *commutation matrices* K s.t. for any matrix $A \in \mathbb{R}^{n\times m}$ it is the case that $K^{(m\times n)}vec(A) = vec(A^T)$, where $K \in \mathbb{R}^{mn\times mn}$

We begin with the error function for PCA

$$f_{PCA}(X,Y) = ||L(E^X - E^Y)L||_F^2$$

By applying the first preliminary, we rearrange this by

$$\begin{aligned} f_{PCA}(X,Y) &= ||L(E^X - E^Y)L||_F^2 \\ f_{PCA}(X,Y) &= ||L(g_X\mathbb{1} + \mathbb{1}g_X^T - 2G_X - g_Y\mathbb{1} - \mathbb{1}g_Y^T + 2G_Y)L||_F^2 \end{aligned}$$

But now notice that $L\mathbb{1} = 0 = \mathbb{1}^T L$. So we can simplify this expression into

$$f_{PCA}(X,Y) = ||L(2G_Y - 2G_X)L||_F^2$$

The squared Frobenius norm $||M||_F^2$ can be rewritten as $tr(M^T M)$. Applying this gives us

$$\begin{aligned} f_{PCA}(X,Y) &= tr\left((L(2G_Y - 2G_X)L)^T L(2G_Y - 2G_X)L\right) \\ &= 4tr\left(L(G_Y - G_X)^T LL(G_Y - G_X)L\right) \\ &= 4tr\left(L(G_Y^T - G_X^T)L(G_Y - G_X)L\right) \\ &= 4tr\left(L(G_Y^T LG_Y - G_X^T LG_Y - G_Y^T LG_X + G_X^T LG_X)L\right) \end{aligned}$$

We now separate this by the linearity of the trace operation to obtain

$$f_{PCA}(X,Y) = 4\left[tr\left(LG_Y^T LG_Y L\right) - tr\left(LG_X^T LG_Y L\right) - tr\left(LG_Y^T LG_X L\right) + tr\left(LG_X^T LG_X L\right)\right]$$

The product is commutative within the trace operation, so we can cancel the outer centering matrices to obtain

$$\begin{aligned} f_{PCA}(X,Y) &= 4\left[tr\left(G_Y^T LG_Y\right) - tr\left(G_X^T LG_Y\right) - tr\left(G_Y^T LG_X\right) + tr\left(G_X^T LG_X\right)\right] \\ f_{PCA}(X,Y) &= 4tr\left(G_Y^T LG_Y - G_X^T LG_Y - G_Y^T LG_X + G_X^T LG_X\right) \\ &\textit{By further commutative swaps and removing the transposes, we get} \rightarrow \\ \rightarrow f_{PCA}(X,Y) &= 4tr\left(L\left(G_Y G_Y - 2G_X G_Y + G_X G_X\right)\right) \\ &= 4tr\left(L\left(G_Y - G_X\right)^2\right) \end{aligned}$$

We now incorporate the fact that the trace of a product of symmetric matrices is the inner product of the vectorizations of those matrices

$$\rightarrow f_{PCA}(X,Y) = 4\langle \text{vec}((G_Y - G_X)^2), \text{vec}(L)\rangle$$

Taking the partial derivative with respect to Y gives:

$$\begin{aligned}
d(f_{PCA}(X,Y)) &= 4\text{vec}(L)^T \text{vec}\left(d\left((G_Y - G_X)^2\right)\right) \\
&= 4\text{vec}(L)^T \text{vec}\left(d\left(G_Y^2\right) - d\left(G_Y G_X\right) - d\left(G_X G_Y\right)\right) \\
&= 4\text{vec}(L)^T \text{vec}\left(d\left(G_Y^2\right) - 2G_X\right) \\
&= 4\text{vec}(L)^T \text{vec}\left(2G_Y d\left(G_Y\right) - 2G_X\right) \\
&= 8\text{vec}(L)^T \text{vec}\left(G_Y(dY^T Y + Y^T dY) - G_X\right) \\
&= 8\text{vec}(L)^T \text{vec}\left(G_Y(dY^T Y + Y^T dY)\right) - 8\text{vec}(L)^T \text{vec}\left(2G_X\right) \\
&= 8\text{vec}(L)^T \left(\text{vec}\left(G_Y dY^T Y\right) + \text{vec}\left(G_Y Y^T dY\right)\right) - 4\text{vec}(L)^T \text{vec}\left(2G_X\right)
\end{aligned}$$

We now extract the dY vectors by performing the $\text{vec}(ABC) = (C^T \otimes A)\text{vec}(B)$ trick:

$$\begin{aligned}
df_{PCA}(X,Y) &= 8\text{vec}(L)^T \left((Y^T \otimes G_Y)K dY + (I \otimes G_Y Y^T)dY\right) - 4\text{vec}(L)^T \text{vec}\left(2G_X\right) \\
&= 8\text{vec}(L)^T \left[\left((Y^T \otimes G_Y)K + (I \otimes G_Y Y^T)\right) dY\right] - 4\text{vec}(L)^T \text{vec}\left(2G_X\right)
\end{aligned}$$