1 Principal Component Analysis

Toy Example

Let's imagine we have a large dataset of noisy, redundant, and intuitively intractable data. We **know** that this data should have some inherent meaning, but we just don't know it. This is crucial in big data applications where each data point (e.g. data of a Facebook user) consists of hundreds or thousands or attributes, and you want to find trends in this data.

As a motivating example, let's say you are a physicist attempting to measure the motion of an oscillating mass on a spring. You know the mass is oscillating in a particular direction, you just don't know what direction that is, and you want to figure it out. You set up 3 cameras to view what is going on.

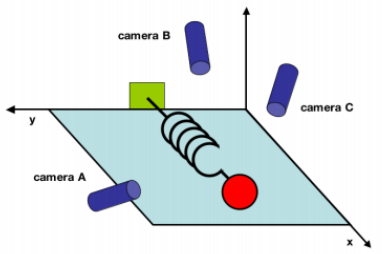


Figure 1: A diagram of the toy example.

Each camera is able to measure the x and y distance on the screen of the mass/spring system. If you had known the direction of the oscillations in the first place, you would only need one value at a given time point, but you have 6 now. You put all of this data into a large matrix of values representing where the spring is at each time point, and you want to do some linear algebra magic to figure out what direction the mass was moving, and how it was moving as a function of x, its linear distance. What kind of linear algebra magic can we do to make this happen? (*Hint:* PCA!)

$$A = \begin{bmatrix} x_{11} & y_{11} & x_{12} & y_{12} & x_{13} & y_{13} \\ x_{21} & y_{21} & x_{22} & y_{22} & x_{23} & y_{23} \\ x_{31} & y_{31} & x_{32} & y_{32} & x_{33} & y_{33} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n1} & y_{n1} & x_{n2} & y_{n2} & x_{n3} & y_{n3} \end{bmatrix}$$

Figure 1: Matrix representing various x and y values. Each row represents 1 time point, with the x and y points of the ball for each camera. Each column represents one attribute recorded over all time steps.

Computing PCA

First let's look at how to find the PCA. There are two major steps in this process: finding the covariance matrix S and diagonalizing it. Eigenvectors of S are exactly the principal components that we need.

Computing the covariance matrix: We start by centering the data matrix A (i.e. for each attribute, we find the average value and subtract it from the column corresponding to the attribute). After this procedure, we would expect the mean of every column to be 0. Centering the data is important because we don't want our results to be affected by simple offsets in data (e.g. one of the cameras adding a constant 1 to all its values). Formally, this can be written as: $\tilde{A} = A - \frac{1}{n}\vec{1}\vec{1}^T A$. Once we have $\tilde{A}$, we can compute $S = \frac{1}{n}\tilde{A}^T\tilde{A}$.

The covariance matrix S has size (attributes $\times$ attributes), and it represents how correlated the attributes are with each other. If each attribute were completely independent of each other, we would expect S to be a diagonal matrix.

Diagonalizing S: This step is quite straightforward: We diagonalize $S = P\Lambda P^T$. Note that we usually diagonalize $S = P\Lambda P^{-1}$, but in this case, S is a symmetric matrix, and so we can find a P, such that $P^T = P^{-1}$ by the Spectral Theorem.

Here is the full procedure:

1. Center the matrix along the attributes (columns in this case), so that $\tilde{A} = A - \frac{1}{n}\vec{1}\vec{1}^T A$.
2. Find the covariance matrix $S = \frac{1}{n}\tilde{A}^T\tilde{A}$
3. Diagonalize the covariance matrix: $S = P\Lambda P^T$
4. The columns of P are the principal components.

Result: The result of this process is the matrix P consisting of the principal components. Changing the data into this basis can reveal many useful insights about the data.

2 Understanding PCA

What is this PCA basis, and why is it useful? There are two equivalent ways of understanding PCA:

1. The principal component vectors $\vec{p}_i$ maximize the variance of data when it is projected on the line corresponding to $\vec{p}_i$.
2. The subspace spanned by k PC vectors (i.e. span$\{\vec{p}_1, \ldots, \vec{p}_k\}$) is the best k-dimensional approximation to the given data.

Variance Maximization perspective: Given any unit vector $\vec{u}$, we can compute the projection of the data along the line defined by $\vec{u}$ by computing $\tilde{A}\vec{u}$. Each sample gets mapped to its "distance along the line". Some choices of $\vec{u}$ will have larger spread (variance) of these distances, while others will have lower spread (i.e. the projections are more concentrated). Formally, this variance is given by $\frac{1}{n}\vec{u}^T\tilde{A}^T\tilde{A}\vec{u}$. The main idea behind PCA is to find trends in the data by choosing the $\vec{u}$ that maximizes this variance.

- Therefore, $\vec{p}_1$ is simply the unit vector that maximizes $\vec{u}^T\tilde{A}^T\tilde{A}\vec{u}$.
- $\vec{p}_2$ is the unit vector that maximizes $\vec{u}^T\tilde{A}^T\tilde{A}\vec{u}$ while **also** being orthogonal to $\vec{p}_1$. Note that the possible choices of $\vec{p}_2$ are more restricted due to orthogonality constraint. This constraint ensures that we get an orthonormal basis.
- $\vec{p}_3$ is the unit vector that maximizes $\vec{u}^T\tilde{A}^T\tilde{A}\vec{u}$ while **also** being orthogonal to $\vec{p}_1$ and $\vec{p}_2$.
- $\vec{p}_k$ is the unit vector that maximizes $\vec{u}^T\tilde{A}^T\tilde{A}\vec{u}$ while **also** being orthogonal to $\vec{p}_1, \ldots, \vec{p}_{k-1}$.

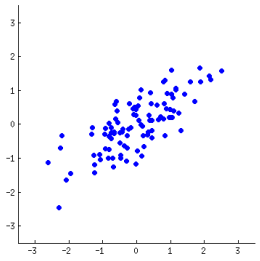
Formally, we can write this as:

$$\vec{p}_k = \underset{u}{\text{maximize}}\ \vec{u}^T\tilde{A}^T\tilde{A}\vec{u}$$

$$\text{subject to } \vec{u}^T\vec{u} = 1, \vec{u}^T\vec{p}_1 = 0, \vec{u}^T\vec{p}_2 = 0, \ldots, \vec{u}^T\vec{p_{k-1}} = 0$$

It turns out that the solutions to this optimization problem are the eigenvectors of $\tilde{A}^T\tilde{A}$, i.e. the eigenvectors of the covariance matrix!

Try it yourself: If the given data looked like the following figure, what would you expect $\vec{p}_1$ and $\vec{p}_2$ to be?



Reconstruction Error Minimization perspective: Let's say we want to compress our high-dimensional data by approximating each sample point $\vec{x}_i$ as a linear combination of a small number of vectors $\vec{v}_1, \ldots, \vec{v}_k$,

That is, $\vec{x}_i \approx \hat{\alpha}_1\vec{v}_1 + \cdots + \alpha_k\vec{v}_k$. Then $\vec{x}_i$ can be (approximately) described by the coefficients $\alpha_1, \ldots, \alpha_k$, and we can just store the α's instead of storing

every attribute of $\vec{x}_i$. These α's can be found by solving a least sqaures problem $V\vec{\alpha} = \vec{x}_i$. The question then is: which vectors $\vec{v}_1, \ldots, \vec{v}_k$ will allow us to make the best reconstructions? These vectors have to minimize the total error in reconstructing the data points $\vec{x}_i$. It can be proven that the first k vectors of the PCA basis $\{\vec{p}_1, \ldots, \vec{p}_k\}$ are the best choice of vectors to use.

This makes PCA an excellent choice of basis when compressing data. Intuitively, the first few vectors in the basis explain most of the variation in the data, and so we can get away with just storing the first k coefficients.

3 PCA and Financial Markets