DIMENSIONALITY REDUCTION

Notes

Matteo Lugli

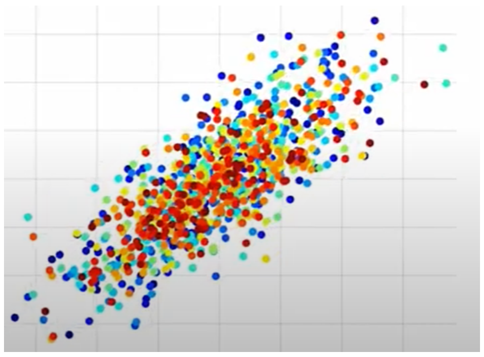
1 Intuition

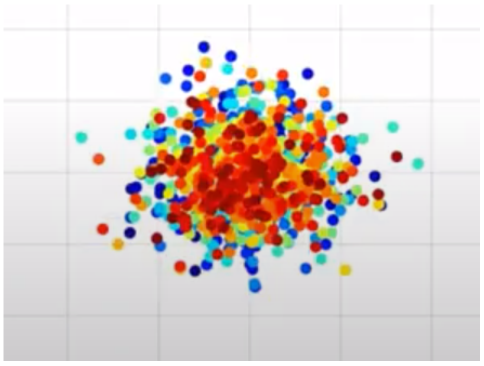
The goal of dimensionality reduction is to shrink the data that needs to be processed later (for example with a classification algorithm) to have better performance in term of computational cost. It is also a way to plot high dimensional data using a 3D visualization. The so called **embedding function** that is used to perform dimensionality reduction can be either linear or non linear. Linear embedding functions can be written in the form $f(x) = w^t x + b$

2 PCA

The PCA algorithm is rather simple:

- Summarize the dataset D in a NxD data matrix (one element for each row);
- center the data, so that we don't need the b parameter anymore. This step is done by subtracting from each row the mean element $\bar{x}$;
- thanks to the centering step we can calculate the covariance matrix by doing $C = x^t x$. Remember that the covariance matrix denotes how the variables interact with each other.
- find the eigenvectors and eigenvalues of C. If we think of the covariance matrix as a linear transformation, the eigenvectors are those vectors that do not change their direction after the transformation. They only change their magnitude. Theese are the candidates for the projection axis. If you apply the covariance matrix of a dataset as a linear transformation, what happens is that all of the vectors get nearer to one of the eigenvectors.
- between all of the eigenvectors, pick the ones with the highest eigenvalues. Theese are the axis on which you get more variance on the projected dataset.


(a) data before transformation

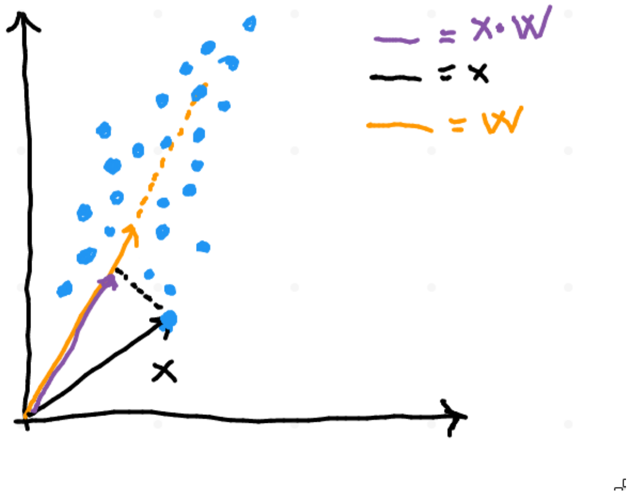

(b) data after transformation

When working with really high dimensional data it's necessary to figure out the dimension of the output space. To to that we check the distribution of the eigenvalues and take the first k eigenvectors that cover around the 90% of the variance.
The PCA algorithm is based on the fact that we consider the data space to be euclidean. Imagine that your points are distributed on a sphere. On such topology computing the

euclidean distance to get the effective distance between two points might not be accurate, as the real distance is represented by an arch. This is why the Laplacian Eigenmap method is taken into account.

2.1 Mathematical demonstration

Let w be the unit vector that coresponds to the projection direction, and $x_i^t \cdot w$ the distance of the projected sample from the origin. In the p dimension space (where p is the new



reduce dimension, let's suppose it's 1 just for simplicity) the actual coordinates of the new point are $(x_i \cdot w^)w$. When we do this sort of projection we create a new representation of our original vectors, so there is always going to be a small error. Let's write the formula for the error:

$$
\begin{aligned}
&||x_i - (w \cdot x_i)w||^2 = (x_i - (w \cdot x_i)w) \cdot (x_i - (w \cdot x_i)w) \\
&= x_i \cdot x_i - x_i \cdot (w \cdot x_i)w - (w \cdot x_i)w \cdot x_i + (w \cdot x_i)w \cdot (w \cdot x_i)w \\
&= ||x_i||^2 - 2(w \cdot x_i)^2 + (w \cdot x_i)^2 w \cdot w \\
&= ||x_i||^2 - (w \cdot x_i)^2
\end{aligned}
$$

since $w \cdot w$ is 1, because we are working with unit vectors. We can use the Mean Squared Error to measure the error considering the whole dataset (so the n vectors):

$$
\begin{aligned}
MSE(w) &= \frac{1}{n} \sum_{i=1}^{n} ||x_i||^2 - (w \cdot x_i)^2 \\
&= \frac{1}{n} (\sum_{i}^{n} ||x_i||^2 - \sum_{i}^{n} (w \cdot x_i)^2) \qquad (1)
\end{aligned}
$$

Considering that we want to minimize the MSE and that $\sum_i^n ||x_i||^2$ is constat, the problem becomes maximizing $\sum_i^n (w \cdot x_i)^2$, which is the sample mean of $(w \cdot x_i)^2$. The mean of a square is the square of the mean plus the variance:

$$\frac{1}{n}\sum_{i}^{n}(w\cdot x_i)^2 = \big(\frac{1}{n}\sum_{i=1}^{n}x_i\cdot w\big)^2 + Var[w\cdot x_i] \tag{2}$$

If we suppose that the original points have mean = 0, then also the projected points have zero mean, so the first part of the equation gets cancelled. This means that we are just left with the variance:

$$\frac{1}{n}\sum_{i}^{n}(w\cdot x_i)^2 = Var[w\cdot x_i] = \sigma^2_{w\cdot x_i} \tag{3}$$

This means that our original problem of minimizing the error made through projecting the points is solved by maximizing the variance, which is exactly what happens in PCA. Let's now see why we need to use the covariance matrix:

$$\begin{aligned}\sigma^2_{w\cdot x_i} &= \frac{1}{n}\sum_{i}^{n}(w\cdot x_i)^2\\ &= \frac{1}{n}(xw)^t(xw) = \frac{1}{n}w^t x^t x w\\ &= w^t\frac{x^t x}{n}w = w^t\Sigma w \end{aligned}\tag{4}$$

where Σ is the covariance matrix of the dataset x. Let's re-write the problem keeping in mind that we are in a constrained scenario:

$$\begin{aligned}\max\quad & f(w) = w^t\Sigma w\\ \text{s.t}\quad & w^t w = 1\end{aligned}\tag{5}$$

To solve our problem now we have to set $\nabla L(w,\lambda) = 0$, meaning that the two components of the gradient need to be null:

$$\begin{aligned}&\frac{dL}{d\lambda} = w^t w - 1\\ &\frac{dL}{dw} = 2\Sigma w - 2\lambda w\\ &\frac{dL}{d\lambda} = 0 \Rightarrow w^t w = 1\\ &\frac{dL}{dw} = 0 \Rightarrow \Sigma w = \lambda w\end{aligned}$$

From the last equation we deduce that w is an **eigenvector** of the covariance matrix Σ! In particular we want the eigenvector with the bigger associated eigenvalue λ. □

3 Laplacian Eigenmap

This method is similar to PCA, but it involves a preliminary step that needs to be performed. We need to compute the Laplacian of the graph associated to the topological structure that is being used.

$$L = D - A \tag{6}$$

where L is the Laplacian, D is the Degree Matrix and A is the Adiacency matrix.

- D is a matrix that counts for each node the number of nodes it is connected to;
- A is simply the adiacency matrix of the graph;

$$L = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 2 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{D}} - \underbrace{\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}}$$

This method though just preserves some properties **locally**, not **globaly**. This method is usefull when working in non euclidean spaces.