

Principal component analysis and singular value decomposition

Oliver W. Layton

CS251: Data analysis and visualization

Lecture 21, Spring 2019

Monday April 1

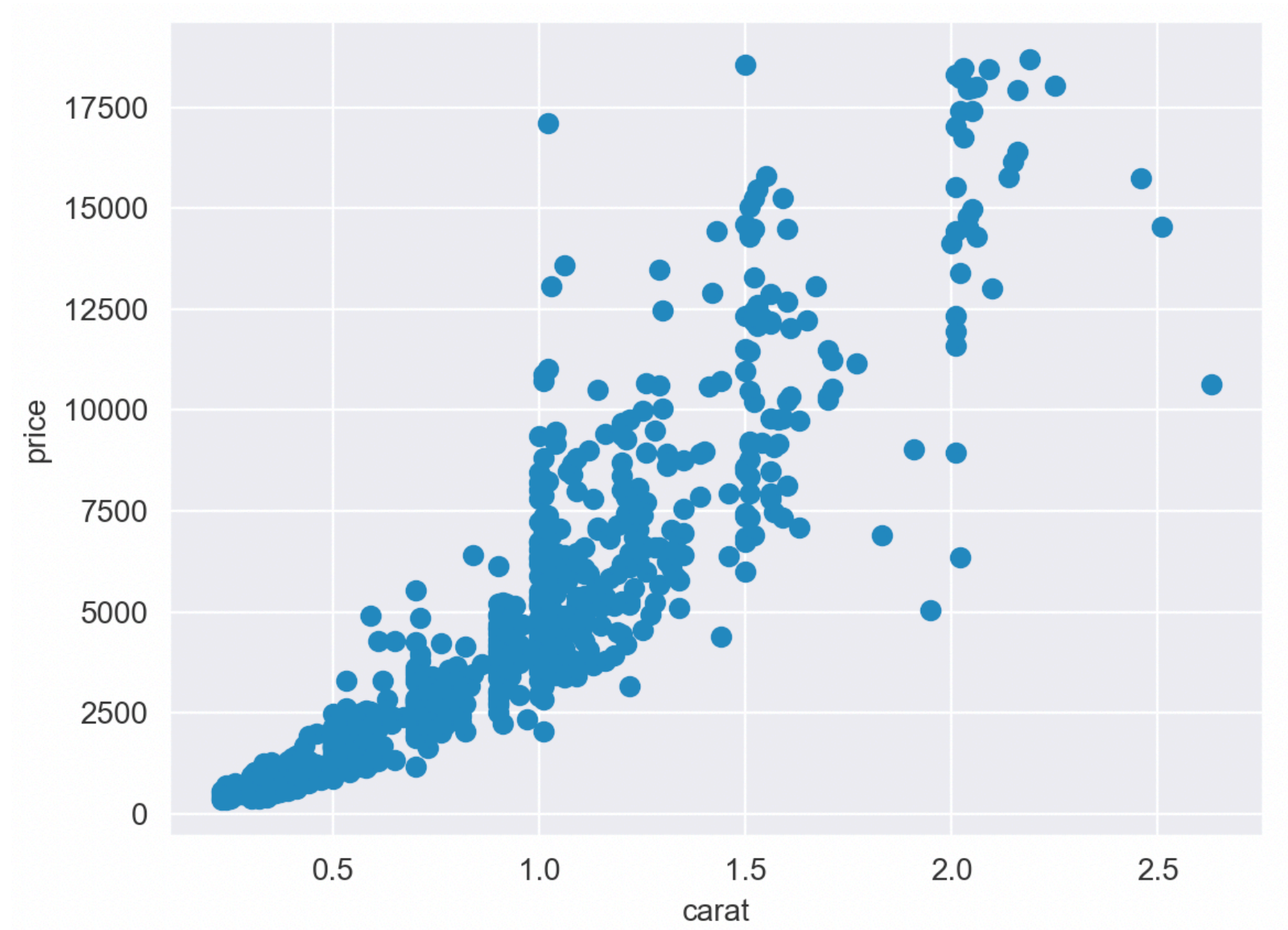
Plan

- Coding up PCA on 3D diamond data
- Reconstructing data from subset of principal components
- Data whitening
- Principal component analysis (singular value decomposition approach)
- Application of SVD PCA: "Eigenfaces"

Reconstruction of data from top principal components

- PCA space useful for analysis, but sometimes we want to ask the question: "If I represent the data according to the top k eigenvectors, what would the data look like in their native space?"
- What might happen in the reconstruction process if we toss out eigenvectors?
 - This reconstruction process is **lossy** because we toss out information by dropping k out of n eigenvectors.
- Let's derive the reconstruction together on the board.

Original diamond data



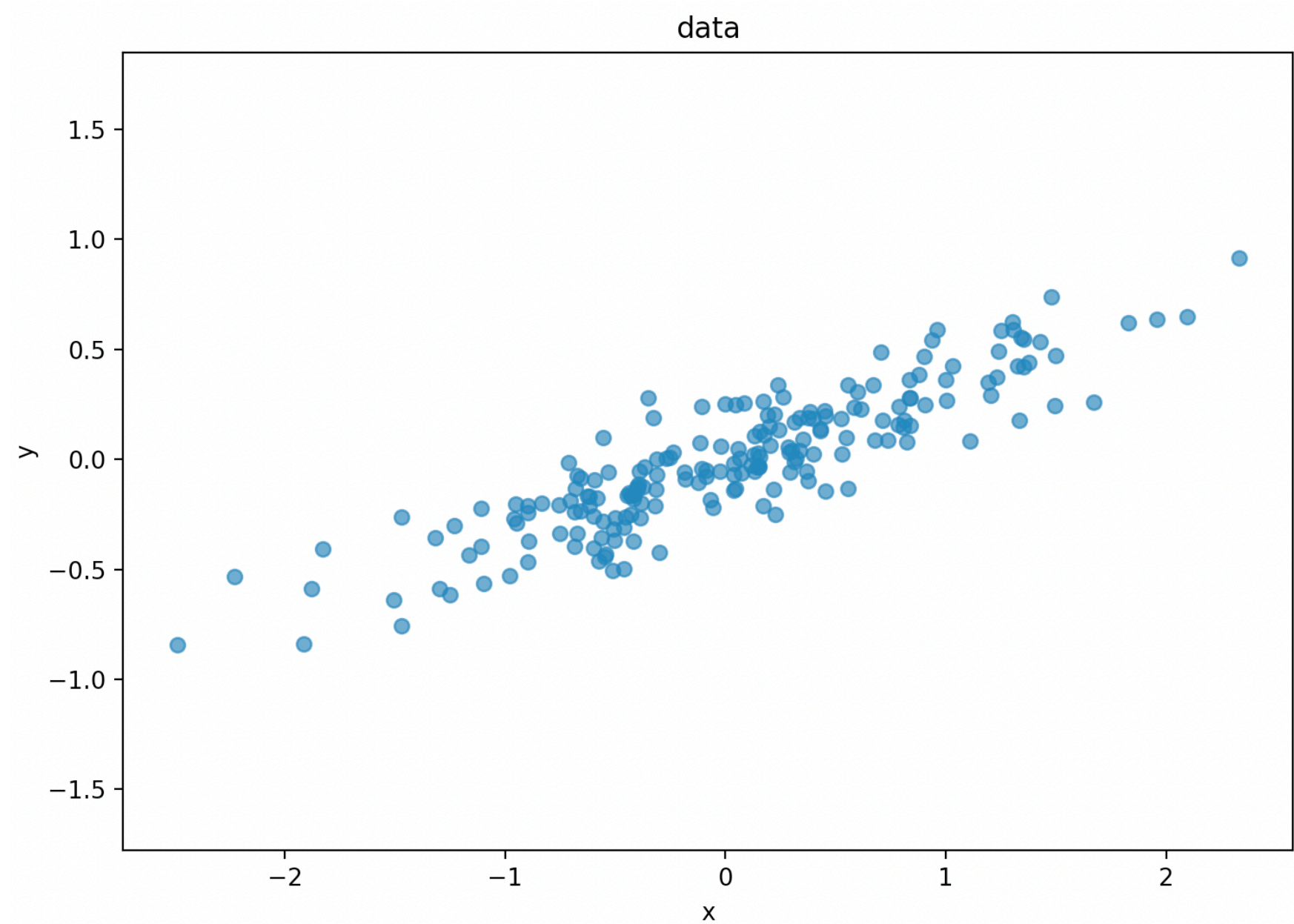
Reconstructed diamond data from top 2 PCs



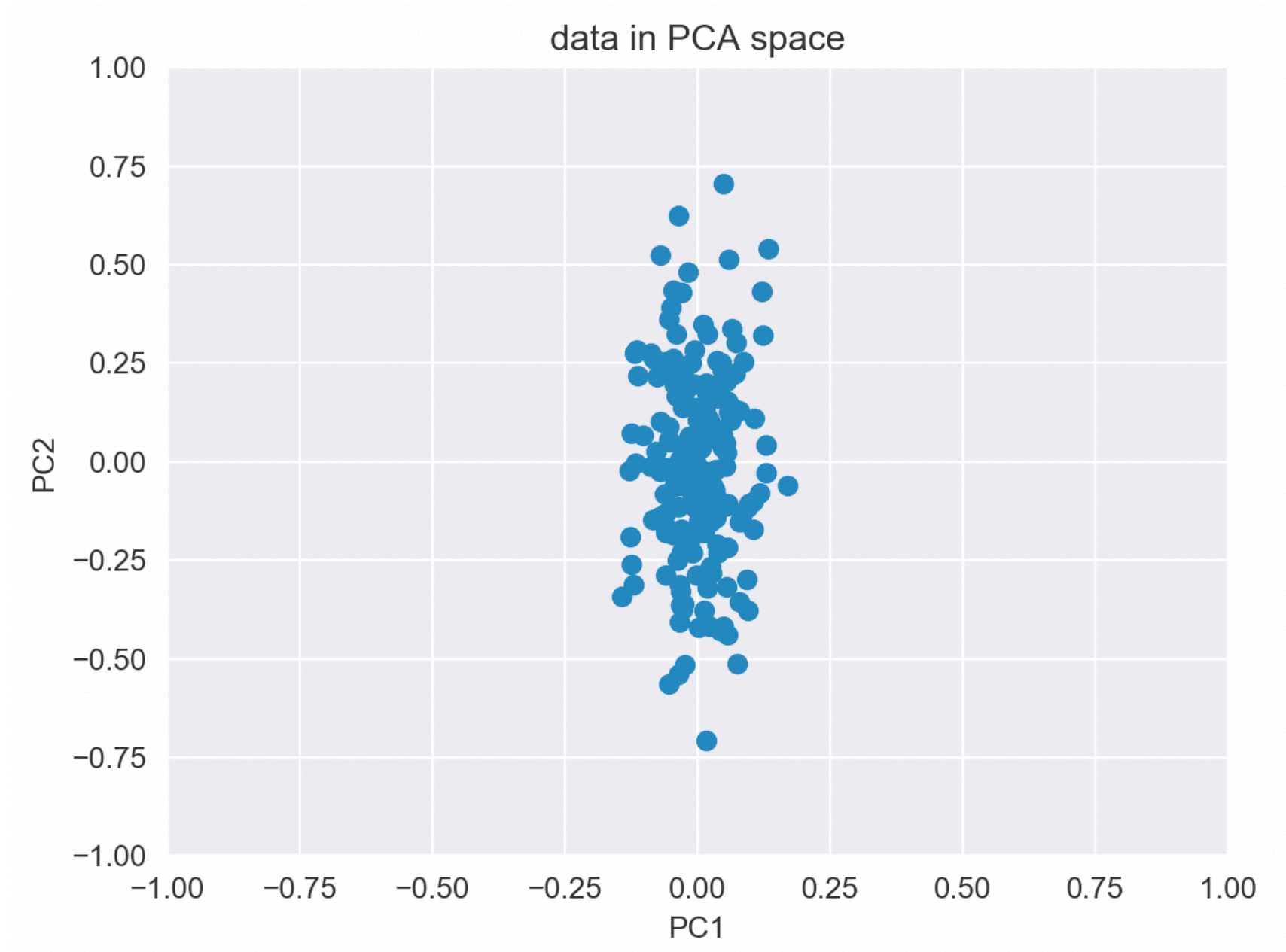
Whitening data

- In some applications (e.g. machine learning), we want to de-correlate the variables when doing dimensionality reduction (via PCA) — i.e. make features as independent from one another as possible.
- **Data whitening/sphering:** Rescaling our data when projecting into PCA space to make the covariance BETWEEN variables 0 and WITHIN variables 1.
- How can we leverage our eigenvalues to whiten the PCA space?

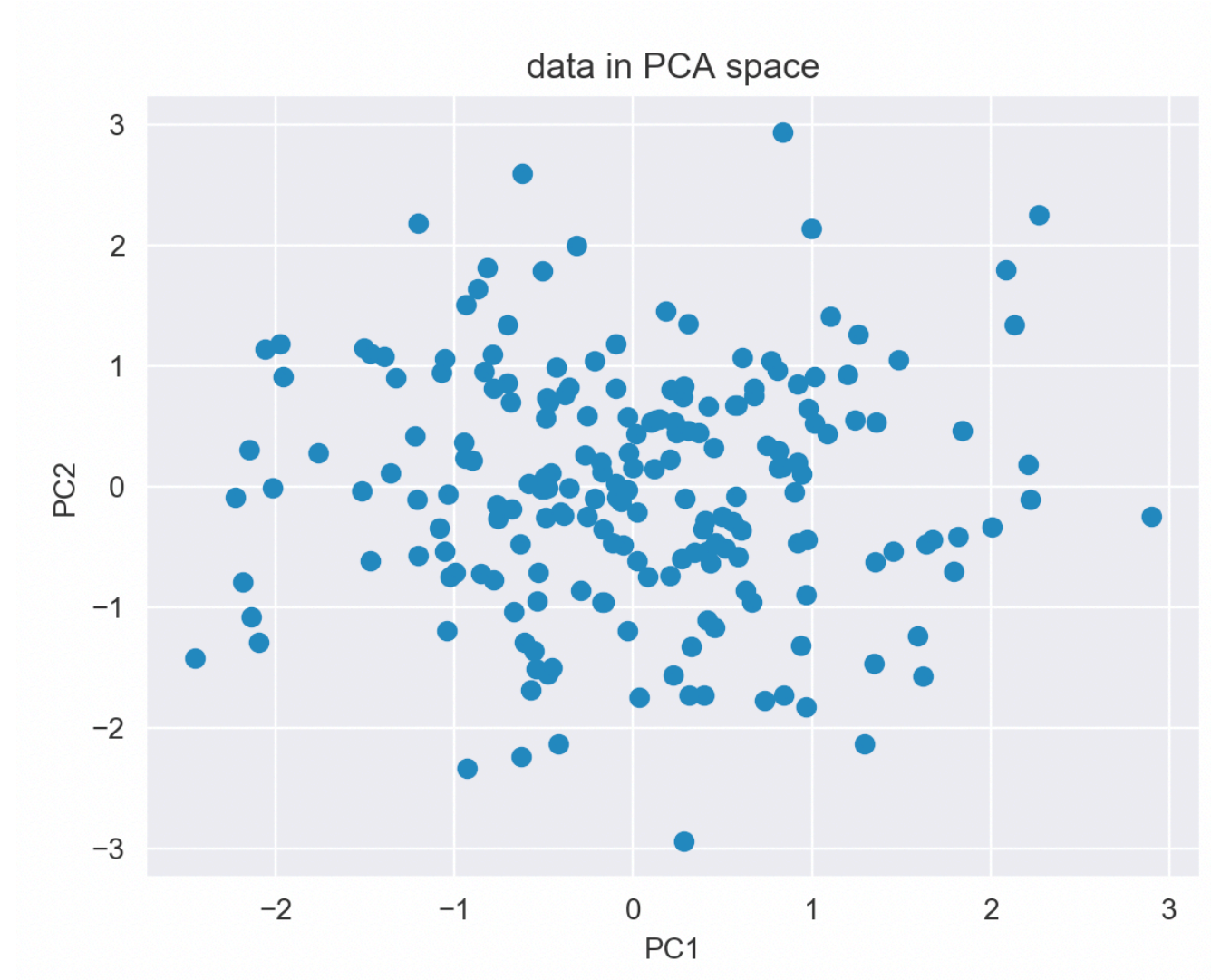
Original data



Un-whitened data in PCA space



Whitened data in PCA space



- Let's derive the whitening process together on the board.

Singular value decomposition

- We can bypass the covariance matrix when computing eigenvectors/eigenvalues using the **singular value decomposition (SVD)**.
- Run SVD on data matrix (original, NOT the covariance, matrix!).
- SVD defined for ANY shape matrix ($n \times n$ or $n \times m$...whatever!).
- Can be more numerically stable than PCA via covariance matrix.
- Let's show how to take the SVD of any matrix.

PCA with SVD

1. Center the data (and optionally normalize) to get X_c .
2. Compute U , S , and V^T via SVD: $USV^T = X_c$
3. Recover eigenvalues $\lambda_i = \frac{s_i^2}{N-1}$ for $N-1$ data points and s_i is the i^{th} diagonal entry of S .
4. Recover eigenvectors as rows of V^T .
5. Project data matrix onto eigenvectors (columns of V): $(VX_c^T)^T$