Unsupervised Dimensionality Reduction via PCA

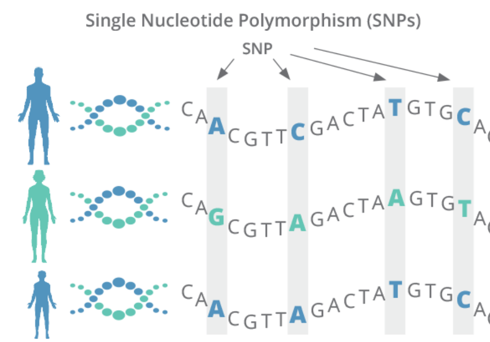
ECE57000: Artificial Intelligence

David I. Inouye

Thursday, September 3, 2020

Very high-dimensional data is becoming ubiquitous

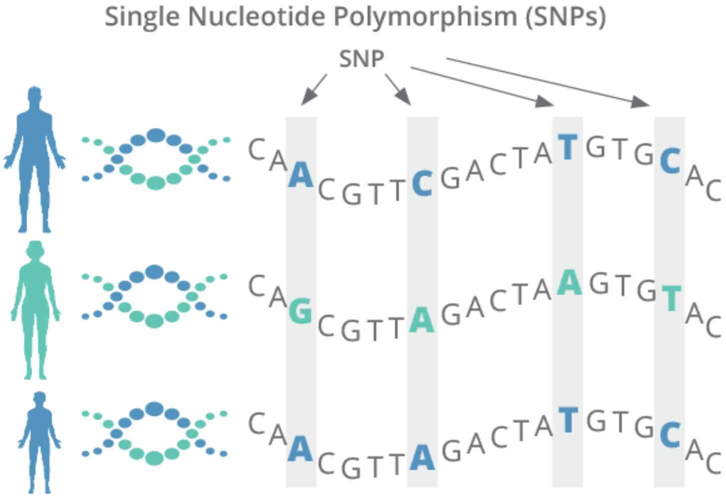
- Images (1 million pixels)
- Text (100k unique words)
- Genetics (4 million SNPs)
- Business data (12 million products)



Why dimensionality reduction? Lower computation costs

- Suppose original dimension is large like $\mathrm{d} = 100000$ (e.g., images, DNA sequencing, or text)

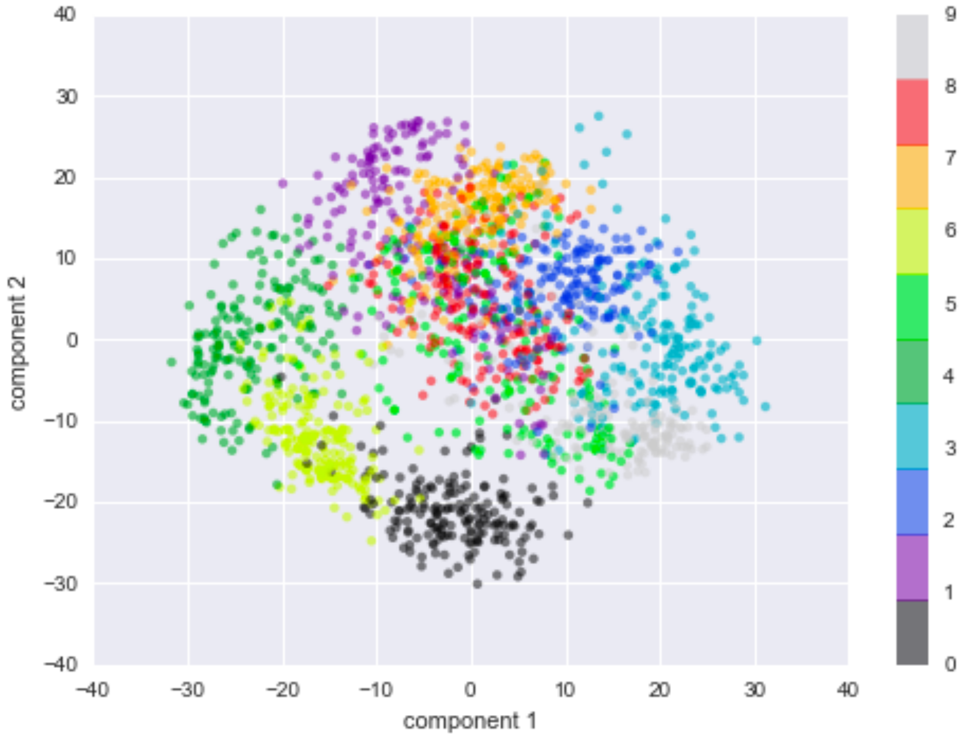
- If we reduce to $k = 100$ dimensions, the training algorithm can be sped up by $1000\times$



4-5 million SNPs in human genome.
https://www.diagnosticsolutionslab.com/tests/genomicinsight

Why dimensionality reduction? Visualization

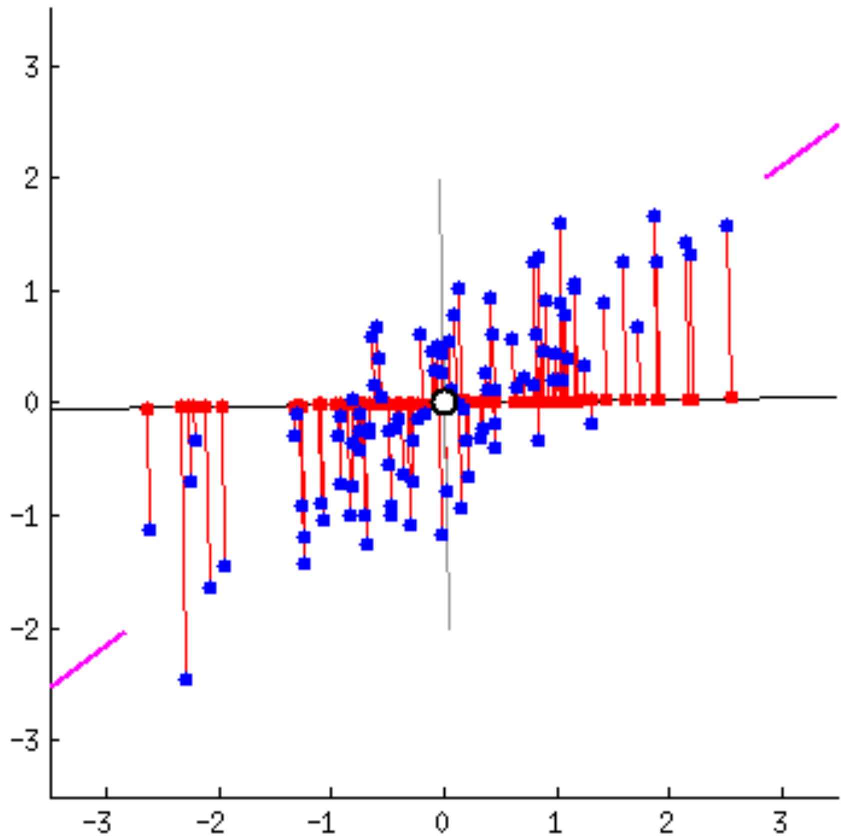
- Allows 2D scatterplot visualizations even of high-dimensional data (2D projection of digits)



https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html

Why dimensionality reduction?
Noise reduction via reconstruction

Principal component analysis finds the **best linear projection** onto a lower-dimensional space



2D to 1D projection: Red lines show the projection error onto 1D lines. PCA finds the line that has the smallest projection error (in this example, when it aligns with the purple).

https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues

Principal Component Analysis (PCA) can be formalized as minimizing the linear reconstruction error of the data using only $k \leq d$ dimensions

- PCA can be formalized as

$$\min_{\mathrm{Z,W}} \|X_c - ZW^T\|_F^2$$

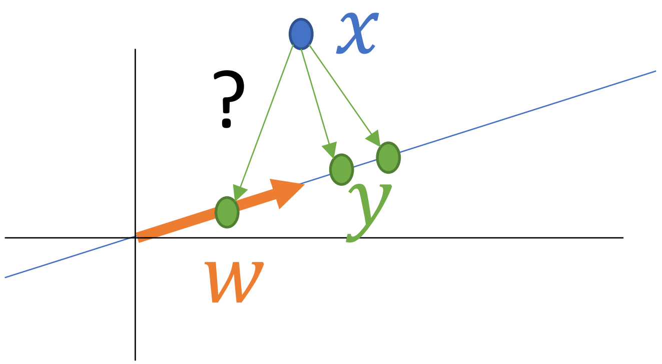
  - where
    $\mathrm{X_c} = \mathrm{X} - \mu_x 1^T \in \mathbb{R}^{n\times d}$ (centered input data)
    $Z \in \mathbb{R}^{n\times k}$ (latent representation or “scores”)
    $W^T \in \mathbb{R}^{k\times d}$ (principal components)
    $w_s^T w_t = 0, w_s^T w_s = \|w_s\|_2 = 1, \forall s, t$
    (orthogonal constraint)
- Solution
  - $W^T = V_{1:k}^T$ where $X_c = USV^T$ is the **SVD** of X_c
  - $Z = X_c W$

Review of linear algebra and introduction to numpy Python library

- See Jupyter notebook, which can be opened and run in Google Colab

The *orthogonal* projection onto a 1D line is the closest projection

- Given a line defined by a unit vector w, what is the *closest* projection onto that line?



- The orthogonal projection! (via dot product)

$$y = (x^T w)w = zw$$

  - Where $z = \|x\|\|w\| \cos\theta = \|x\| \cos\theta$ is the distance from the origin (cos = adj/hyp)

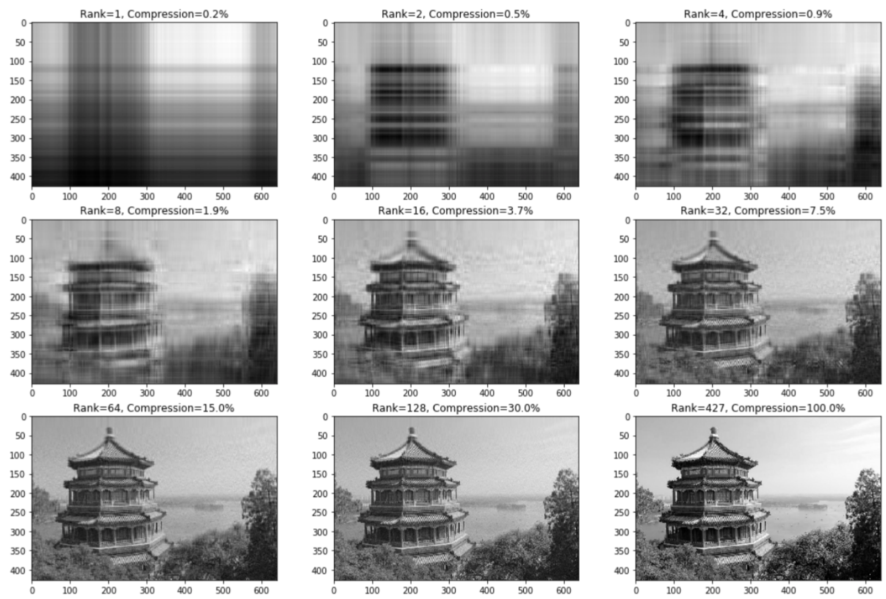
Formulate problem as minimizing reconstruction error

- Squared distance of point to best projection
$$\|x - y\|_2^2 = \|x - (x^T w) w\|_2^2$$
- Minimize the reconstruction error for all points in the dataset
$$\min_{w:\|w\|=1} \sum_i \left\|x_i - \left(x_i^T w\right) w\right\|_2^2 = \|X_c - (X_c w) w^T\|_F^2$$
- PCA generalized to more dimensions
$$\min_W \|X_c - (X_c W) W^T\|_F^2 \quad \text{s.t.} \quad W^T W = I$$

The PCA solution is the top k right singular vectors via SVD

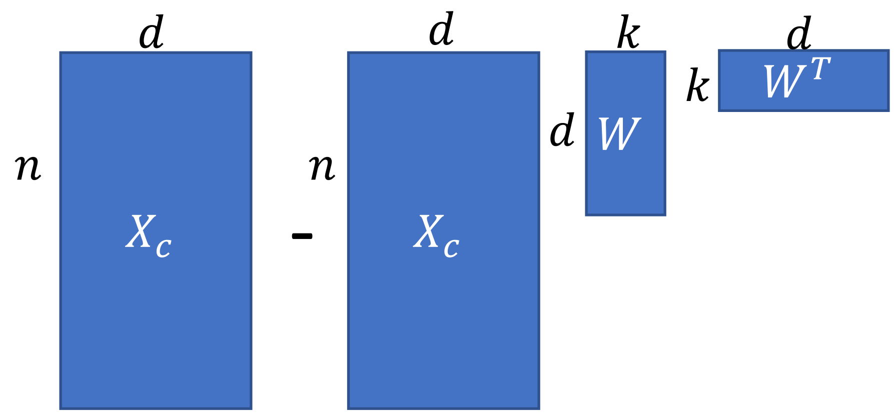
- If $X_c = USV^T$, then the solution to the previous problem is simply $W^* = V_{1:k}$

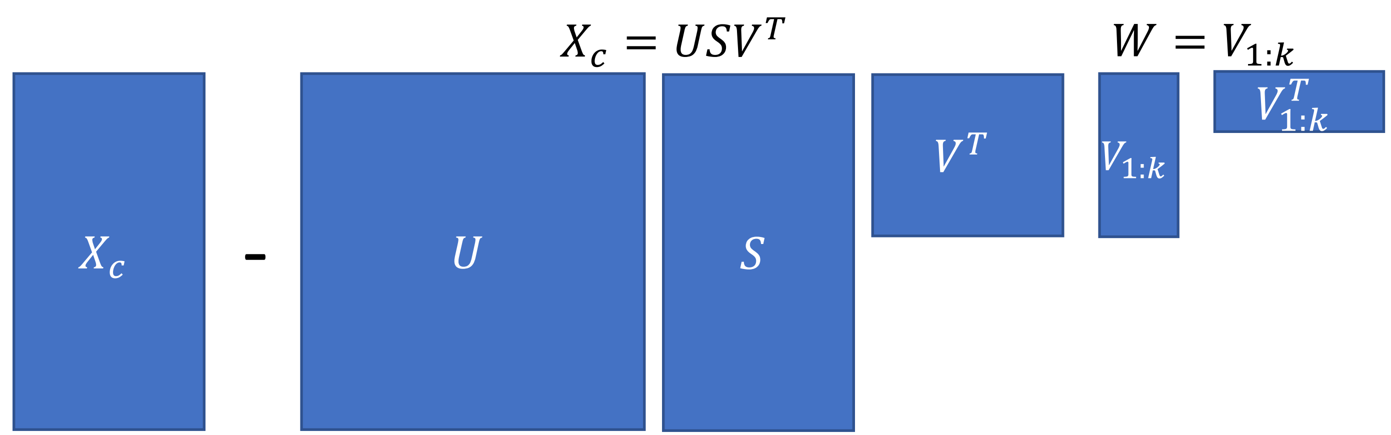
- Remember: SVD is best k dim. approximation



The solution reveals the truncated SVD as best approximation

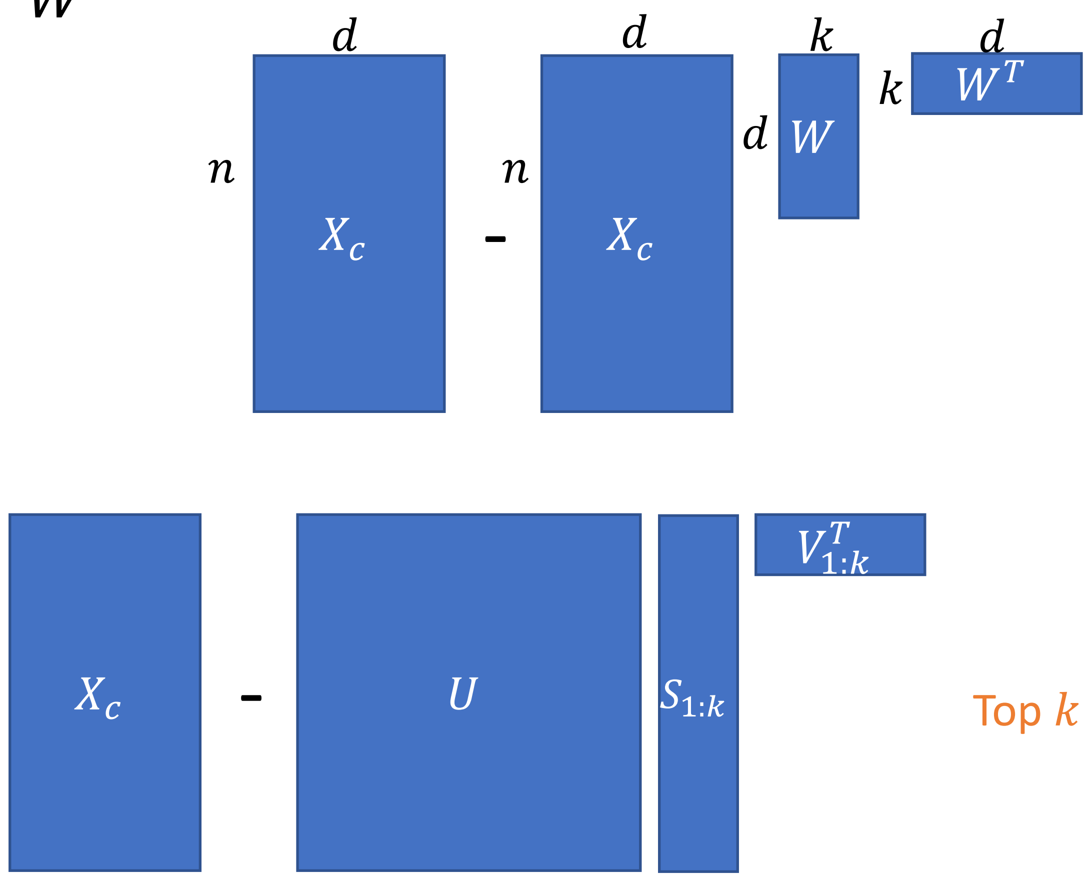
$$\min_{W} \|X_c - (X_c W)W^T\|_F^2 \quad \text{s.t.} \quad W^T W = I$$





The solution reveals the truncated SVD as best approximation

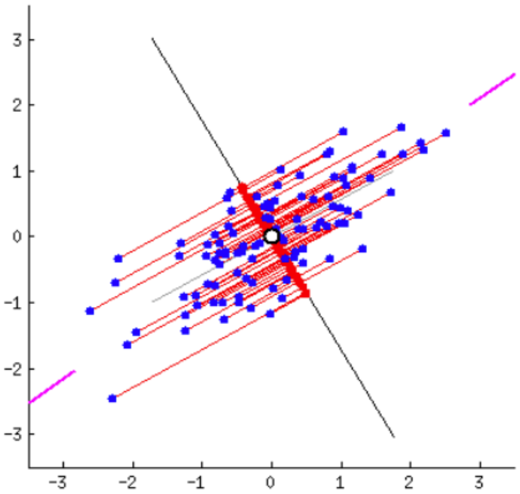
$$\min_{W} \|X_c - (X_c W) W^T\|_F^2 \quad \text{s.t.} \quad W^T W = I$$

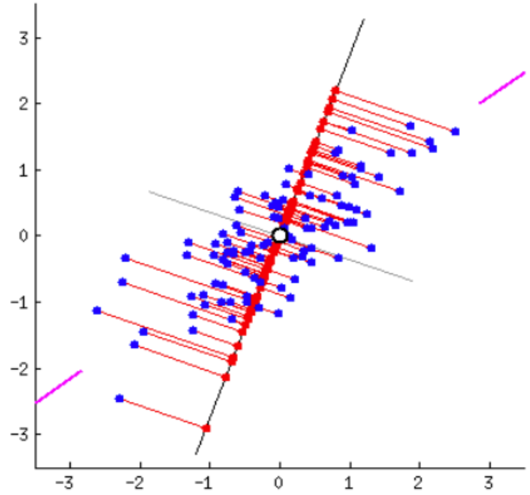


Top k truncated SVD

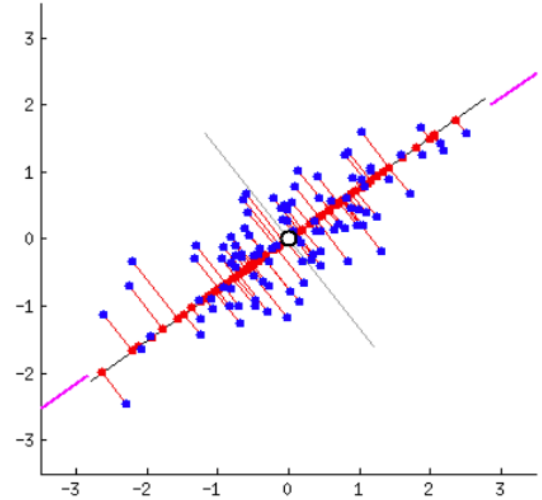
Claim: Minimizing reconstruction error (red lines) is equivalent to maximizing the variance of projection (spread of red points)

Max reconstruction error
Min variance





Min reconstruction error
Max variance



Derivation of min error equivalent to max variance

- Simplify squared distance
- $\left\| x_i - (x_i^T w) w \right\|_2^2$
- $= \left(x_i - (x_i^T w) w \right)^T \left(x_i - (x_i^T w) w \right)$
- $= x_i^T x_i - 2(x_i^T w) w^T x_i + \left(x_i^T w \right)^2 w^T w$
- $= \|x_i\|^2 - 2\left(x_i^T w \right)^2 + \left(x_i^T w \right)^2 \|w\|^2$
- $= \|x_i\|^2 - \left(x_i^T w \right)^2$

Derivation of min error equivalent to max variance

- Equivalence of optimization in 1D
- $\arg\min_w \sum_i \left\| x_i - (x_i^T w) w \right\|_2^2$
- $= \arg\min_w \sum_i \|x_i\|^2 - \left(x_i^T w\right)^2 = \arg\min_w \sum_i -\left(x_i^T w\right)^2$
- $= \arg\max_w \frac{1}{n} \sum_i \left(x_i^T w\right)^2 = \arg\max_w \frac{1}{n} \sum_i z_i^2$
- $= \arg\max_w \ \sigma_z^2$

Note z is already centered so mean of squares is variance

The solution is the eigenvector with the largest eigenvalue of the covariance matrix $\hat{\Sigma}_x$

- Suppose $\hat{\Sigma}_x = \frac{1}{n} X_c^T X_c = Q\Lambda \mathrm{Q}^{\mathrm{T}}$, where $\lambda_1 \geq \lambda_2 \geq \cdots \geq 0$
  - The solution is top eigenvector $w^* = q_1$
- The more general case

$$\arg \max_{W:W^T W = I_k} \sum_{j=1}^{k} w_j^T \hat{\Sigma}_x w_j = \arg \max_{W:W^T W = I_k} \sum_{j=1}^{k} \sigma_{z_j}^2$$

  - The solution is the top k eigenvectors of $\hat{\Sigma}_x$

$$W^* = Q_{1:k}$$

The solution to both problems is the top k right singular vectors of X_c

- Minimize reconstruction error
  - Singular value decomposition (SVD) of $X_c = USV^T$
  - Solution: $W^* = V_{1:k}$

- Maximize variance of latent projection
  - Equivalence solution

$$n\,\Sigma_x = X_c^T X_c = (USV^T)^T(USV^T) = (VSU^T)(USV^T)$$
$$= VS(U^T U)SV^T = VS^2V^T = Q\Lambda Q^T$$

  - Solution: $W^* = Q_{1:k} \equiv V_{1:k}$