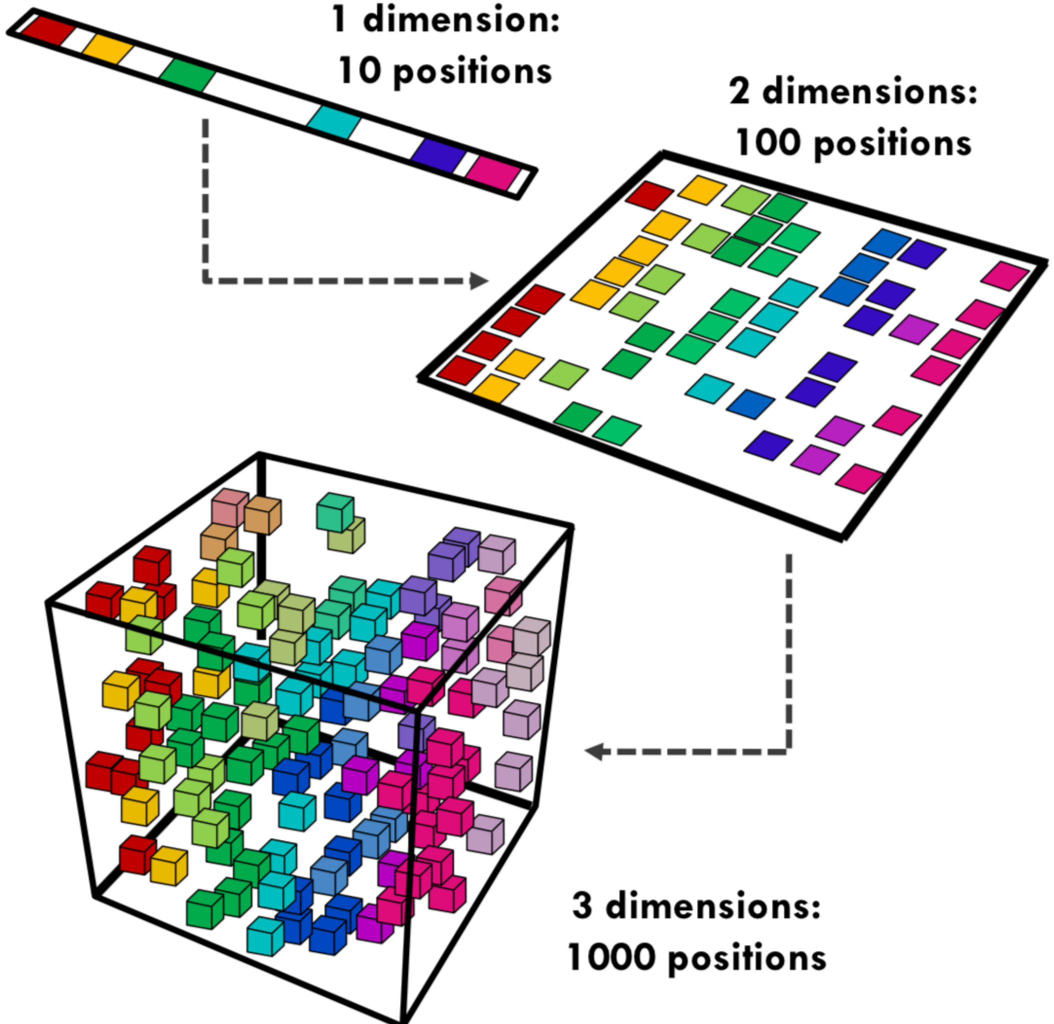
Dimension reduction

Kookjin Lee

(kookjin.Lee@asu.edu)

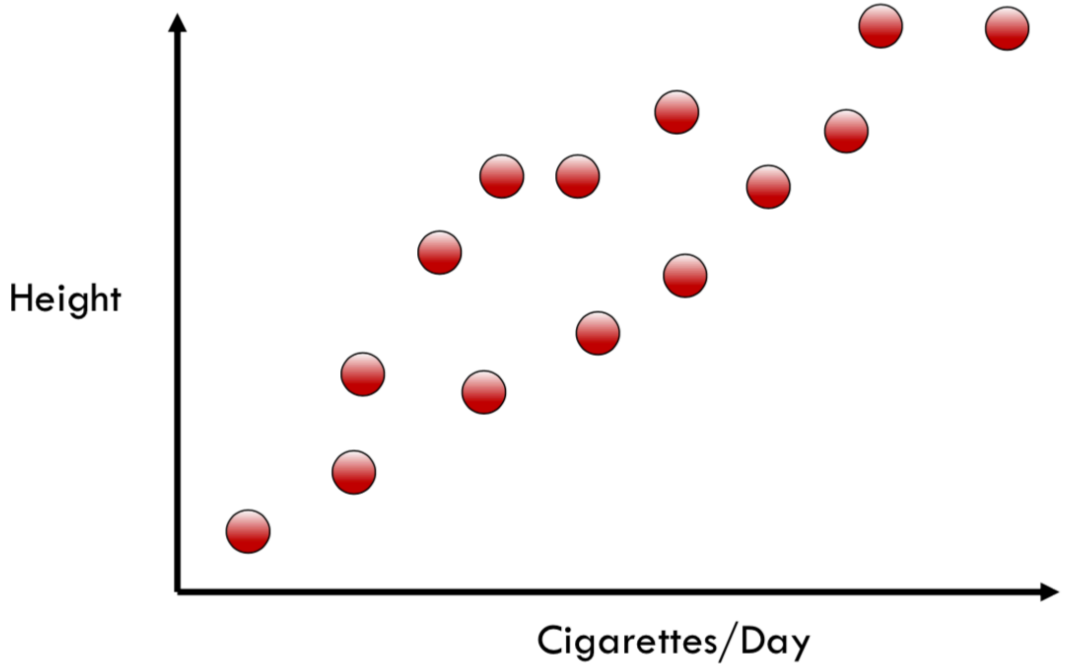
Curse of dimensionality

- Theoretically, increasing features should improve performance
- In practice, too many features leads to worse performance
- Number of training examples required increases exponentially with dimensionality



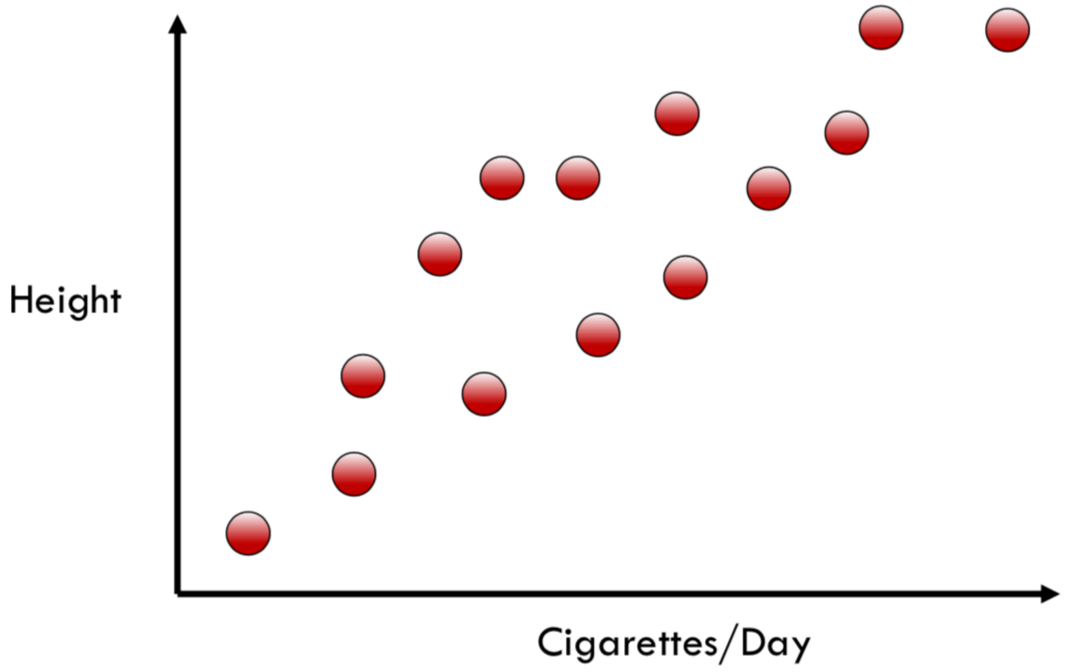
Solution: dimension reduction

- Data can be represented by fewer dimensions (features)
- Reduce dimensionality by selecting subset (feature elimination)
- Combine with linear and non-linear transformations



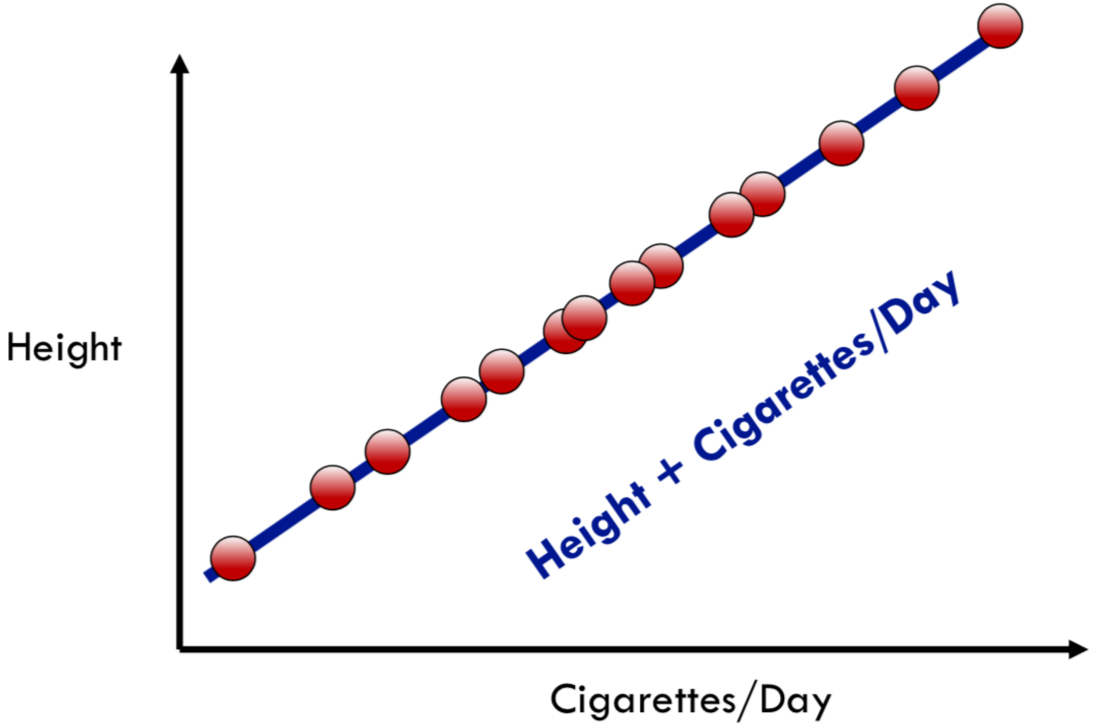
Solution: dimension reduction

- Two features: height and cigarettes per day
- Both features increase together (correlated)
- Can we reduce number of features to one?



Solution: dimension reduction

- Create single feature that is combination of height and cigarettes
- Principal Component Analysis (PCA)

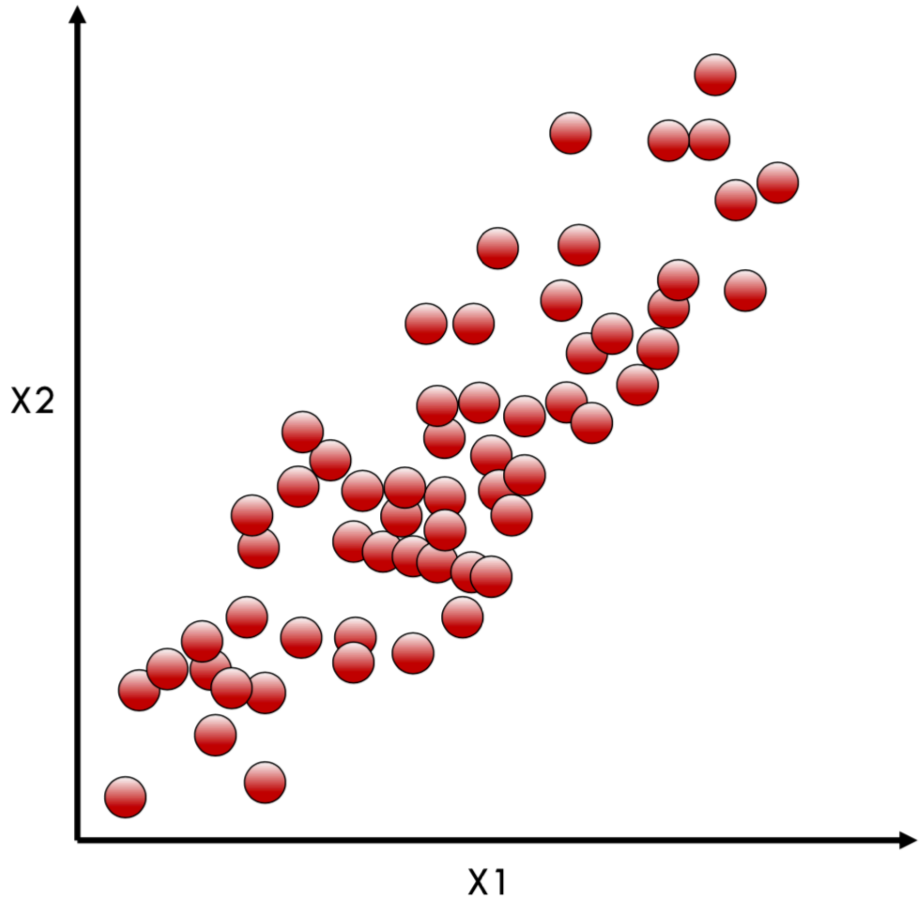


Dimensionality reduction

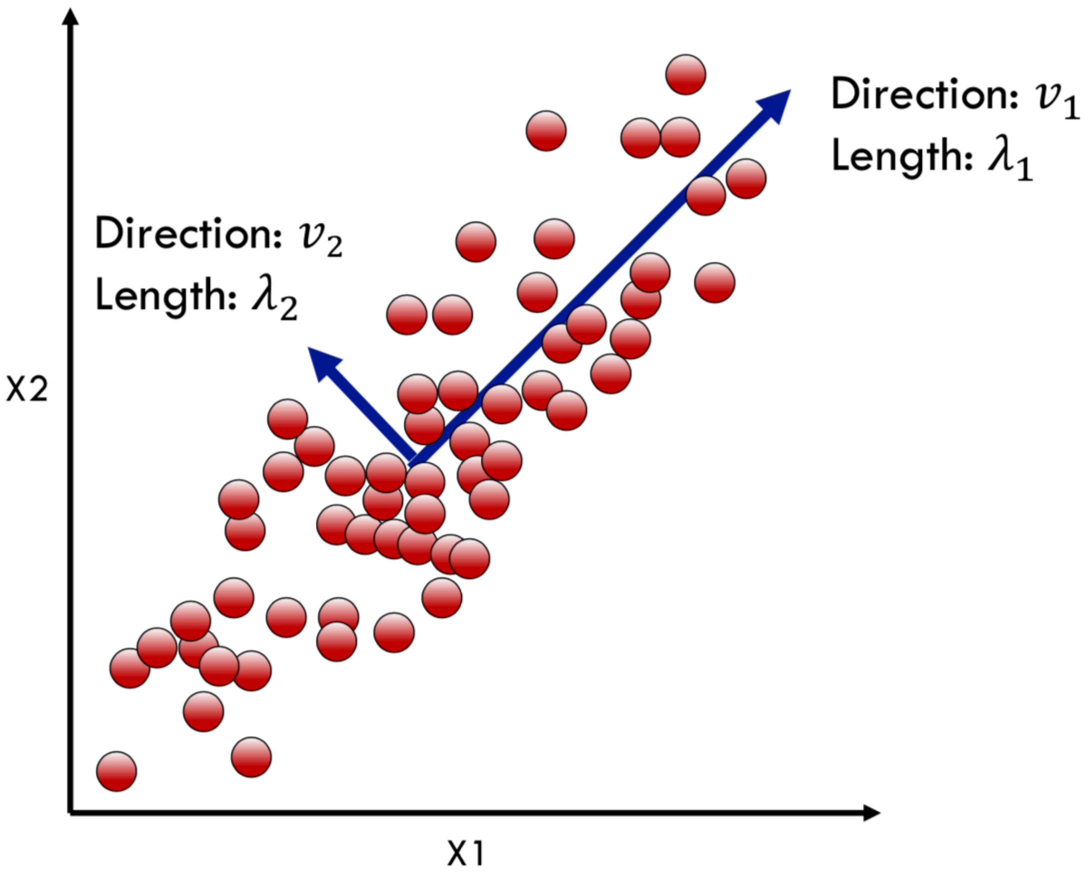
- Given an N-dimensional data set, find an N x k matrix:

$$y^{(i)} = U^\intercal x^{(i)}$$

Principal Component Analysis (PCA)



Principal Component Analysis (PCA)



Mathematical derivation

- A simple example

Mathematical derivation

- Projection

$$uu^\intercal x^{(i)}$$

  - Distance from origin to projected quantity

    $$u^\intercal x^{(i)}$$

  - Maximize the variance?

    $$\frac{1}{n}\sum_{i=1}^{n}(u^\intercal x^{(i)})^2$$

Mathematical derivation

- Projection

$$uu^\intercal x^{(i)}$$

  - Distance from origin to projected quantity

$$u^\intercal x^{(i)}$$

  - Maximize the variance?

$$\frac{1}{n}\sum_{i=1}^{n}(u^\intercal x^{(i)})^2 = \frac{1}{n}\sum_{i=1}^{n} u^\intercal x(i) x^{(i)\intercal} u$$

$$= u^\intercal \left(\frac{1}{n}\sum_{i=1}^{n} x(i) x^{(i)\intercal} \right) u$$

Mathematical derivation

- The principal component = principal eigenvector

$$u^\intercal \left(\frac{1}{n} \sum_{i=1}^{n} x(i) x^{(i)\intercal} \right) u = u^\intercal \Sigma u$$

$$\Sigma u = \lambda u$$

Singular Value Decomposition

- SVD is a numerical algorithm commonly used for PCA (e.g., scikit-learn)

$$\underbrace{\begin{bmatrix} \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \end{bmatrix}}_{A_{m\times n}} = \underbrace{\begin{bmatrix} \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \end{bmatrix}}_{U_{m\times m}} \underbrace{\begin{bmatrix} \star & 0 & 0 \\ 0 & \star & 0 \\ 0 & 0 & \star \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{S_{m\times n}} \underbrace{\begin{bmatrix} \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \end{bmatrix}}_{V^T_{n\times n}}$$

Singular Value Decomposition

- Truncated SVD

$$\underset{A_{m\times n}}{\begin{bmatrix} \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \end{bmatrix}} \approx \underset{U_{m\times k}}{\begin{bmatrix} \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \end{bmatrix}} \underset{S_{k\times k}}{\begin{bmatrix} \mathbf{9} & 0 & 0 \\ 0 & \mathbf{7} & 0 \\ 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}} \underset{V^T_{k\times n}}{\begin{bmatrix} \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \end{bmatrix}}$$

3D example

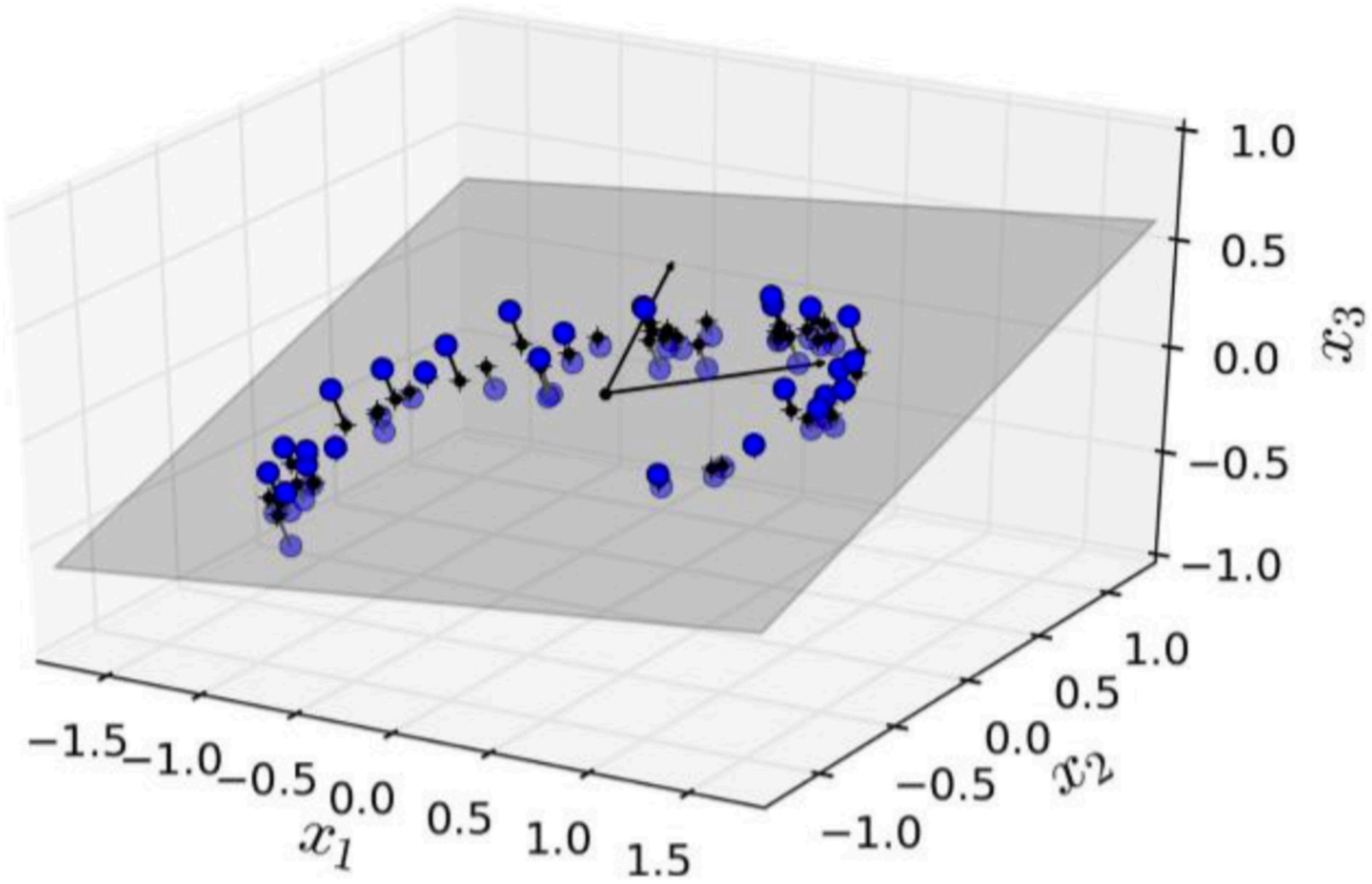


Figure 8-2. A 3D dataset lying close to a 2D subspace

3D example

- Projected data

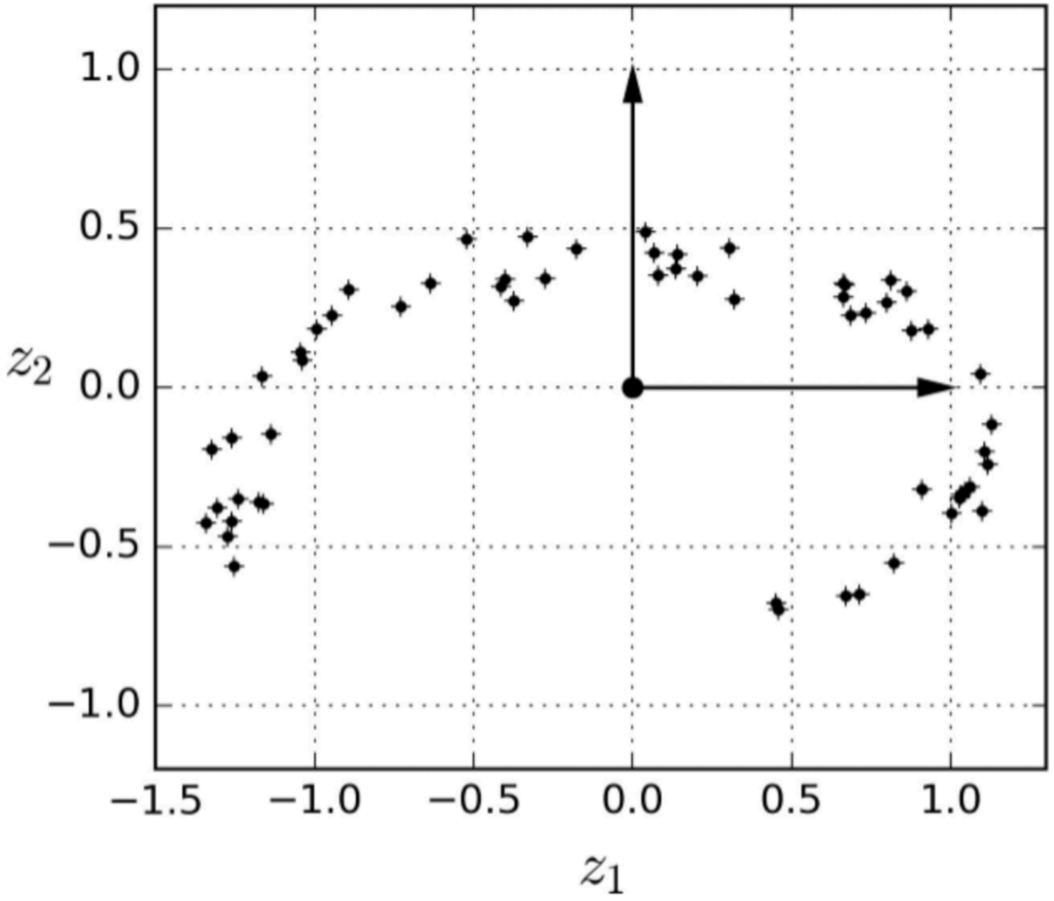
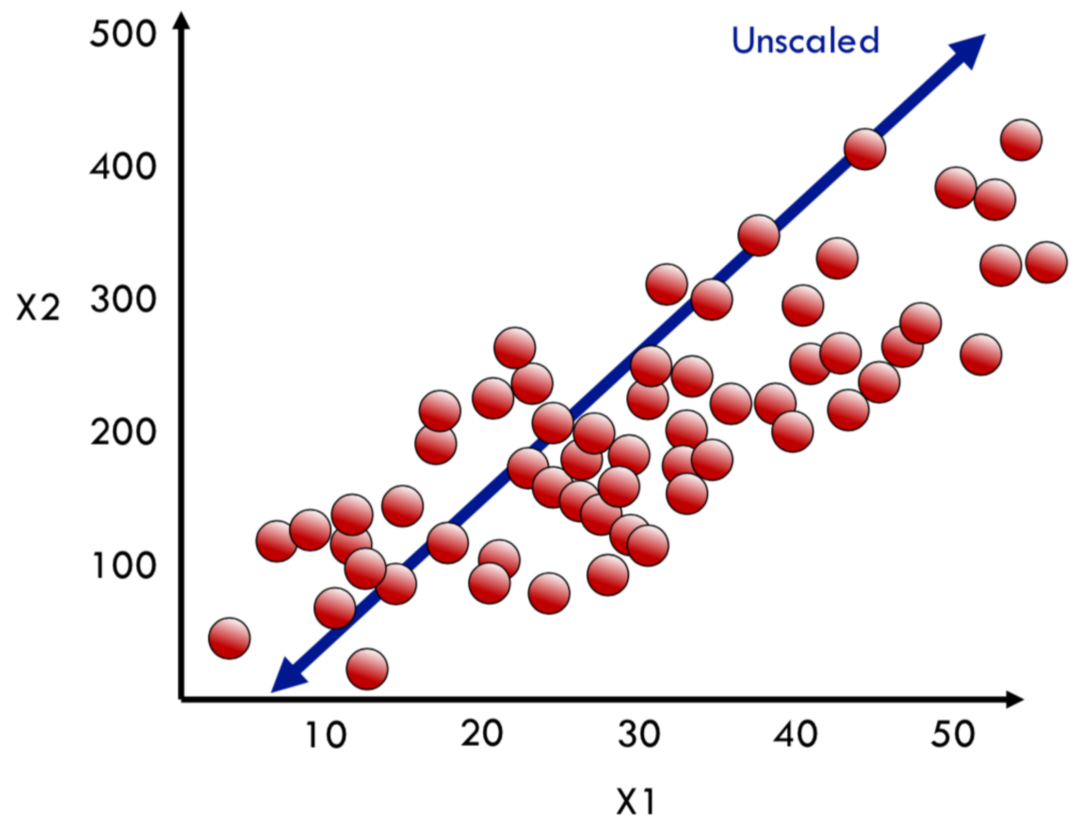


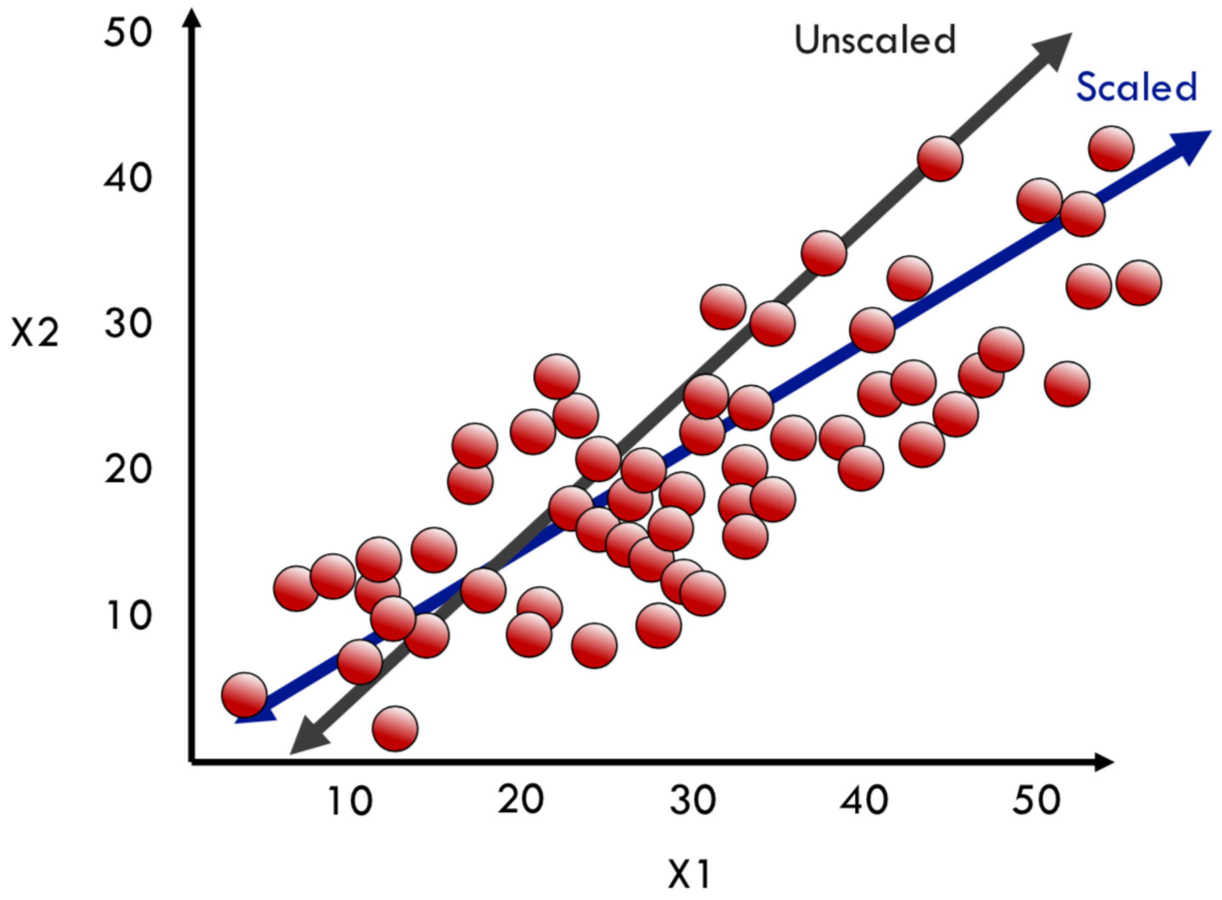
Figure 8-3. The new 2D dataset after projection

Importance of feature scaling

- PCA (and SVD) seek to find the vectors that capture the most variance
- Variance is sensitive to axis scale
- Must scale data

Importance of feature scaling

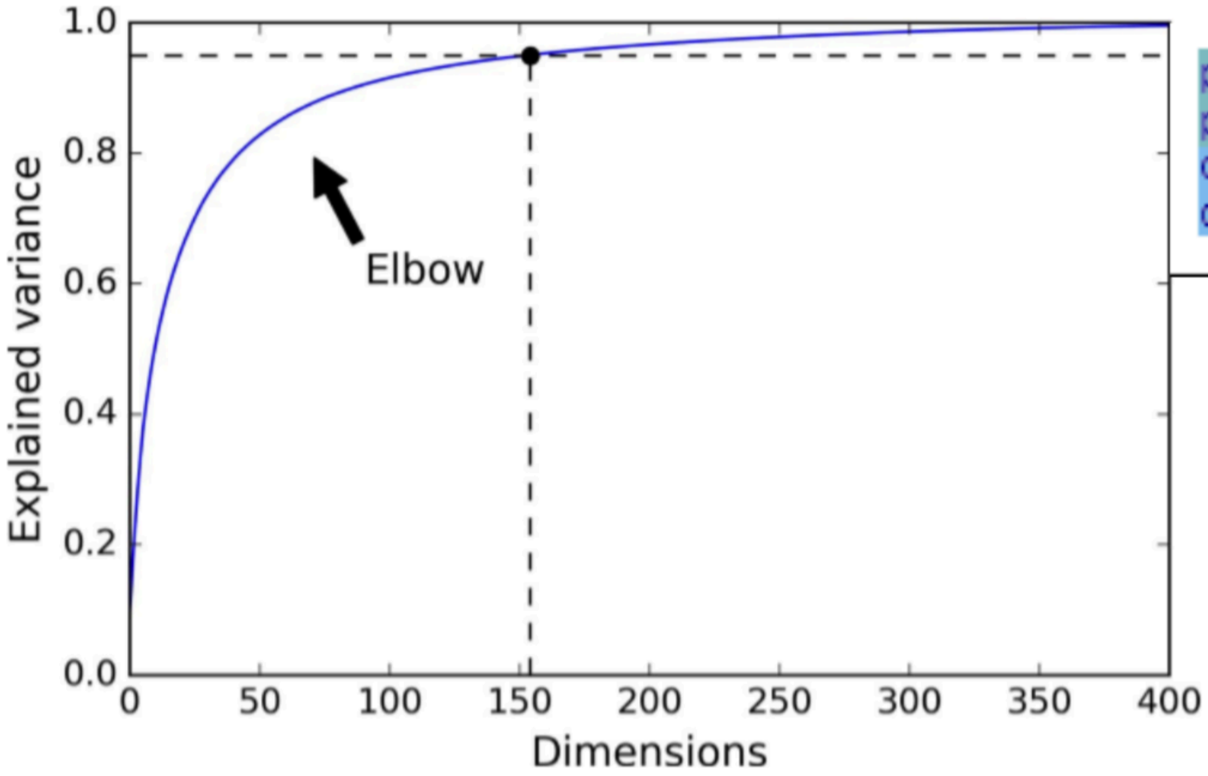




How to scale?

- To have zero mean and unit variance
  - Standard Scaler

Explained variance




```
pca = PCA()
pca.fit(X)
cumsum = np.cumsum(pca.explained_variance_ratio_)
d = np.argmax(cumsum >= 0.95) + 1
```