Dimensionality Reduction

Intro

- The Curse of Dimensionality
  - Distances between points grow very fast
  - Analogy: Finding penny on a line, a football field, in a building
- Ways to reduce dimensionality
  - Feature Subset Selection
    - $O(2^n)$if we try all
    - Forward selection – add feature that decreases the error the most
    - Backward selection – remove feature that decreases the error the most (or increases it only slightly)
    - But selection is greedy not necessarily optimal
  - Feature Extraction
    - PCA
    - LDA
    - FA
    - MDS

Principal Component Analysis (PCA)

What It Does

- Comes up with a new coordinate system
- Performs a rotation of your dataset that decorrelates the features
- Allows you to reduce the dimensionality of your data

Uses

- Dimensionality reduction
- Pattern recognition (e.g. Eigenfaces)

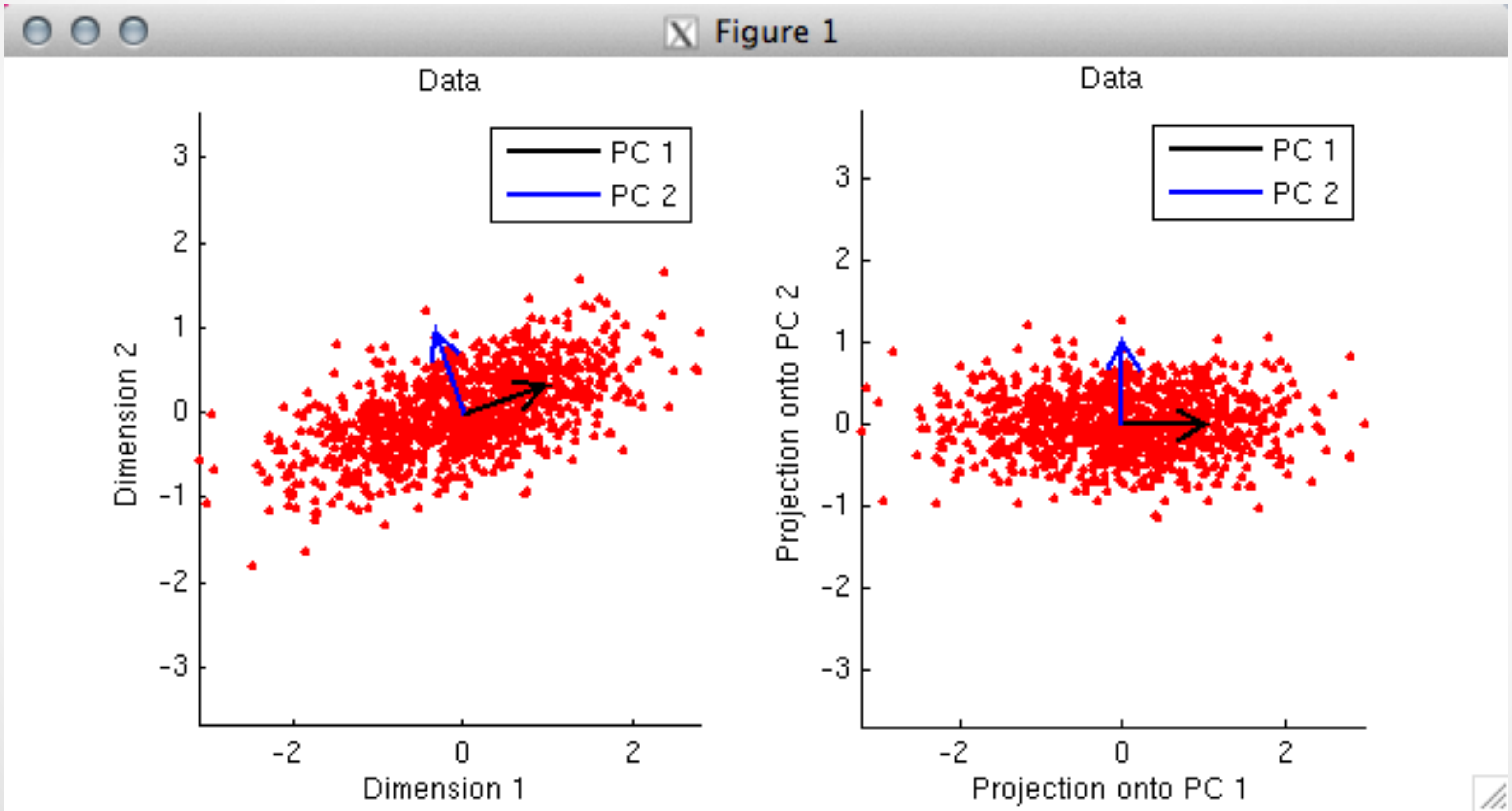
Cinematography

The Pareto Principle

PCA

- Creates new features that are linear combinations of the original features
- New features are orthogonal to each other
- Keep the new features that account for a large amount of the variance in the original dataset
- Re-base the dataset's coordinate system in a new space defined by its lines of greatest variance

Visualization

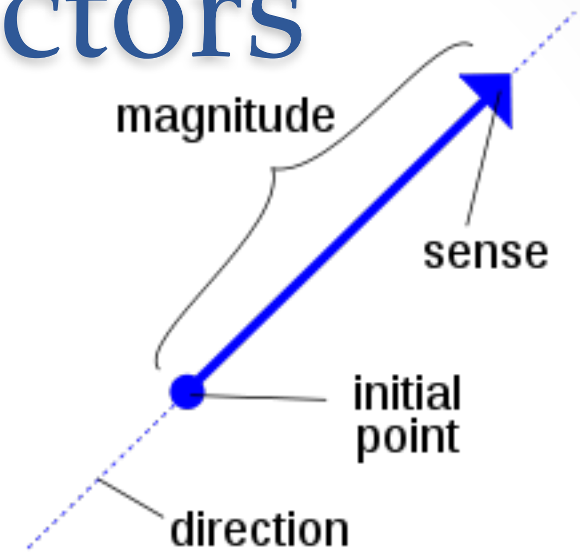


Principal Components

- Linearly uncorrelated variables
- 1st principal component has the largest possible variance
- Each succeeding component has highest possible variance. Constraint: Must be orthogonal to all the preceding components

Observation About Vectors

$$[x_1\ x_2\ \dots\ x_m]^{\mathrm{T}} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix},$$



- Almost all vectors change direction when multiplied by a matrix
- Certain exceptional vectors (which are called ***eigenvectors***) remain in the same direction

Eigenvector

- A vector that when multiplied by a given matrix gives a scalar multiple of itself
- The **0** vector is never considered an eigenvector
- The scalar multiple is called its ***eigenvalue*** λ.

Eigenvalue

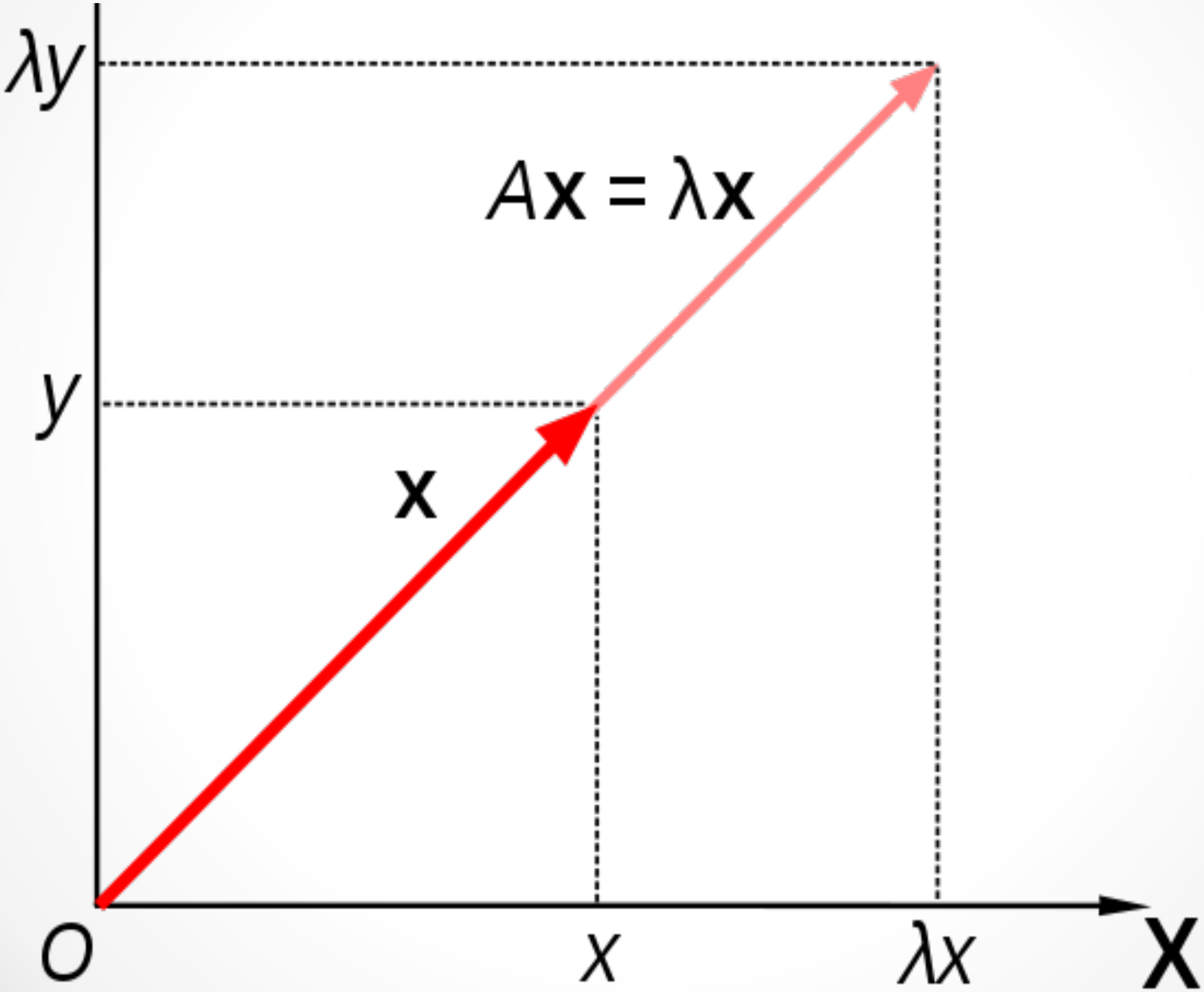
- A scalar
- Scale factor corresponding to a particular eigenvector
- Merely elongates or shrinks or reverses **v**, or leaves it unchanged

Eigens Expressed As An Equation

- A: a square matrix
- $\mathbf{x}$: a nonzero vector ("eigenvector")
- λ: a nonzero scalar ("eigenvalue of A")

$$A\mathbf{x} = \lambda\mathbf{x}$$

Graphical Depiction



Example of Eigenvalue & Eigenvector Pair

$$A = \begin{bmatrix} 1 & -1 \\ 2 & 4 \end{bmatrix}, \quad \lambda = 3, \quad \boldsymbol{x} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

$$A\boldsymbol{x} = \lambda\boldsymbol{x}$$

$$\begin{bmatrix} 1 & -1 \\ 2 & 4 \end{bmatrix}\begin{bmatrix} 1 \\ -2 \end{bmatrix} = 3\begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

Identity Matrix

- A square matrix that looks like this:

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Get the pattern?

It's ones down this main diagonal and zeros everywhere else.

Eigen

- German for "very own"
- "My very own apartment":
  »Meine eigene Wohnung«

Characteristic Polynomial

- Start with $A\lambda = \lambda\mathbf{v}$
- I is the identity matrix
- Noting that $I\mathbf{v} = \mathbf{v}$, rewrite as $(A - \lambda I)\mathbf{v} = 0$
- $A - \lambda I$ is square matrix

$$p_A(\lambda) = |A - \lambda I|$$

- | | is notation for determinant

Characteristic Equation of A

Since v is nonzero, $A-\lambda I$ is singular (non invertible); therefore its determinant is 0.

$$p_A(\lambda) = 0$$

The roots of this n-degree polynomial are the eigenvalues of A.

Example Calculation of Eigenvalues

$$|\mathbf{A} - \lambda \cdot \mathbf{I}| = \left| \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \right| = 0$$

$$\left| \begin{bmatrix} -\lambda & 1 \\ -2 & -3-\lambda \end{bmatrix} \right| = \lambda^2 + 3\lambda + 2 = 0$$

Solutions are $\lambda = -1$ and $\lambda = -2$

Eigenvalue Equation

- Use to obtain the corresponding eigenvector **v** for each eigenvalue λ

$$(A - \lambda I)\boldsymbol{v} = 0$$

Eigenvalues and Eigenvectors in R

- > eigen(x)
- demo

R Functions for PCA

- `prcomp()` - Uses SVD, the preferred method
- Display shows standard deviations of the components
- `> pr<-prcomp(dataset, scale=TRUE)`

- Transform the data into the new coordinate system:

```
> new<-pr$x[,1:2]
```

`princomp()` uses covariance matrix—for compatibility with S-PLUS

Tips

- Dataset can have numeric values only. Need to exclude nonnumeric features with brackets or subset.
- `modelname<-princomp(dataset)`
- `summary(modelname)` gives proportion of the total variance explained by each component.
- Modelname$loadings
- Modelnames$scores

Options

- `center`
- `scale`
- `scale.`

Component Loadings

- Eigenvectors •√Eigenvalues
- Correlation between the component and the original features: how much variation in a feature is explained by a component

Preparation

- Center the variables
- Scale the variables
- Skewness transformation: makes the distribution more symmetrical
- Box-Cox transformation: makes the distribution more normal like

Scoring

- *V*: the matrix whose columns are the eigenvectors of the covariance matrix. Each eigenvector is normalized to have unit length.
- V^T now defines a rotation
- Start with original dataset **x**.
- Calculate mean of each column to obtain row vector **m**.
- Subtract **m** from each row of **x** to obtain **z**. Multiply **z** by V^T to obtain the new matrix of new coordinates **c**.

Covariance

$$COV(X,Y) = \frac{\sum_{i=1}^{n}\left(X_i - \bar{X}\right)\left(Y_i - \bar{Y}\right)}{n-1}$$

(Numerator can be expressed as $\mathbf{X}\mathbf{X}^{T}$)

Covariance Matrix

$$\begin{bmatrix} \mathrm{Var}(X) & \mathrm{Cov}(X,Y) & \mathrm{Cov}(X,Z) \\ \mathrm{Cov}(X,Y) & \mathrm{Var}(Y) & \mathrm{Cov}(Y,Z) \\ \mathrm{Cov}(X,Z) & \mathrm{Cov}(Y,Z) & \mathrm{Var}(Z) \end{bmatrix}$$

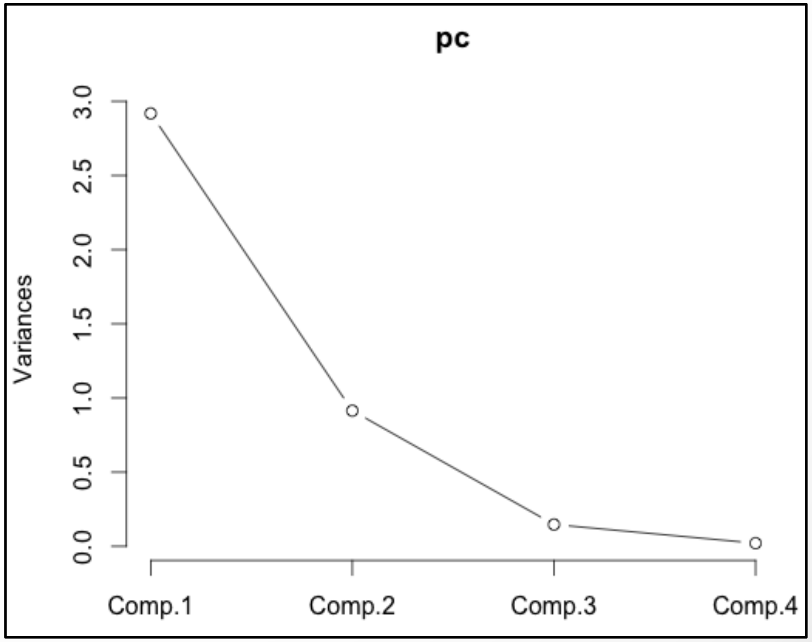
Principal Components

- The first principal component is the eigenvector of the covariance matrix that has the largest eigenvalue
- This vector points into the direction of the largest variance of the data
- The magnitude of this vector equals the corresponding eigenvalue.
- The second largest eigenvector is orthogonal to the largest eigenvector, and points into the direction of the second largest spread of the data.

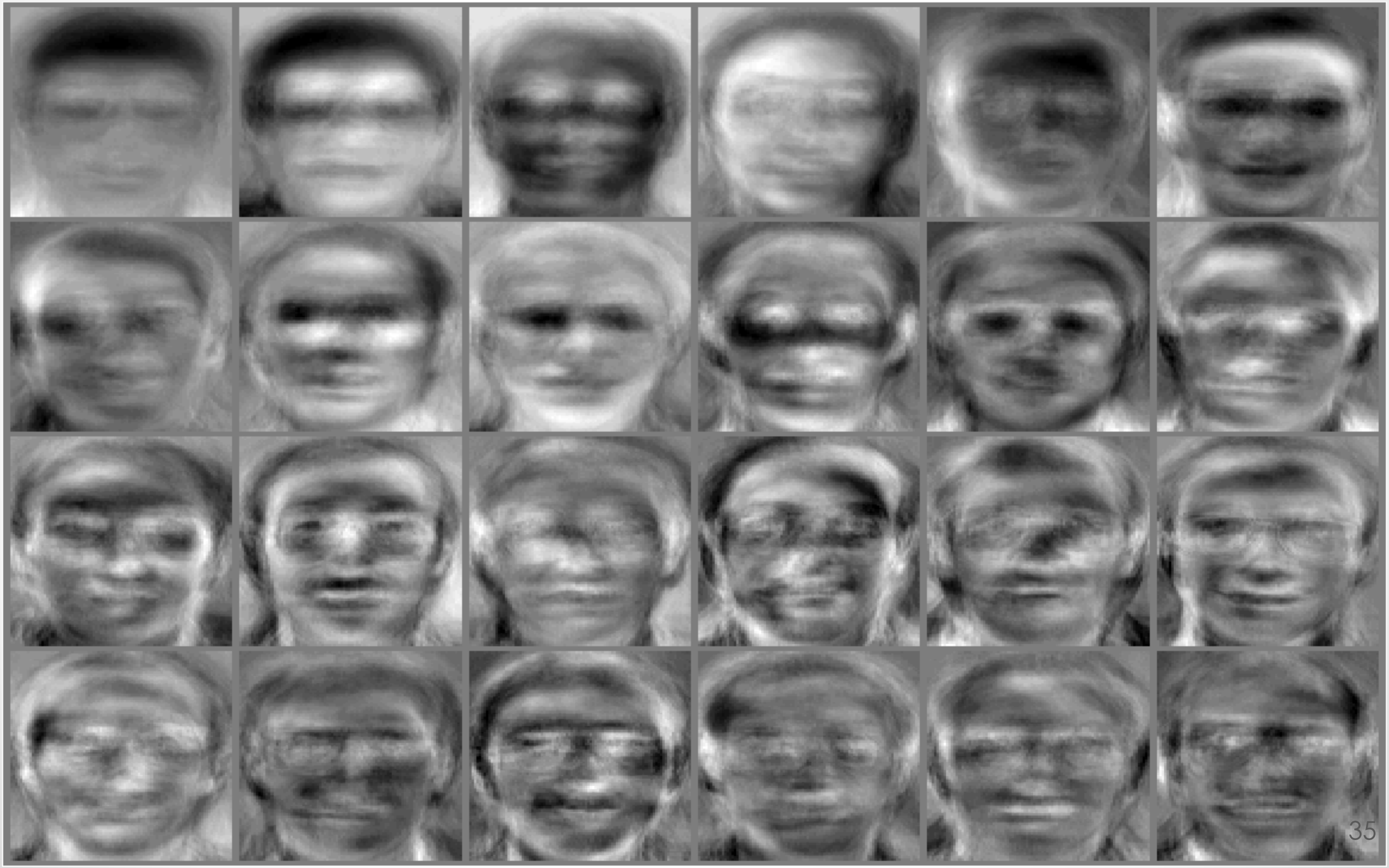
Scree Plot

- Plots the variances against the number of the principal component
- Used to visiually assess which components explain most of the variability
- In R: `fit <- princomp(dataset)`

```
screeplot(fit)
```



Eigenfaces



Multidimensional Scaling

Overview of MDS

- You are given pairwise relationships between cases, e.g.
  - Distance between cities
  - Measures of similarity/dissimilarity
  - Importance
  - Preferences
- MDS lays these cases out as points in an *n*-dimensional space
- Traditional use is with n=2 or n=3 to visualize relationships in the data for exploratory purposes
- In ML we can also use to reduce dimensionality of data

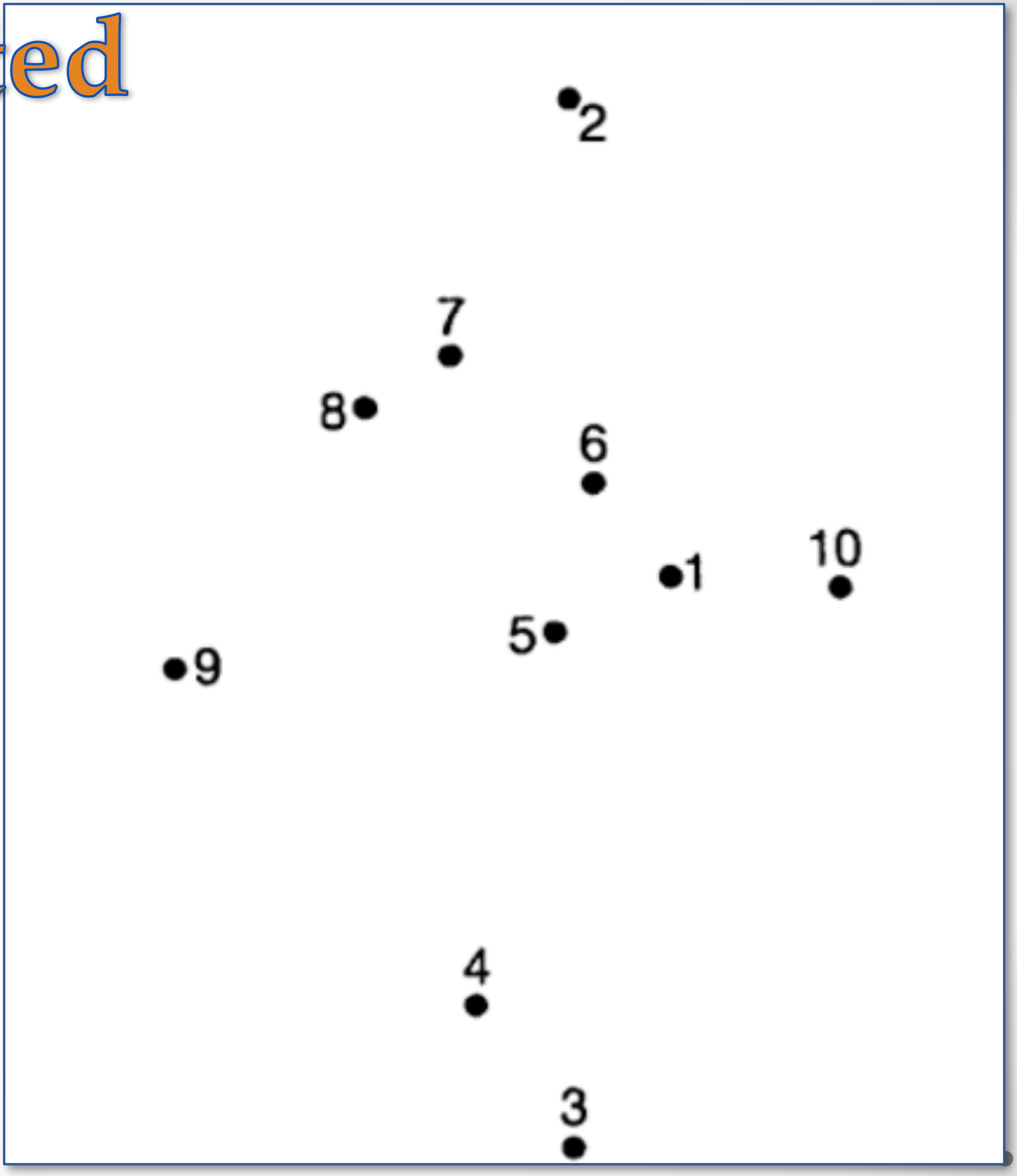
Contrast with PCA

- PCA reduces dimensionality while retaining variance
- MDS
  - Can be used to introduce dimensionality
  - Can be used to reduces dimensionality while retaining the relative distances

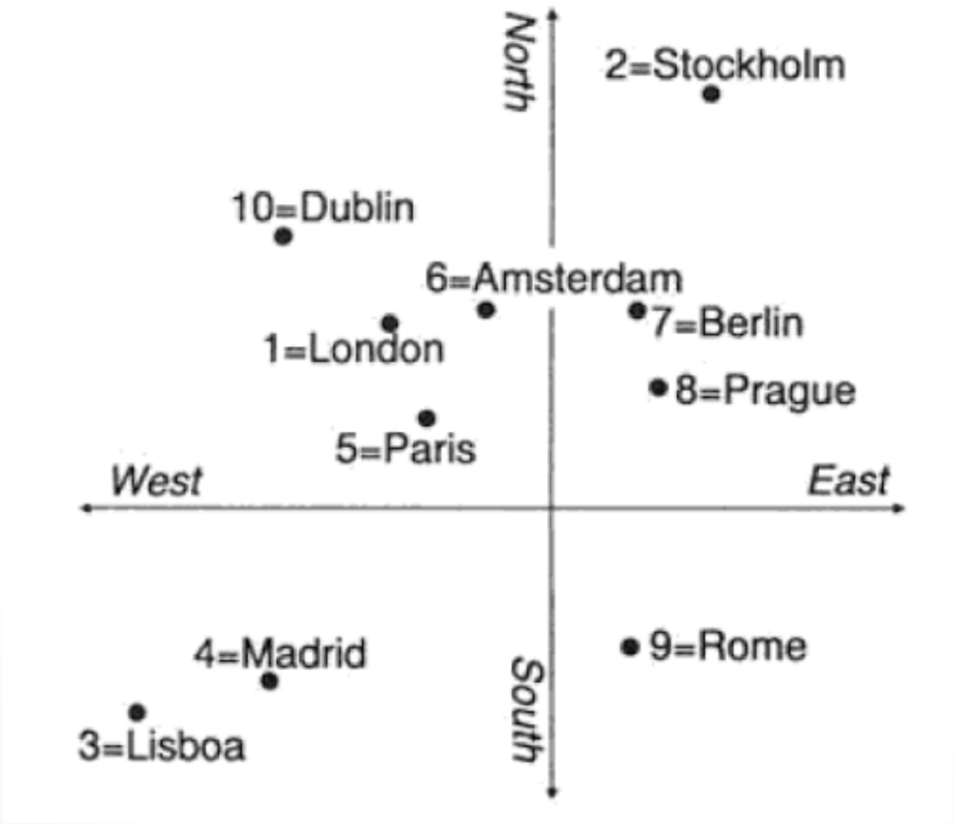
Distances Between Some European Cities

	1	2	3	4	5	6	7	8	9	10
1	0	569	667	530	141	140	357	396	570	190
2	569	0	1212	1043	617	446	325	423	787	648
3	667	1212	0	201	596	768	923	882	714	714
4	530	1043	201	0	431	608	740	690	516	622
5	141	617	596	431	0	177	340	337	436	320
6	140	446	768	608	177	0	218	272	519	302
7	357	325	923	740	340	218	0	114	472	514
8	396	423	882	690	337	272	114	0	364	573
9	569	787	714	526	436	519	472	364	0	755
10	190	648	714	622	320	302	514	573	755	0

Constructed Space



Actual locations



For Dimensionalty Reduction

- Original data matrix – Each column represents a feature. Each of the *N* rows represents a point.
- Create Dissimilarity Matrix storing the **`dist()`** distances d_0 between the points. The R function does this.

$$\begin{pmatrix} d_0(X_1, X_1) & \cdots & d_0(X_N, X_1) \\ \vdots & \ddots & \vdots \\ d_0(X_1, X_N) & \cdots & d_0(X_N, X_N) \end{pmatrix}$$

Performing MDS in R

- ***k*** is number of dimensions you want for the reconstructed space
- ***d*** is full symmetric Dissimilarity Matrix

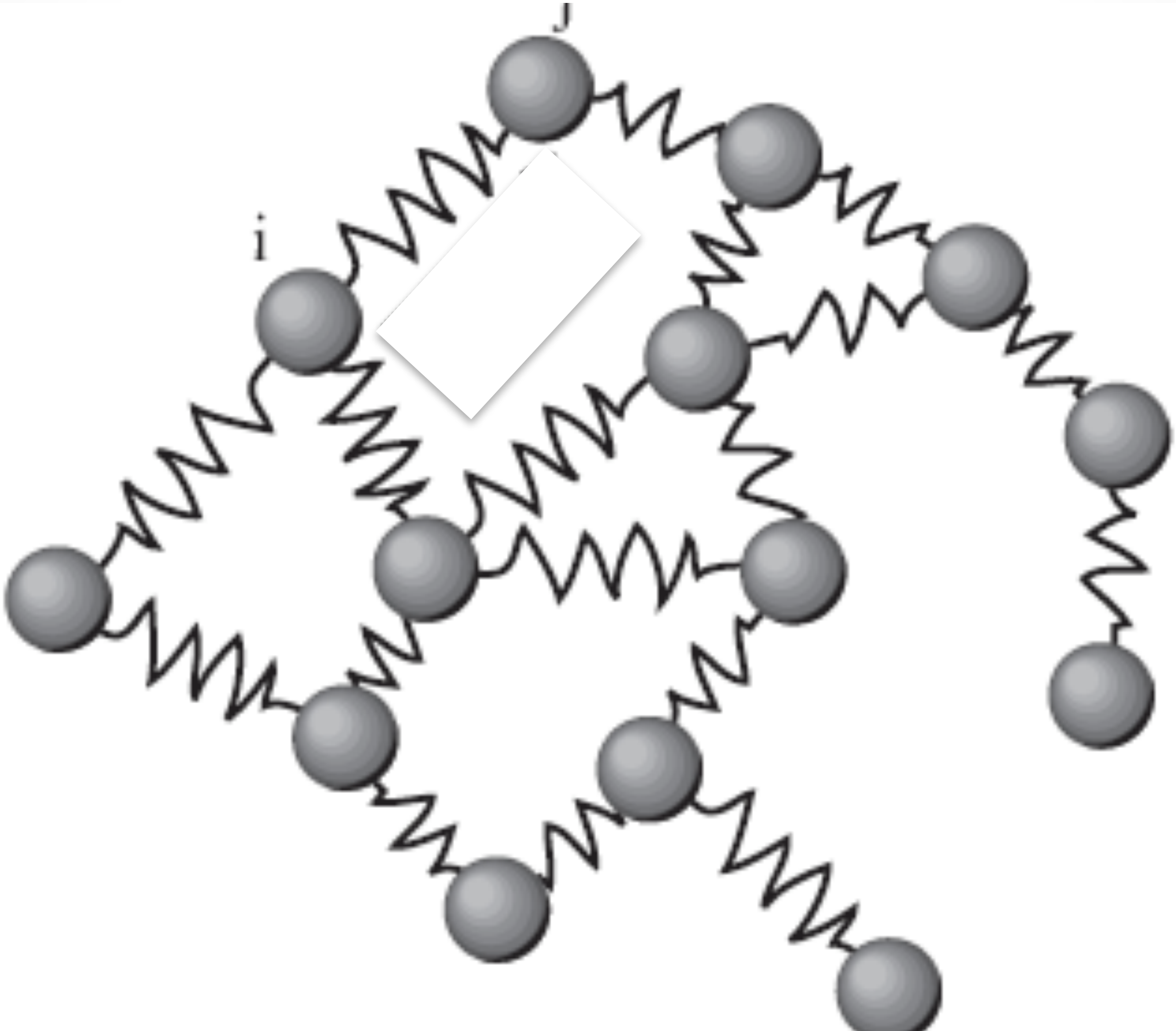
- `cmdscale(d, k=3)`

- The **`c`** in **`cmdscale`** stands for "classical"

Non-Metric MDS

- Exact distances in the reconstructed space can be off a little
- Imagine springs of the original distance between the points. Want to minimize the overall stress

Spring model



Common Stress Metrics

$$STRESS1 = \left(\frac{\sum_{i<j}(d_{ij} - \hat{d}_{ij})^2}{\sum_{i<j} d_{ij}^2} \right)^{\frac{1}{2}}$$

$$STRESS2 = \left(\frac{\sum_{i<j}(d_{ij} - \hat{d}_{ij})^2}{\sum_{i<j}(d_{ij} - \bar{d})^2} \right)^{\frac{1}{2}}$$

NMDS in R

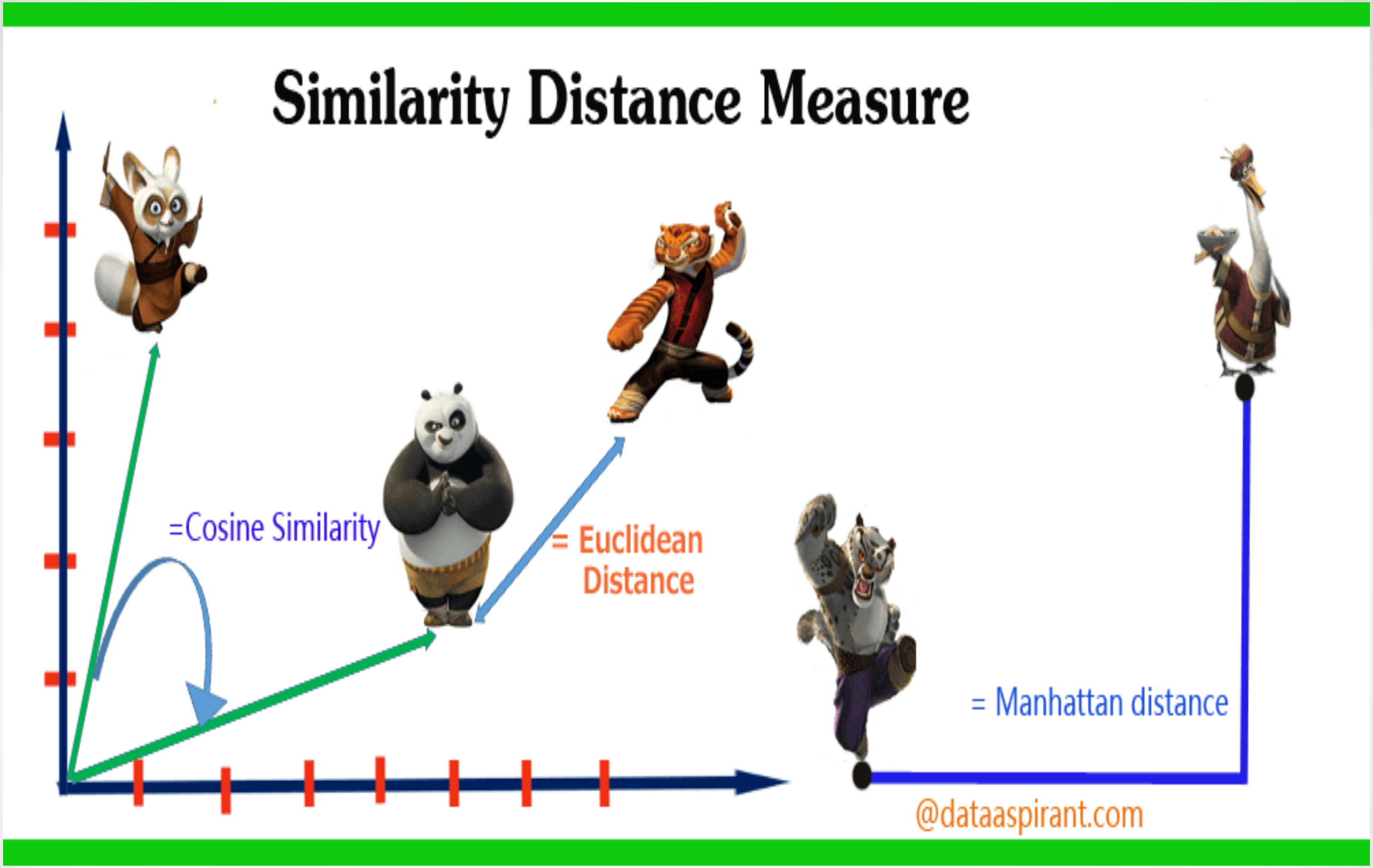
- `dmat` is lower-triangle Dissimilarity Matrix
- `nmds(dmat)`

tsne

t-Distributed Stochastic Neighbor Embedding

Basic Idea

- Maps points in *n* dimensions onto a visualizable 2 or 3 dimensions
- Computes probability distribution of each point being similar to each other point
- Does this for both the hi dim and lo dim representations
- Then minimizes divergence between the high dimension and low dimension distributions

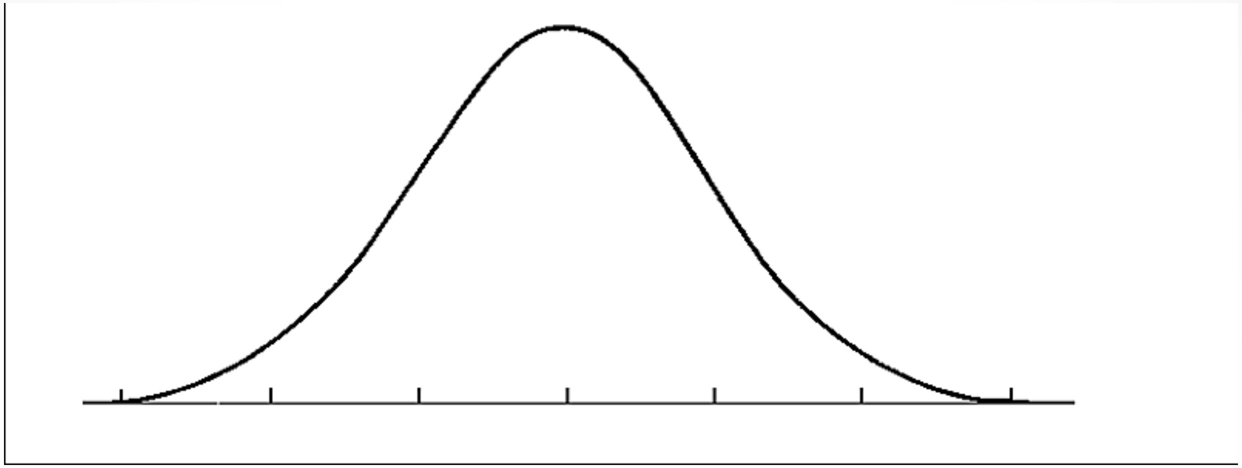


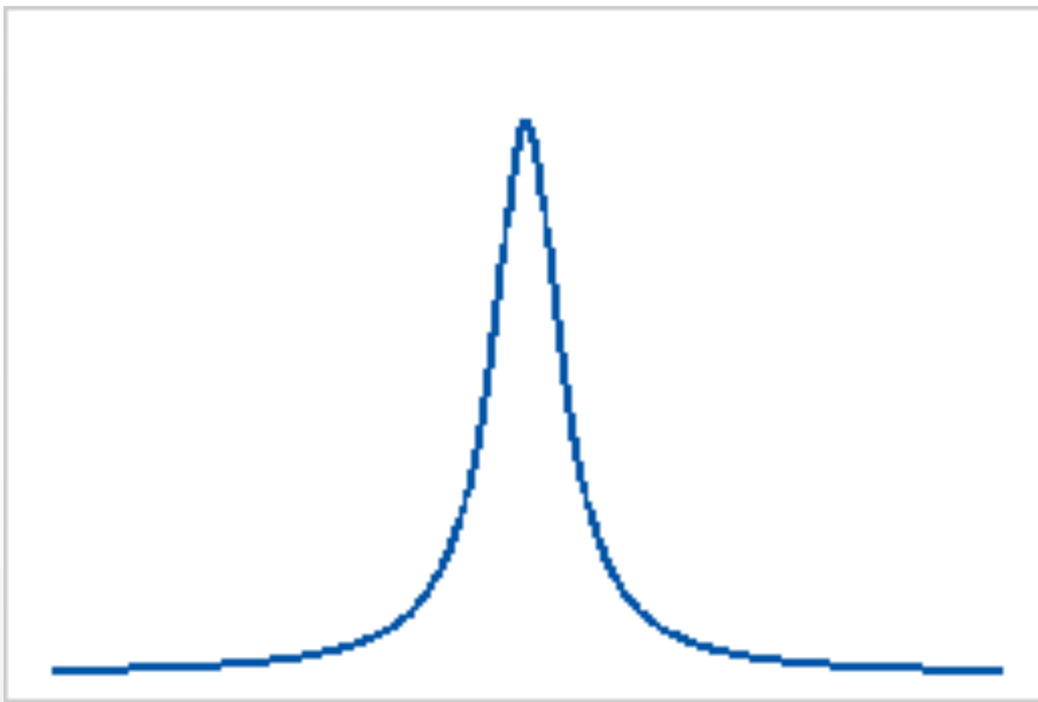
- **Euclidean distance**

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2}.$$

tsne takes a Probabilistic Approach to Similarity

- Model the distance as having a probability distribution
- Treat measured distance the peak of a symmetric bell curve
- Assign lower variances to points that are in denser areas
- Originally the low dim and hi dim representations each used Gaussian distributions
- Refinement of the technique now uses the heavier-tailed Student's *t*-distribution for the lo dim





tsne in R

- Performs tsne dimensionality reduction on an R matrix or a "dist" object

```
tsne(X, initial_config = NULL, k = 2,
    initial_dims = 30, perplexity = 30,
    max_iter = 1000, min_cost = 0,
    epoch_callback = NULL, whiten = TRUE,
    epoch=100)
```

R Demo of tsna and Comparison with PCA

Factor Analysis

Factor Analysis

- Discovers latent factors that influence the features
- Each original feature is a linear combination of the factors