Graph Theory

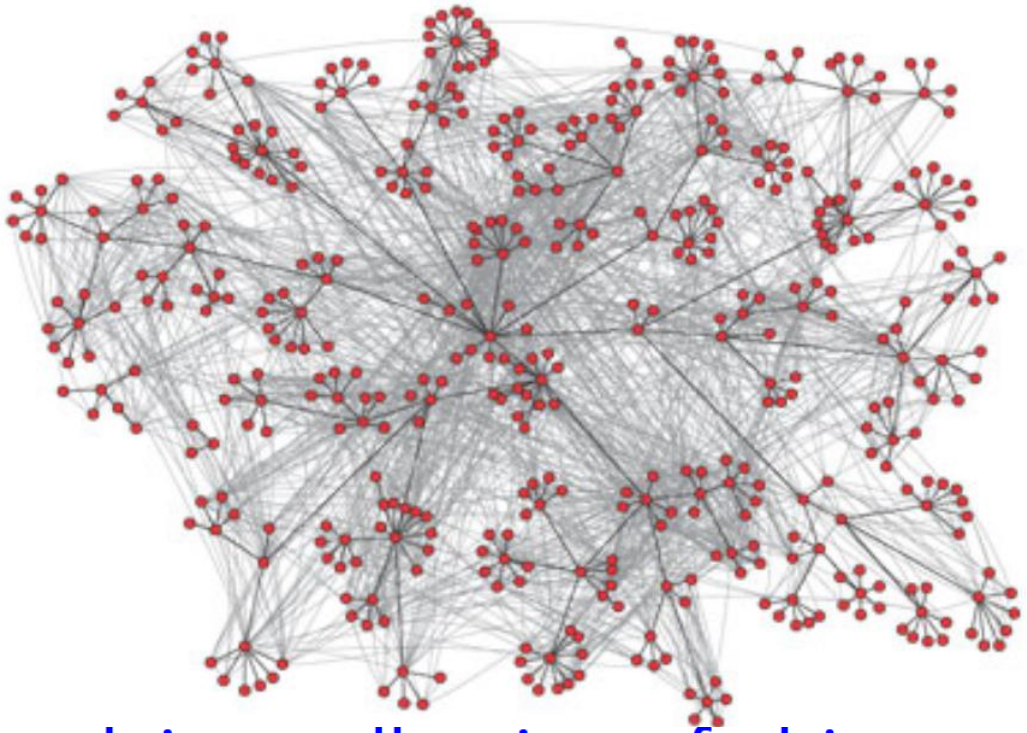
CE642: Social and Economic Networks

Maryam Ramezani
Sharif University of Technology
maryam.ramezani@sharif.edu

01

Introduction

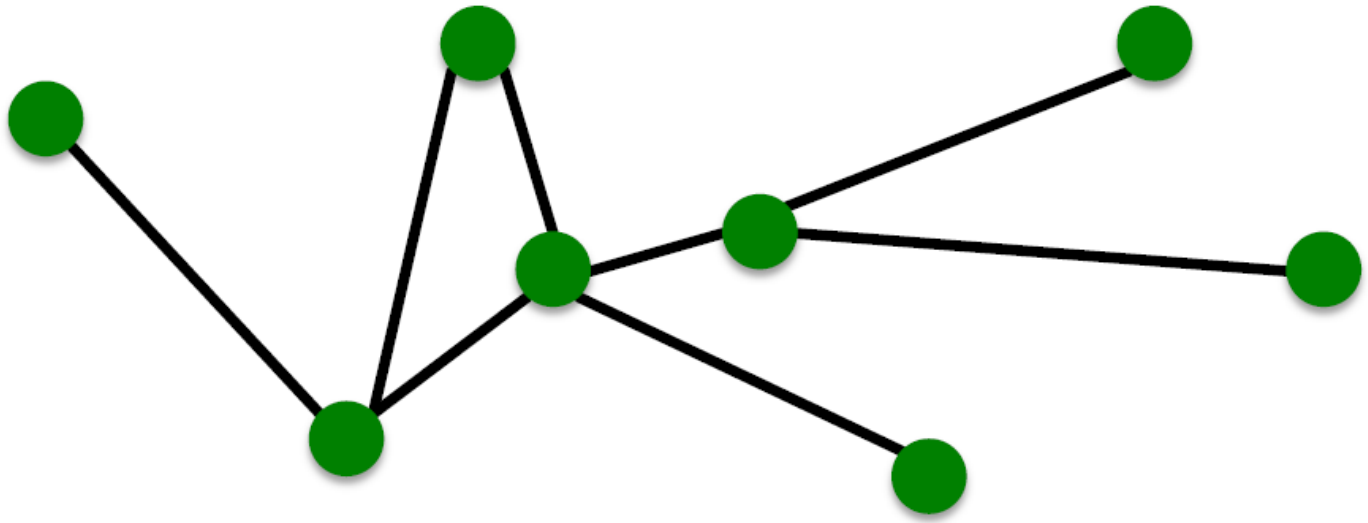
Structure of Networks



A network is a collection of objects where some pairs of objects are connected by links

What is the structure of the network?

Components of a Network



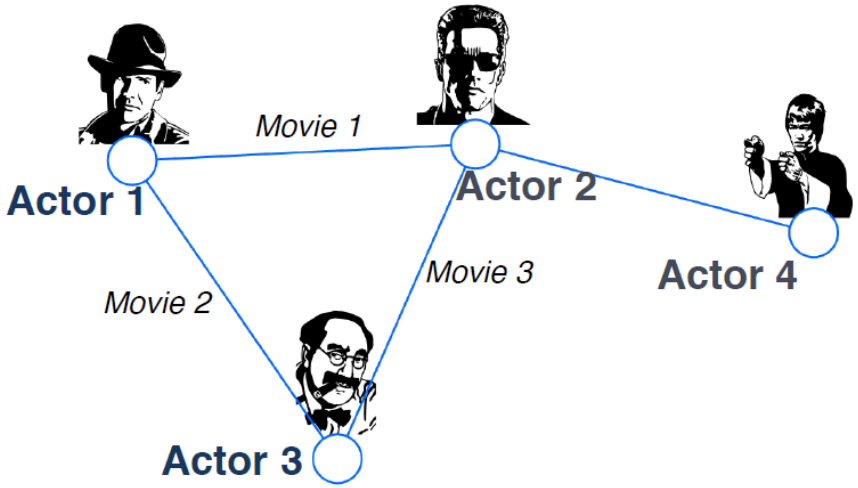
- Objects: nodes, vertices N
- Interactions: links, edges E
- System: network, graph G(N,E)

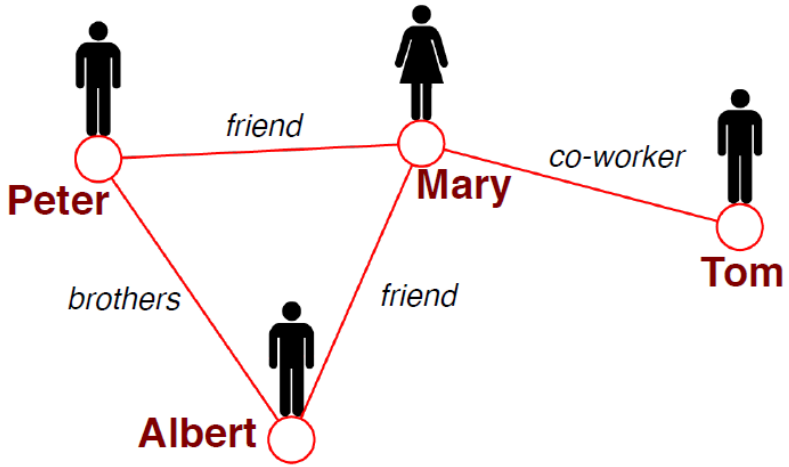
Networks or Graphs?

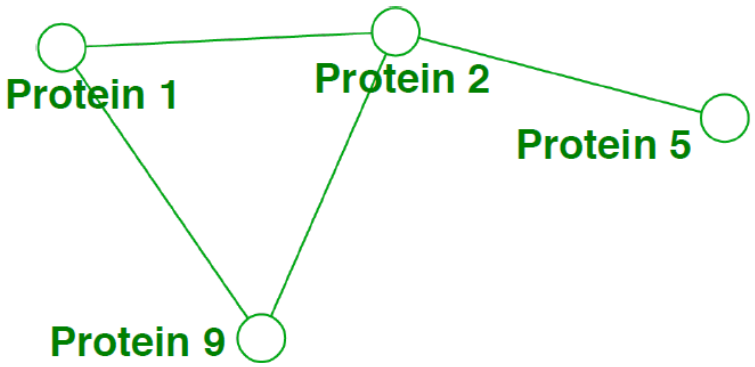
- Network often refers to real systems
  - Web, Social network, Metabolic network
  - Language: Network, node, link
- Graph is a mathematical representation of a network
  - Web graph, Social graph, Knowledge Graph
  - Language: Graph, vertex, edge

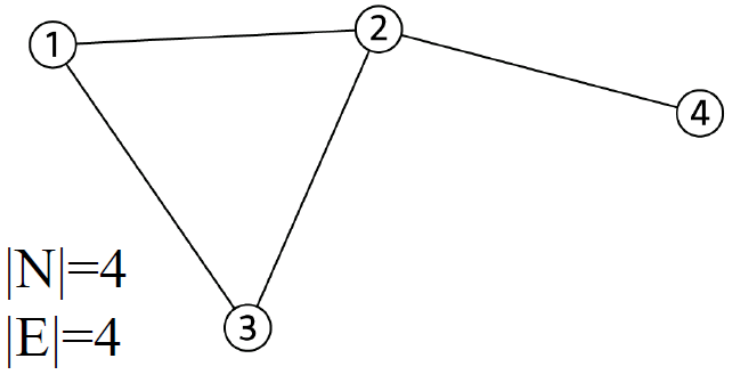
We will try to make this distinction whenever it is appropriate, but in most cases we will use the two terms interchangeably

Networks: Common Language









How do you define a network?

- **How to build a graph:**
  - **What are nodes?**
  - **What are edges?**
- **Choice of the proper network representation of a given domain/problem determines our ability to use networks successfully:**
  - In some cases there is a unique, unambiguous representation
  - In other cases, the representation is by no means unique
  - The way you assign links will determine the nature of the question you can study

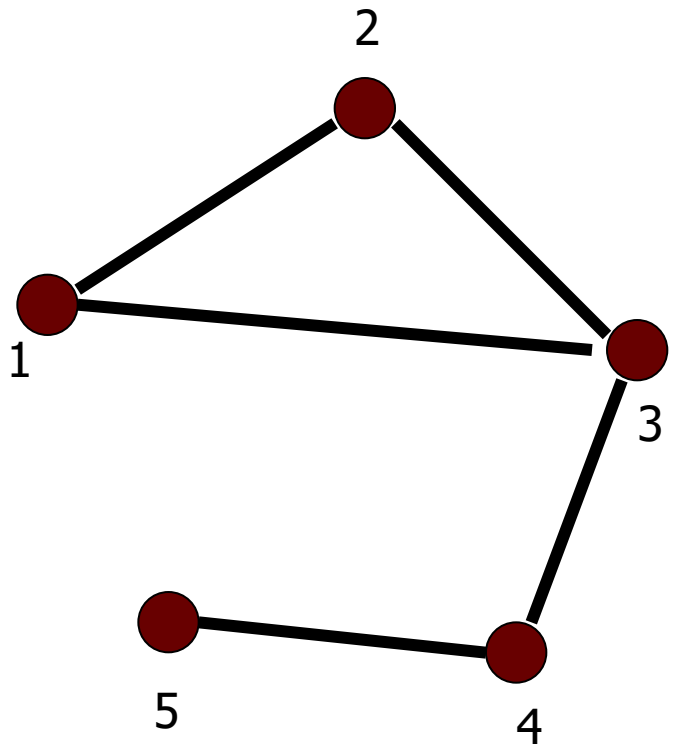
02

Types of Graphs

Undirected Graph

- Graph G=(V,E)
  - V = set of vertices
  - E = set of edges

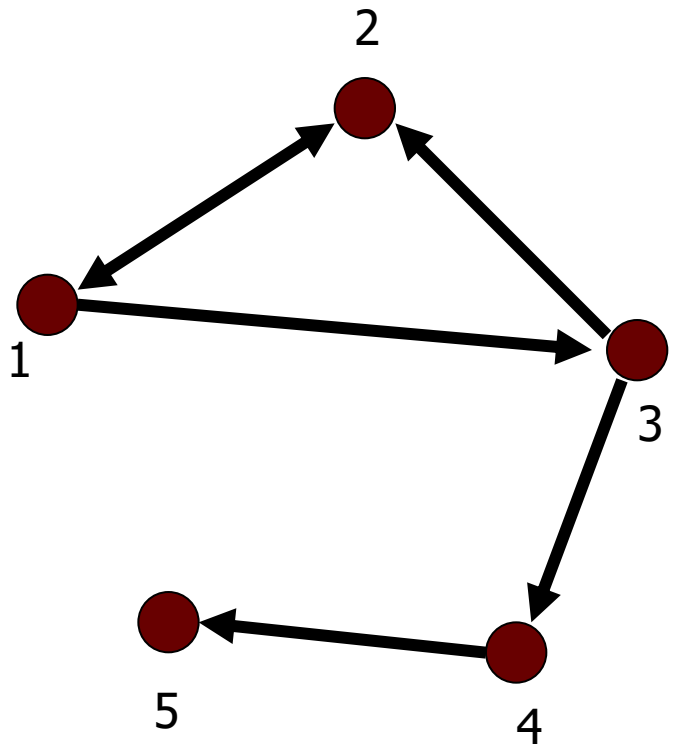
undirected graph
V = {1, 2, 3, 4, 5}
E={(1,2),(1,3),(2,3),(3,4),(4,5)}



Directed Graph

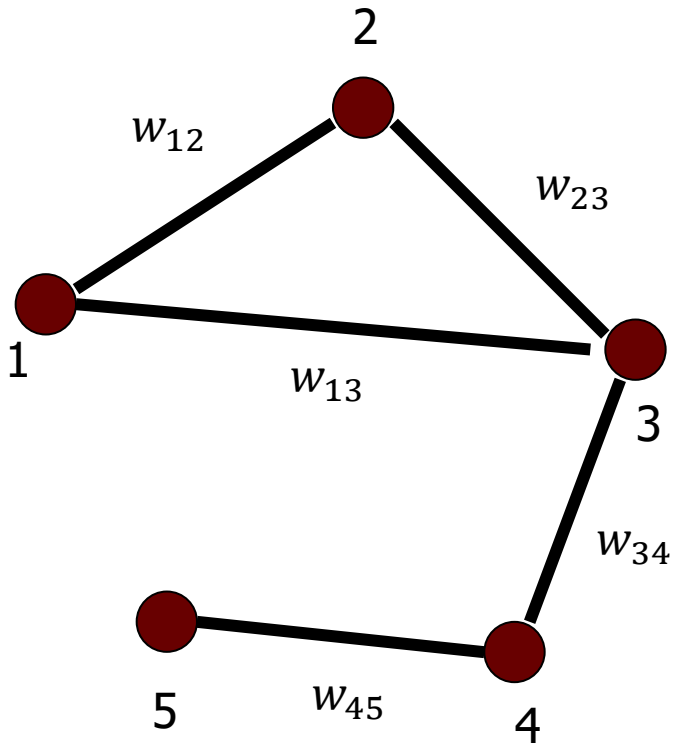
- Graph G=(V,E)
  - V = set of vertices
  - E = set of edges

directed graph
V = {1, 2, 3, 4, 5}
E={‹1,2›, ‹2,1› ‹1,3›, ‹3,2›, ‹3,4›, ‹4,5›}



Weighted Graph

- Graph G=(V,E)
  - V = set of vertices
  - E = set of edges and their weights

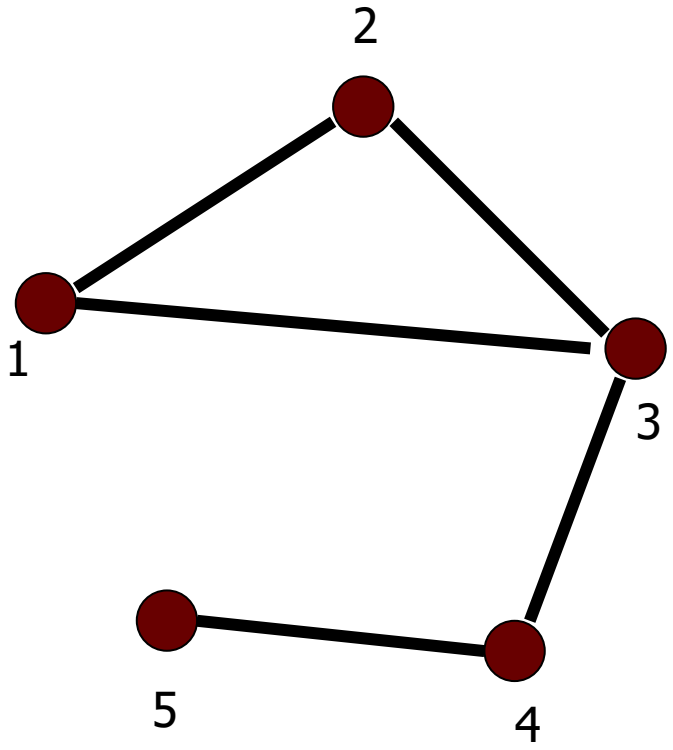


Weights can be either distances or similarities

weighted graph
V = {1, 2, 3, 4, 5}
E={(1,2,w_{12}),(1,3, w_{12}),(2,3, w_{12}),(3,4, w_{12}),(4,5, w_{12})}

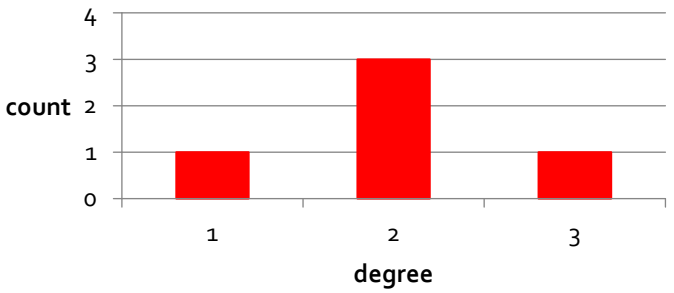
Undirected graph

- Neighborhood N(i) of node i
  - Set of nodes adjacent to i
- degree d(i) of node i
  - Size of N(i)
  - number of edges incident on i

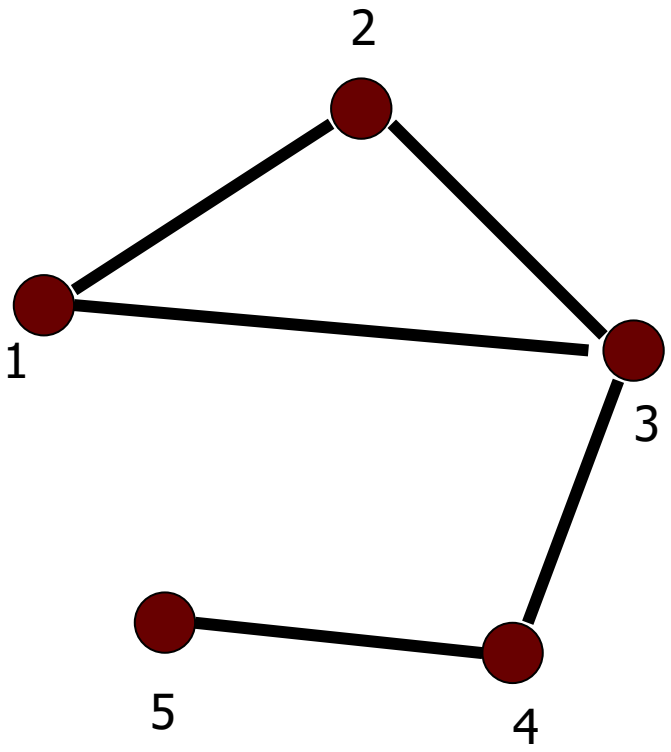


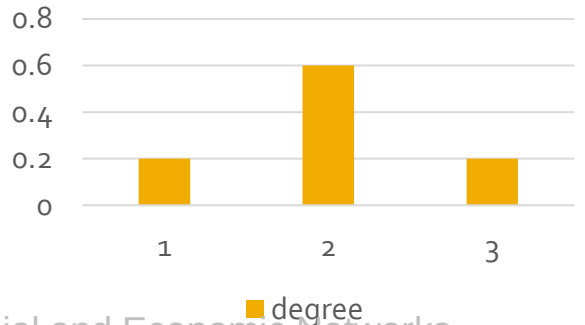
Undirected graph

- degree sequence
  - [d(1),d(2),d(3),d(4),d(5)]
  - [2,2,3,2,1]
- degree histogram
  - [(1:1),(2:3),(3,1)]



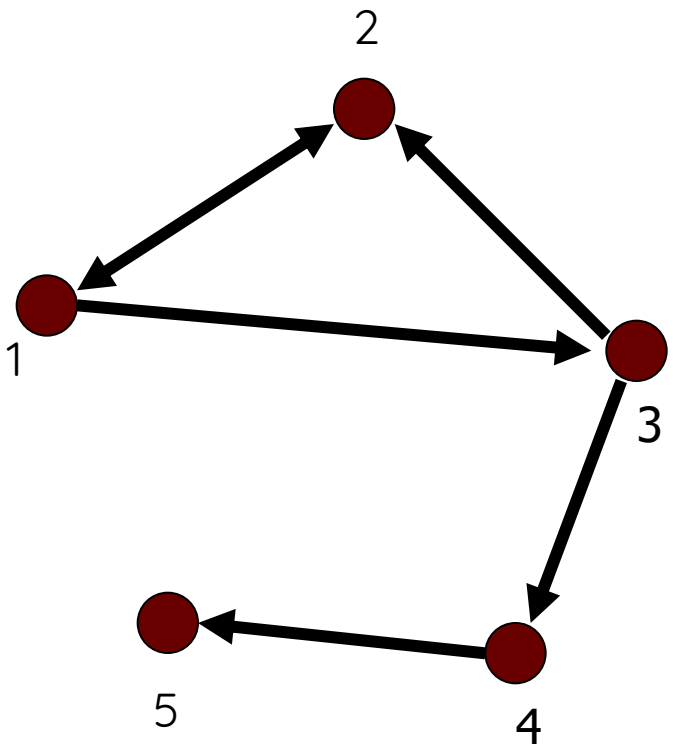
- degree distribution
  - [(1:0.2),(2:0.6),(3,0.2)]





Directed Graph

- in-degree $d_{in}(i)$ of node i
  - number of edges incoming to node i
- out-degree $d_{out}(i)$ of node i
  - number of edges leaving node i

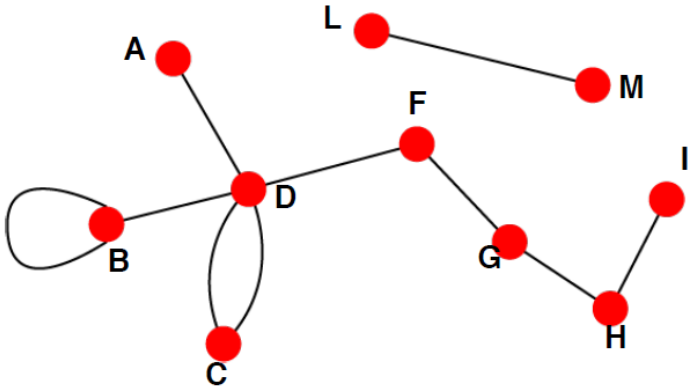


- in-degree sequence
  - [1,2,1,1,1]
- out-degree sequence
  - [2,1,2,1,0]
- in-degree histogram
  - [(1:4),(2:1)]
- out-degree histogram
  - [(0:1),(1:2),(2:2)]

Directed vs Undirected Graphs

Undirected

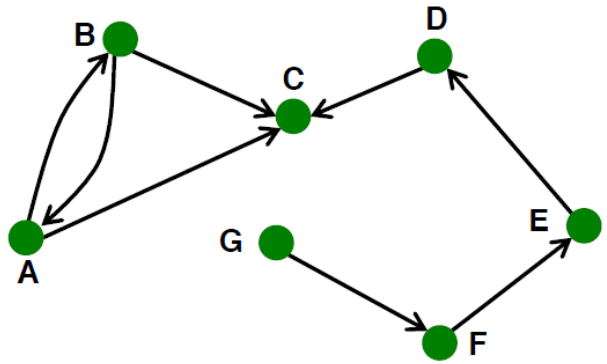
- **Links:** undirected (symmetrical, reciprocal)



- **Examples:**
  - Collaborations
  - Friendship on Facebook

Directed

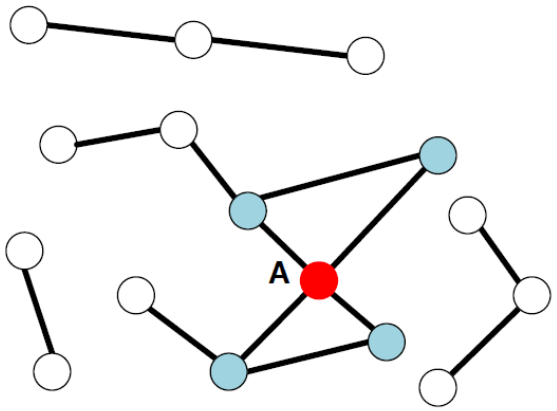
- **Links:** directed (arcs)



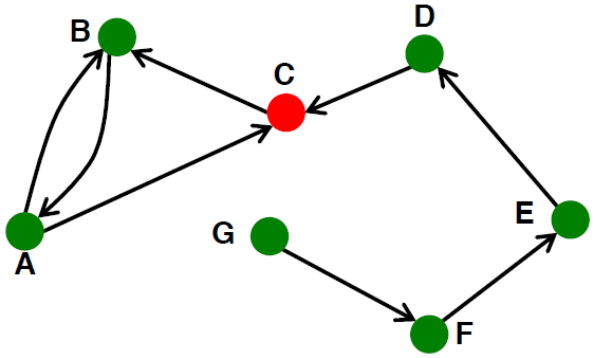
- **Examples:**
  - Phone calls
  - Following on Twitter

Node Degrees

Undirected



Directed



Source: Node with $k^{in} = 0$

Sink: Node with $k^{out} = 0$

Node degree, k_i: the number of edges adjacent to node ***i***

$$k_A = 4$$

Avg. degree: $\bar{k} = \langle k \rangle = \frac{1}{N}\sum_{i=1}^{N} k_i = \frac{2E}{N}$

In directed networks we define an **in-degree** and **out-degree.**

The (total) degree of a node is the sum of in- and out-degrees.

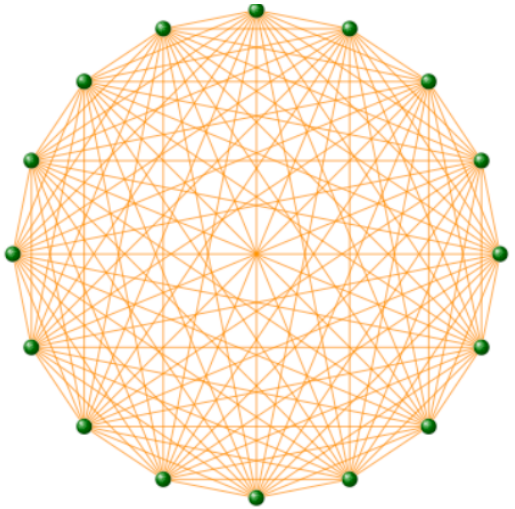
$$k_C^{in} = 2 \qquad k_C^{out} = 1 \qquad k_C = 3$$

$$\bar{k} = \frac{E}{N} \qquad \overline{k^{in}} = \overline{k^{out}}$$

Complete Graph

The **maximum number of edges** in an undirected graph on N nodes is

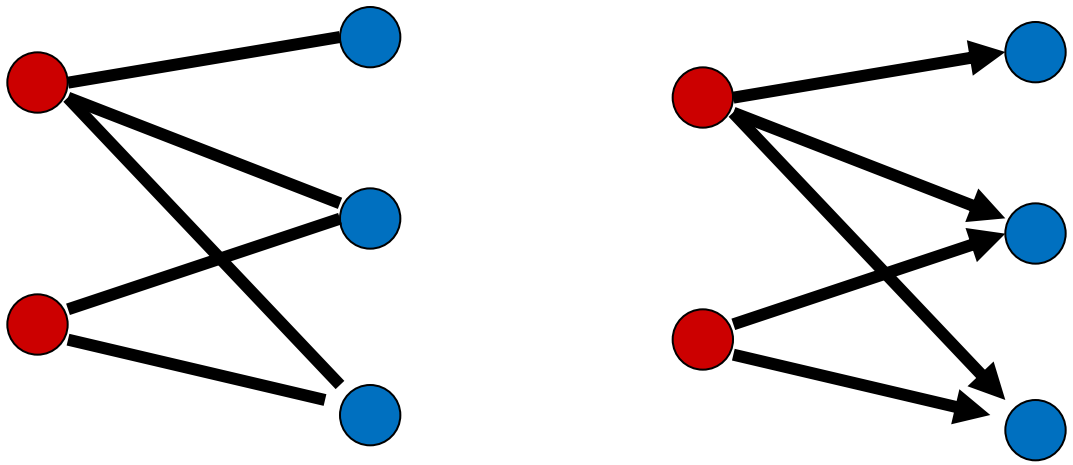
$$E_{\max} = \binom{N}{2} = \frac{N(N-1)}{2}$$



An undirected graph with the number of edges $E = E_{max}$ is called a **complete graph**, and its average degree is $N-1$

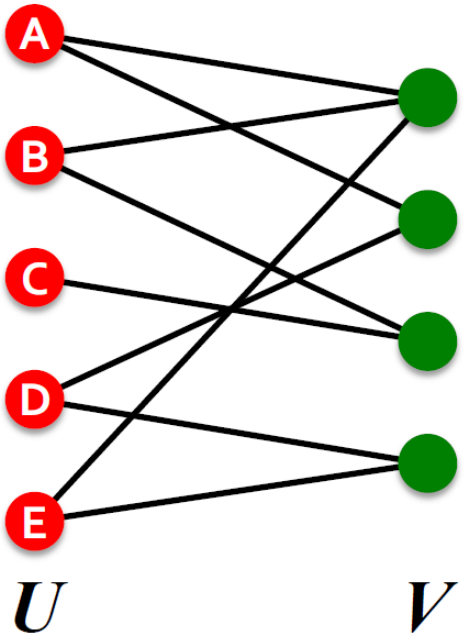
Bipartite graphs

- Graphs where the set of nodes V can be partitioned into two sets L and R, such that there are edges only between nodes in L and R, and there is no edge within L or R



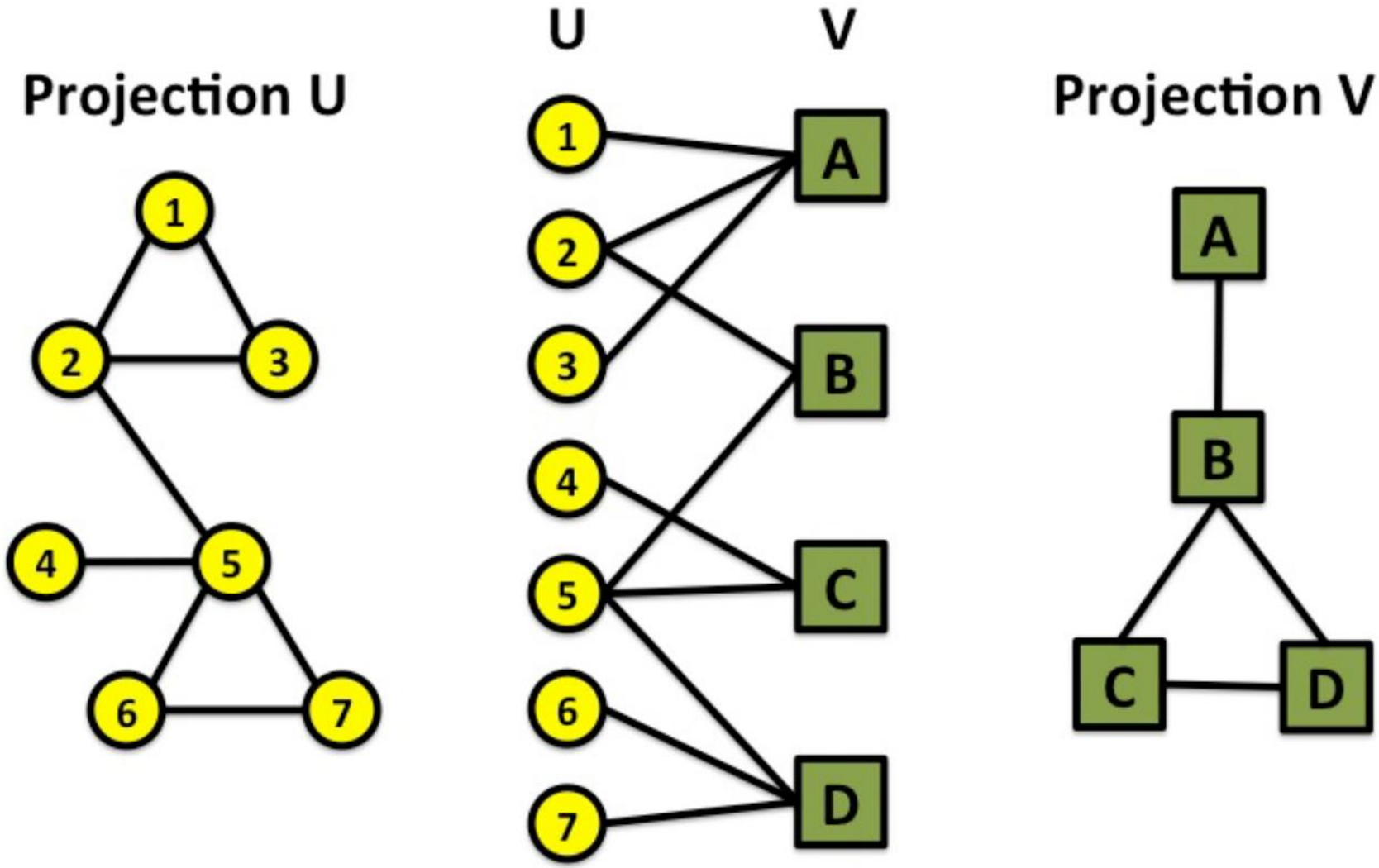
Bipartite Graph

- **Bipartite graph** is a graph whose nodes can be divided into two disjoint sets U and V such that every link connects a node in U to one in V; that is, U and V are **independent sets**



- **Examples:**
  - Authors-to-Papers (they authored)
  - Actors-to-Movies (they appeared in)
  - Users-to-Movies (they rated)
  - Recipes-to-Ingredients (they contain)
- **"Folded" networks:**
  - Author collaboration networks
  - Movie co-rating networks

Folded/Projected Bipartite Graph



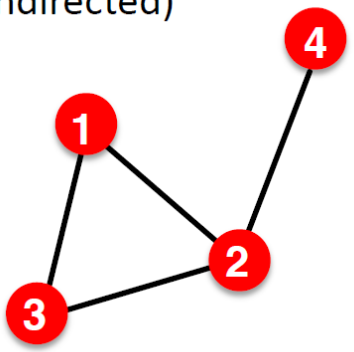
Edge Attributes

Possible options:

- Weight (e.g. frequency of communication)
- Ranking (best friend, second best friend…)
- Type (friend, relative, co-worker)
- Sign: Friend vs. Foe, Trust vs. Distrust
- Properties depending on the structure of the rest of the graph: number of common friends

More Types of Graphs

- **Unweighted**
  (undirected)



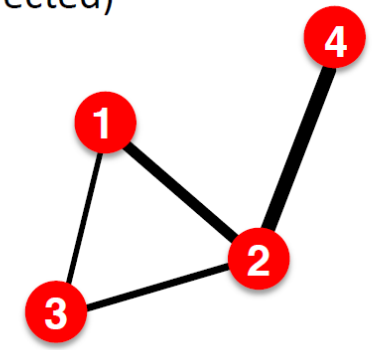
$$A_{ij} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

$$A_{ii} = 0 \qquad A_{ij} = A_{ji}$$

$$E = \frac{1}{2}\sum_{i,j=1}^{N} A_{ij} \qquad \bar{k} = \frac{2E}{N}$$

Examples: Friendship, Hyperlink

- **Weighted**
  (undirected)



$$A_{ij} = \begin{pmatrix} 0 & 2 & 0.5 & 0 \\ 2 & 0 & 1 & 4 \\ 0.5 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{pmatrix}$$

$$A_{ii} = 0 \qquad A_{ij} = A_{ji}$$

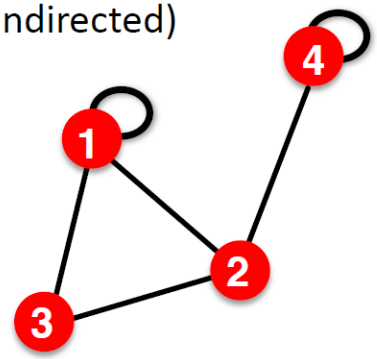
$$E = \frac{1}{2}\sum_{i,j=1}^{N} nonzero(A_{ij}) \qquad \bar{k} = \frac{2E}{N}$$

Examples: Collaboration, Internet, Roads

More Types of Graphs

- **Self-edges (self-loops)**

(undirected)



$$A_{ij} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

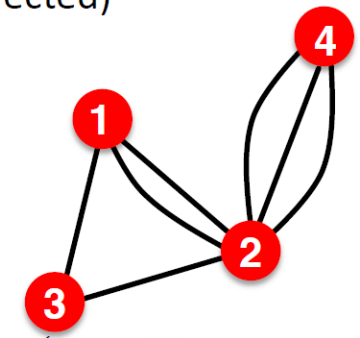
$$A_{ii} \neq 0 \qquad A_{ij} = A_{ji}$$

$$E = \frac{1}{2}\sum_{i,j=1,i\neq j}^{N} A_{ij} + \sum_{i=1}^{N} A_{ii}$$

Examples: Proteins, Hyperlinks

- **Multigraph**

(undirected)

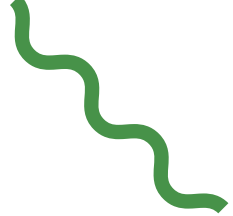


$$A_{ij} = \begin{pmatrix} 0 & 2 & 1 & 0 \\ 2 & 0 & 1 & 3 \\ 1 & 1 & 0 & 0 \\ 0 & 3 & 0 & 0 \end{pmatrix}$$

$$A_{ii} = 0 \qquad A_{ij} = A_{ji}$$

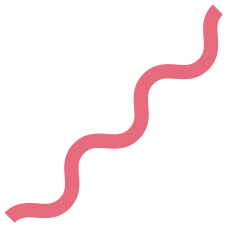
$$E = \frac{1}{2}\sum_{i,j=1}^{N} nonzero(A_{ij}) \qquad \bar{k} = \frac{2E}{N}$$

Examples: Communication, Collaboration

03

Graph Traversals

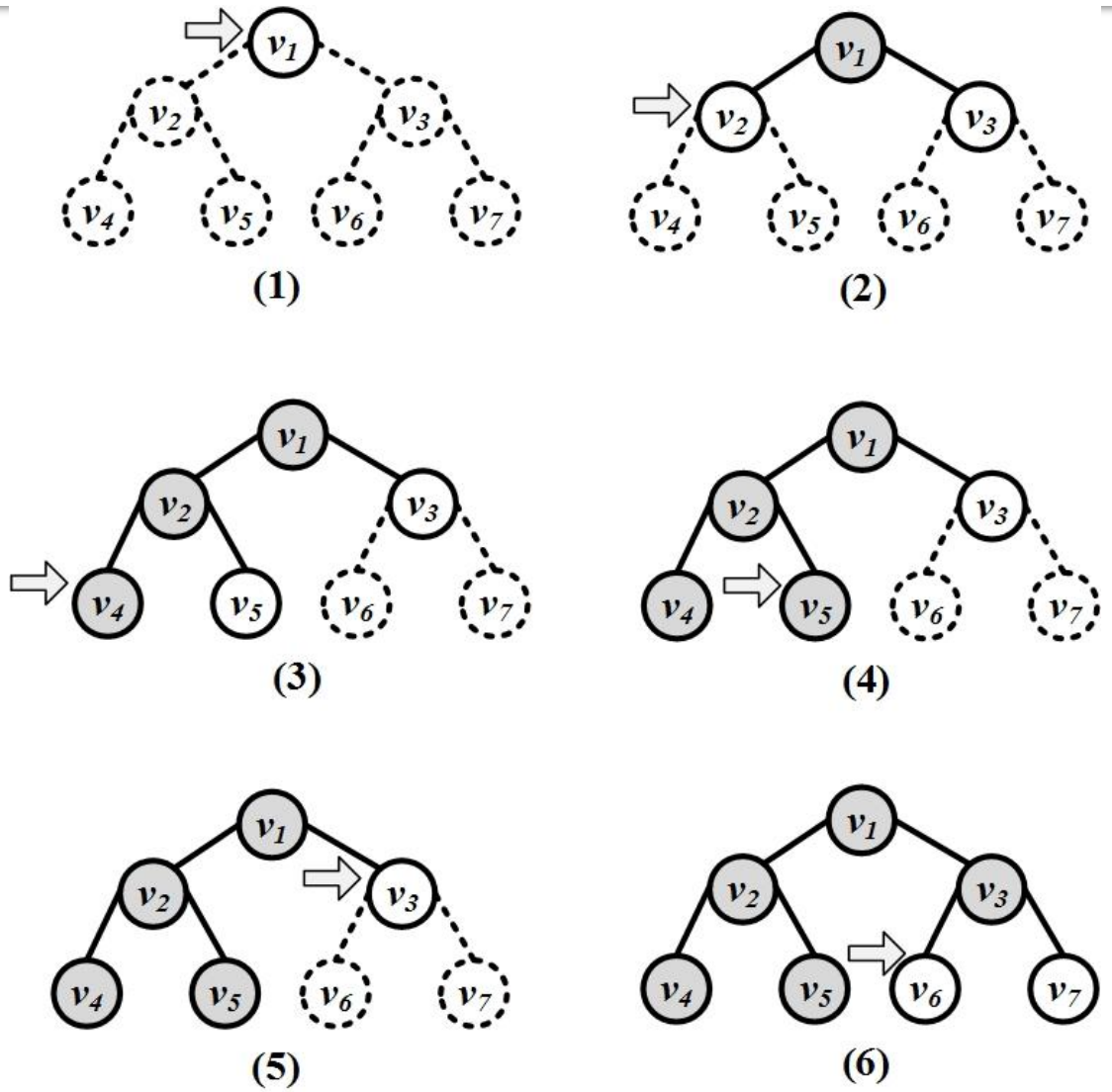
Graph Traversals

- A traversal is a procedure for visiting (going through) all the nodes in a graph:
  - Depth First Search (DFS)
  - Breadth First Search (BFS)

Depth First Search Traversal

- Depth-First Search (DFS) starts from a node i, selects one of its neighbors j from N(i) and performs Depth-First Search on j before visiting other neighbors in N(i).
  - The algorithm can be implemented using a *stack structure*

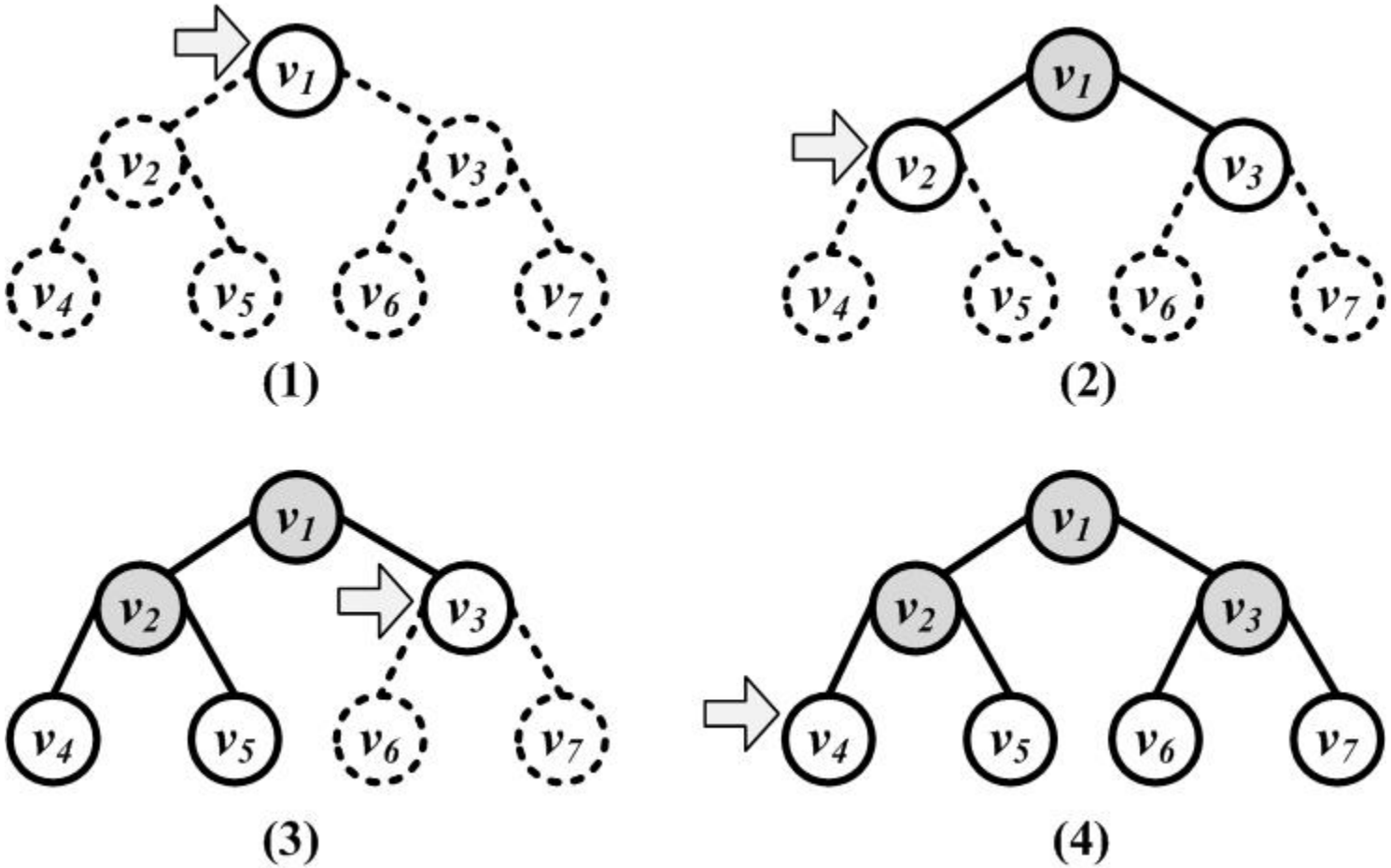
Example of DFS



Breadth First Search Traversal

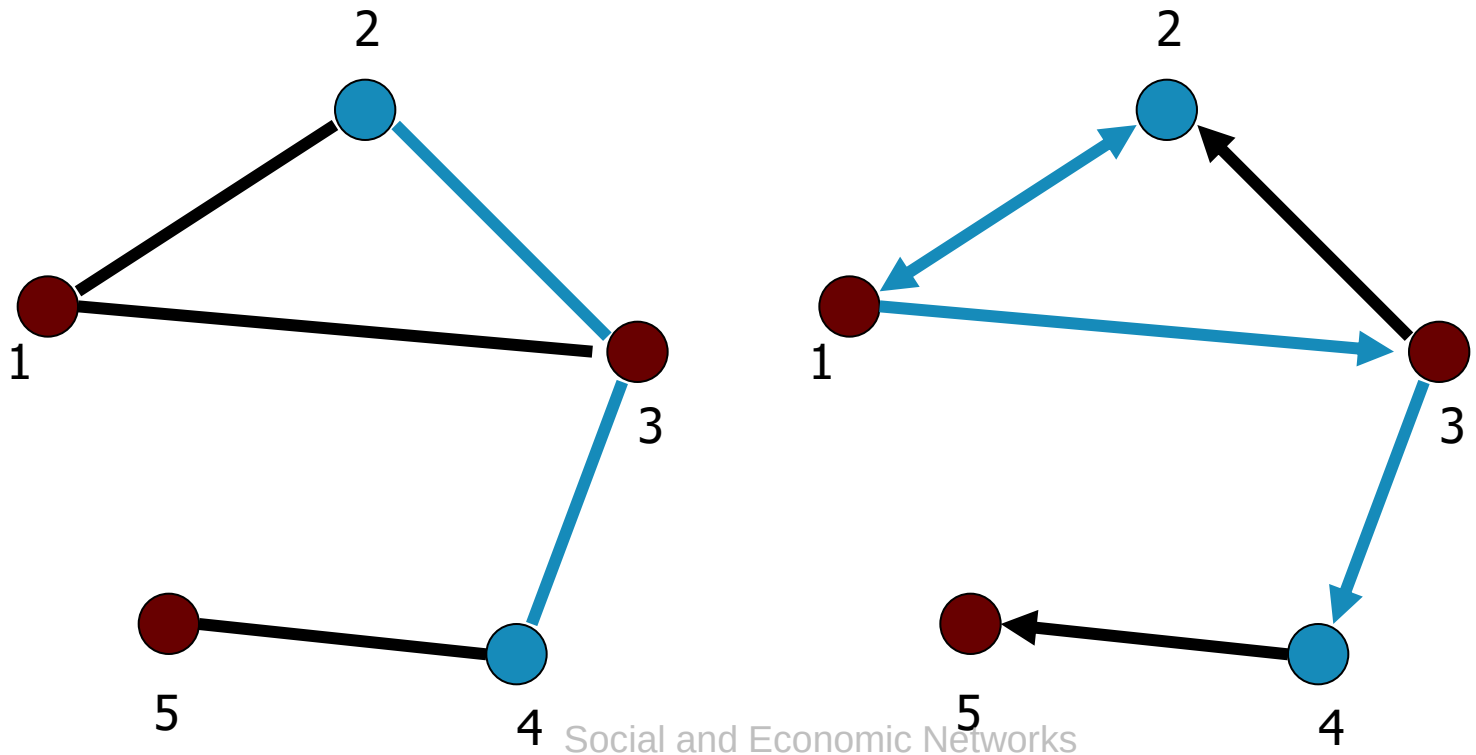
- Breadth-First-Search (BFS) starts from a node, visits all its immediate neighbors first, and then moves to the second level by traversing their neighbors.
  - The algorithm can be implemented using a *queue structure*

Example of BFS



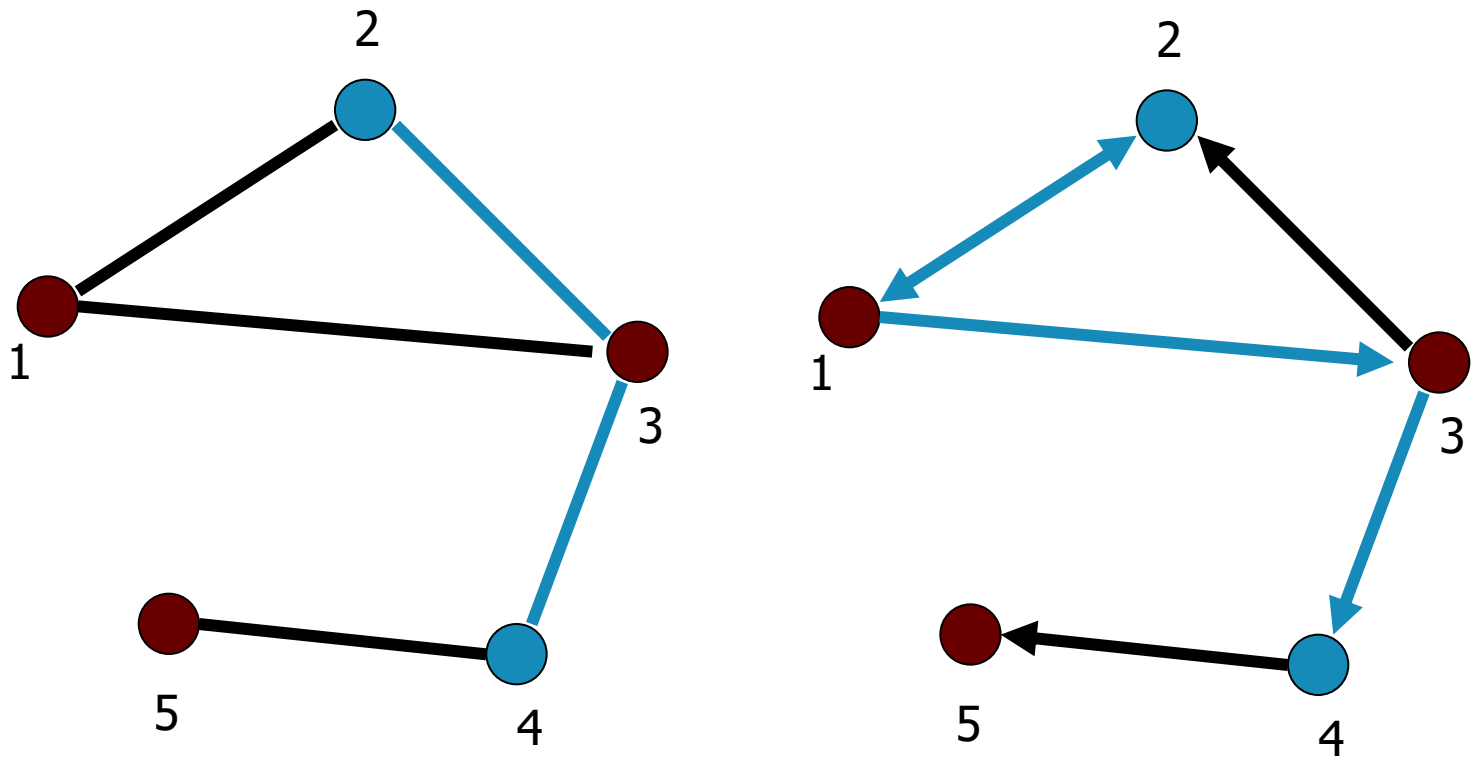
Paths

- Path from node i to node j: a sequence of edges (directed or undirected from node i to node j)
  - **path length**: number of edges on the path nodes i and j are connected
  - **cycle**: a path that starts and ends at the same node



Shortest Paths

- Shortest Path from node i to node j
  - also known as BFS path, or geodesic path

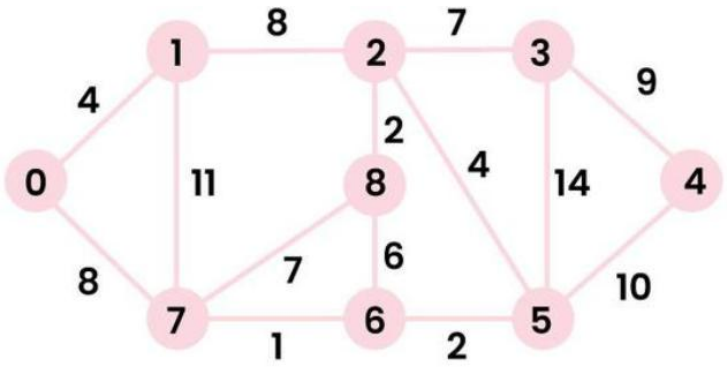


Shortest paths on weighted graphs

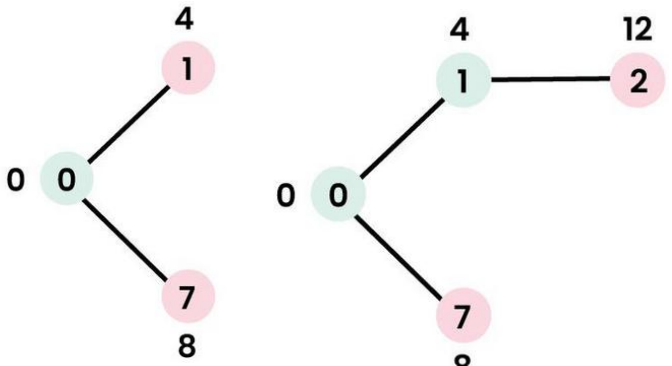
- Shortest paths on weighted graphs are harder to construct
  - There are several well known algorithms for finding single-source, or all-pairs shortest paths
  - For example: Dijkstra's Algorithm

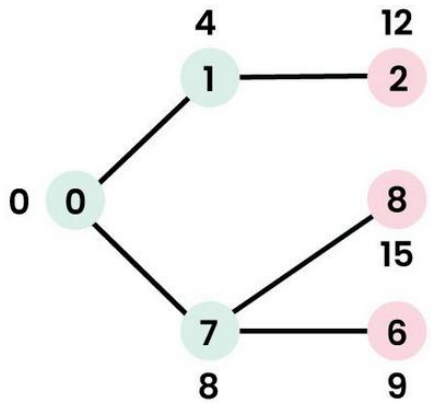
Dijkstra's Algorithm

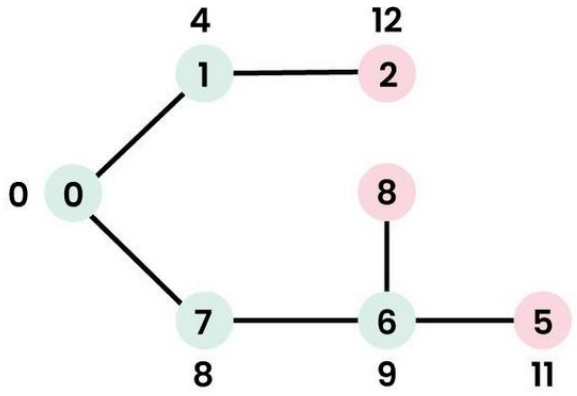
- To understand the Dijkstra's Algorithm lets take a graph and find the shortest path from source to all nodes. Consider below graph and src = 0.

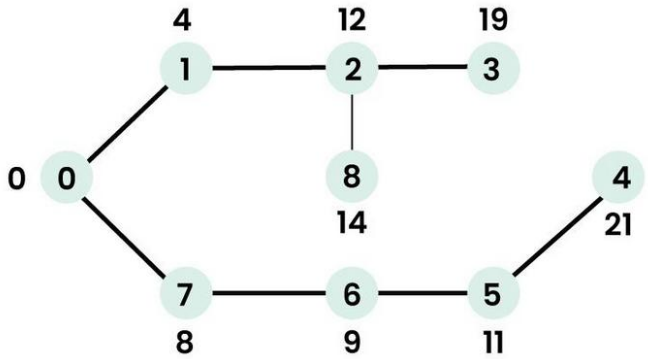


sptSet={0, INF, INF, INF, INF, INF, INF, INF}



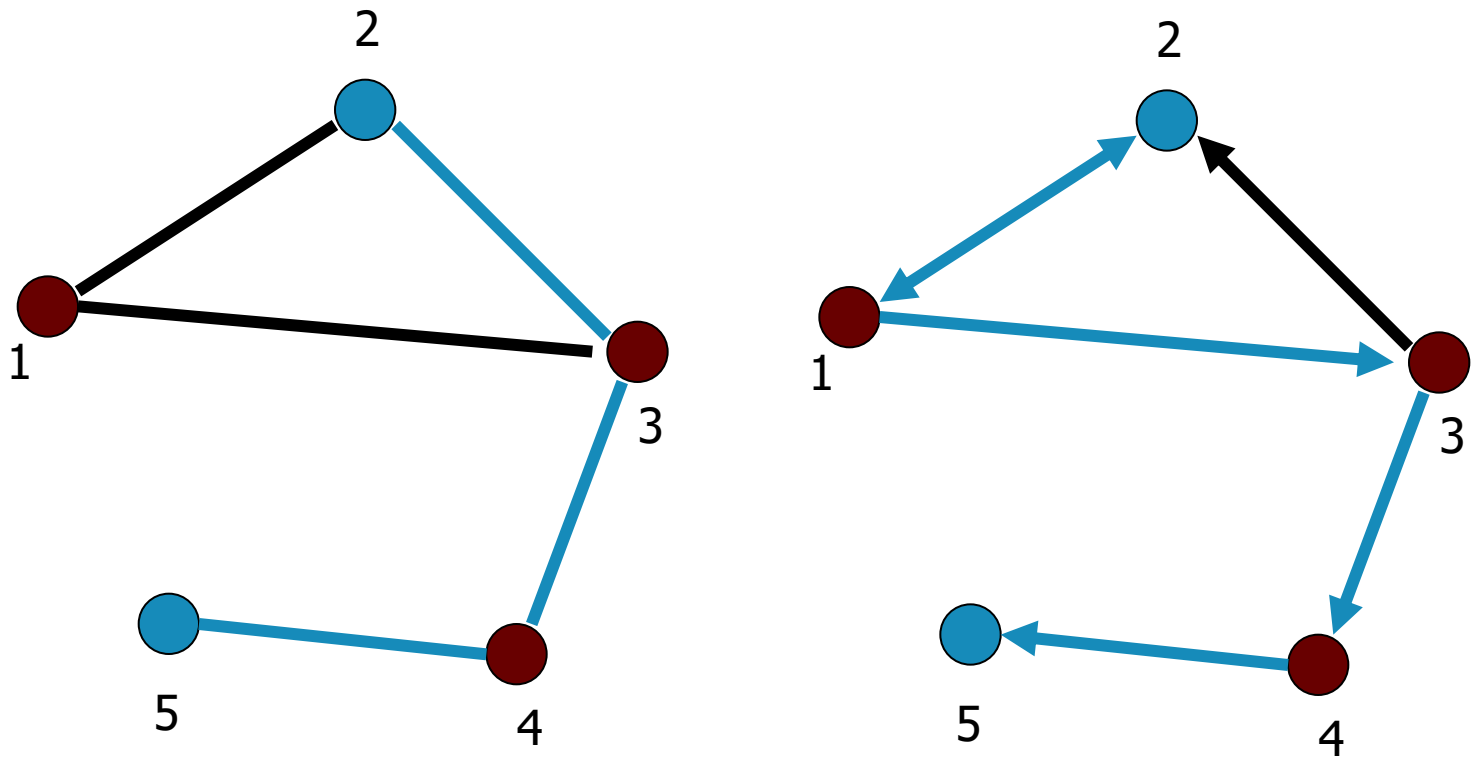






Diameter

- The longest shortest path in the graph

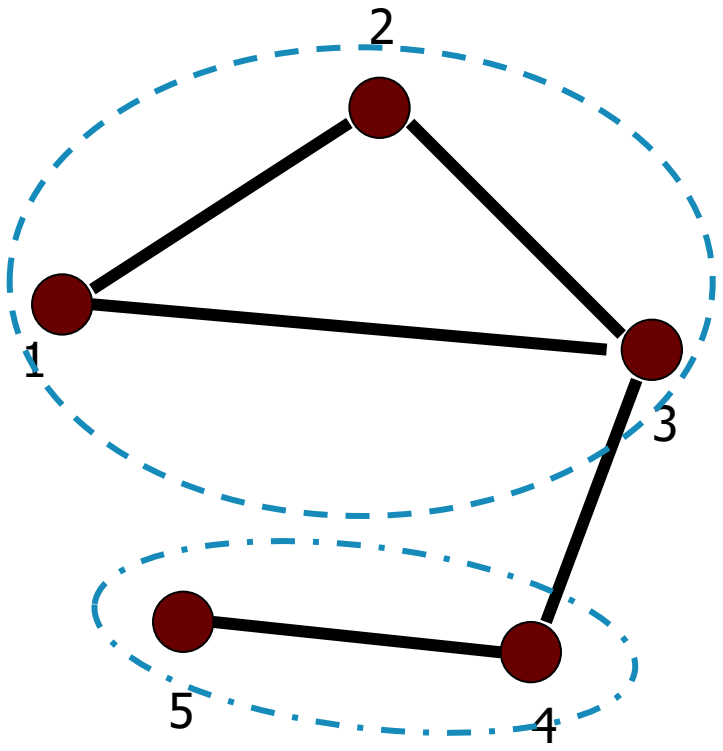


04

Graph Connectivity

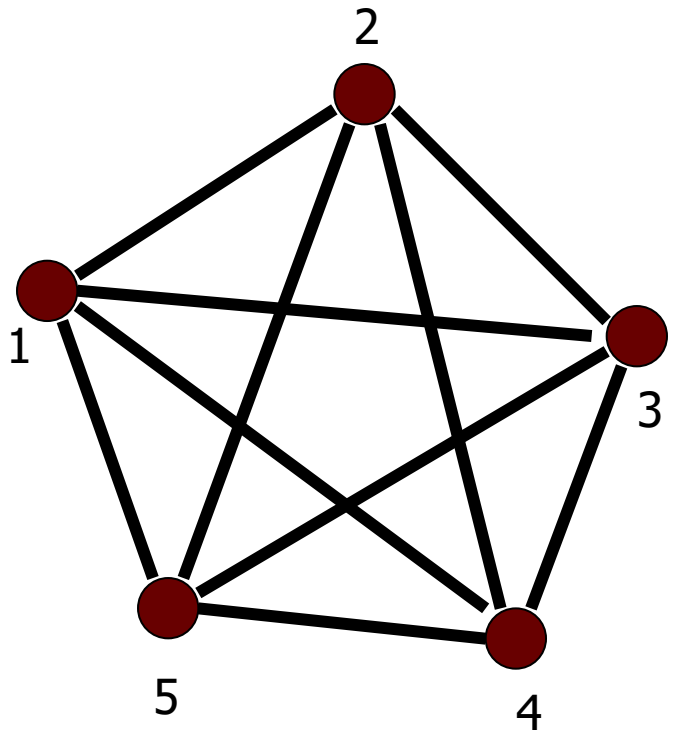
Undirected graph

- **Connected graph**: a graph where there every pair of nodes is connected
- **Disconnected graph**: a graph that is not connected
- **Connected Components**: subsets of vertices that are connected



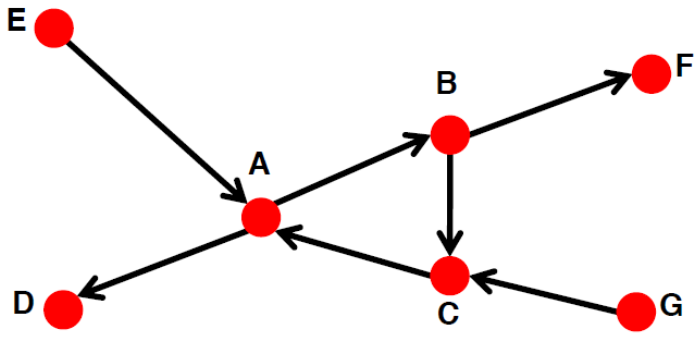
Fully Connected Graph

- Clique K_n
- A graph that has all possible n(n-1)/2 edges

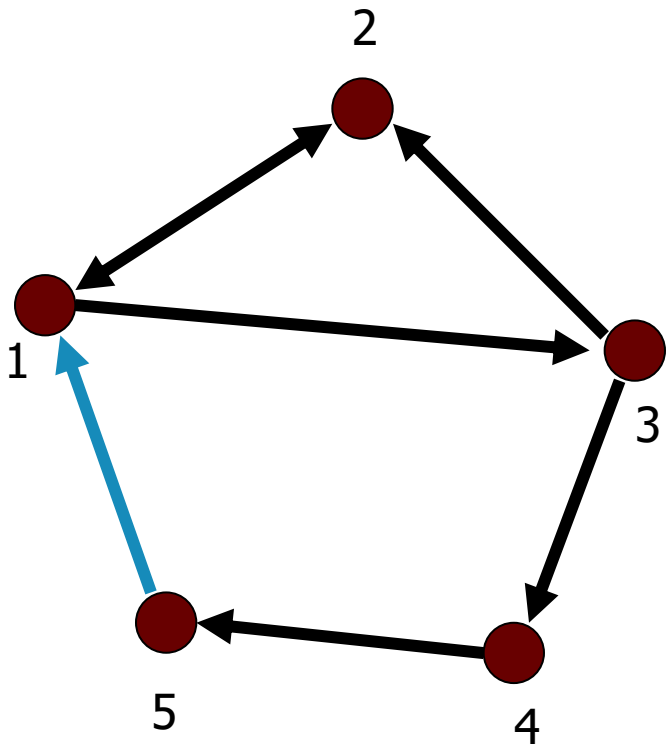


Connectivity of Directed Graph

- **Strongly connected graph**: there exists a path from every i to every j. has a path from each node to every other node and vice versa (e.g., A-B path and B-A path)
- **Weakly connected graph**: If edges are made to be undirected the graph is connected

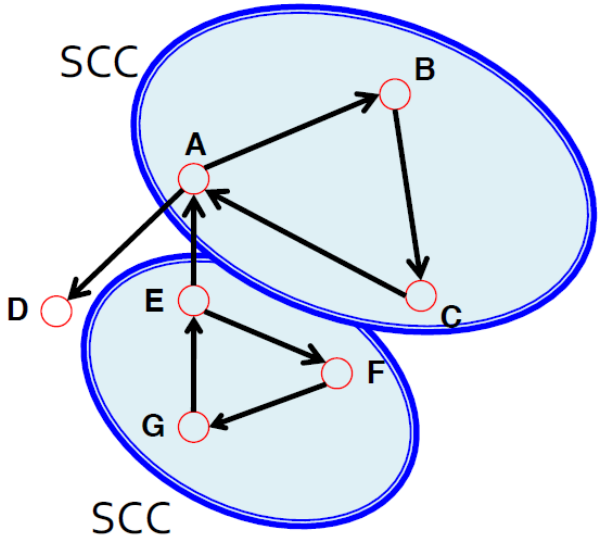


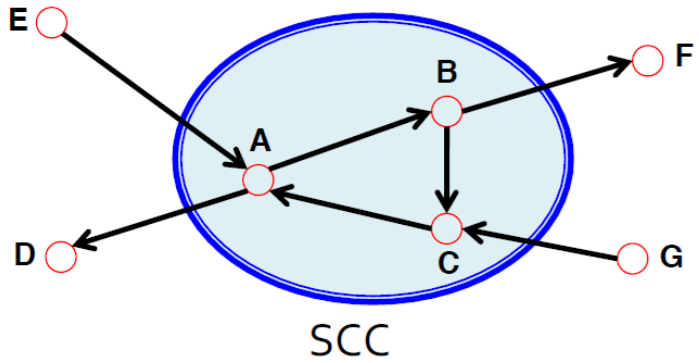
Graph on the left is connected but not strongly connected (e.g., there is no way to get from F to G by following the edge directions).



Connectivity of Directed Graph

- **Strongly connected components (SCCs) can be identified, but not every node is part of a nontrivial strongly connected component.**

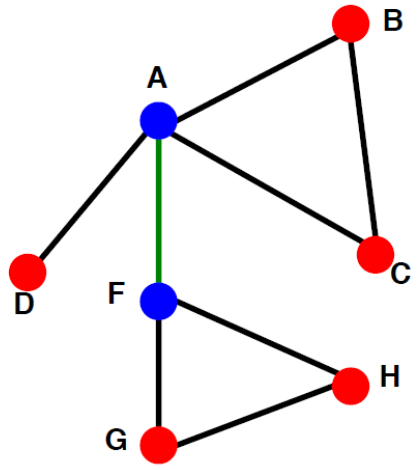


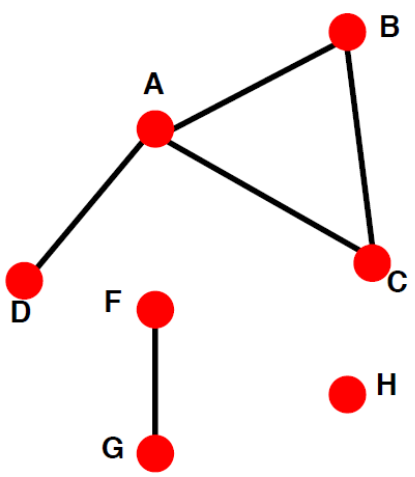


- In-component: nodes that can reach the SCC,
- Out-component: nodes that can be reached from the SCC.

Connectivity of Undirected Graphs

- **Connected (undirected) graph:**
  - Any two vertices can be joined by a path
- A disconnected graph is made up by two or more connected components





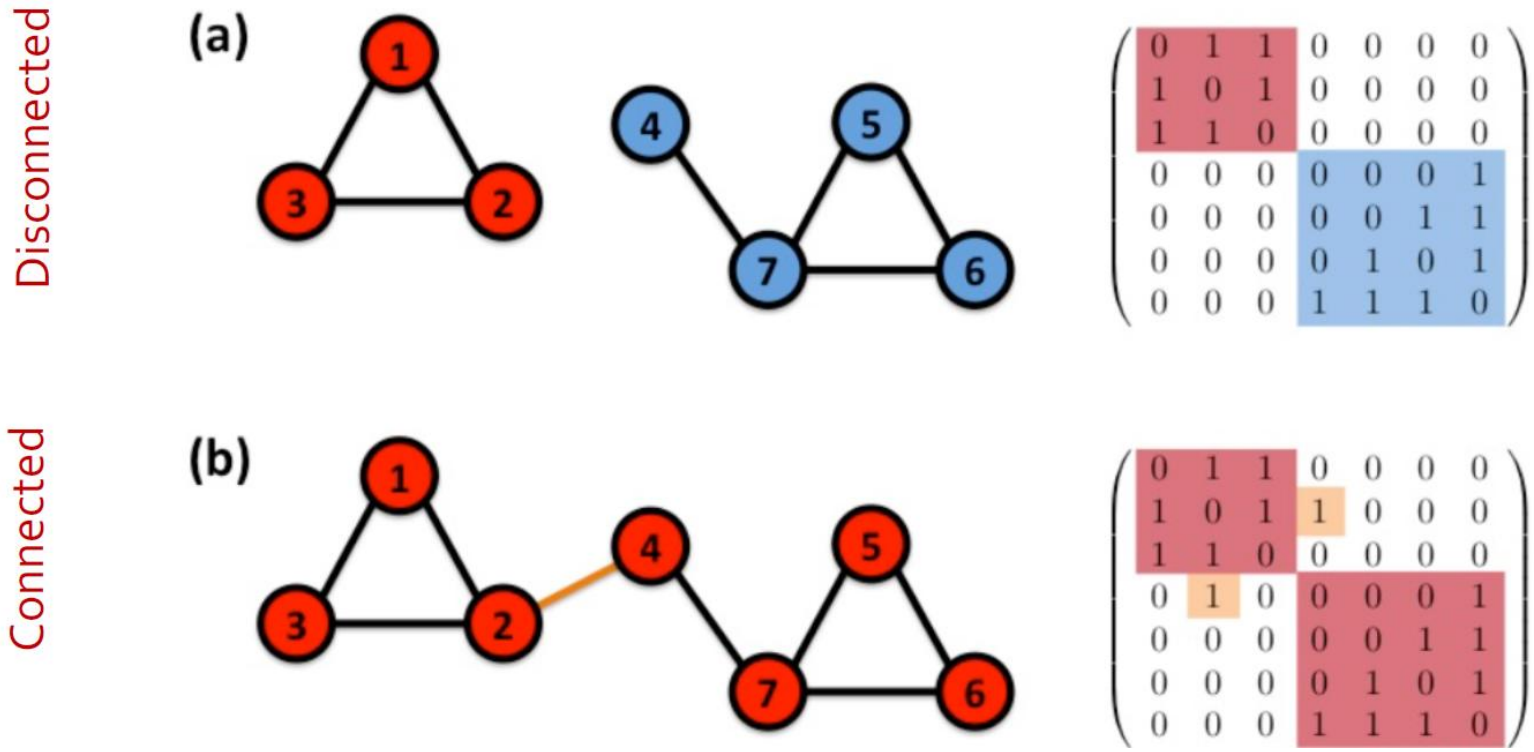
Largest Component:
Giant Component

Isolated node (node H)

- **Bridge edge:** If we erase the **edge**, the graph becomes disconnected
- **Articulation node:** If we erase the **node**, the graph becomes disconnected

Connectivity Example

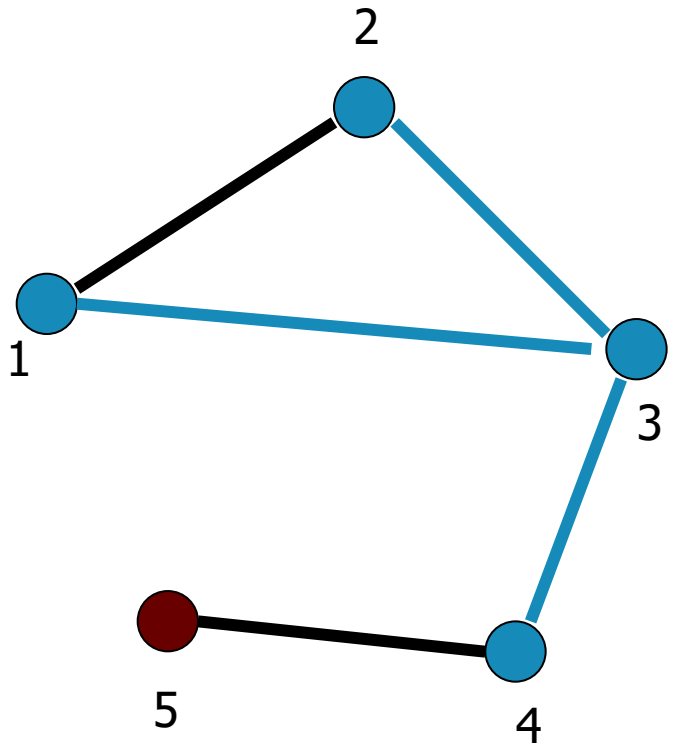
- The adjacency matrix of a network with several components can be written in a block- diagonal form, so that nonzero elements are confined to squares, with all other elements being zero:



Subgraphs

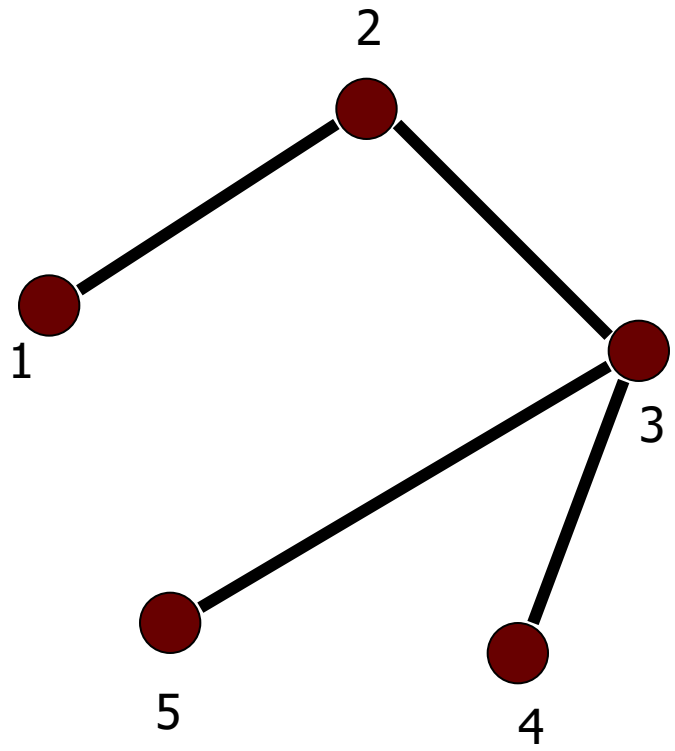
- Subgraph: Given $V' \subseteq V$, and $E' \subseteq E$, the graph $G'=(V',E')$ is a subgraph of G.

- Induced subgraph: Given $V' \subseteq V$, let $E' \subseteq E$ is the set of all edges between the nodes in V'. The graph $G'=(V',E')$, is an induced subgraph of G



Trees

- Connected Undirected graphs without cycles

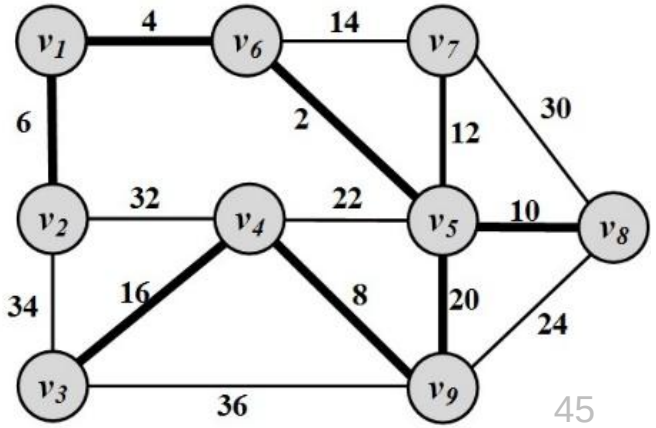


Trees Properties

- Edges and Vertices Relationship: A tree with n vertices has exactly n-1 edges.
- Unique Path: There is a unique path between any two vertices in a tree.
- All Edges Are Bridges: In a tree, every edge is a bridge; removing any edge will disconnect the graph.
- At Least Two Leaves: Every tree with at least two vertices has at least two vertices of degree one, known as leaves.

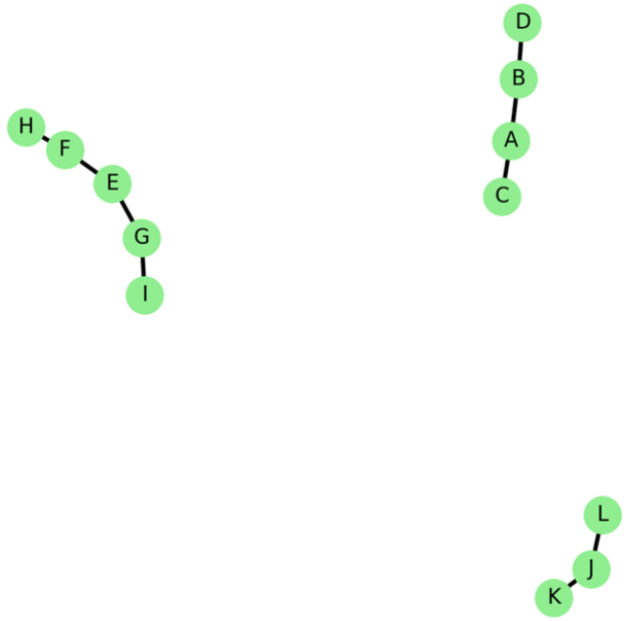
Spanning Tree

- For any connected graph, the spanning tree is a subgraph and a tree that includes all the nodes of the graph
- There may exist multiple spanning trees for a graph.
- For a weighted graph and one of its spanning tree, the weight of that spanning tree is the summation of the edge weights in the tree.
- Among the many spanning trees found for a weighted graph, the one with the minimum weight
  is called the minimum spanning tree (MST)



Forest

- A simple, undirected graph with no cycles. It consists of a collection of disjoint trees, where each connected component is a tree.



Forest Properties

- Number of Edges: A forest with n vertices and k connected components has exactly n - k edges. This is because each tree with m vertices contains m - 1 edges; thus, the total number of edges in the forest is the sum of the edges in all its trees.
- Acyclic Nature: Forests contain no cycles; consequently, every connected subgraph within a forest is also acyclic.
- Connected Components: Each connected component in a forest is a tree; therefore, a forest can be viewed as a collection of separate trees.

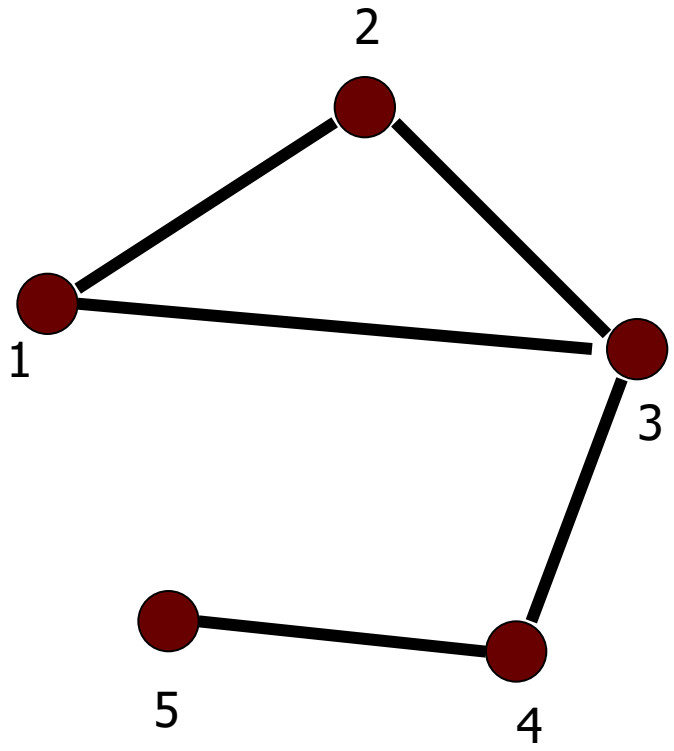
04

Graph Representation

Graph Representation

- Adjacency Matrix
  - symmetric matrix for undirected graphs

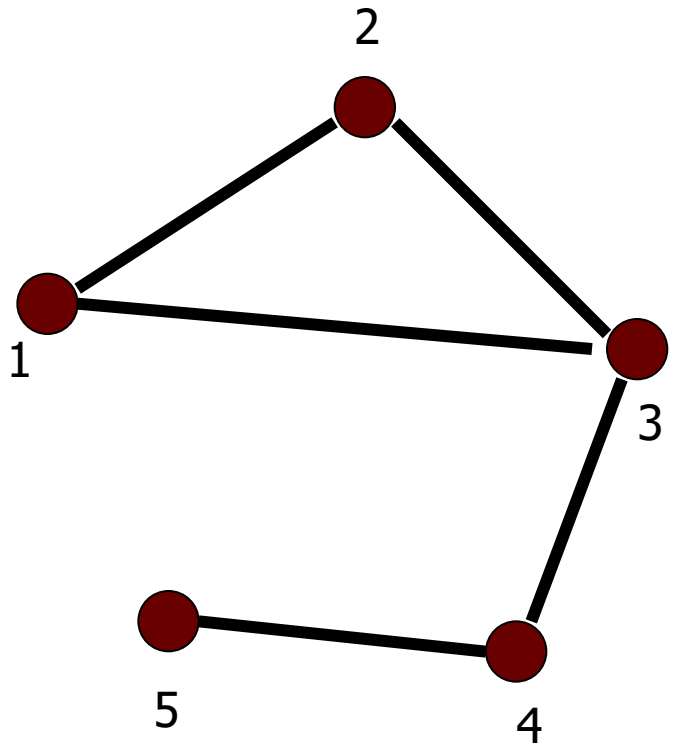
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



Graph Representation

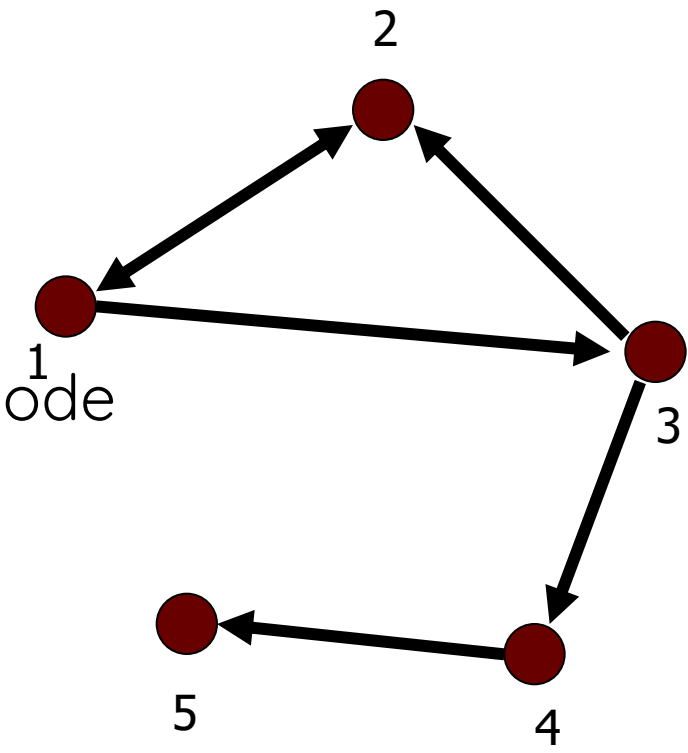
- Adjacency List
  - For each node keep a list with neighboring nodes

1: [2, 3]
2: [1, 3]
3: [1, 2, 4]
4: [3, 5]
5: [4]



Graph Representation

- Adjacency List
  - For each node keep a list of the nodes it points to
  - Easier to work with if network is
    - Large
    - Sparse
  - Allows us to quickly retrieve all neighbors of a given node

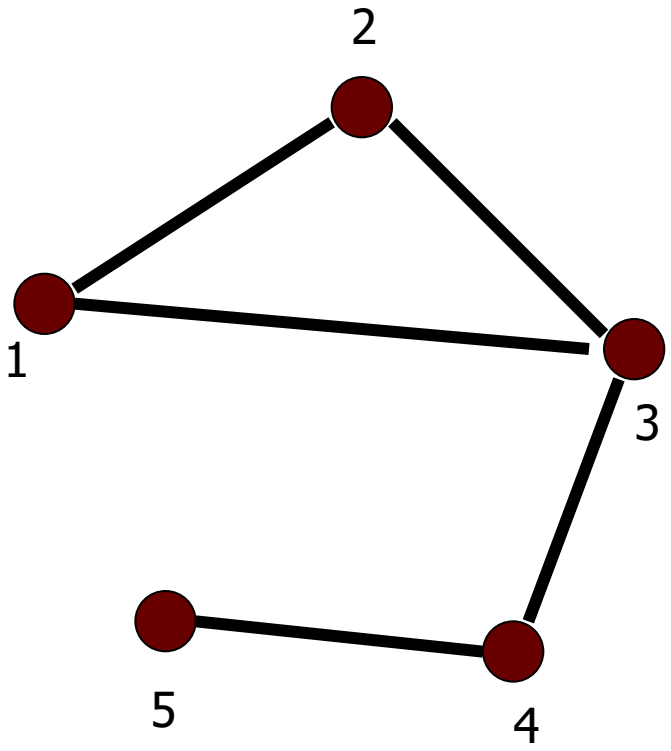


1: [2, 3]
2: [1]
3: [2, 4]
4: [5]
5: [null]

Graph Representation

- List of Edges
  - Keep a list of all the edges in the graph

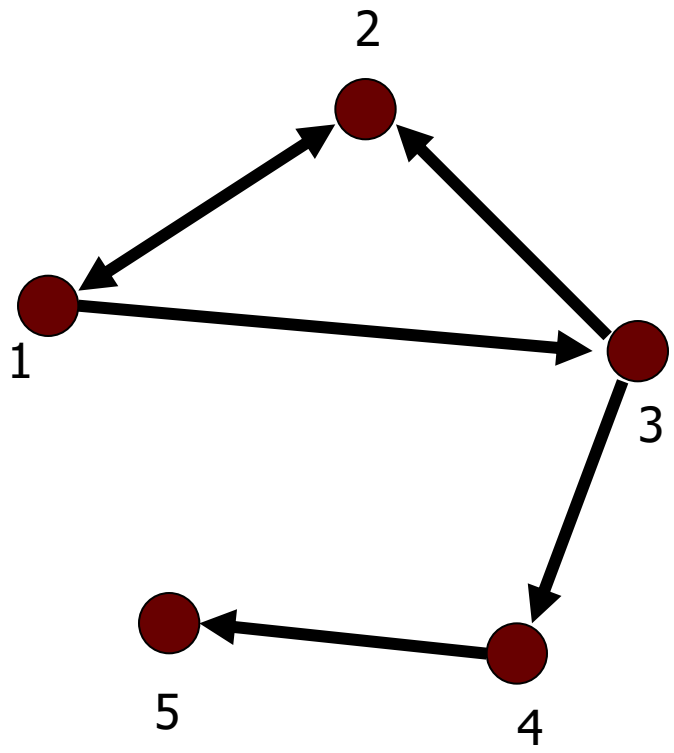
(1,2)
(2,3)
(1,3)
(3,4)
(4,5)



Graph Representation

- List of Edges
  - Keep a list of all the directed edges in the graph

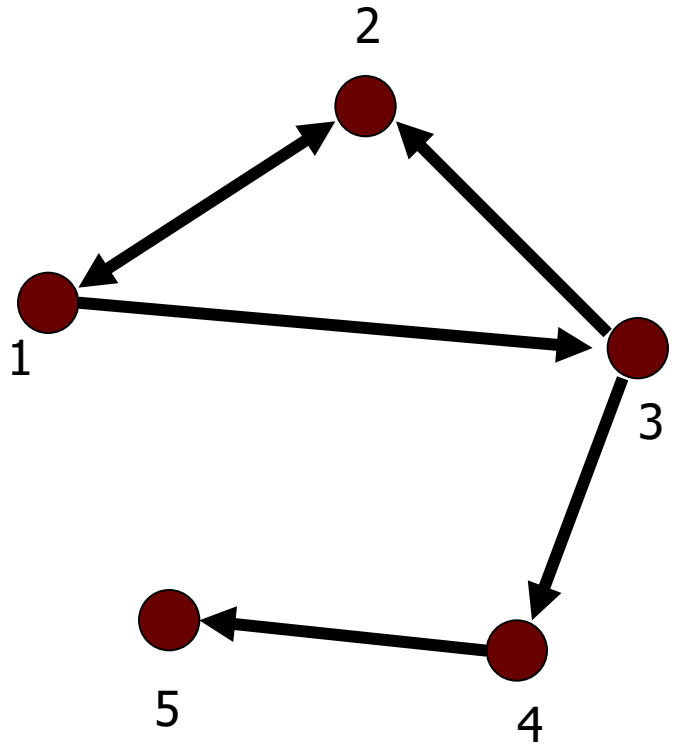
(1,2)
(2,1)
(1,3)
(3,2)
(3,4)
(4,5)



Graph Representation

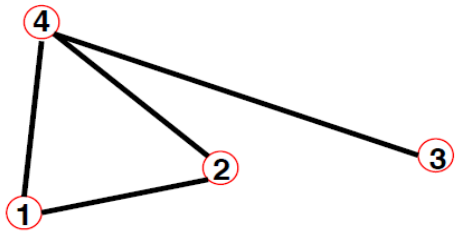
- Adjacency Matrix
  - unsymmetric matrix for undirected graphs

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



Adjacency Matrix

Undirected



$$A_{ij} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

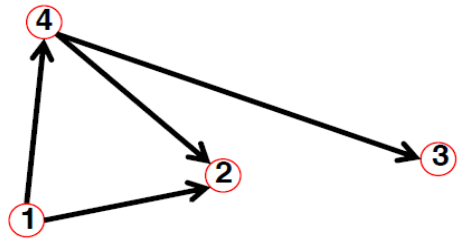
$$k_i = \sum_{j=1}^{N} A_{ij}$$

$$k_j = \sum_{i=1}^{N} A_{ij}$$

$$A_{ij} = A_{ji}$$
$$A_{ii} = 0$$

$$L = \frac{1}{2}\sum_{i=1}^{N} k_i = \frac{1}{2}\sum_{ij} A_{ij}$$

Directed



$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

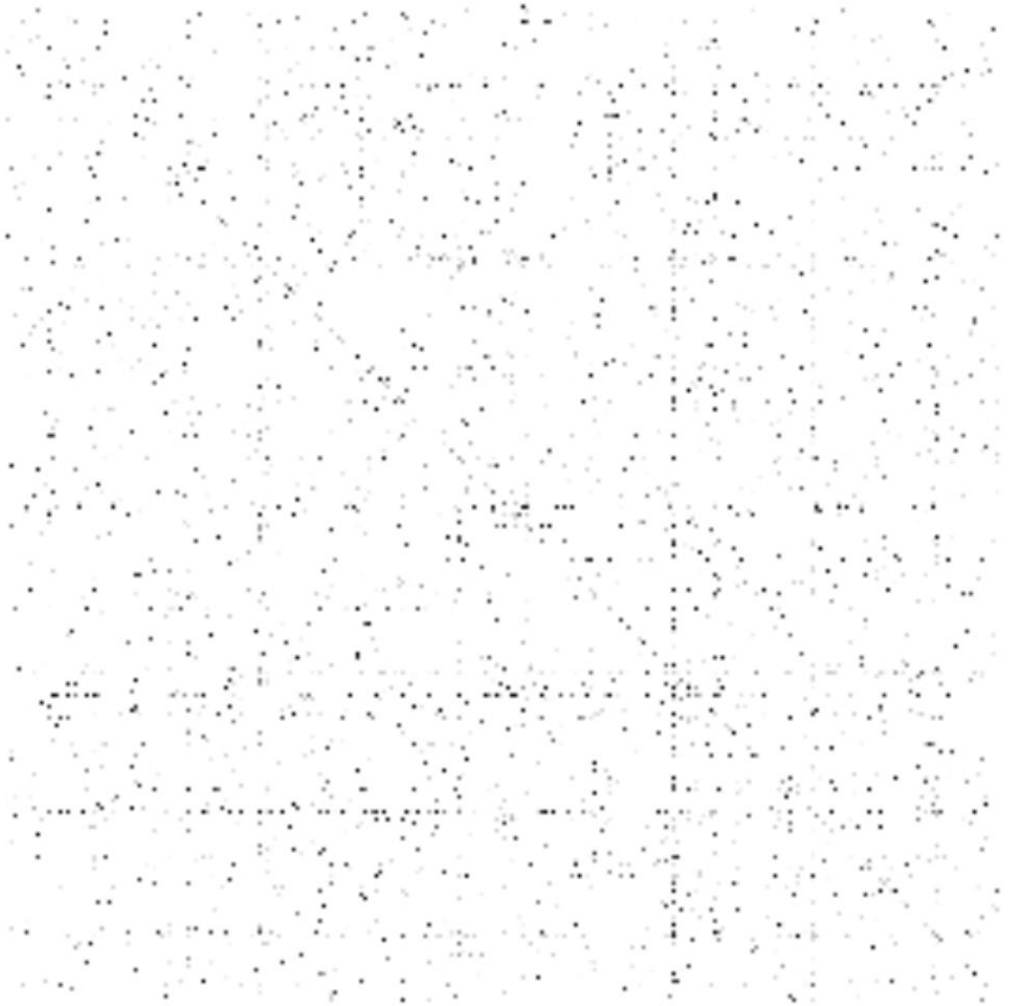
$$k_i^{out} = \sum_{j=1}^{N} A_{ij}$$

$$k_j^{in} = \sum_{i=1}^{N} A_{ij}$$

$$A_{ij} \neq A_{ji}$$
$$A_{ii} = 0$$

$$L = \sum_{i=1}^{N} k_i^{in} = \sum_{j=1}^{N} k_j^{out} = \sum_{i,j}^{N} A_{ij}$$

Adjacency Matrices are Sparse



Networks are Sparse Graphs

Most real-world networks are sparse

$$E << E_{max} \quad (\text{or } \bar{k} << N-1)$$

WWW (Stanford-Berkeley):	N=319,717	⟨k⟩=9.65
Social networks (LinkedIn):	N=6,946,668	⟨k⟩=8.87
Communication (MSN IM):	N=242,720,596	⟨k⟩=11.1
Coauthorships (DBLP):	N=317,080	⟨k⟩=6.62
Internet (AS-Skitter):	N=1,719,037	⟨k⟩=14.91
Roads (California):	N=1,957,027	⟨k⟩=2.82
Proteins (S. Cerevisiae):	N=1,870	⟨k⟩=2.39

(Source: *Leskovec et al., Internet Mathematics, 2009*)

Consequence: Adjacency matrix is filled with zeros!

(**Density of the matrix (E/N^2):** WWW=1.51×10^{-5}, MSN IM = 2.27×10^{-8})

Network Representations

Email network >> directed multigraph with self-edges

Facebook friendships >> undirected, unweighted

Citation networks >> unweighted, directed, acyclic

Collaboration networks >> undirected multigraph or weighted graph

Mobile phone calls >> directed, (weighted?) multigraph

Protein Interactions >> undirected, unweighted with self-interactions

05

Linear Algebra Review
Eigenvalue & Eigenvector

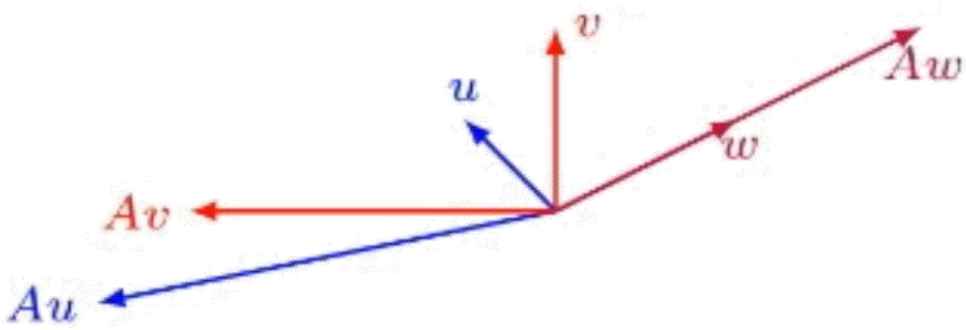
Motivation

- $A = \begin{bmatrix} 3 & -2 \\ 1 & 0 \end{bmatrix}$

$u = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \Rightarrow Au = \begin{bmatrix} 3 & -2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -5 \\ -1 \end{bmatrix}$

$v = \begin{bmatrix} 0 \\ 2 \end{bmatrix} \Rightarrow Av = \begin{bmatrix} 3 & -2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \end{bmatrix} = \begin{bmatrix} -4 \\ 0 \end{bmatrix}$

$w = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \Rightarrow Aw = \begin{bmatrix} 3 & -2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$



- Vector “w” keeps the straight, but changes the scale.

Definition

Definition

An **eigenvector** of a square $n \times n$ matrix A is nonzero vector v such that $Av = \lambda v$ for some scalar λ. A scalar λ is called an **eigenvalue** of A if there is a nontrivial solution v of $Av = \lambda v$; such an v is called an *eigenvector corresponding to* λ.

- **An eigenvector must be nonzero, by definition, but an eigenvalue may be zero.**

Example

- $A = \begin{bmatrix} 3 & -2 \\ 1 & 0 \end{bmatrix}$, $v = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$, $\lambda = 2$
- Show that 7 is an eigenvalue of matrix B, and find the corresponding eigenvectors.

$$B = \begin{bmatrix} 1 & 6 \\ 5 & 2 \end{bmatrix}$$

Eigenspace

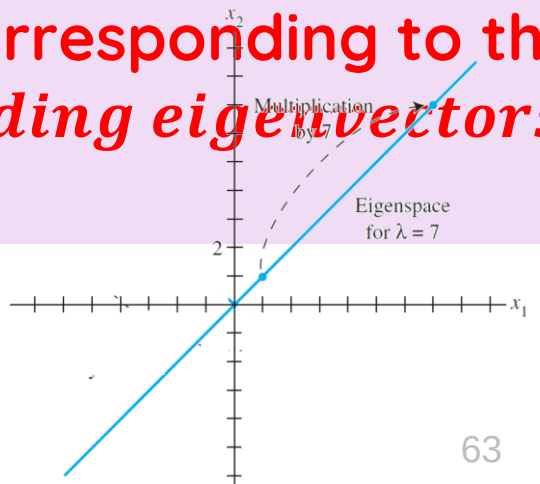
Note

λ is an eigenvalue of an $n \times n$ matrix:

$$Av = \lambda v \Rightarrow (A - \lambda I)v = 0$$

The set of all solutions of above is just the null space of the matrix $A - \lambda I$. So this set is the *subspace* of $\mathbb{R}^n$ and is called the **eigenspace** of A corresponding to λ . The eigenspace consists of the zero vector and all the eigenvectors corresponding to λ.

Eigenspace: A vector space formed by eigenvectors corresponding to the same eigenvalue and the origin point. $\boldsymbol{span\{corresponding\ eigenvectors\}}$



Definitions

Theorem

Let A be a $m \times n$ matrix:

$$Nullity(A) + Rank(A) = n$$

Note

- $Av = \lambda v \Rightarrow Av - \lambda vI = 0 \Rightarrow (A - \lambda I)v = 0 \quad v \neq 0$
  - $v \in N(A - \lambda I)$
  - $A - \lambda I$ must be singular.
  - Proof that for finding the eigenvalue we should solve the determinate zero equation. Look at nullspace, rank and nullity theorem, singular matrix, and det zero!
- **Characteristic polynomial** $\det(A - \lambda I)$
- **Characteristic equation** $\det(A - \lambda I) = 0$
- If λ is an eigenvalue of A, then the subspace E_λ = {span{v} | Av = λv} is called the eigenspace of A associated with λ. (This subspace contains all the span of eigenvectors with eigenvalue λ, and also the zero vector.)
- **Eigenvector is basis for eigenspace.**
- Set of all eigenvalues of matrix is $\sigma(A)$ *named spectrum of a matrix*

Definitions

Note

- Instead of $\det(A - \lambda I)$, we will compute $\mathbf{det(\lambda I - A)}$. Why?
  - $\det(A - \lambda I) = (-1)^{n}\det(\lambda I - A)$
  - Matrix $n \times n$ with real values has …… eigenvalues.

Finding Eigenvalues and Eigenvectors

Let A be an $n \times n$ matrix.

1. First, find the eigenvalues λ of A by solving the equation $\det(\lambda I - A) = 0$.
2. For each λ, find the basic eigenvectors $X \neq 0$ by finding the basic solutions to $(\lambda I - A)\, X = 0$.

To verify your work, make sure that $AX = \lambda X$ for each λ and associated eigenvector X.

Example

Find eigenvalues and eigenvectors, eigenspace (E), and *spectrum* of matrix $A = \begin{bmatrix} 3 & -2 \\ 1 & 0 \end{bmatrix}$:

$$\det(A - \lambda \mathrm{I}) = \begin{vmatrix} 3-\lambda & -2 \\ 1 & -\lambda \end{vmatrix} = \lambda^2 - 3\lambda + 2 = 0 \Rightarrow \begin{cases} \lambda_1 = 1 \\ \lambda_2 = 2 \end{cases}$$

$$\left.\begin{matrix} \lambda_1 = 1 \\ (A - \lambda_1 I) q_1 = 0 \end{matrix}\right\} \Rightarrow q_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 3 & -2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 1 \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\left.\begin{matrix} \lambda_2 = 2 \\ (A - \lambda_2 I) q_2 = 0 \end{matrix}\right\} \Rightarrow q_2 = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 3 & -2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = 2 \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Eigenvalues={1,2}

Eigenvectors=$\{\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}\}$

$E_1(A) = span\{\begin{bmatrix} 1 \\ 1 \end{bmatrix}\}$ $E_2(A) = span\{\begin{bmatrix} 2 \\ 1 \end{bmatrix}\}$

$\sigma(A)$={1,2}

$$\boldsymbol{AQ = Q\Lambda} \Rightarrow \begin{bmatrix} 3 & -2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

Eigenvalues Properties

- Are eigenvectors unique?
  - If $\mathbf{v}$ is an eigenvector, then $\beta\mathbf{v}$ is also an eigenvector

$$A(\beta v) = \beta(Av) = \beta(\lambda v) = \lambda(\beta v)$$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

  - For a 2 × 2 matrix, this is a simple quadratic equation with two solutions (maybe complex)

$$\lambda = (a_{11} + a_{22}) \pm \sqrt{\frac{(a_{11} + a_{22})^2}{4(a_{11}a_{22} - a_{12}a_{21})}}$$

Eigenvalues Properties

- If A is an n × n matrix:
  - The sum of the n eigenvalues of A is the trace of A.
  - The product of the n eigenvalues is the determinant of A.
  - $0 \in \sigma(A) \Leftrightarrow |A|=0$
  - If A is symmetric, then any two eigenvectors from different eigenspace are orthogonal.

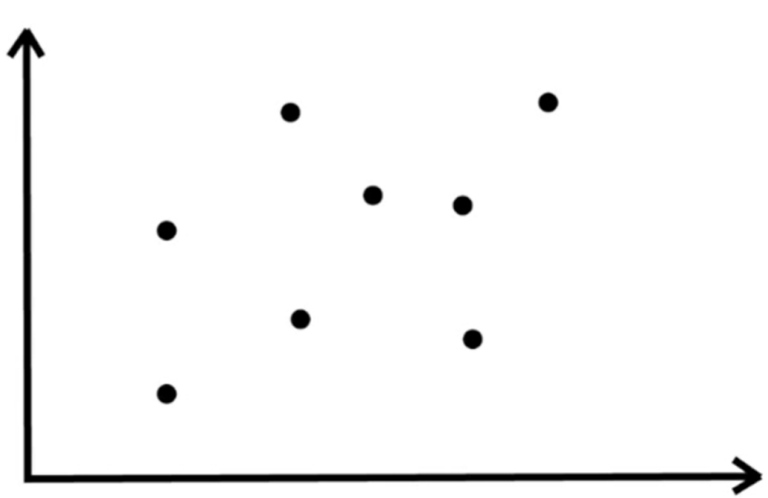
$$\left.\begin{array}{c} Av_1 = \lambda_1 v_1 \\ Av_2 = \lambda_2 v_2 \\ \lambda_1 \neq \lambda_2 \end{array}\right\} \Rightarrow {v_1}^T v_2 = 0$$

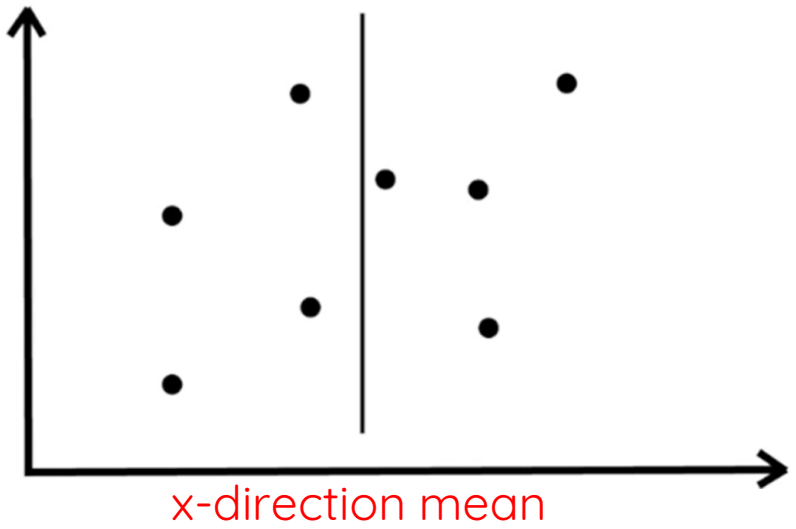
06

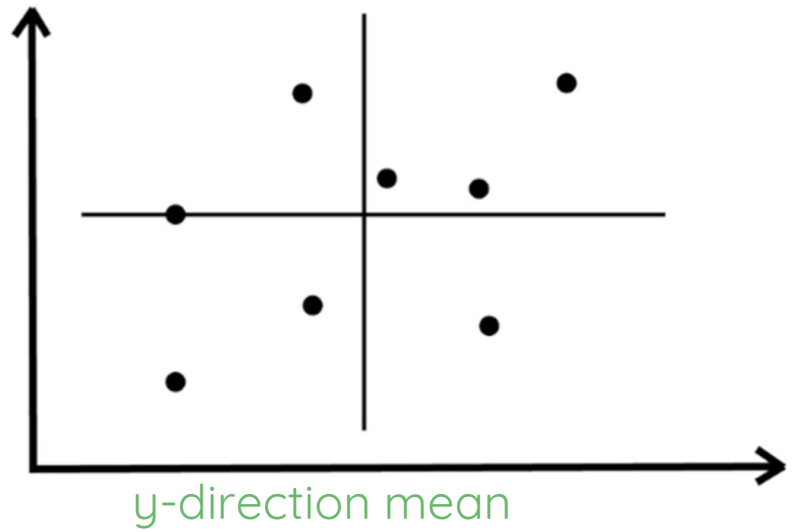
Covariance Matrix

Covariance Matrix

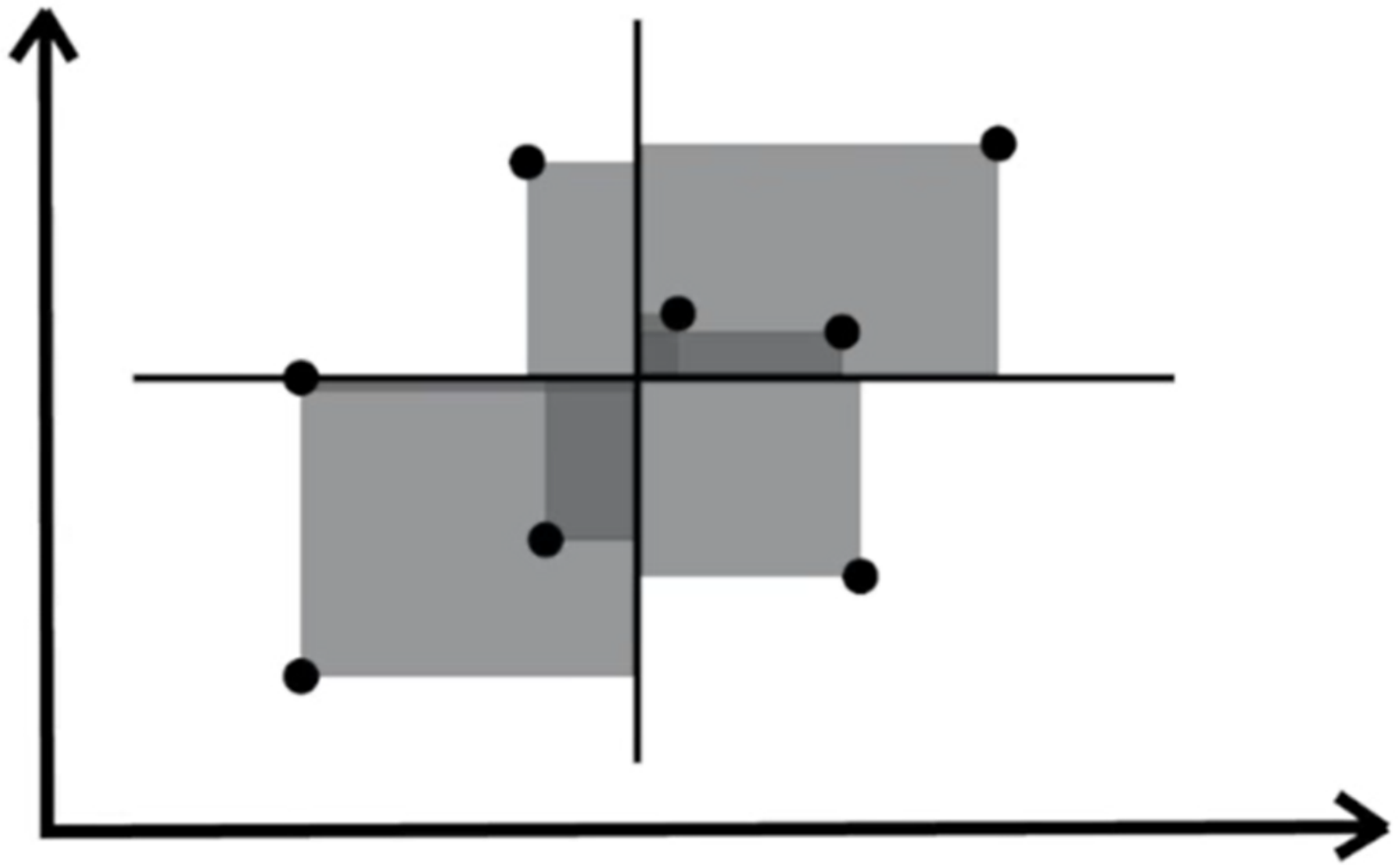
- Measures how much two variables change together.
- Look at how much is the distance of each point from the x-direction mean & y-direction mean.



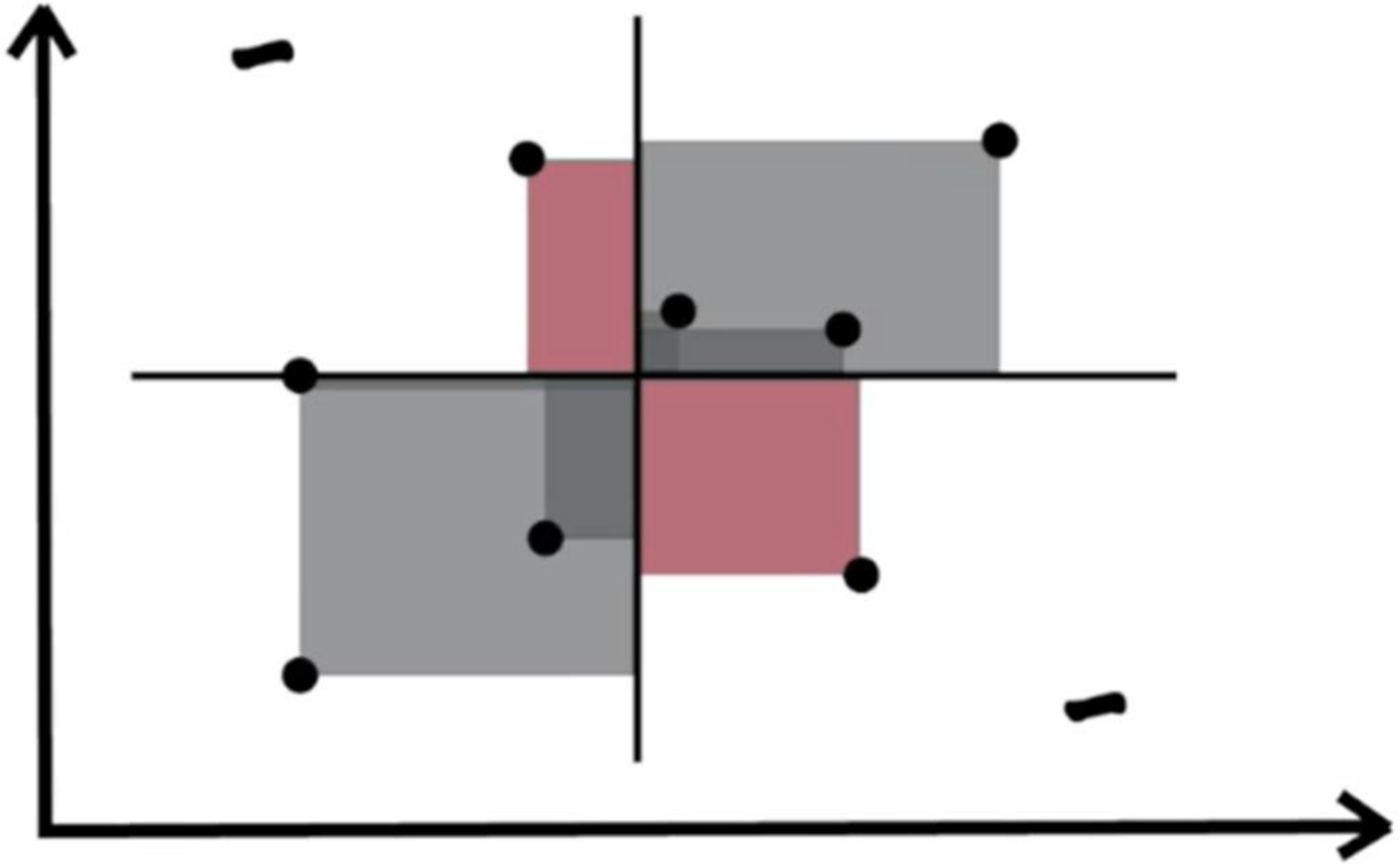




Covariance Matrix

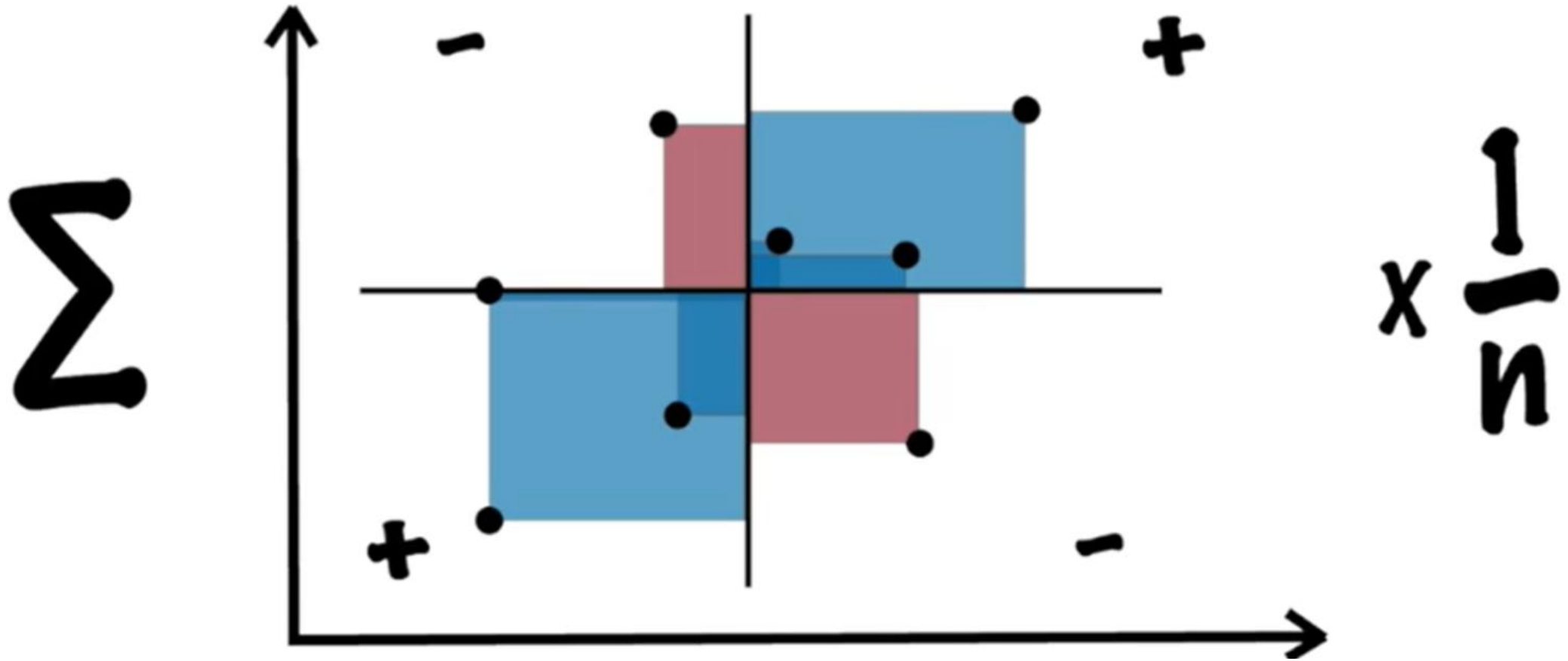


Covariance Matrix

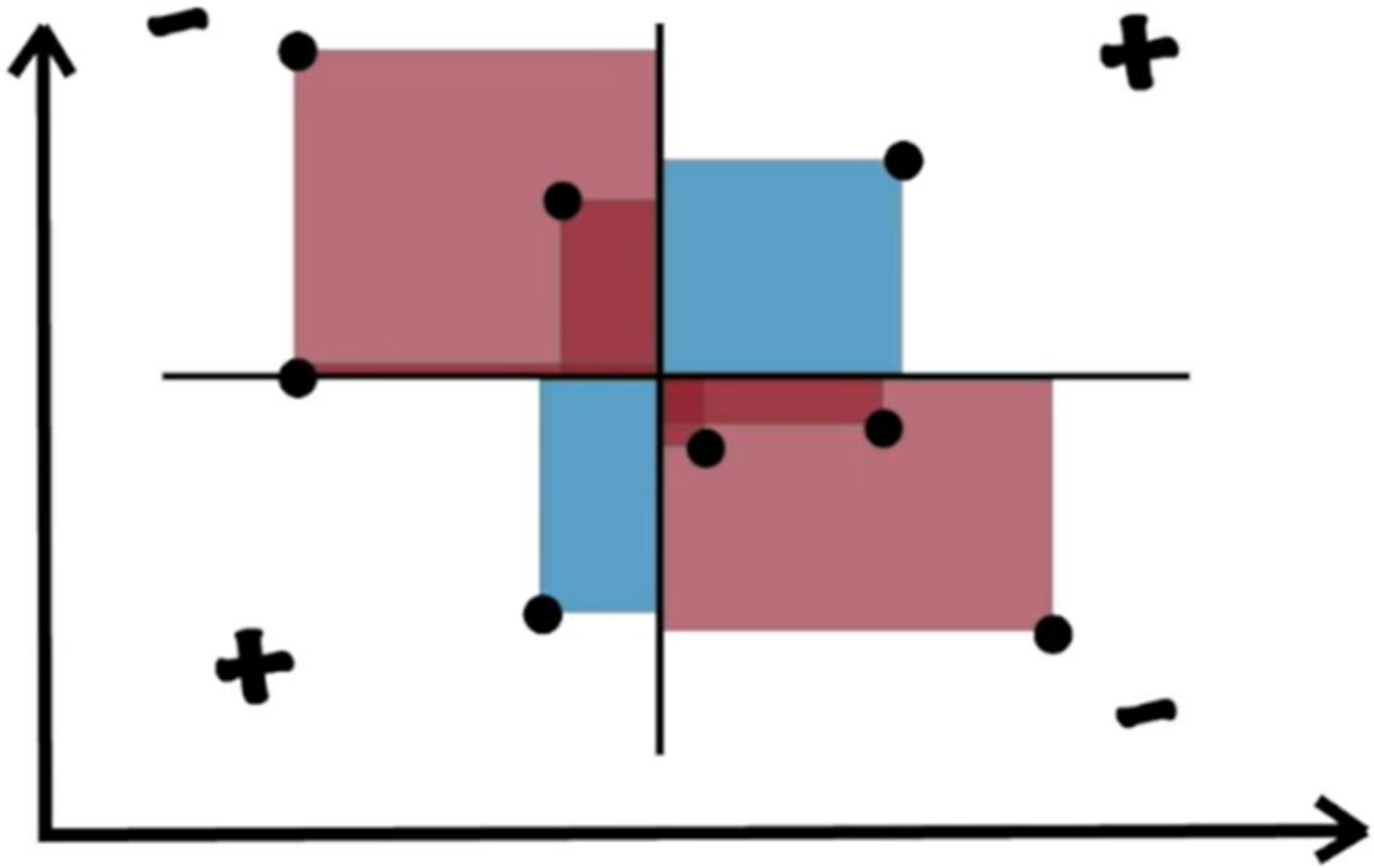


Covariance Matrix

$$Cov(X,Y) = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{n}$$

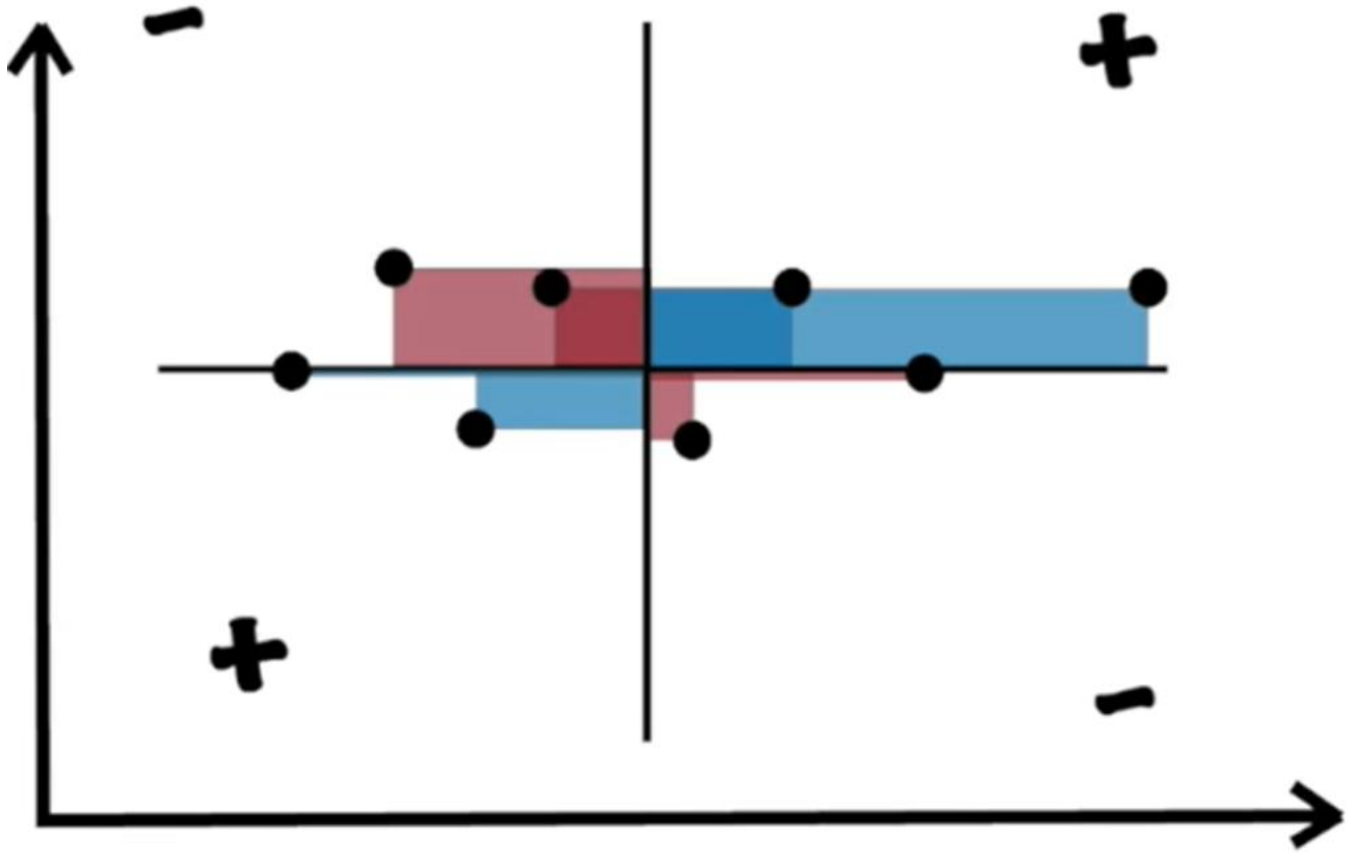


Negative Covariance



Low Covariance

- Dataset with spread only in one dimension will have a low covariance



Covariance Conclusion

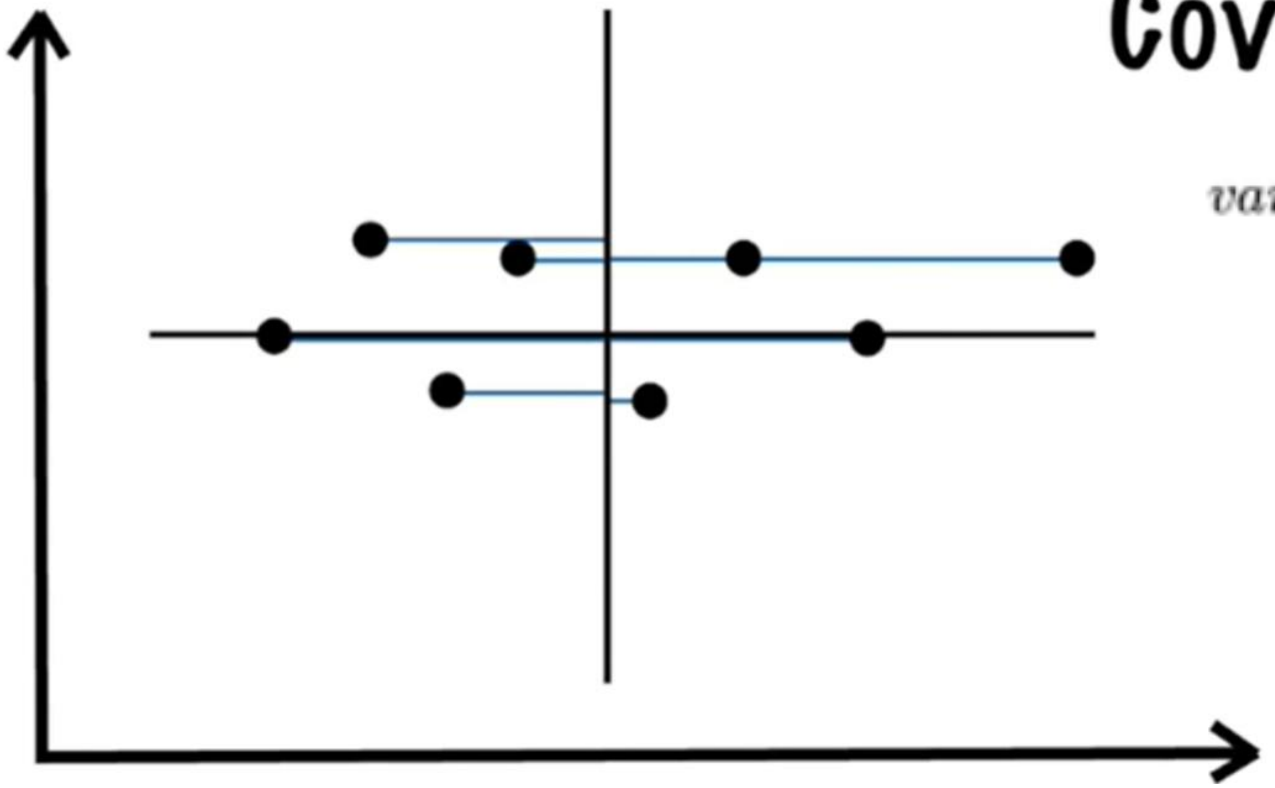
Large Negative Covariance

Near Zero Covariance

Large Positive Covariance

Variance

- Covariance of a dimension with itself.



Cov(x,x)

$$var(x) = \frac{\sum_i^n (x_i - \mu)^2}{N}$$

Covariance Matrix

- Any covariance matrix is **symmetric** and **positive semi-definite** and its main diagonal contains variances.
  - covariance is a symmetric function, i.e. Cov(X,Y)=Cov(Y,X)

$$\Sigma = \begin{bmatrix} \mathrm{Var}(x,x) & \mathrm{Cov}(x,y) & \mathrm{Cov}(x,z) \\ \mathrm{Cov}(y,x) & \mathrm{Var}(y,y) & \mathrm{Cov}(y,z) \\ \mathrm{Cov}(z,x) & \mathrm{Cov}(z,y) & \mathrm{Var}(z,z) \end{bmatrix}$$

Covariance Matrix for Graph

- $C = \frac{1}{n}(X_{\text{centered}})^{\top}(X_{\text{centered}})$

1. **Data Matrix** X: Let $X = A$.

$$X = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

2. **Optionally, mean-center** each column. (Calculate the mean of each column and subtract it from each entry in that column.)

3. **Covariance Matrix**:

$$C = \frac{1}{4} X_{\text{centered}}^{\top} X_{\text{centered}}.$$

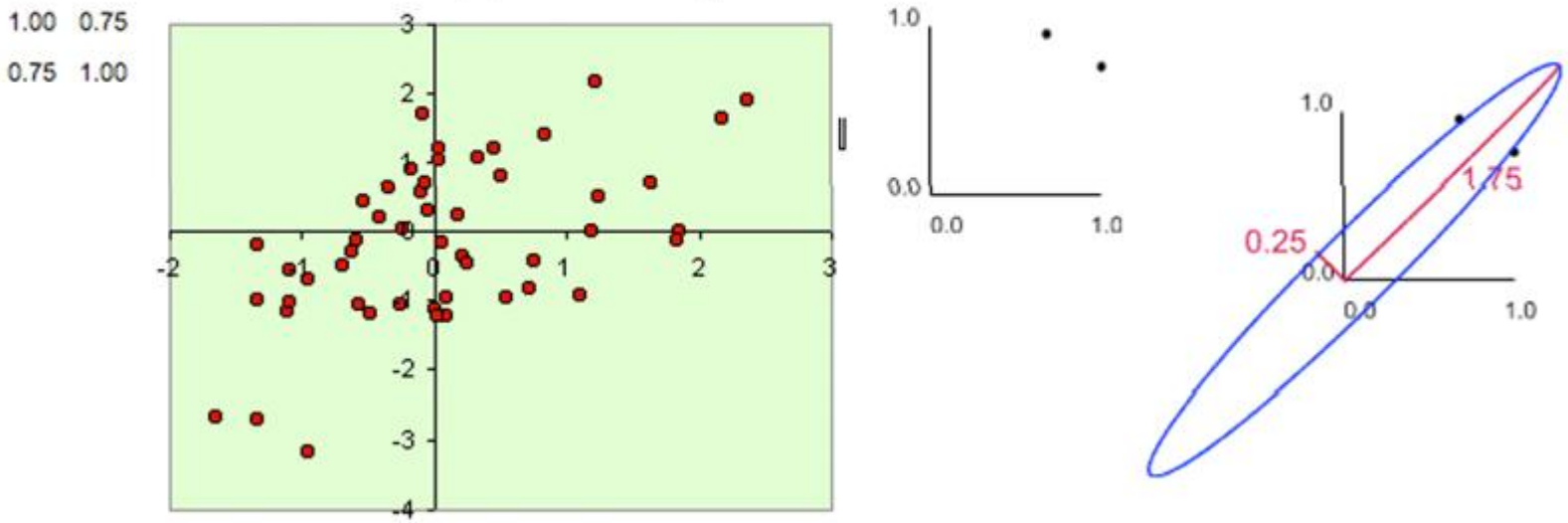
07

Physical interpretation

Physical interpretation

- Consider a covariance matrix, **A**, i.e., A = 1/n $S^T S$ for some S

$$\mathbf{A} = \begin{bmatrix} 1 & .75 \\ .75 & 1 \end{bmatrix} \Rightarrow \lambda_1 = 1.75, \lambda_2 = 0.25$$



- Error ellipse with the major axis as the larger eigenvalue and the minor axis as the smaller eigenvalue

Eigenvalues and Eigenvectors

- The value λ is an eigenvalue of matrix A if there exists a non-zero vector x, such that $Ax=\lambda x$. Vector x is an eigenvector of matrix A
  - The largest eigenvalue is called the principal eigenvalue
  - The corresponding eigenvector is the principal eigenvector
  - Corresponds to the direction of maximum change

08

Principal Component

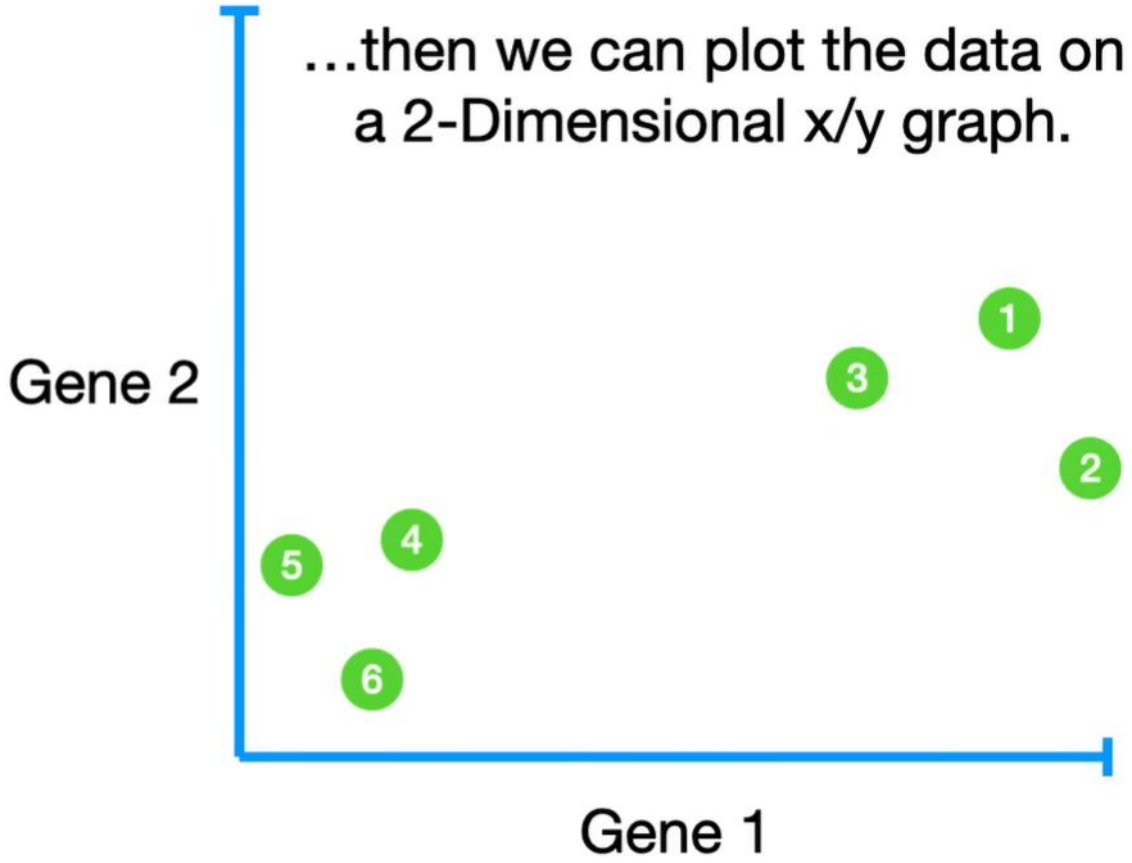
Introduction

	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	1	2

Even though it's a simple graph, it shows us that mice 1, 2 and 3 are more similar to each other than they are to mice 4, 5 6.

Introduction

	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	1	2
Gene 2	6	4	5	3	2.8	1



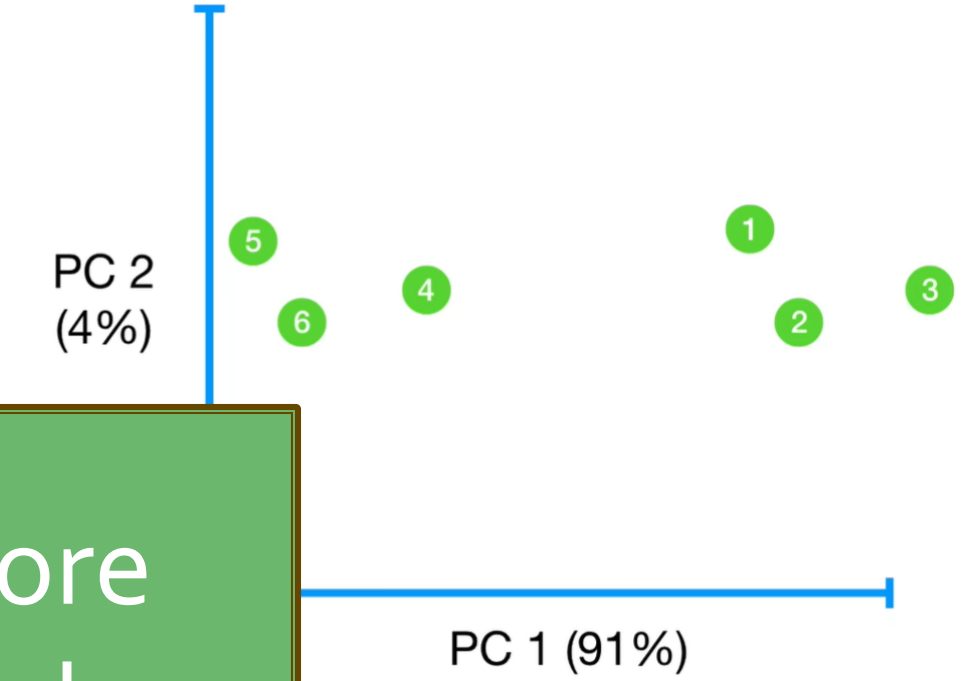
Introduction

	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1
Gene 3	12	9	10	2.5	1.3	2
Gene 4	5	7	6	2	4	7

If we measured 4 genes, however, we can no longer plot the data - 4 genes require 4 dimensions.

Introduction

PCA might tell us that Gene 3 is responsible for separating samples along the x-axis.

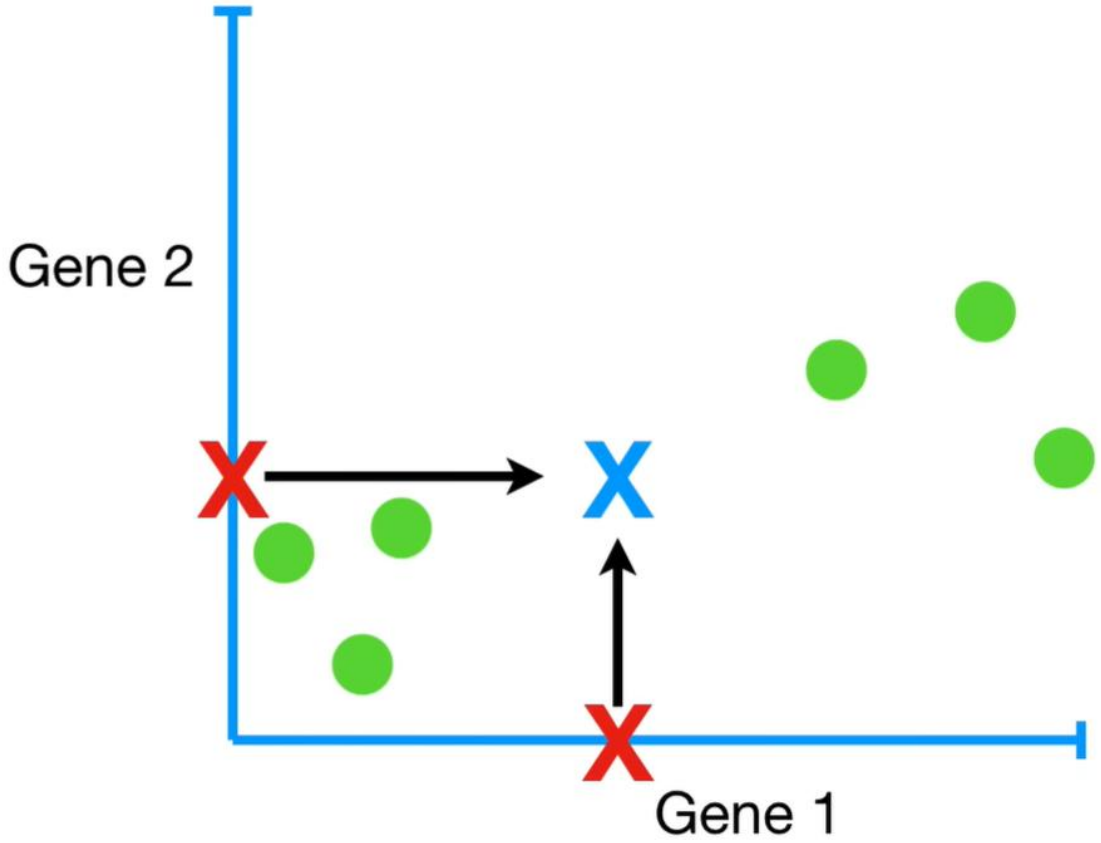


How PCA can take 4 or more gene measurements and make a 2-D PCA Plot?

What PCA Does

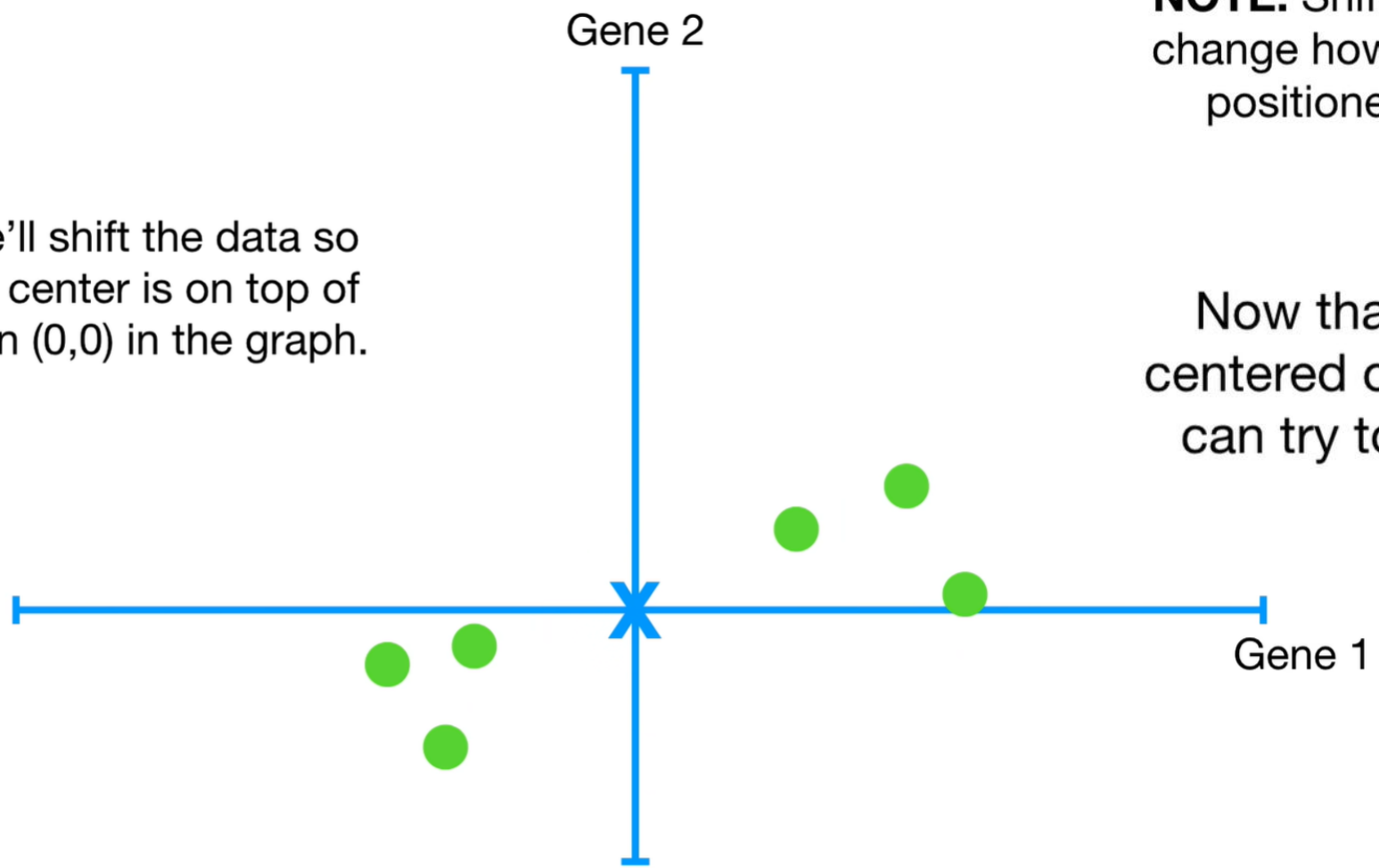
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1

From this point on, we'll focus on what happens in the graph; we no longer need the original data…



What PCA Does

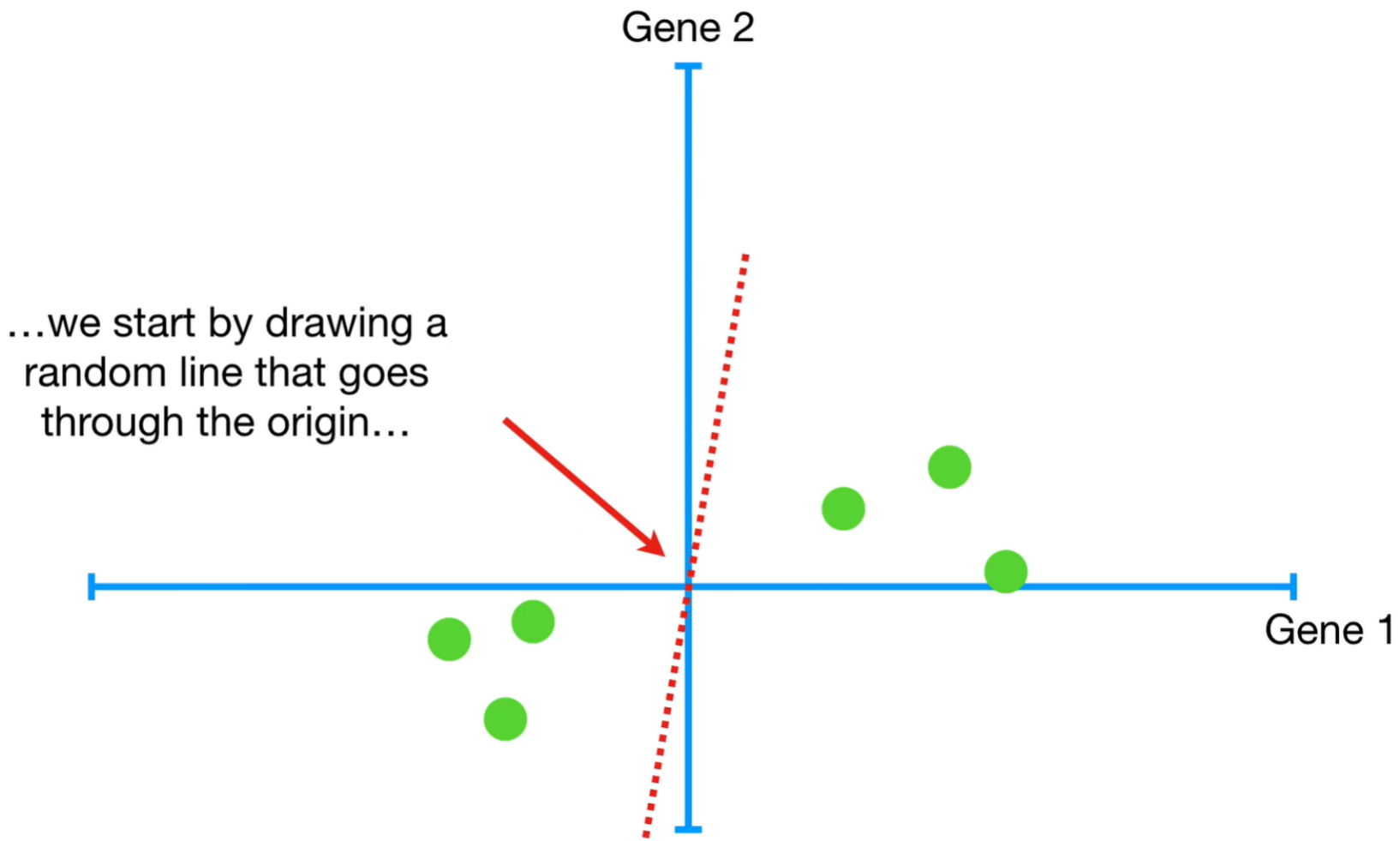
Now we'll shift the data so that the center is on top of the origin (0,0) in the graph.



NOTE: Shifting the data did not change how the data points are positioned *relative to each other*.

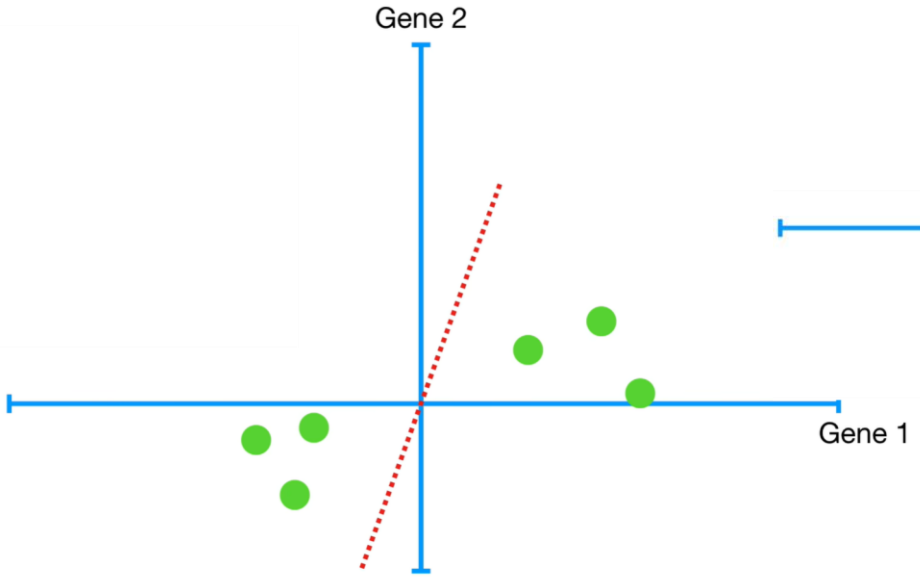
Now that the data are centered on the origin, we can try to fit a line to it.

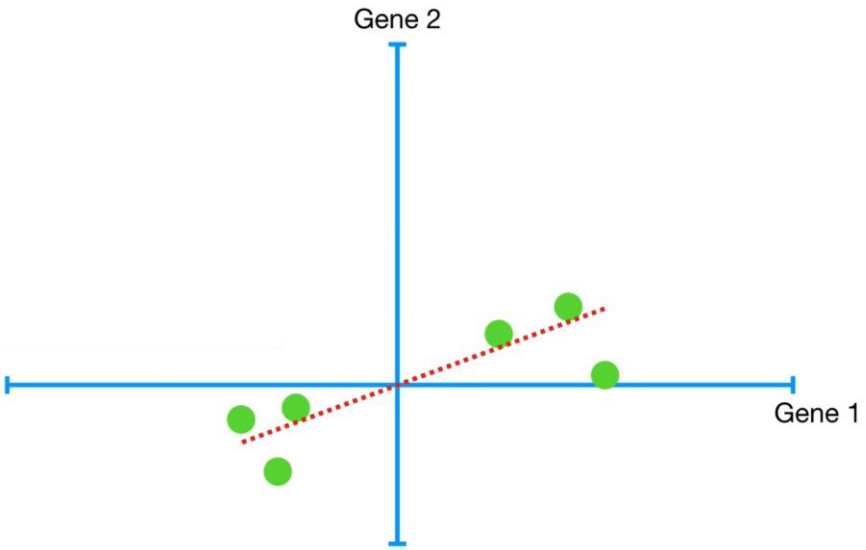
What PCA Does

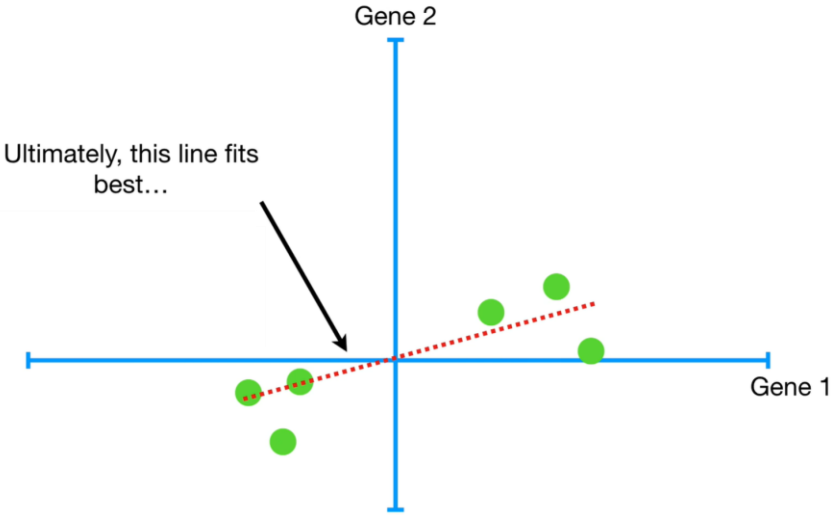


What PCA Does

…then we rotate the line until it fits the data as well is it can, given that it has to go through the origin.

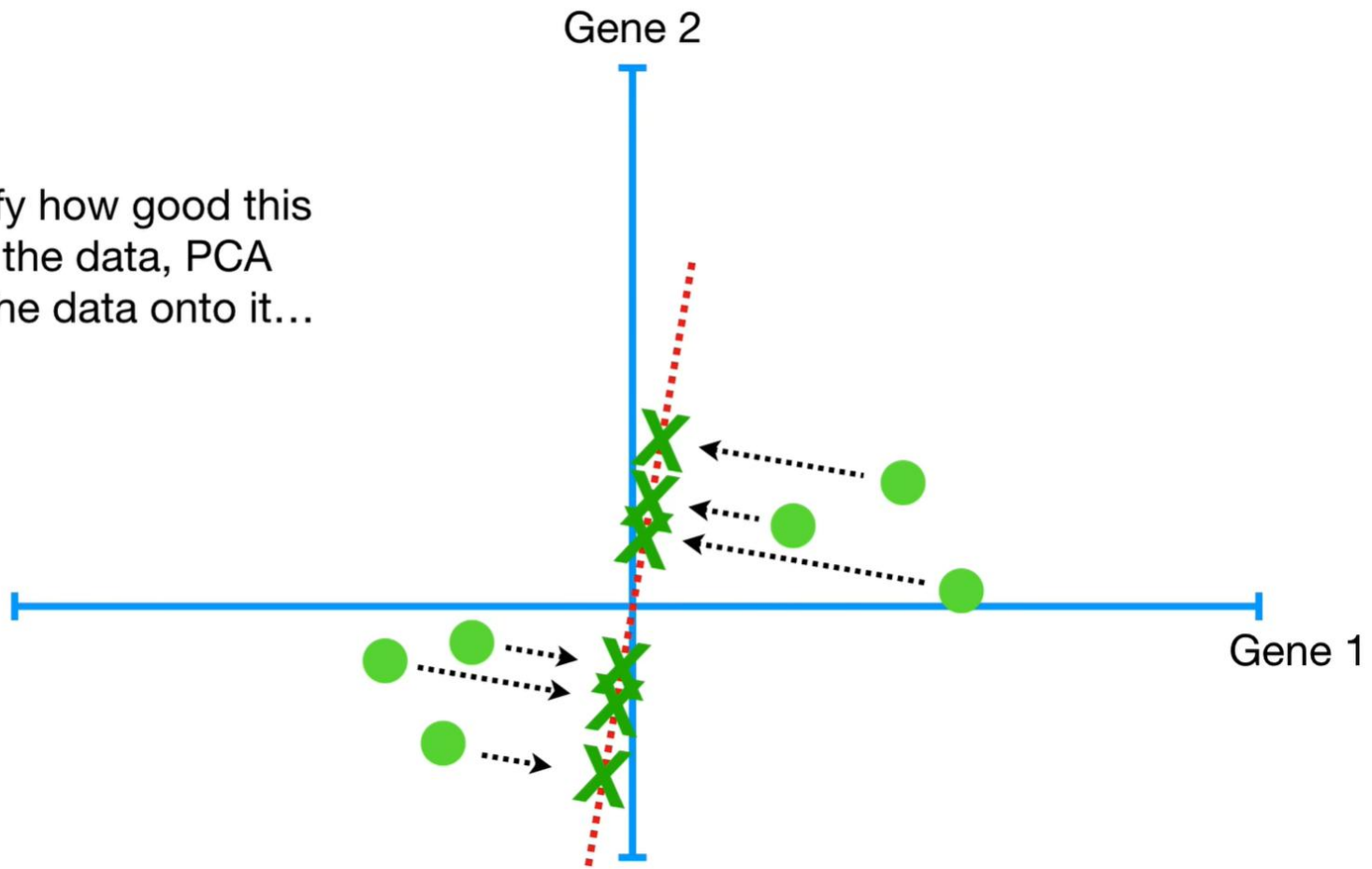




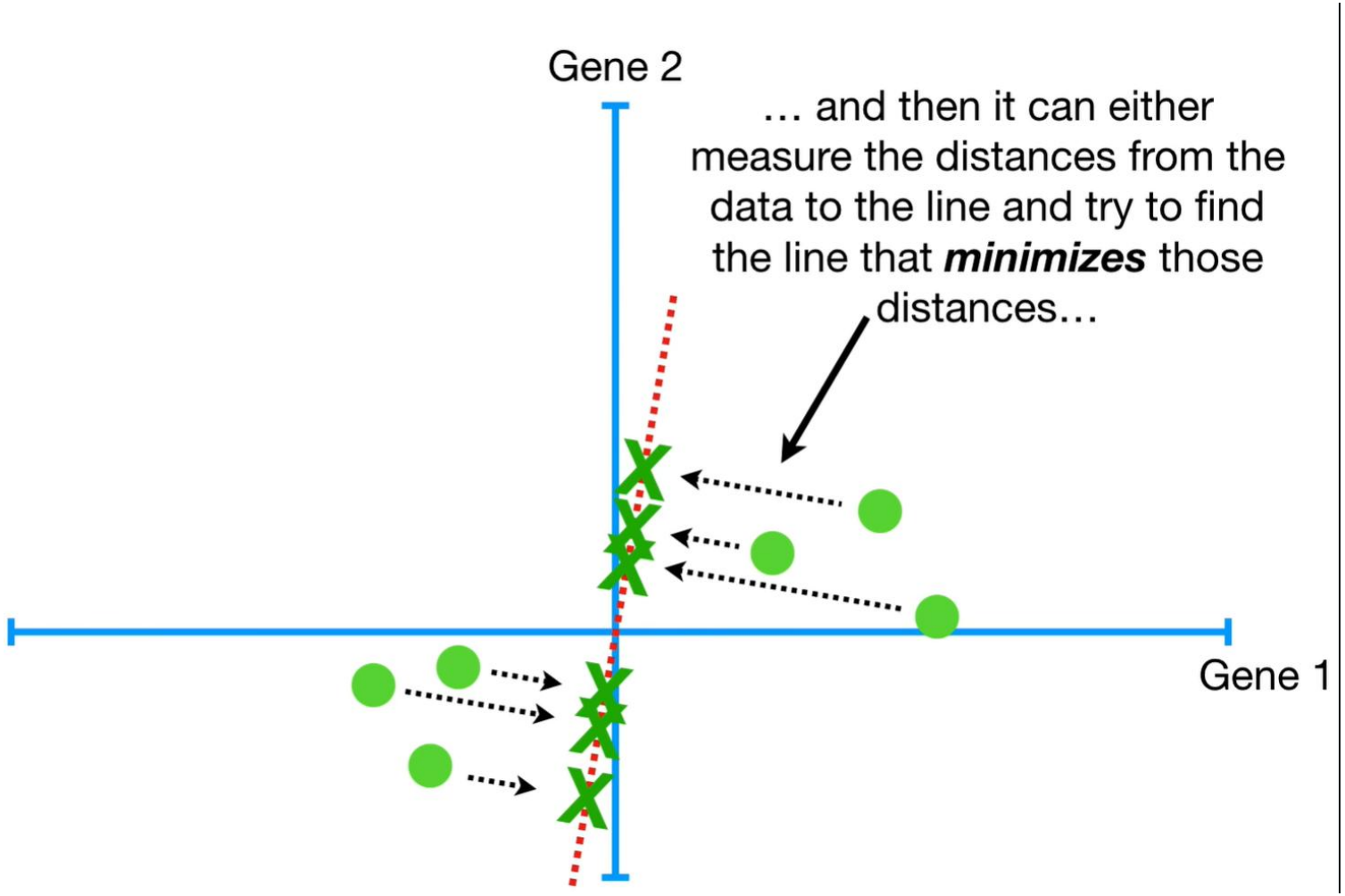


How PCA Decides the Best Line?

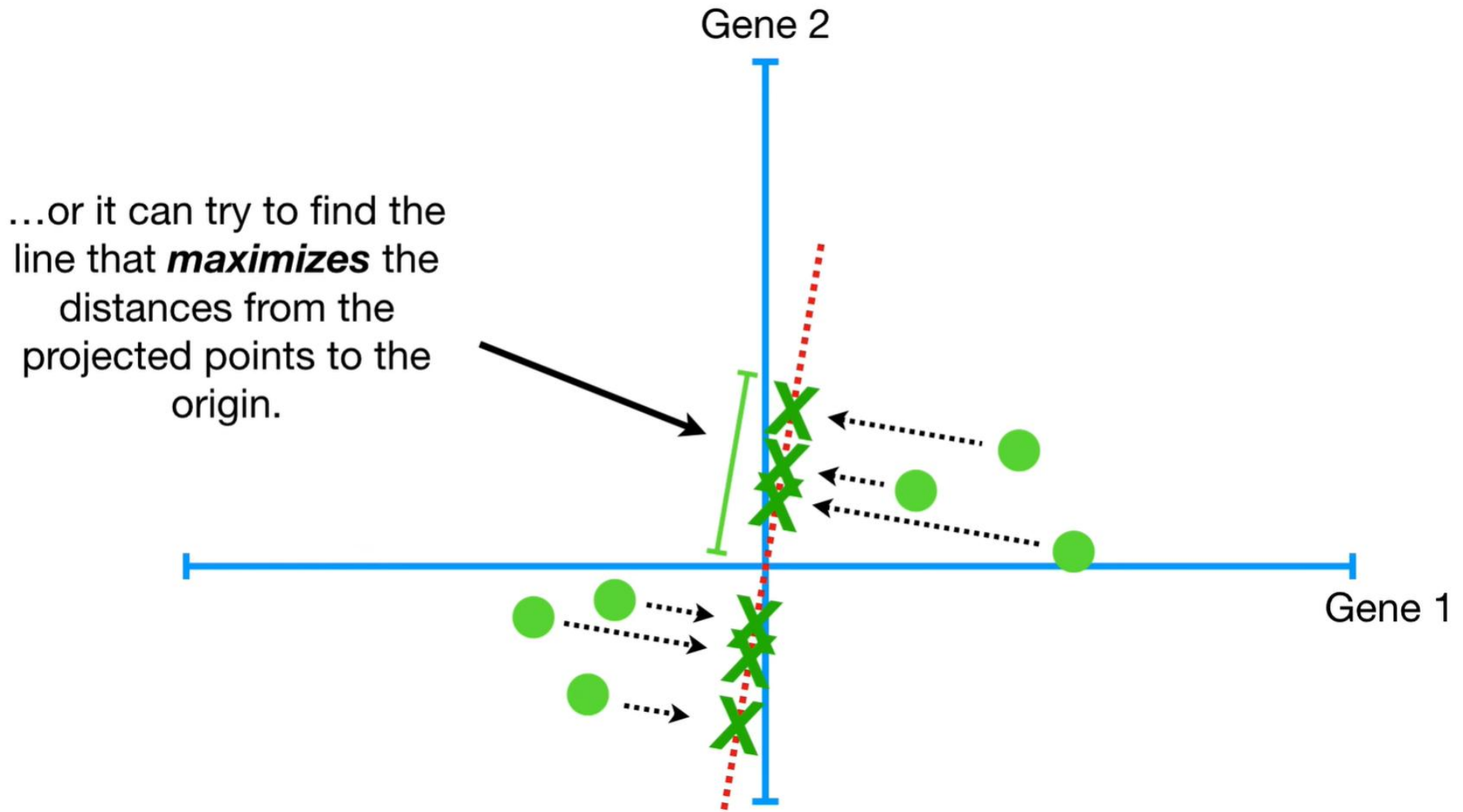
To quantify how good this line fits the data, PCA projects the data onto it…



How PCA Decides the Best Line?

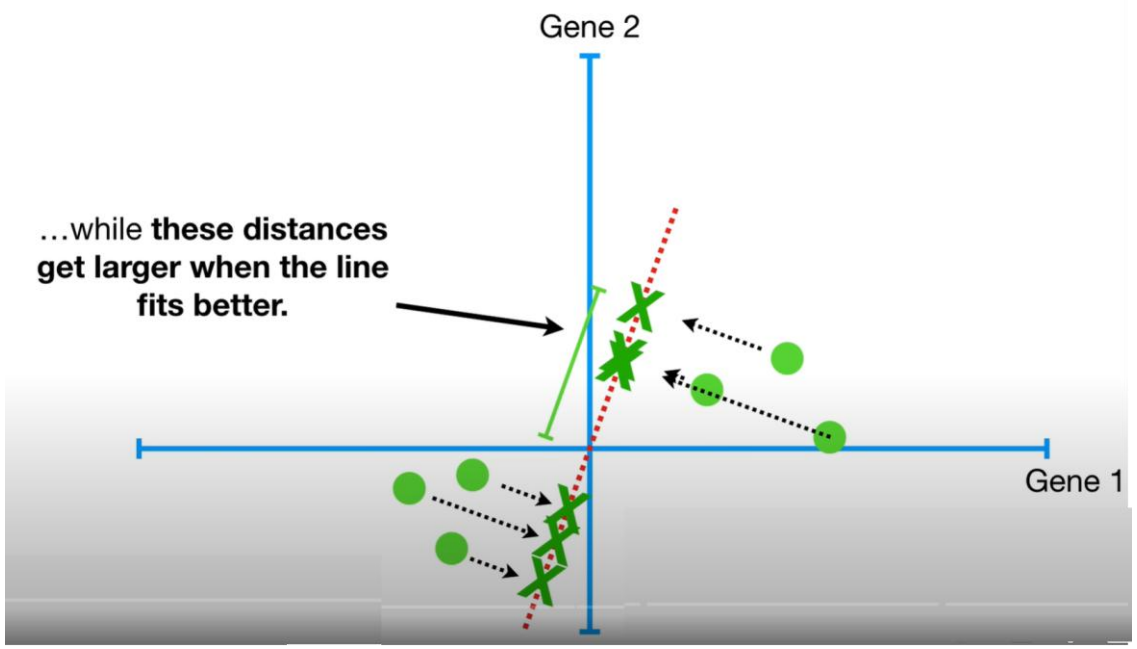


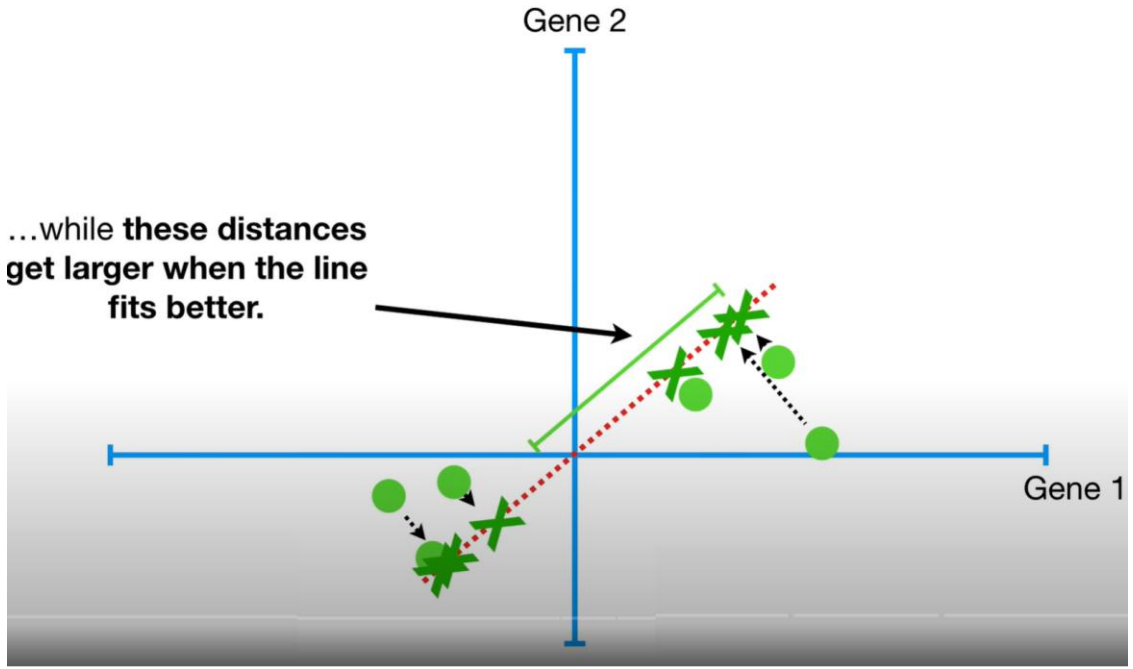
How PCA Decides the Best Line?



Let's Think

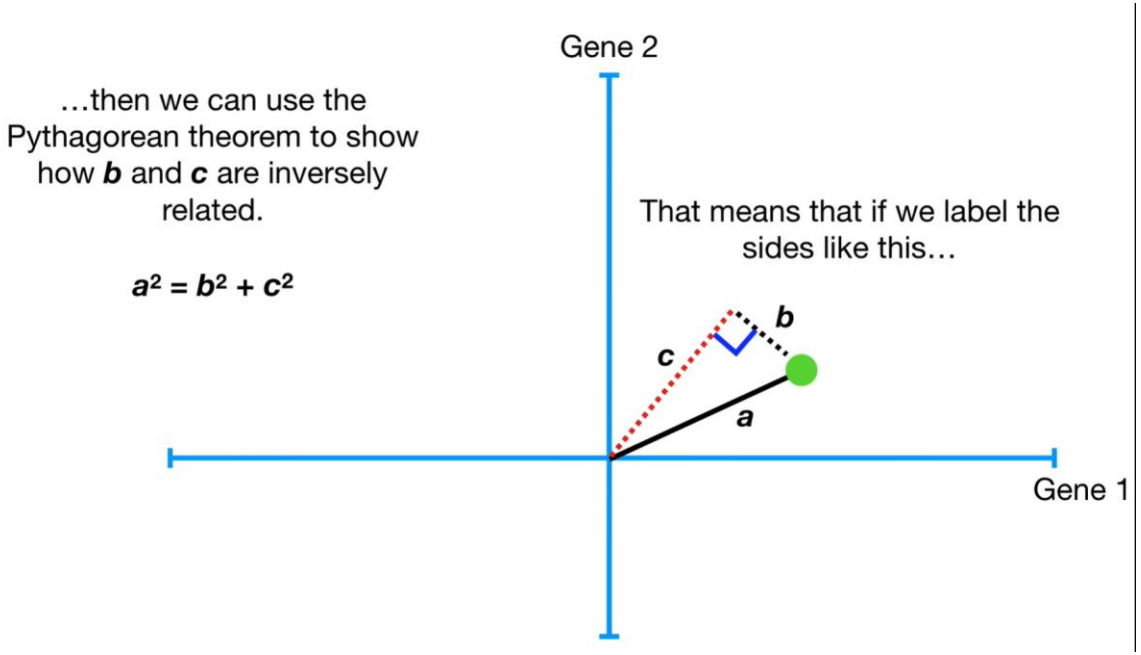
- min(distances of points from line) = max(distances of projected points to the origin)

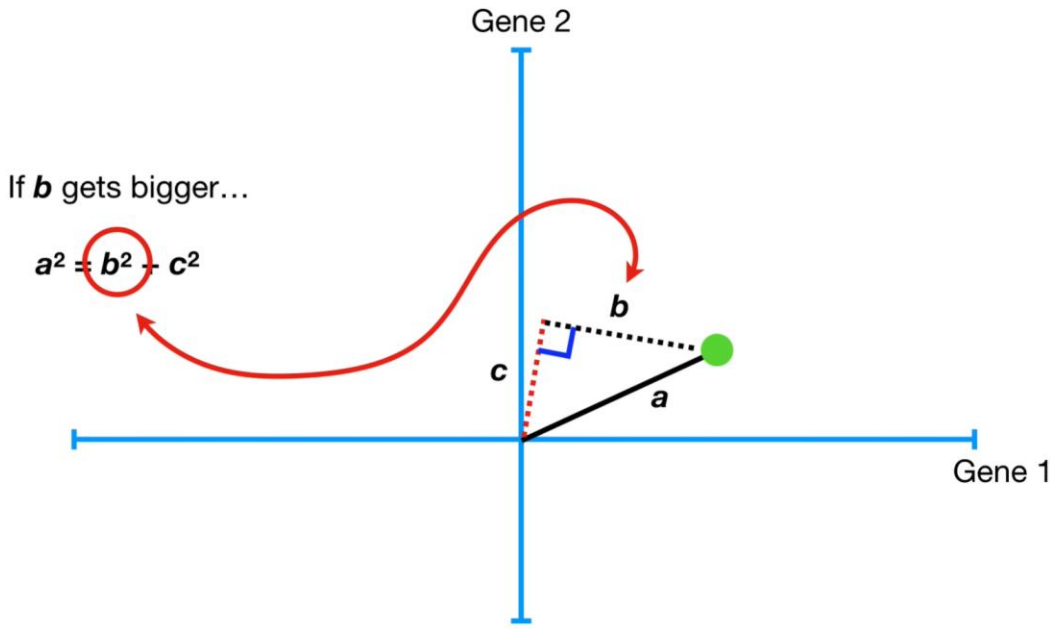




Let's Think

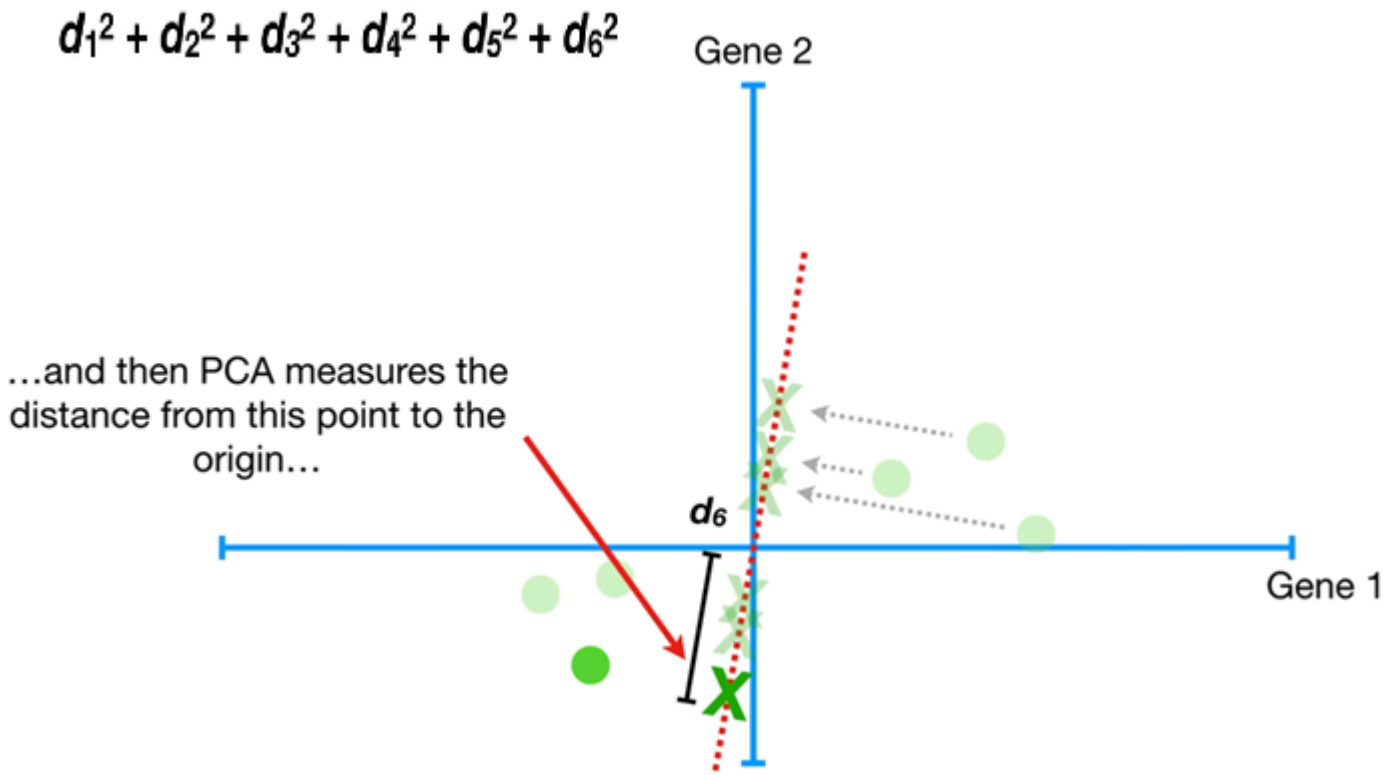
- Consider one data point.
- The distance from the point to the origin doesn't change when the red dotted line rotates.
- Project the point onto the line
- It is usually easier to calculate "c".





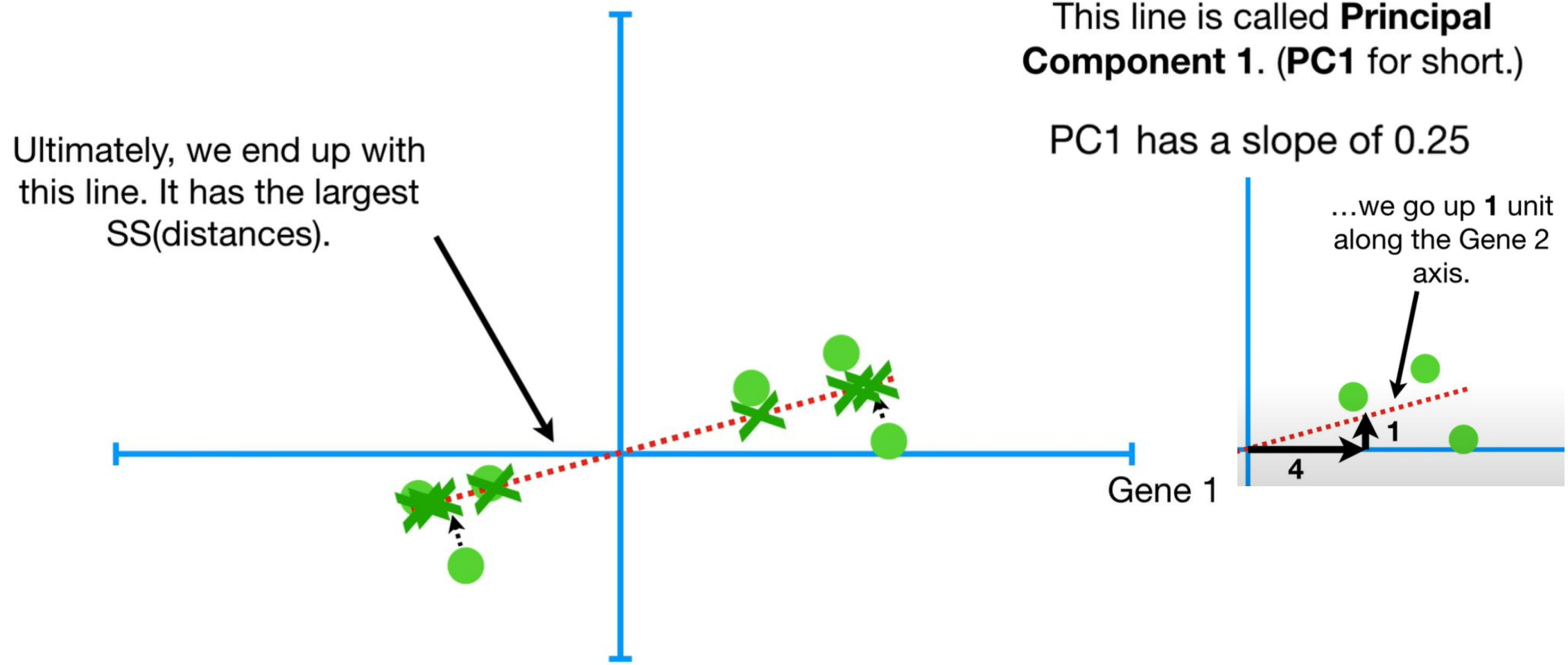
PCA

- PCA finds the best fitting line by maximizing the sum of the squared distances from the projected points to the origin.

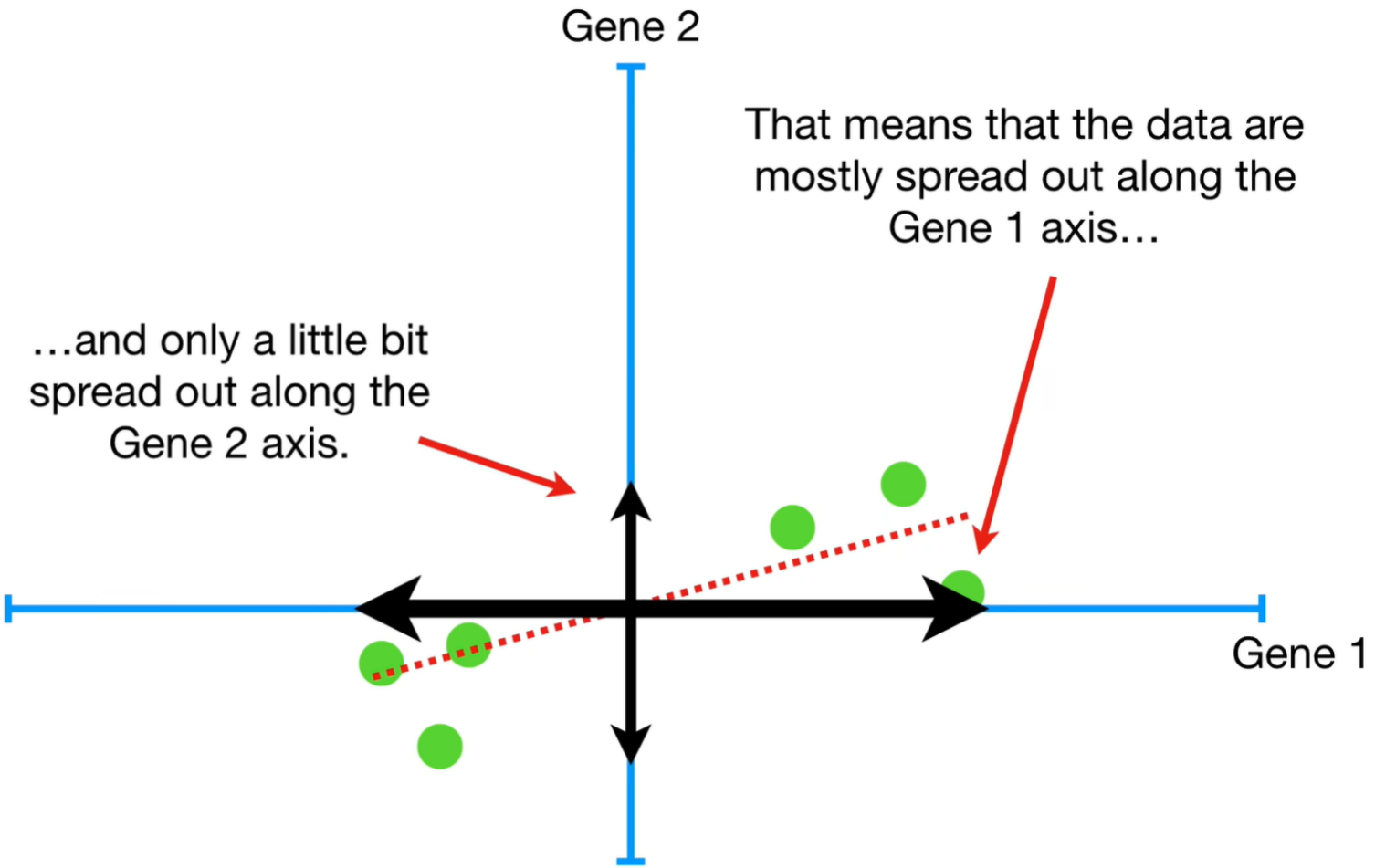


PC1

$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2$ = sum of squared distances = SS(distances)



PC1

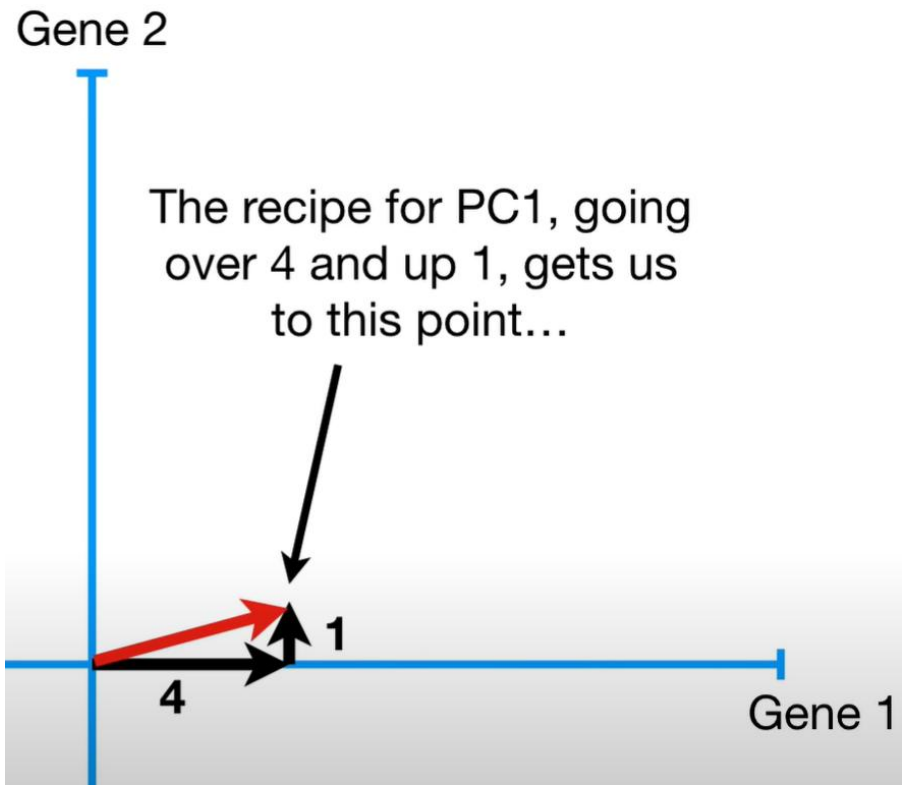


To make PC1
Mix **4** parts Gene 1 with **1** part Gene 2

The ratio of Gene 1 to Gene 2 tells you that Gene 1 is more important when it comes to describing how the data are spread out..

a "*linear combination*" of Genes 1 and 2.

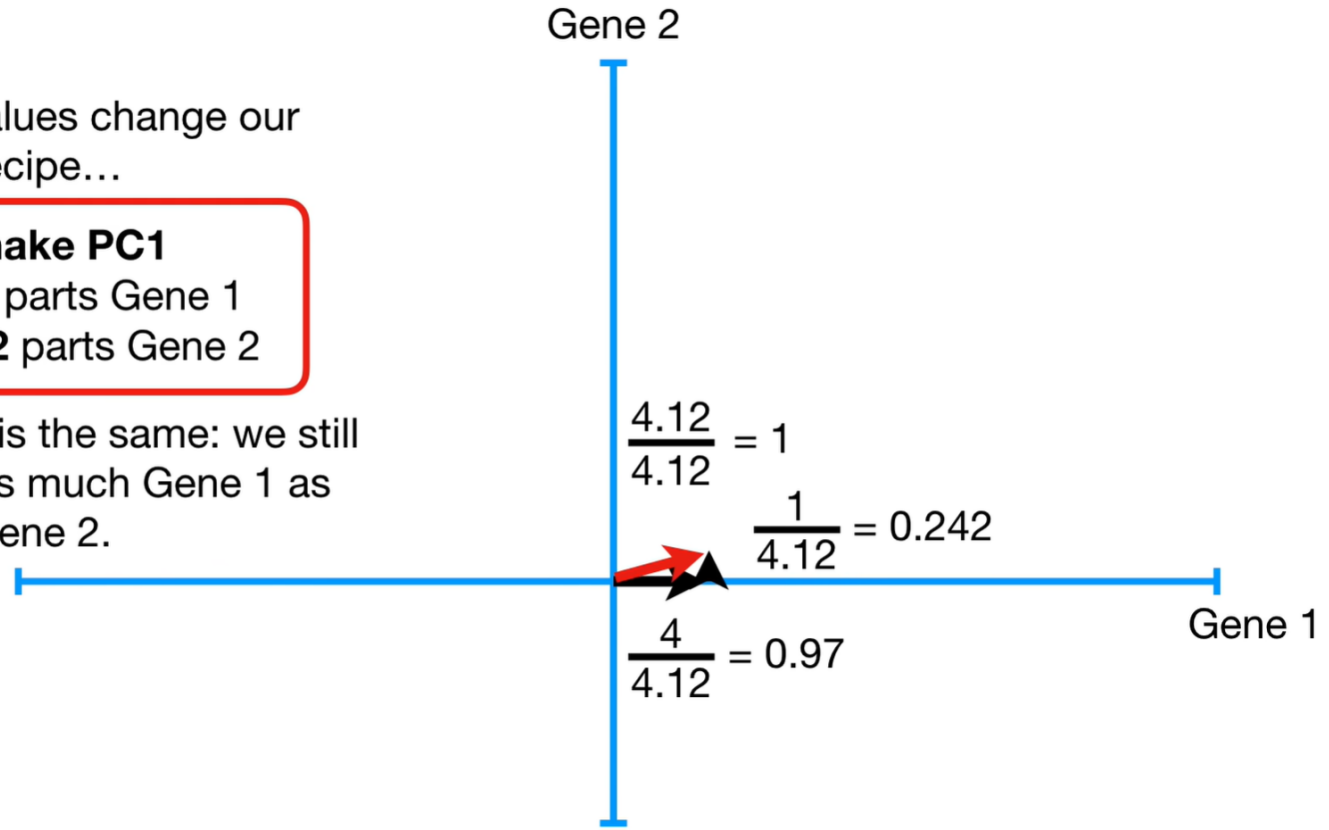
PC1



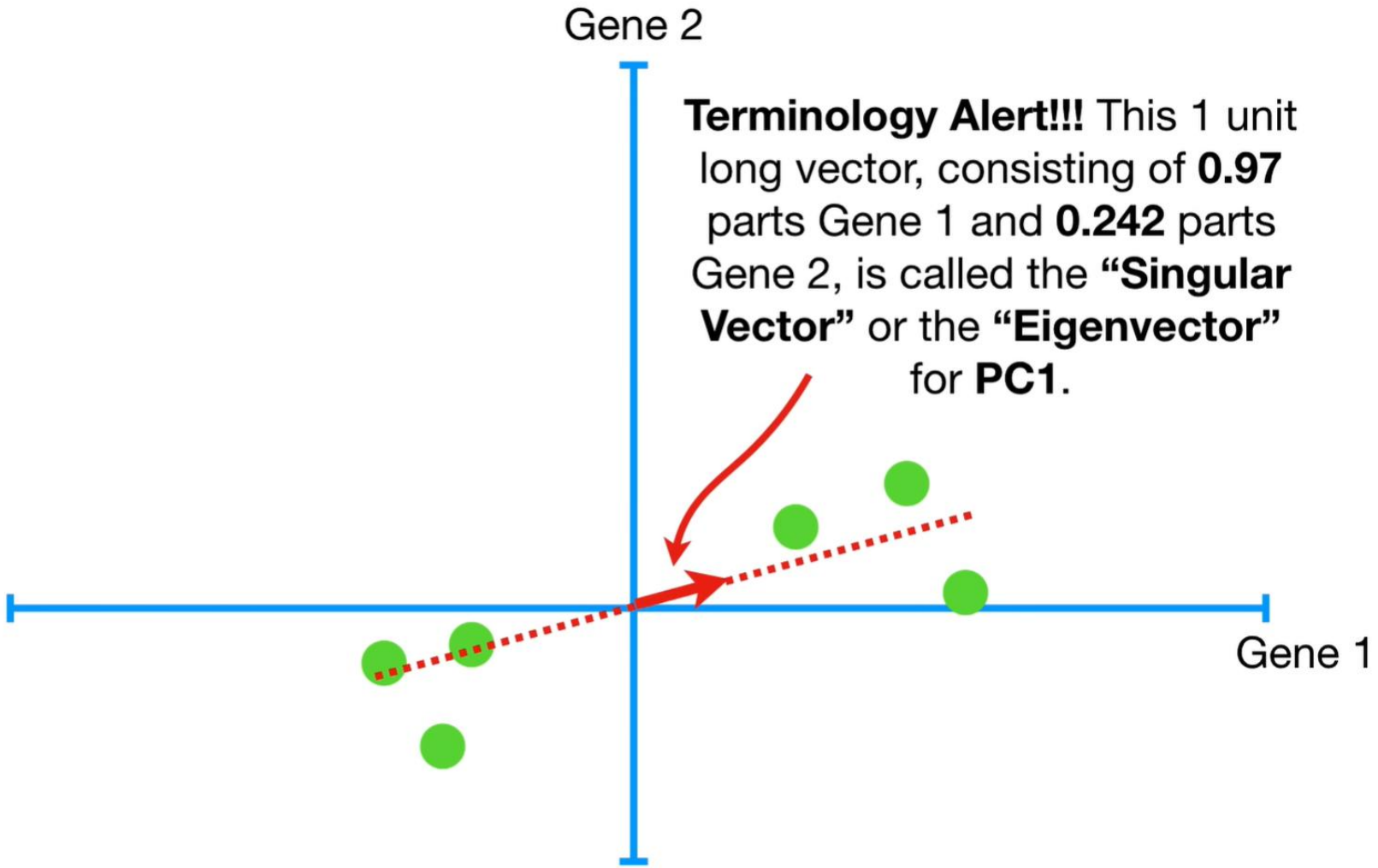
The new values change our recipe…

To make PC1
Mix **0.97** parts Gene 1
with **0.242** parts Gene 2

…but the ratio is the same: we still use 4 times as much Gene 1 as Gene 2.



PC1


Gene 2
Terminology Alert!!! This 1 unit long vector, consisting of 0.97 parts Gene 1 and 0.242 parts Gene 2, is called the "Singular Vector" or the "Eigenvector" for PC1.
Gene 1

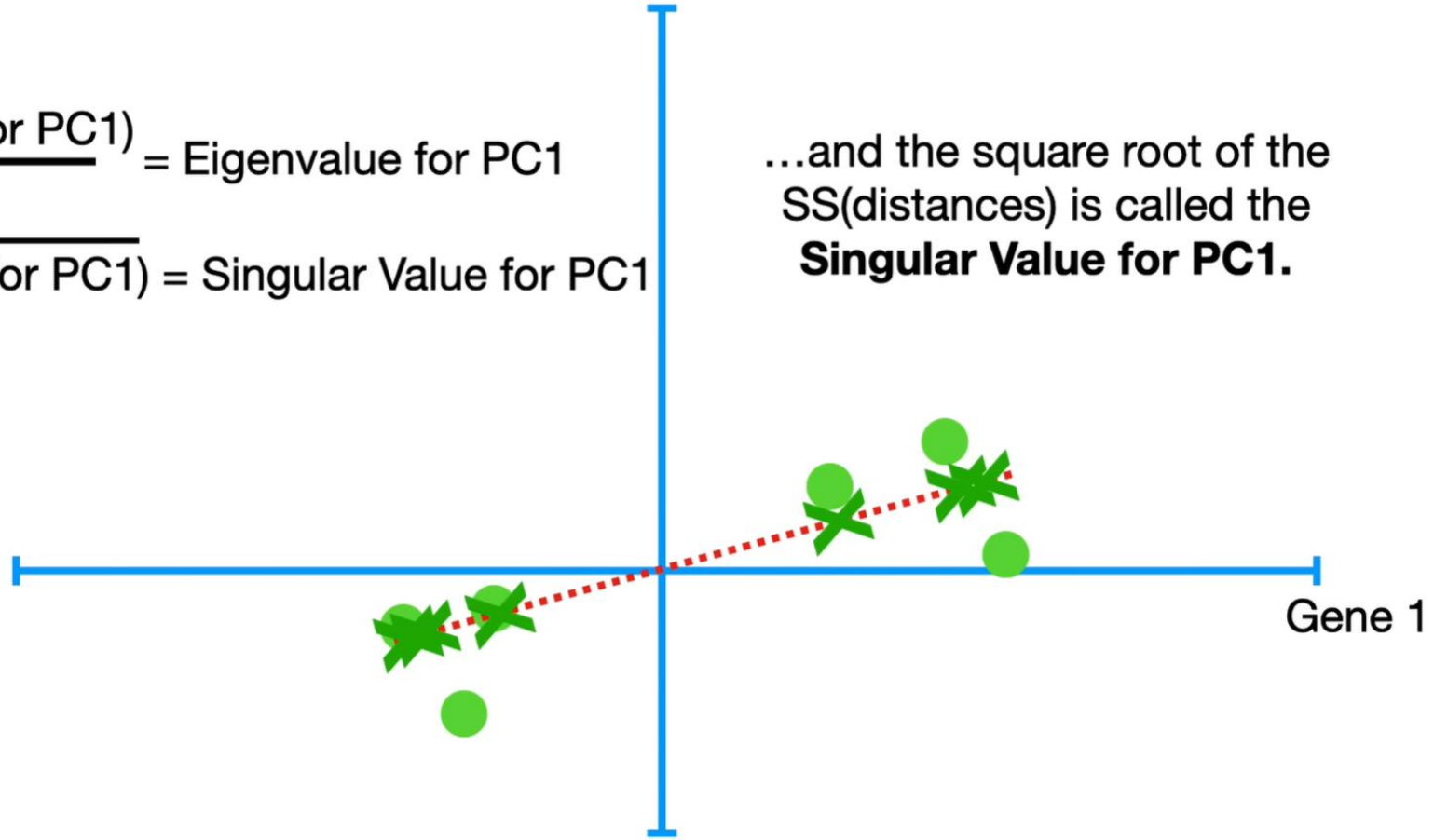
PC1

$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2$ = sum of squared distances = SS(distances)

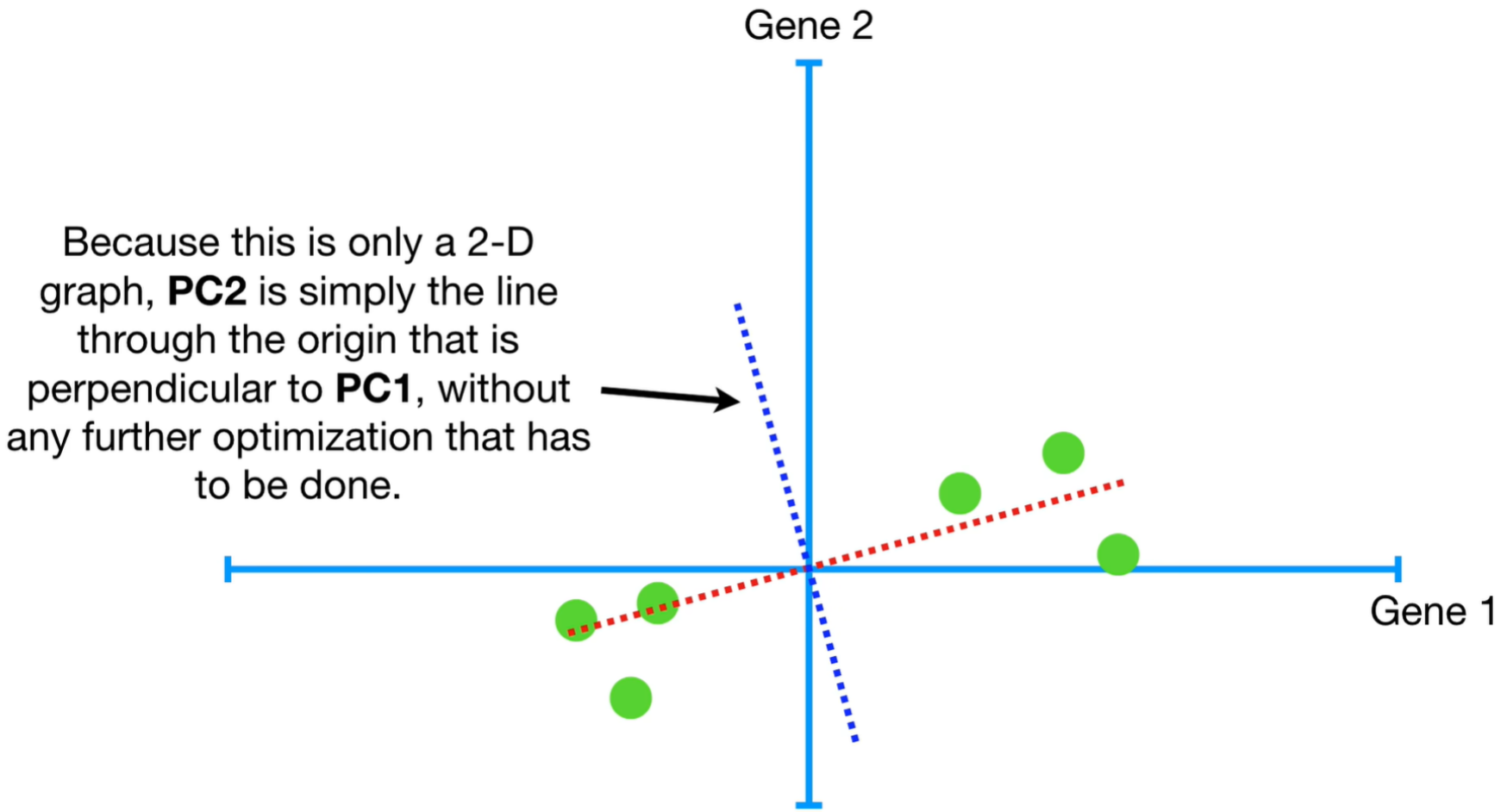
$\frac{\text{SS(distances for PC1)}}{n}$ = Eigenvalue for PC1

$\sqrt{\text{SS(distances for PC1)}}$ = Singular Value for PC1

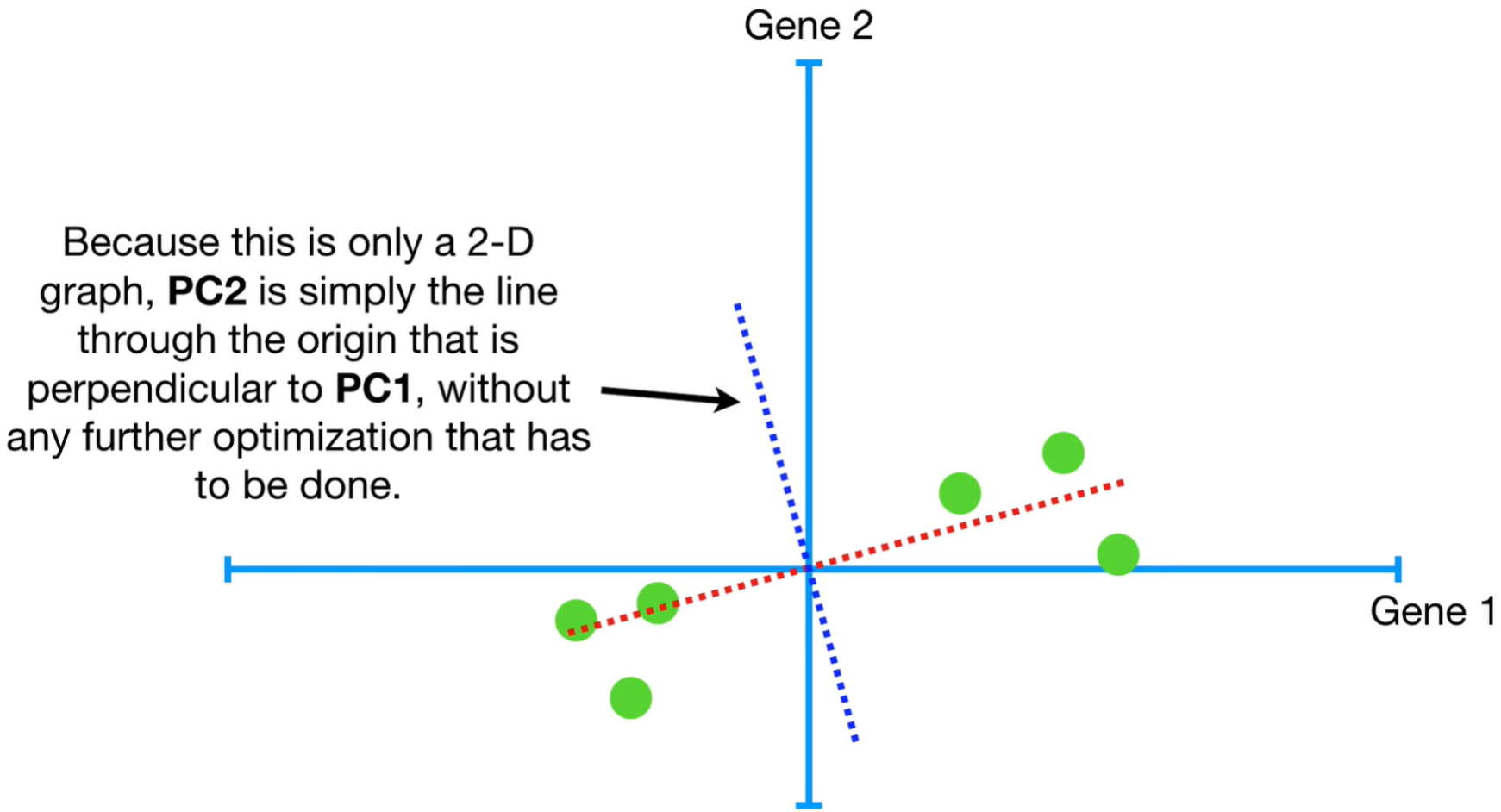
…and the square root of the SS(distances) is called the **Singular Value for PC1.**



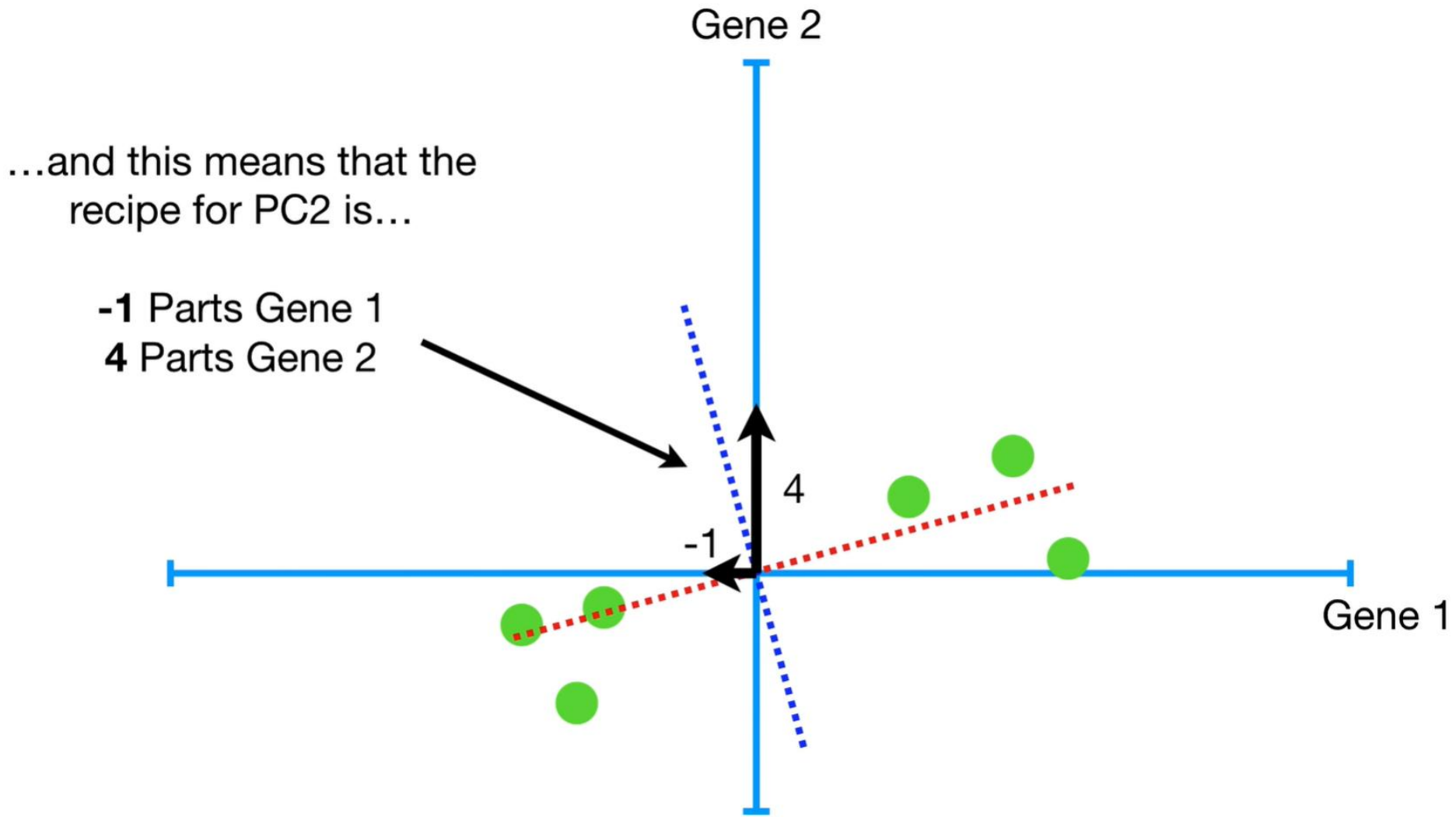
PC2


Gene 2
Because this is only a 2-D graph, **PC2** is simply the line through the origin that is perpendicular to **PC1**, without any further optimization that has to be done.
Gene 1

PC2


Because this is only a 2-D graph, **PC2** is simply the line through the origin that is perpendicular to **PC1**, without any further optimization that has to be done.
Gene 2
Gene 1

PC2


Gene 2
…and this means that the recipe for PC2 is…
-1 Parts Gene 1
4 Parts Gene 2
4
-1
Gene 1

Conclusion

- $$\underbrace{\text{Eigen-decomposition}(C)}_{C=X_{\text{centered}}^{\top}X_{\text{centered}}} \quad \longleftrightarrow \quad \underbrace{\text{SVD}(X_{\text{centered}})}_{X_{\text{centered}}=U\,\Sigma\,V^{\top}}.$$

- PCA is the eigen decomposition of covariance matrix.
- PCA is the SVD decomposition of matrix.

- **PCA on $C = X^{\top}X$:**
  - Eigenvalues of $C \rightarrow$ variances in principal directions.
  - Eigenvectors of $C \rightarrow$ principal axes (PCs).
- **SVD on X:**
  - Right singular vectors of $X \rightarrow$ same directions as eigenvectors of C.
  - Singular values are $\sqrt{\text{eigenvalues of } C}$.

Example

- $$X = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

- Eigen vector of covariance matrix

Covariance Matrix $C = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & 3 & 1 & 0 \\ 1 & 1 & 2 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$.

PCA on C means we look for eigen-decomposition:

$$C\,\mathbf{v} \;=\; \lambda\,\mathbf{v}.$$

- The eigenvalues λ indicate the variance captured.
- The eigenvectors $\mathbf{v}$ are the principal components.

Approximate Largest Eigenvalue & Eigenvector

- **Largest eigenvalue** $\lambda_{\max} \approx 4.70$.
- **Corresponding eigenvector** $\mathbf{v}_{\max} \approx (0.529,\ 0.597,\ 0.529,\ 0.291)$.

This eigenvector is **PC1** (the first principal component) of C.

Example

- $X = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$

- **SVD of X**

Performing **Singular Value Decomposition** (SVD) on the same matrix X:

$$X = U \Sigma V^\top,$$

where

- U is 4×4,
- Σ is 4×4 diagonal (singular values),
- V is 4×4.

From our eigen-decomposition of C, the largest eigenvalue was ≈ 4.70. Its square root is $\sqrt{4.70} \approx 2.17$. In the SVD of X:

- The **largest singular value** $\sigma_{\max}$ is about 2.17.
- The **corresponding right singular vector** is $\mathbf{v}_{\max} \approx (0.529,\ 0.597,\ 0.529,\ 0.291)$.

This is exactly the same vector (up to a possible sign) as the principal component from the eigen-decomposition of C.

Any Question?