DSC 140B
Representation Learning

Lecture 07 | Part 1

Recall: PCA

Goal

- **Given**: centered data $\vec{x}^{(1)}, \ldots, \vec{x}^{(n)} \in \mathbb{R}^d$
- **Map**: each data point $\vec{x}^{(i)} \in \mathbb{R}^d$ to a new feature vector $\vec{z}^{(i)} = (z_1^{(i)}, \ldots, z_k^{(i)})$ in $\mathbb{R}^k$.
- This is **dimensionality reduction** when $k < d$.

PCA

- **Given**: centered data $\{\vec{x}^{(1)}, \dots, \vec{x}^{(n)}\} \in \mathbb{R}^d$, number of components k.

- Compute covariance matrix C, top $k \leq d$ eigenvectors $\vec{u}^{(1)}$, $\vec{u}^{(2)}$, …, $\vec{u}^{(k)}$.

- For any vector $\vec{x} \in \mathbb{R}$, its new representation in $\mathbb{R}^k$ is $\vec{z} = (z_1, z_2, \dots z_k)^T$, where:

$$\begin{aligned} z_1 &= \vec{x} \cdot \vec{u}^{(1)} \\ z_2 &= \vec{x} \cdot \vec{u}^{(2)} \\ &\vdots \\ z_k &= \vec{x} \cdot \vec{u}^{(k)} \end{aligned}$$

PCA: Matrix Formulation

- **Given**: $n \times d$ centered data matrix X.
- Let U be matrix whose k columns are top k eigenvectors of C.
- Compute new data matrix $Z = XU$.
  - Result is $n \times k$ matrix.

DSC 140B

Representation Learning

Lecture 07 | Part 2

Interpreting PCA

Three Interpretations

- What is PCA doing?

- PCA is an **orthogonal projection** of the data onto a lower-dimensional subspace.

- Three interpretations:
  1. Maximizing total variance
  2. Finding the best reconstruction
  3. Decorrelating features

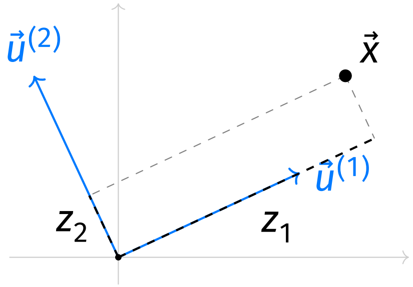
Orthogonal Projections

- **Recall:** we change basis by projecting points onto new basis vectors.

$$z_1 = \vec{x} \cdot \vec{u}^{(1)} \quad , \quad z_2 = \vec{x} \cdot \vec{u}^{(2)}$$

- If U is the matrix with columns $\vec{u}^{(1)}, \vec{u}^{(2)}$, and X is the data matrix, then the new representation is:

$$Z = XU$$



Orthogonal Projections

- If X is $n \times d$ and we use d basis vectors, then XU just changes basis.
  - Still d dimensions, no information lost.

- But if we use $k < d$ basis vectors, then XU projects points onto a k-dimensional subspace.
  - Some information is lost.
  - This is dimensionality reduction, and it is what PCA does.

Visualization

```
http://dsc140b.com/static/vis/pca-3d-spheroid/
```

PCA

- Any choice of k orthonormal basis vectors defines a projection onto a k-dimensional subspace.
- PCA chooses the basis vectors to be the top k eigenvectors of the covariance matrix C.
- In a sense, PCA is the **best** choice of basis.

View #1: Maximizing Total Variance

- The basis chosen by PCA maximizes the “total variance” of the new data.

Deriving PCA

- We derived PCA by maximizing variance.
- **Idea**: the direction of max variance is most interesting; let's use it as our first basis vector.
- We found that the direction of max variance is the top eigenvector of the covariance matrix C.

Deriving PCA

- We then iterated.
- **Idea**: the *second* most interesting direction is the one with max variance *orthogonal* to the first.
- We found that this is the *second* eigenvector of C.

Deriving PCA

- **Idea**: the k most interesting directions are the k directions with maximum variance, orthogonal to each other.

- We found that these are the top k eigenvectors of C.

- They capture the “shape” of the data.

Maximizing Variance

- At each step, we chose the direction that maximized variance.
- Overall, PCA chooses the k-dimensional subspace that maximizes the “total variance”.

Total Variance

- Consider a data set $\vec{x}^{(1)}, \ldots, \vec{x}^{(n)} \in \mathbb{R}^d$ with features $x_1, x_2, \ldots, x_d$.
- The **total variance** of the data is
$$\sum_{j=1}^{d} \text{Var}(\text{feature } x_j)$$
- Example: with phone width, height as features, the total variance is:
$$\text{Var}(\text{widths}) + \text{Var}(\text{heights})$$

Total Variance

- Now say you do an orthogonal projection of the data onto a k-dimensional subspace: $Z = XU$.
- This is a new data set with new features.
- The **total variance** of the new data set is

$$\sum_{j=1}^{k} \text{Var}(\text{feature } z_j)$$

Claim

- Out of all orthogonal projections onto k-dimensional subspaces, PCA **maximizes** the total variance of the new data.

Visualization

`http://dsc140b.com/static/vis/pca-3d-spheroid/`

Variance and Eigenvalues

- **Recall:** variance in direction of unit vector $\vec{u}$ is

$$\text{Var}(\vec{u}) = \vec{u}^T C \vec{u}$$

- If $\vec{u}$ is an eigenvector of C with eigenvalue λ, then

$$\text{Var}(\vec{u}) = \vec{u}^T C \vec{u} = \vec{u}^T(\lambda \vec{u}) = \lambda(\vec{u}^T \vec{u}) = \lambda$$

- Thus, the variance of the kth new PCA feature is the kth largest eigenvalue λ_k.

Total Variance of PCA Features

- If we use PCA to project onto k dimensions, the total variance of the new data is:

$$\sum_{j=1}^{k} \text{Var}(\text{feature } z_j) = \sum_{j=1}^{k} \lambda_j$$

- I.e., the sum of the top k eigenvalues of C.

Exercise

Imagine we throw a perfectly thin chocolate chip pancake in the air.

While it is up there, you measure the 3D coordinates of n chocolate chips embedded in the pancake and compute the sample covariance matrix of the data.

What is the third eigenvalue of the covariance matrix?

Main Idea

PCA maximizes the total variance of the new data. I.e., chooses the most “interesting” new features which are not redundant.

View #2: Minimizing Reconstruction Error

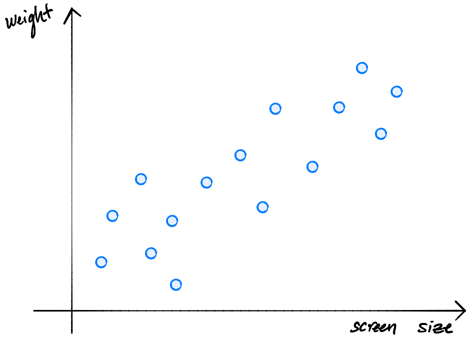
- PCA can also be viewed as minimizing the “reconstruction error”.

Reconstructing Points

- PCA reduces dimensionality from $\mathbb{R}^d \to \mathbb{R}^k$
- Suppose we have the “new” representation in $\mathbb{R}^k$.
- Can we “reconstruct” the original point in $\mathbb{R}^d$?

Back to $\mathbb{R}^d$

- Suppose new representation of $\vec{x}$ is z.
- $z = \vec{x} \cdot \vec{u}^{(1)}$
- Idea: $\vec{x} \approx z\vec{u}^{(1)}$



Reconstructions

- Suppose we map $\vec{x} \in \mathbb{R}^d$ to $\vec{z} \in \mathbb{R}^k$ by projecting onto k orthonormal basis vectors $\vec{u}^{(1)}, \ldots, \vec{u}^{(k)}$:

$$z_1 = \vec{x} \cdot \vec{u}^{(1)} \quad , \quad z_2 = \vec{x} \cdot \vec{u}^{(2)} \quad , \quad \ldots \quad , \quad z_k = \vec{x} \cdot \vec{u}^{(k)}$$

- Equivalently, $\vec{z} = U^T \vec{x}$ where U is the $d \times k$ matrix with columns $\vec{u}^{(1)}, \ldots, \vec{u}^{(k)}$.

- The **reconstruction** of $\vec{x}$ is

$$z_1 \vec{u}^{(1)} + z_2 \vec{u}^{(2)} + \ldots + z_k \vec{u}^{(k)} = U\vec{z} = UU^T\vec{x}$$

Reconstruction Error

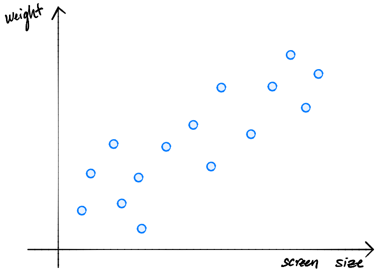
- The reconstruction $UU^T\vec{x}$ *approximates* the original point, $\vec{x}$.

- The **reconstruction error** for a single point, $\vec{x}$:

$$\|\vec{x} - U\vec{U}^T\vec{x}\|^2$$

- Total reconstruction error:

$$\sum_{i=1}^{n} \|\vec{x}^{(i)} - UU^T\vec{x}^{(i)}\|^2$$



Goal

- Here's another way to frame dimensionality reduction.
- **Given**: centered data $\vec{x}^{(1)}, \ldots, \vec{x}^{(n)} \in \mathbb{R}^d$, dimensionality k.
- **Goal**: find orthonormal $d \times k$ projection matrix U that **minimizes** total reconstruction error:

$$\sum_{i=1}^{n} \| \vec{x}^{(i)} - U\vec{z}^{(i)} \|^2$$

Visualization

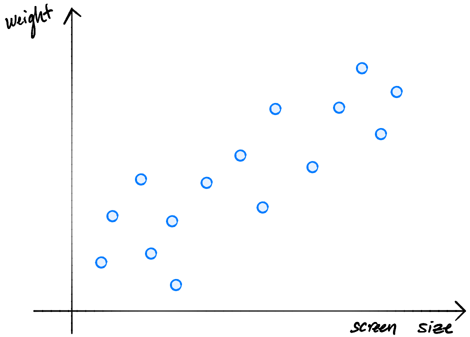
`http://dsc140b.com/static/vis/pca-max_variance/`

View #2: Reconstruction Error

- Claim: PCA **minimizes** the total reconstruction error out of all orthogonal projections onto k-dimensional subspaces.

Lines of Best Fit

- Both **least squares regression** and **PCA** find "lines/planes of best fit".
- They differ in how they measure error.
- **Least squares**: vertical distance to line/plane.
- **PCA**: perpendicular distance to line/plane.



Exercise

The chocolate chip pancake is back in the air!

We quickly perform PCA on the 3D coordinates of the chocolate chips to reduce the data to 2D.

What is the reconstruction error?

Main Idea

PCA minimizes the reconstruction error. It is the “best” projection of points onto a linear subspace of dimensionality k. When $k = d$, the reconstruction error is zero.

View #3: Decorrelation

- PCA has the effect of “decorrelating” the features.

Exercise

Suppose we perform PCA to get new features $z_1, z_2, \ldots, z_k$ and we compute the sample covariance matrix of the new data.

True or False: the covariance matrix will be diagonal.

Covariance of PCA Features

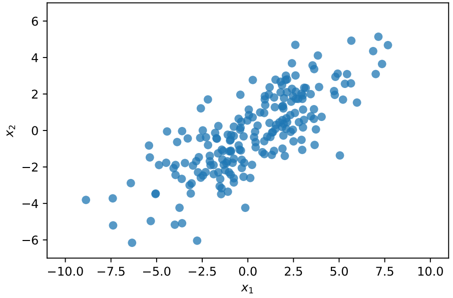
- The covariance matrix of the new features is diagonal:

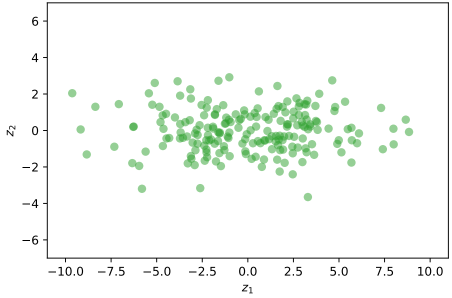
$$\begin{pmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_k \end{pmatrix}$$

- This is because we have changed to the eigenbasis of the covariance matrix.

Decorrelated Features

- A diagonal covariance matrix means the features are **uncorrelated**.





Main Idea

PCA learns a new representation by rotating the data into a basis where the features are uncorrelated (not redundant).

That is: the natural basis vectors are the principal directions (eigenvectors of the covariance matrix). PCA changes the basis to this natural basis.

DSC 140B
Representation Learning

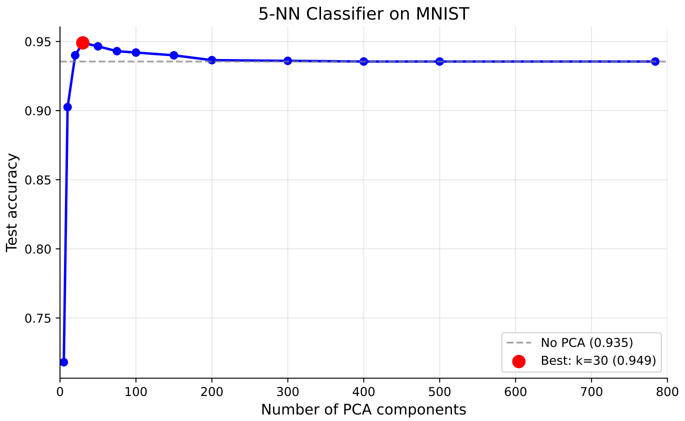
Lecture 07 | Part 3

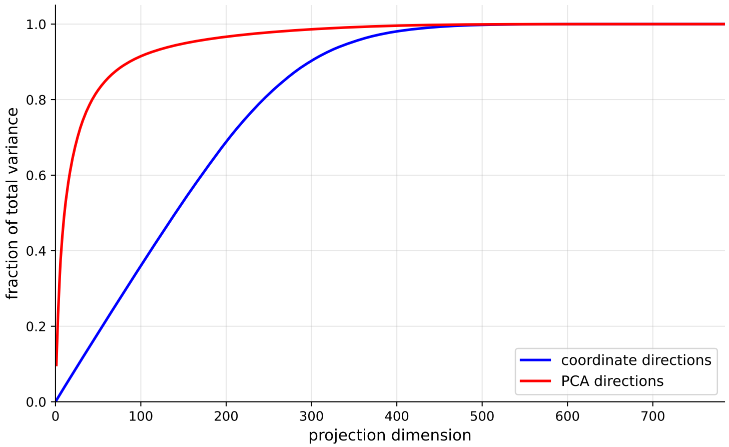
PCA in Practice

PCA in Practice

- PCA is often used in **preprocessing** before classifier is trained, etc.
- Must choose number of dimensions, k.
- One way: cross-validation.
- Another way: the elbow method.

Example: PCA Before k-NN Classifier

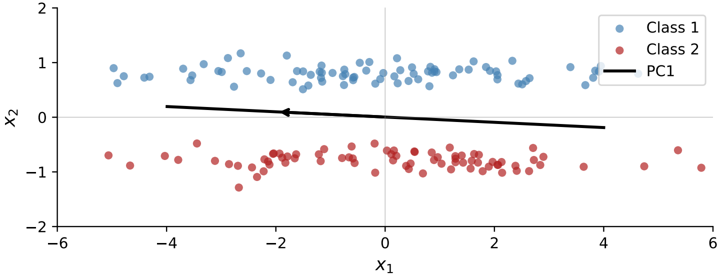



fraction of total variance
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
600
700
projection dimension
coordinate directions
PCA directions

Caution

- PCA's assumption: variance is interesting
- PCA is totally unsupervised
- The direction most meaningful for classification may not have large variance!

Example



DSC 140B

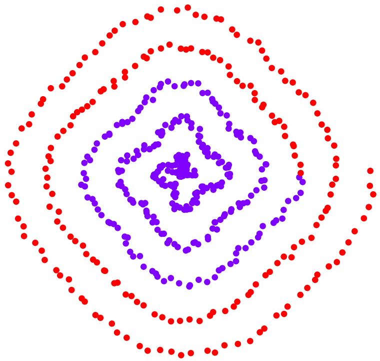
Representation Learning

Lecture 07 | Part 4

Nonlinear Dimensionality Reduction

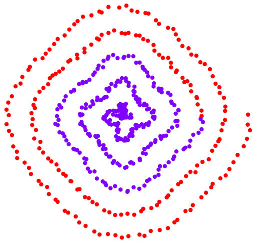
Scenario

- You want to train a classifier on this data.
- It would be easier if we could “unroll” the spiral.
- Data seems to be one-dimensional, even though in two dimensions.
- Dimensionality reduction?



PCA?

- Does PCA work here?
- Try projecting onto one principal component.



No

PCA?

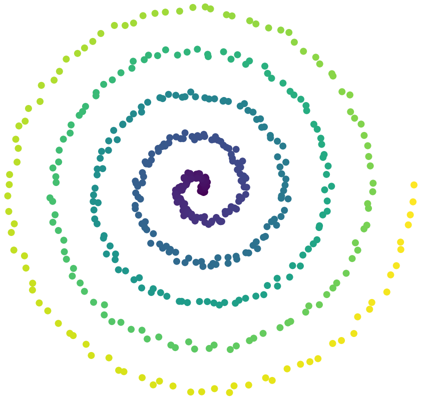
- PCA simply “rotates” the data.
- No amount of rotation will “unroll” the spiral.
- We need a fundamentally different approach that works for non-linear patterns.

Today

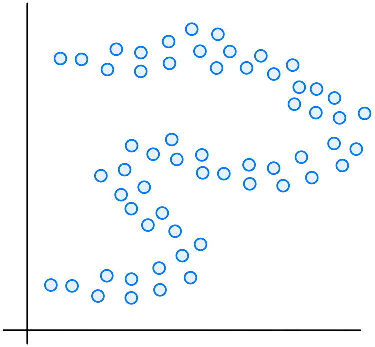
- Non-linear dimensionality reduction via **spectral embeddings**.

Rethinking Dimensionality

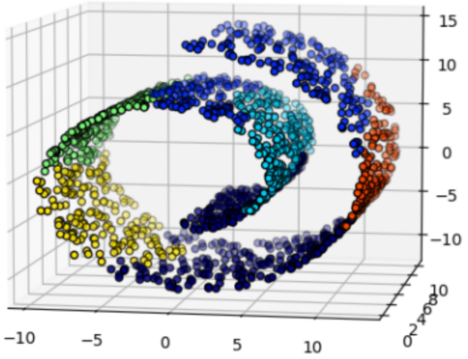
- Each point is an (x, y) coordinate in two dimensional space
- But the structure is one-dimensional
- Could (roughly) locate point using one number: distance from end.



Rethinking Dimensionality



Rethinking Dimensionality

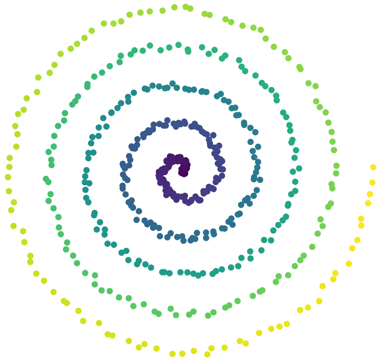


Rethinking Dimensionality

- Informally: data expressed with d dimensions, but its *really* confined to k-dimensional region
- This region is called a **manifold**
- d is the **ambient** dimension
- k is the **intrinsic** dimension

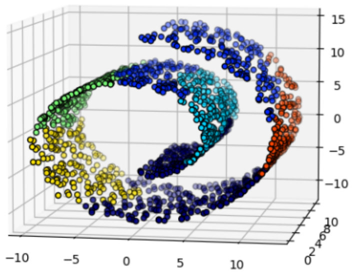
Example

- Ambient dimension: 2
- Intrinsic dimension: 1



Example

- Ambient dimension: 3
- Intrinsic dimension: 2



Example

- Ambient dimension:
- Intrinsic dimension:

Manifold Learning

- **Given**: data in high dimensions
- **Recover**: the low-dimensional manifold

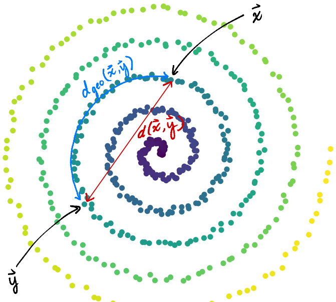
Types of Manifolds

- Manifolds can be linear
  - E.g., linear subpaces – hyperplanes
  - Learned by PCA

- Can also be non-linear (locally linear)
  - Example: the spiral data
  - Learned by **Laplacian eigenmaps**, among others

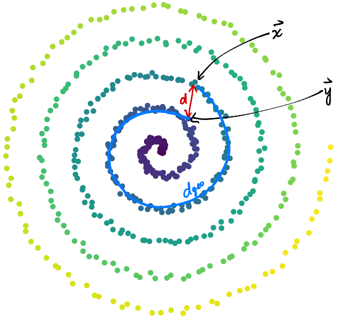
Euclidean vs. Geodesic Distances

- **Euclidean distance**: the “straight-line” distance
- **Geodesic distance**: the distance along the manifold



Euclidean vs. Geodesic Distances

- **Euclidean distance**: the “straight-line” distance
- **Geodesic distance**: the distance along the manifold



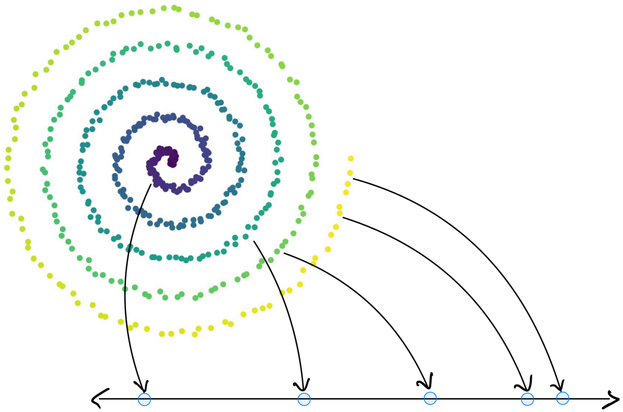
Euclidean vs. Geodesic Distances

- If data is close to a linear manifold, geodesic ≈ Euclidean
- Otherwise, can be very different

Non-Linear Dimensionality Reduction

- **Goal**: Map points in $\mathbb{R}^d$ to $\mathbb{R}^k$
- **Such that**: if $\vec{x}$ and $\vec{y}$ are close in **geodesic** distance in $\mathbb{R}^d$, they are close in **Euclidean** distance in $\mathbb{R}^k$

Embeddings



DSC 140B
Representation Learning

Lecture 07 | Part 5

Embedding Similarities

Similar Netflix Users

- Suppose you are a data scientist at Netflix
- You’re given an $n \times n$ **similarity matrix** W of users
  - entry (i, j) tells you how *similar* user i and user j are
  - 1 means “very similar”, 0 means “not at all”
- **Goal**: visualize to find patterns

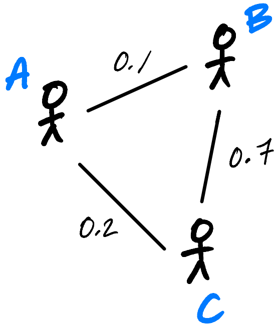
Idea

- We like scatter plots. Can we make one?
- Users are **not** vectors / points!
- They are **nodes** in a **similarity graph**

Similarity Graphs

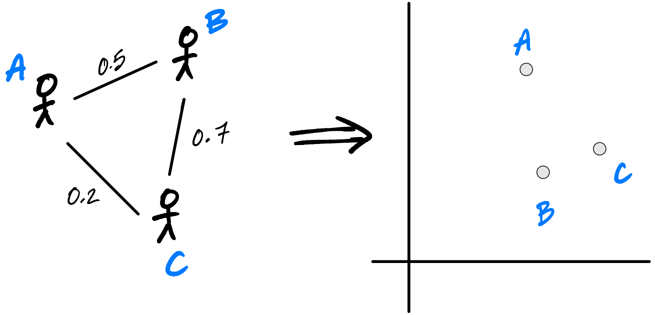
- Similarity matrices can be thought of as weighted graphs, and *vice versa*.

$$\begin{array}{c|ccc} & A & B & C \\ \hline A & 1 & 0.1 & 0.2 \\ B & 0.1 & 1 & 0.7 \\ C & 0.2 & 0.7 & 1 \end{array}$$



Goal

- **Embed** nodes of a similarity graph as points.
- Similar nodes should map to nearby points.



Today

- We will design a graph embedding approach:
  - **Spectral embeddings** via **Laplacian eigenmaps**

More Formally

- **Given**:
  - A **similarity graph** with n nodes
  - a number of dimensions, k

- **Compute**: an **embedding** of the n nodes into $\mathbb{R}^k$ so that similar objects are placed nearby

To Start

- **Given**:
  - A **similarity graph** with n nodes
  - a number of dimensions, k
- **Compute**: an **embedding** of the n nodes into $\mathbb{R}^1$ so that similar objects are placed nearby

Vectors as Embeddings into $\mathbb{R}^1$

- Suppose we have *n* nodes (objects) to embed
- Assume they are numbered 1, 2, …, *n*
- Let $f_1, f_2, \ldots, f_n \in \mathbb{R}$ be the embeddings
- We can pack them all into a vector: $\vec{f}$.
- **Goal**: find a good set of embeddings, $\vec{f}$.

Example

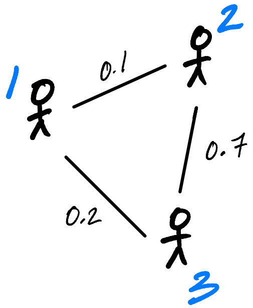
$\vec{f} = (1, 3, 2, -4)^T$

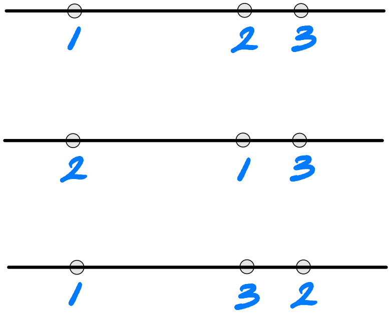
An Optimization Problem

- We'll turn it into an optimization problem:
- **Step 1**: Design a cost function quantifying how good a particular embedding $\vec{f}$ is
- **Step 2**: Minimize the cost

Exercise

Which of the following embeddings is intuitively the best, given the similarity graph below?





Cost Function for Embeddings

- **Idea**: cost is low if similar points are close
- Here is one approach:

$$\text{Cost}(\vec{f}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(f_i - f_j)^2$$

- where w_{ij} is the weight between i and j.

Interpreting the Cost

$$\text{Cost}(\vec{f}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(f_i - f_j)^2$$

- If $w_{ij} \approx 0$, that pair can be placed very far apart without increasing cost
- If $w_{ij} \approx 1$, the pair should be placed close together in order to have small cost.

Exercise

Do you see a problem with the cost function?

$$\text{Cost}(\vec{f}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(f_i - f_j)^2$$

Hint: what embedding $\vec{f}$ minimizes it?

Problem

- The cost is **always** minimized by taking $\vec{f} = 0$.
- This is a "**trivial**" solution. Not useful.
- **Fix**: require $\|\vec{f}\| = 1$
    - Really, any number would work. 1 is convenient.

Exercise

Do you see **another** problem with the cost function, even if we require $\vec{f}$ to be a unit vector?

$$\text{Cost}(\vec{f}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(f_i - f_j)^2$$

Hint: what other choice of $\vec{f}$ will **always** make this zero?

Problem

- The cost is **always** minimized by taking $\vec{f} = \frac{1}{\sqrt{n}}(1, 1, \dots, 1)^T$.

- This is a "**trivial**" solution. Again, not useful.

- **Fix**: require $\vec{f}$ to be orthogonal to $(1, 1, \dots, 1)^T$.
  - Written: $\vec{f} \perp (1, 1, \dots, 1)^T$
  - Ensures that solution is not close to trivial solution
  - Might seem strange, but it will work!

The New Optimization Problem

- **Given**: an $n \times n$ similarity matrix W
- **Compute**: embedding vector $\vec{f}$ minimizing

$$\text{Cost}(\vec{f}) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(f_i - f_j)^2$$

subject to $\|\vec{f}\| = 1$ and $\vec{f} \perp (1, 1, \dots, 1)^T$

How?

- This looks difficult.
- Let’s write it in matrix form.
- We’ll see that it is actually (hopefully) familiar.