Dimensionality Reduction

Recap

- **Unsupervised learning**
  - Learning from unlabeled data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}_{i=1}^{m} \subseteq \mathbb{R}^d$
  - Easy to scale up — necessary for large-scale training

  - **Clustering**
    - Learning a mapping $\boldsymbol{\Phi}(\,\cdot\,) : \mathbb{R}^d \rightarrow \{1,\ldots,k\}$
    - Each k may be represented by some mean $\mu_i \in \mathbb{R}^d$ (and variance, and so on …)
      - K-Means
      - Gaussian Mixture Models

Today

- **Dimensionality Reduction**
  - Learning a mapping $\mathbf{\Phi}(\,\cdot\,): \mathbb{R}^d \to \mathbb{R}^k \ (k < d)$
  - In particular, we focus on the case of **linear** $\Phi(\,\cdot\,)$
    - Precisely, we discuss Principal Component Analysis (PCA)
  - Other examples
    - ICA (Independent Component Analysis)
    - Autoencoders

Motivations

Dealing with high-dimensional data

- Many datasets are extremely high-dimensional, in its raw form

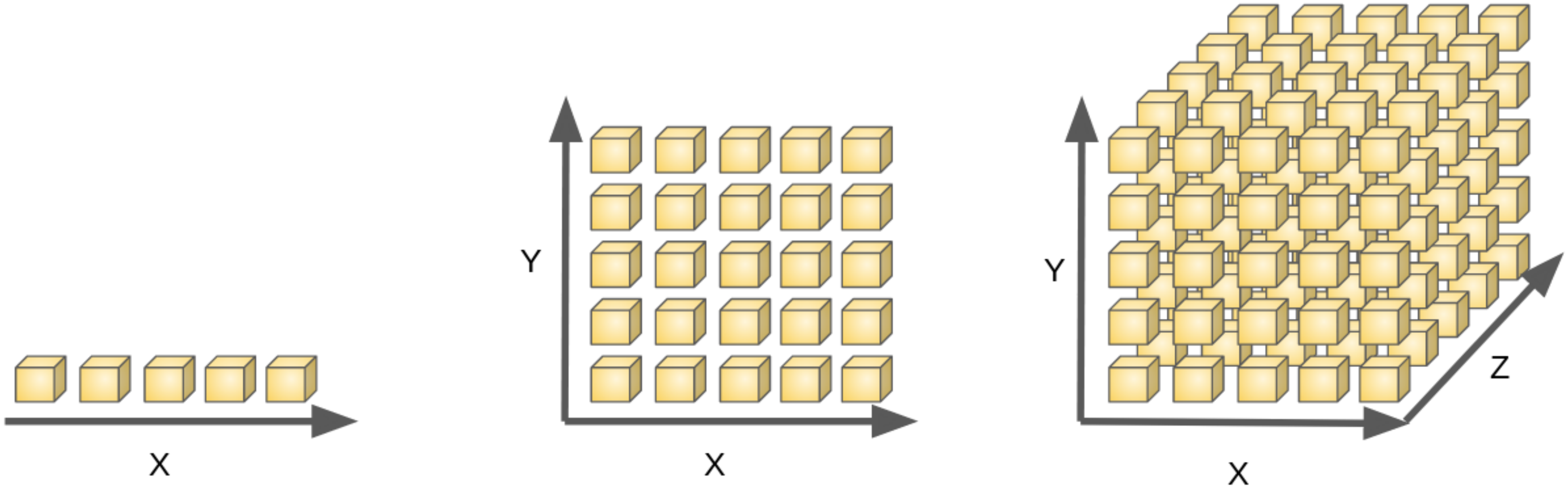
- **Example.** Suppose you are an ML engineer at Google
  - Goal. A model that detect copyrighted clips from Youtube shorts

  - The dimensionality of Youtube shorts $\mathbf{x} \in \mathbb{R}^d$ are:

    1920 x 1080 x RGB x 60FPS x 60 Seconds
    =22.4 Billion dimension

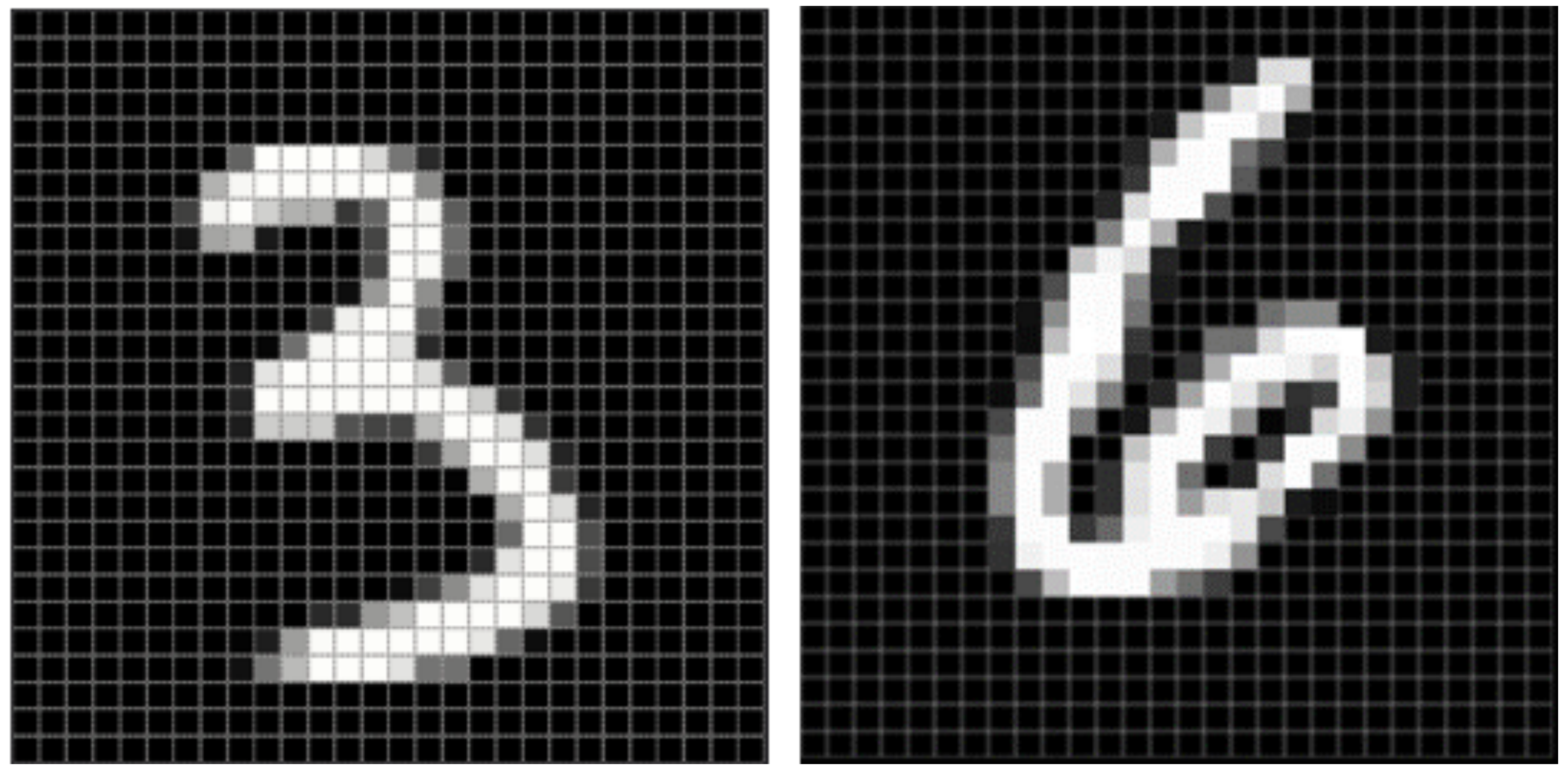
Curse of dimensionality

- Learning from high-dimensional data is challenging
  - Computation
  - Higher chance of noise
  - Difficult to visualize — for human insights
  - Difficult to find generalizable patterns (important)

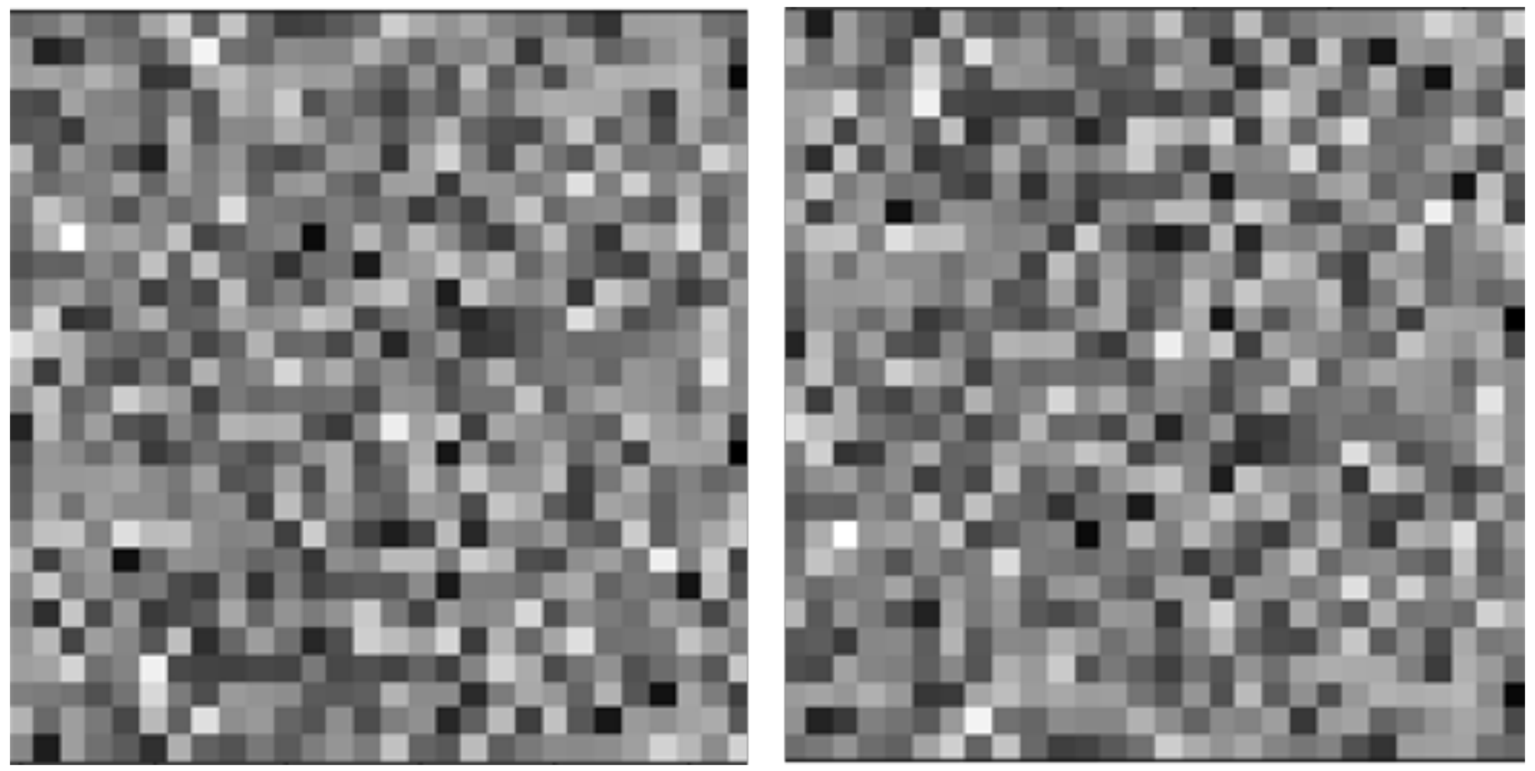


Nominal dimensionality vs. True

- But do we really need all these dimensions?
- **Example.** Handwritten digit recognition (MNIST, 28x28)



only looks like this



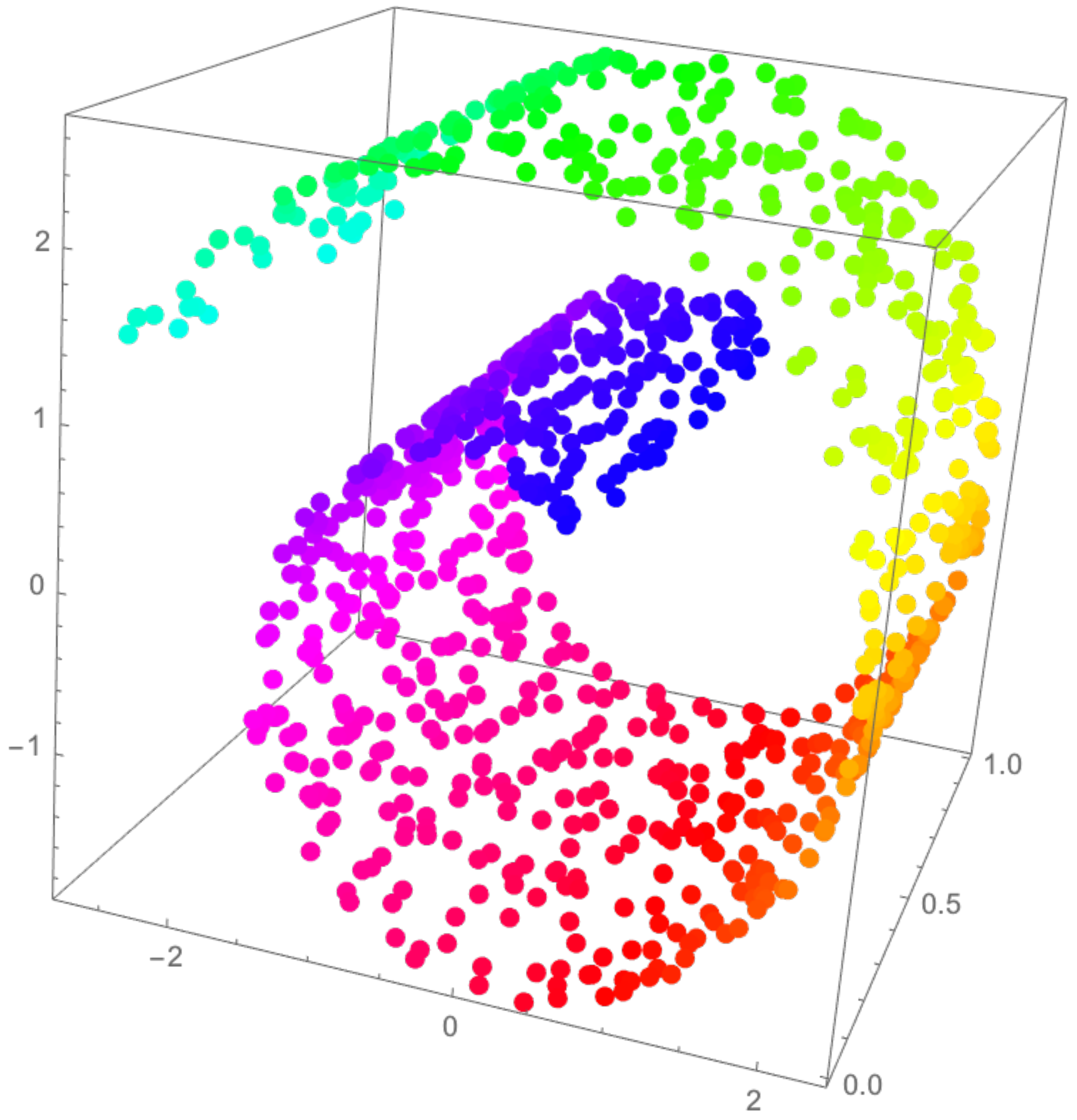
… and not like this

- That is, we are not fully utilizing $\mathbb{R}^{28\times 28} = \mathbb{R}^{784}$

Nominal dimensionality vs. True

- **Hypothesis.**
  There exists some low-dim. subspace (or submanifold) in the high-dim. space where the real data lies in

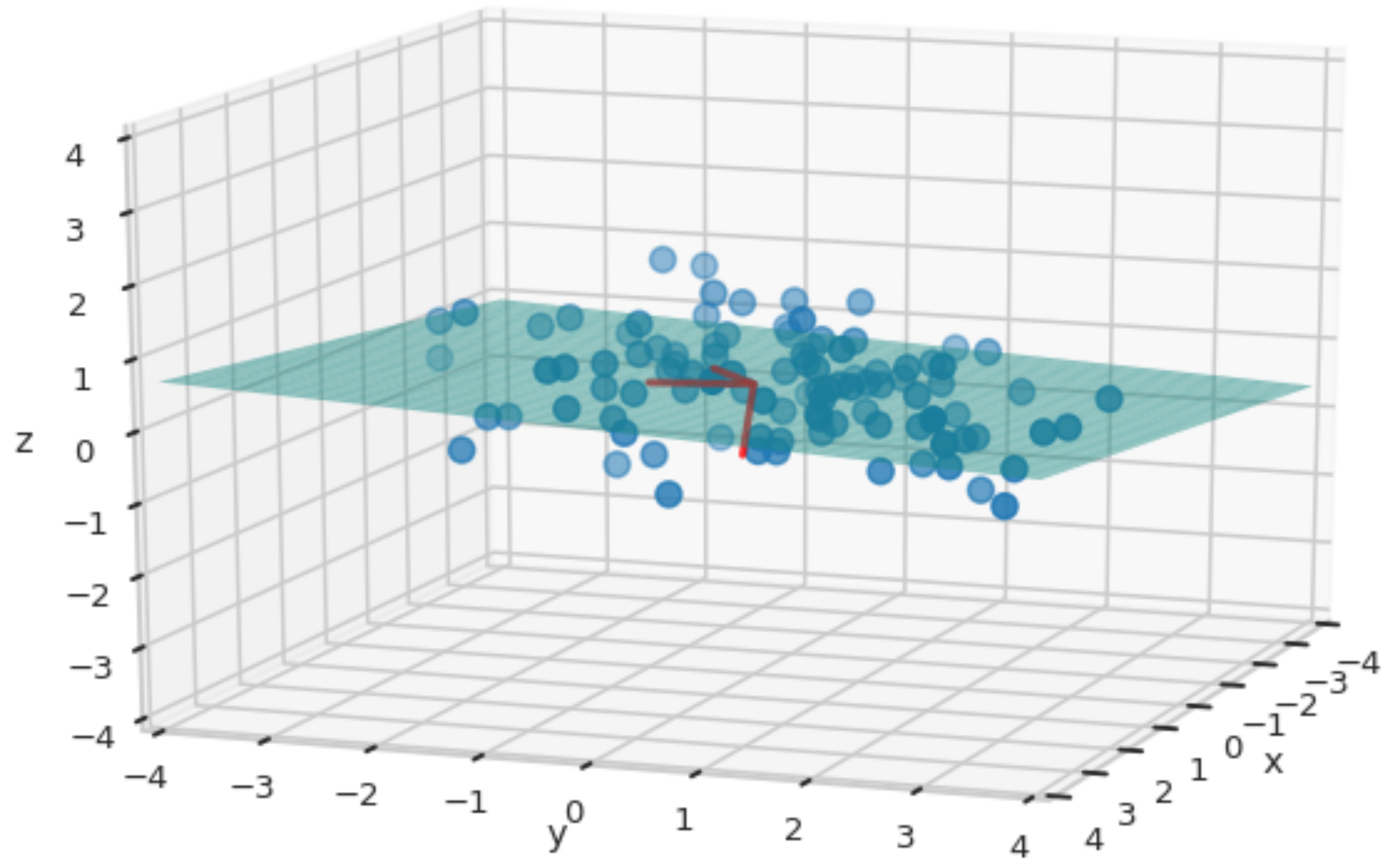
- **Dimensionality Reduction**
  Using unlabeled data to find the right mapping b/w high-dim & low-dim spaces
  - <u>Caveat</u>. Data could be noisy



Principal Component Analysis

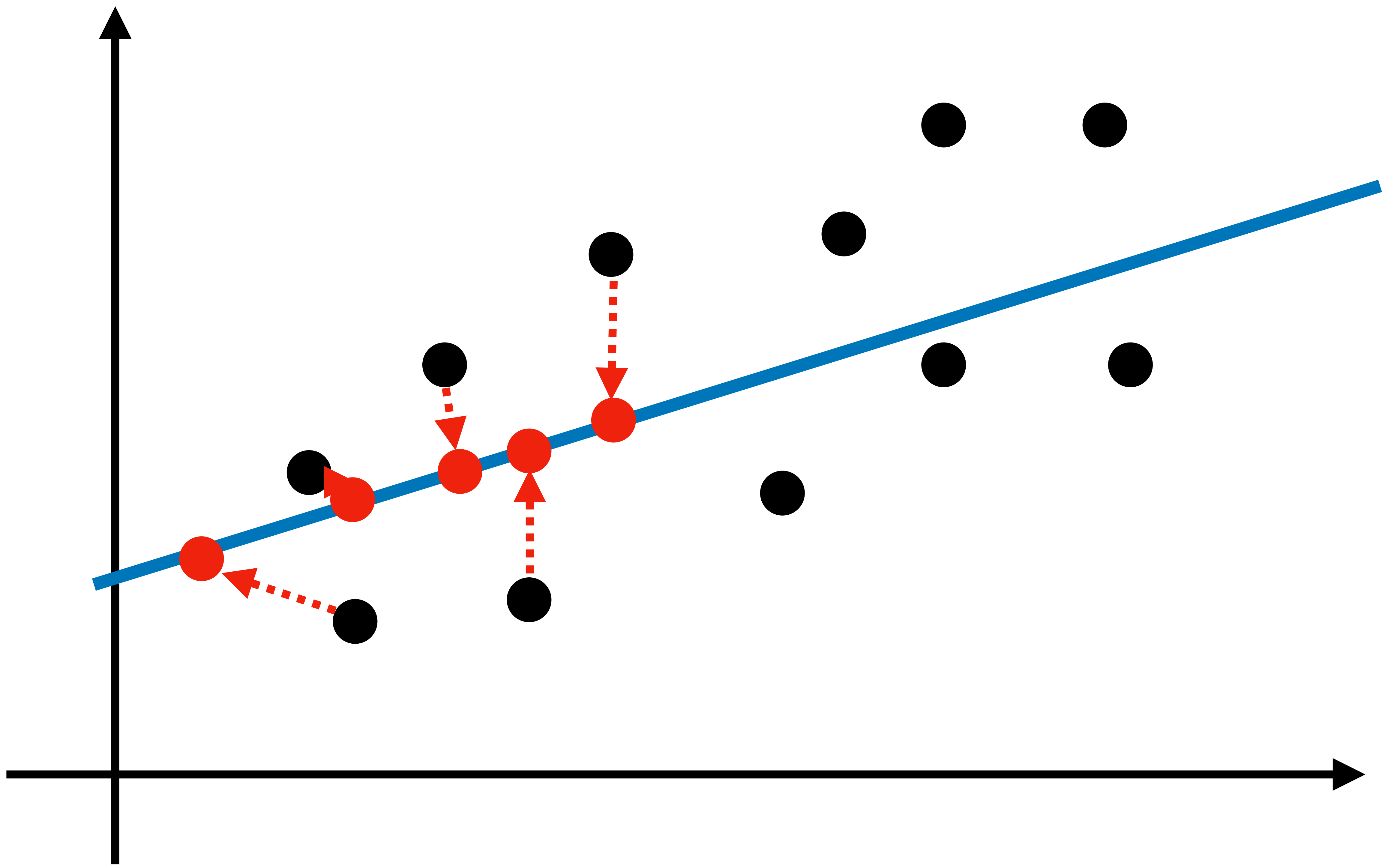
Overview

- A dimensionality reduction technique, invented by Karl Pearson (1909)
  - Uses an affine subspace of the original space
  - Many aliases — e.g., Karhunen-Loève Transform



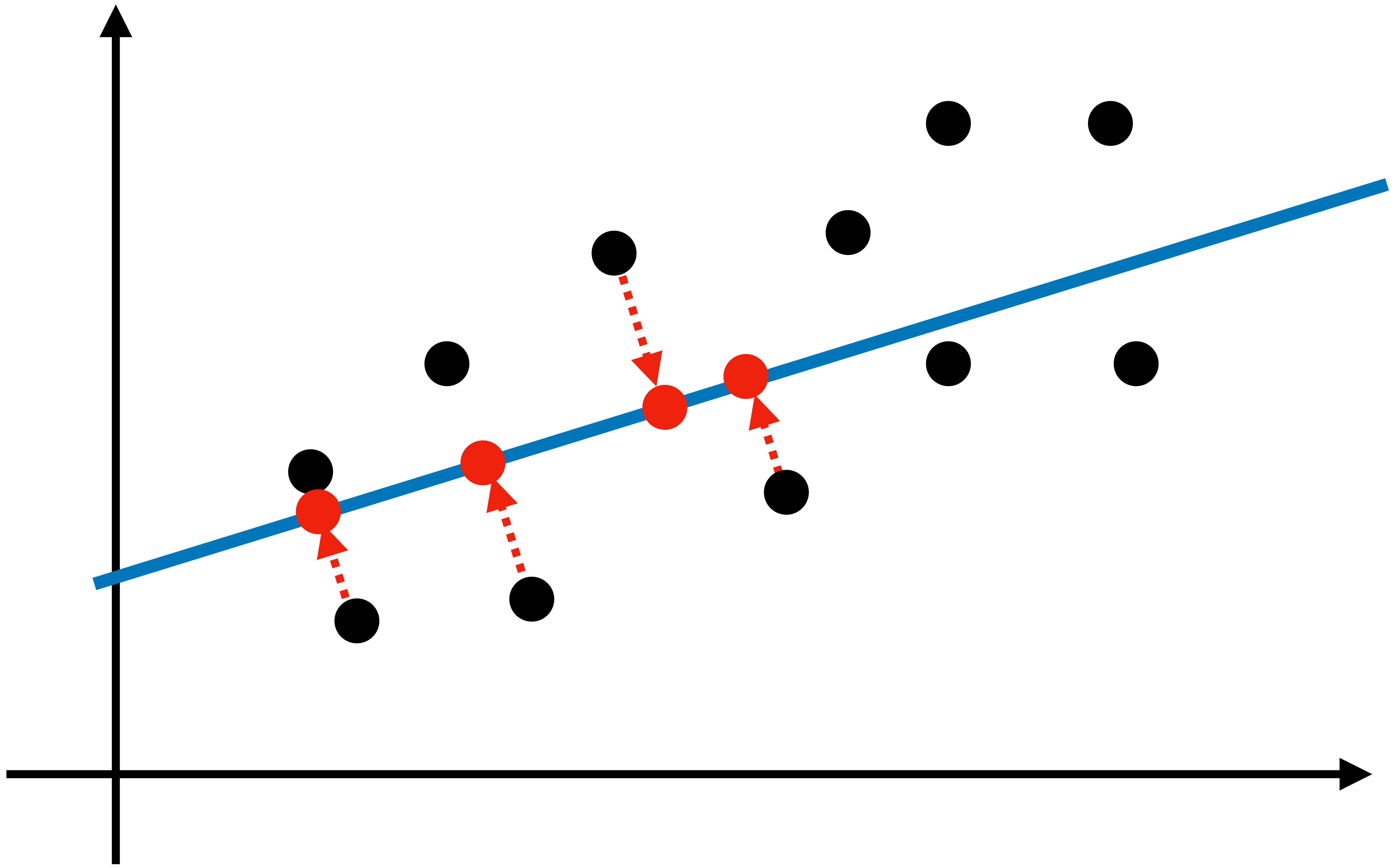
Motivating PCA: Toy Example

- Suppose that we are given a 2D dataset
- **Goal.** Find a nice **1d subspace** and the corresponding **mappings**, such that the mapped data have **desirable properties**



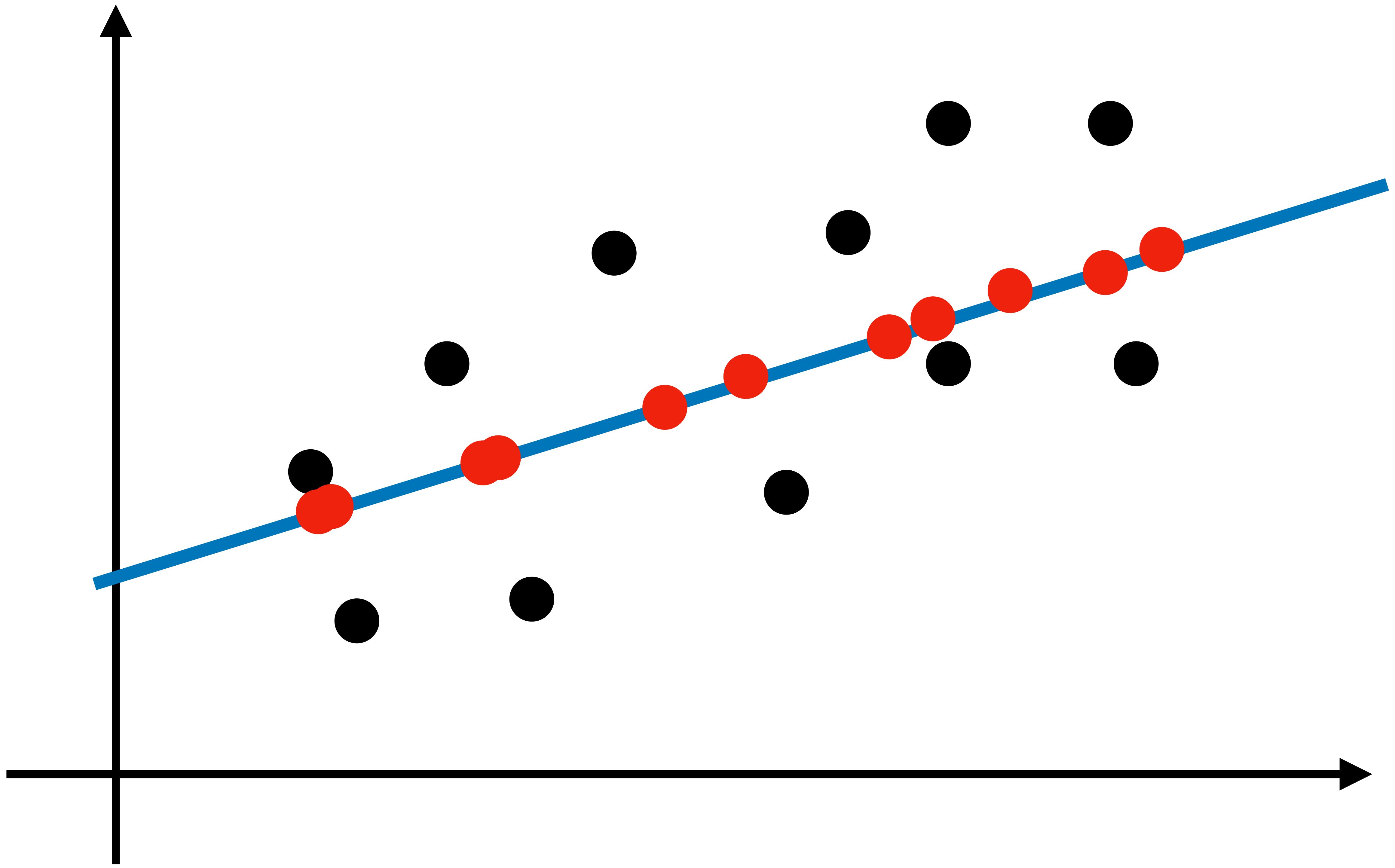
Motivating PCA: Toy Example

- Let's simplify a bit
  - We confine the mapping to be an orthogonal projection
  - Given a subspace, the mapping is uniquely determined.



Motivating PCA: Toy Example

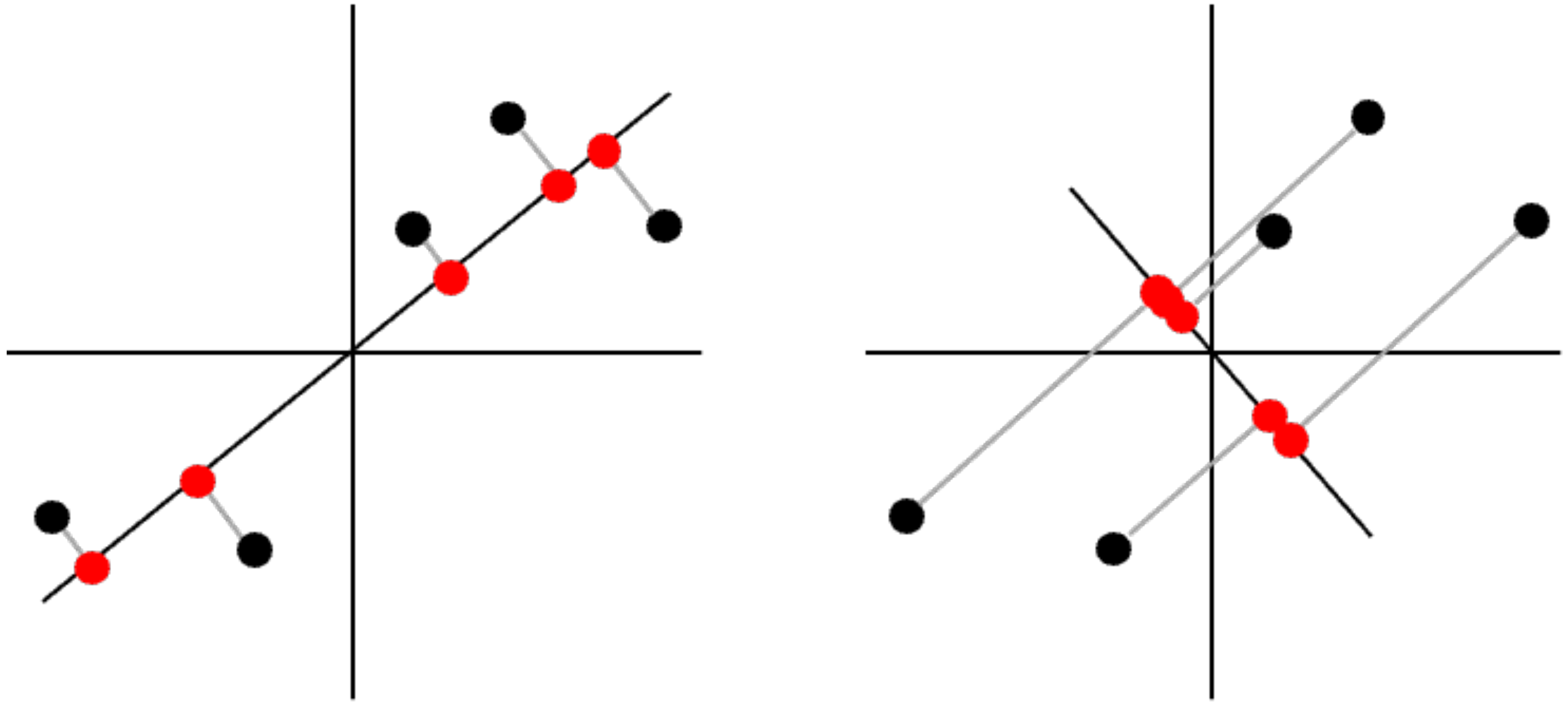
- **Goal (restated).** Find a nice 1D subspace such that the projected data have desirable properties
  - Exactly what properties do we need?



Motivating PCA: Toy Example

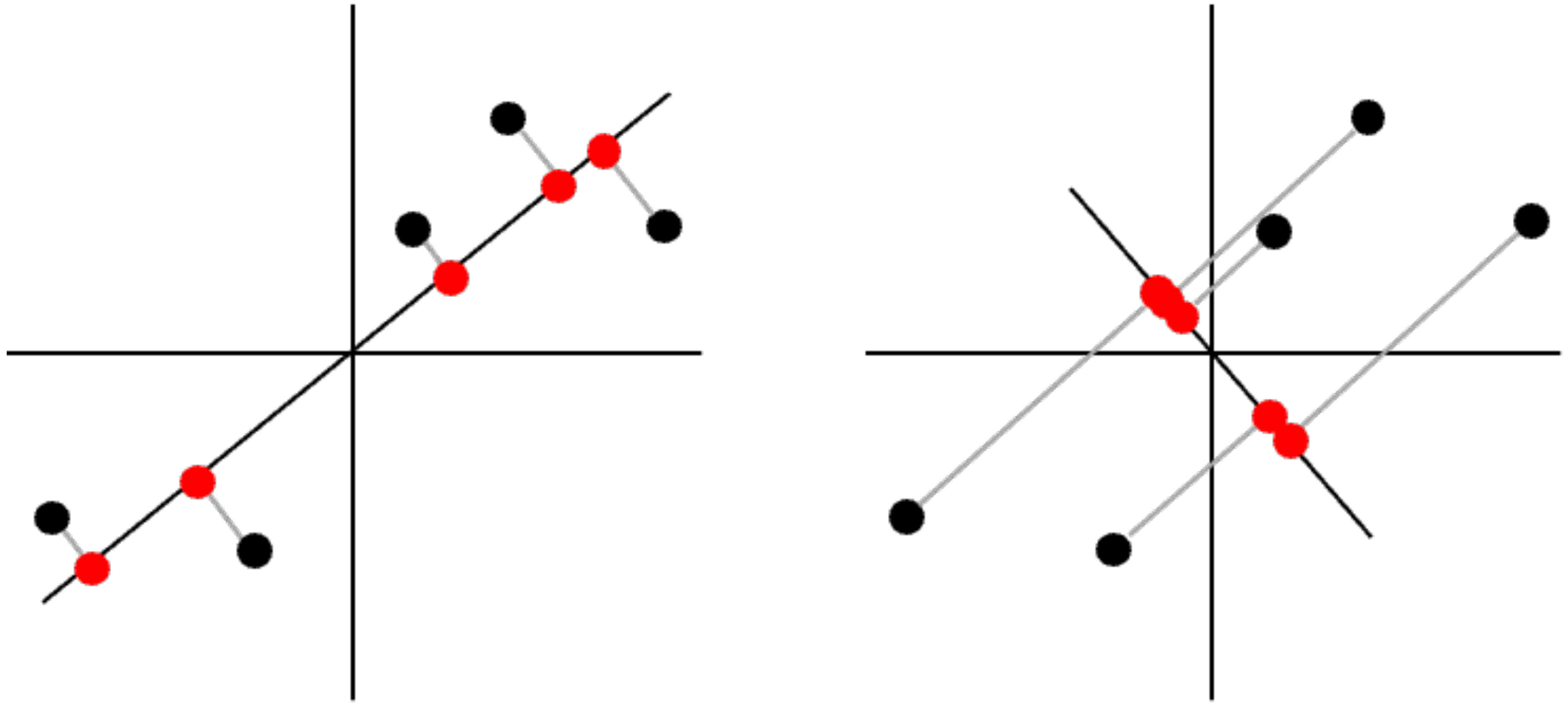
- **Answer.** Preserve task-relevant information as much as possible
  - However, this is a difficult task
    - task-relevance: no label given to us!
    - information: usual metrics, e.g., entropy is hard to estimate

  - **Simpler approach.** Which projection is more informative?



Motivating PCA: Toy Example

- **Answer.** Left is considered informative, for two reasons
  - (A) Projected points are more **well-spread**
    - Does not ignore differences b/w points
    - Noise-robust
  - (B) Projected points (●) are **closer** to their original data (●)
    - That is, more accurate reconstruction is possible



Motivating PCA: Toy Example

- **Answer.** Left is considered informative, for two reasons
  - (A) Projected points are more **well-spread**
    - Does not ignore differences b/w points
    - Noise-robust
  - (B) Projected points (●) are **closer** to their original data (●)
    - That is, more accurate reconstruction is possible

- Interestingly, these two criteria are equivalent!

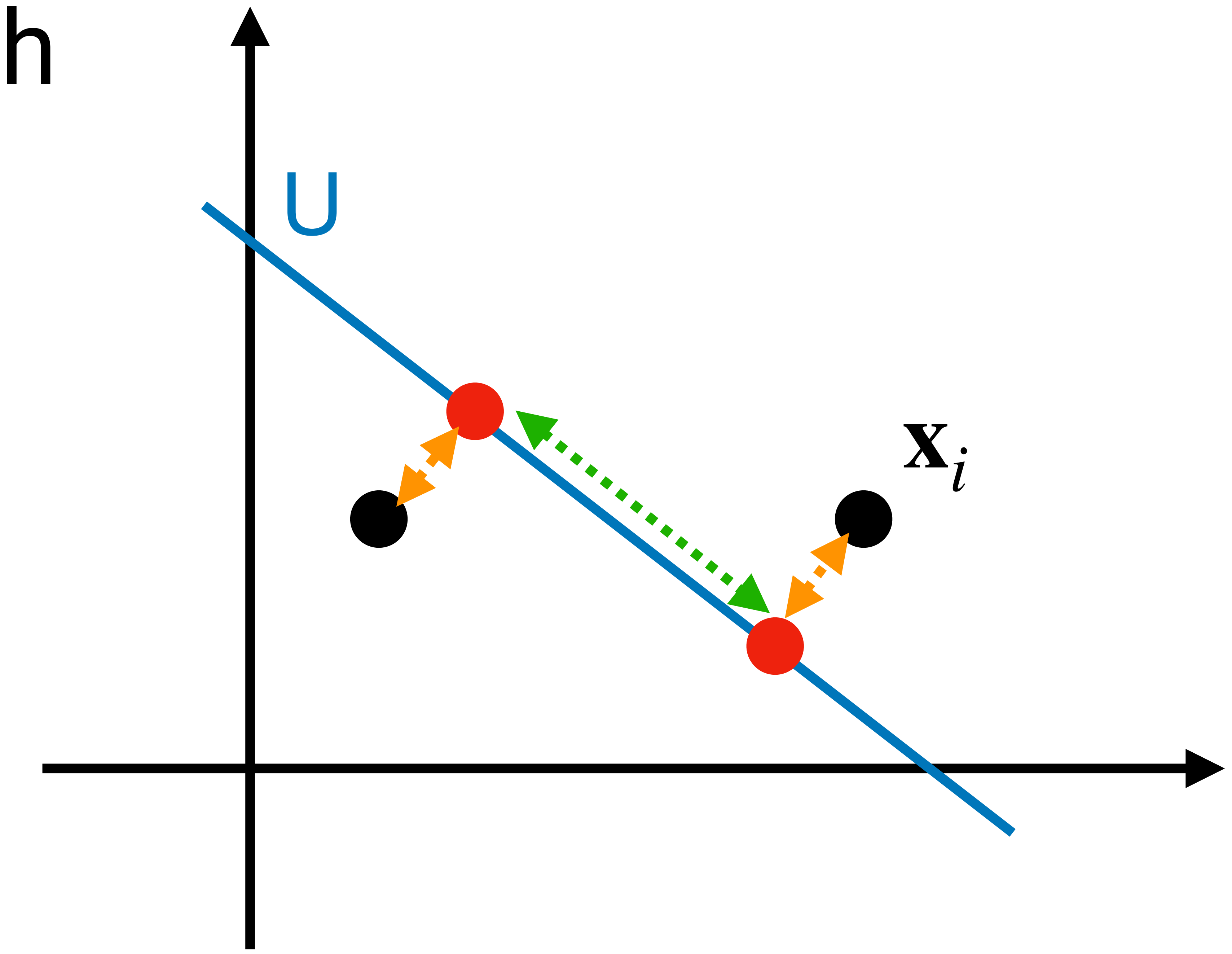
Key Result

- We are given a dataset $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$
- **Goal.** Find a k-dimensional subset $\mathsf{U} \subseteq \mathbb{R}^d$ with
  - (A) Maximum **variance** of projected points

$$\max_{\mathsf{U}} \mathrm{Var}(\pi_{\mathsf{U}}(\mathrm{x}_1), \ldots, \pi_{\mathsf{U}}(\mathbf{x}_n))$$

  - (B) Minimum ℓ^2 **distortion** from projection

$$\min_{\mathsf{U}} \sum_{i=1}^{n} \|\mathbf{x}_i - \pi_{\mathsf{U}}(\mathbf{x}_i)\|_2^2$$



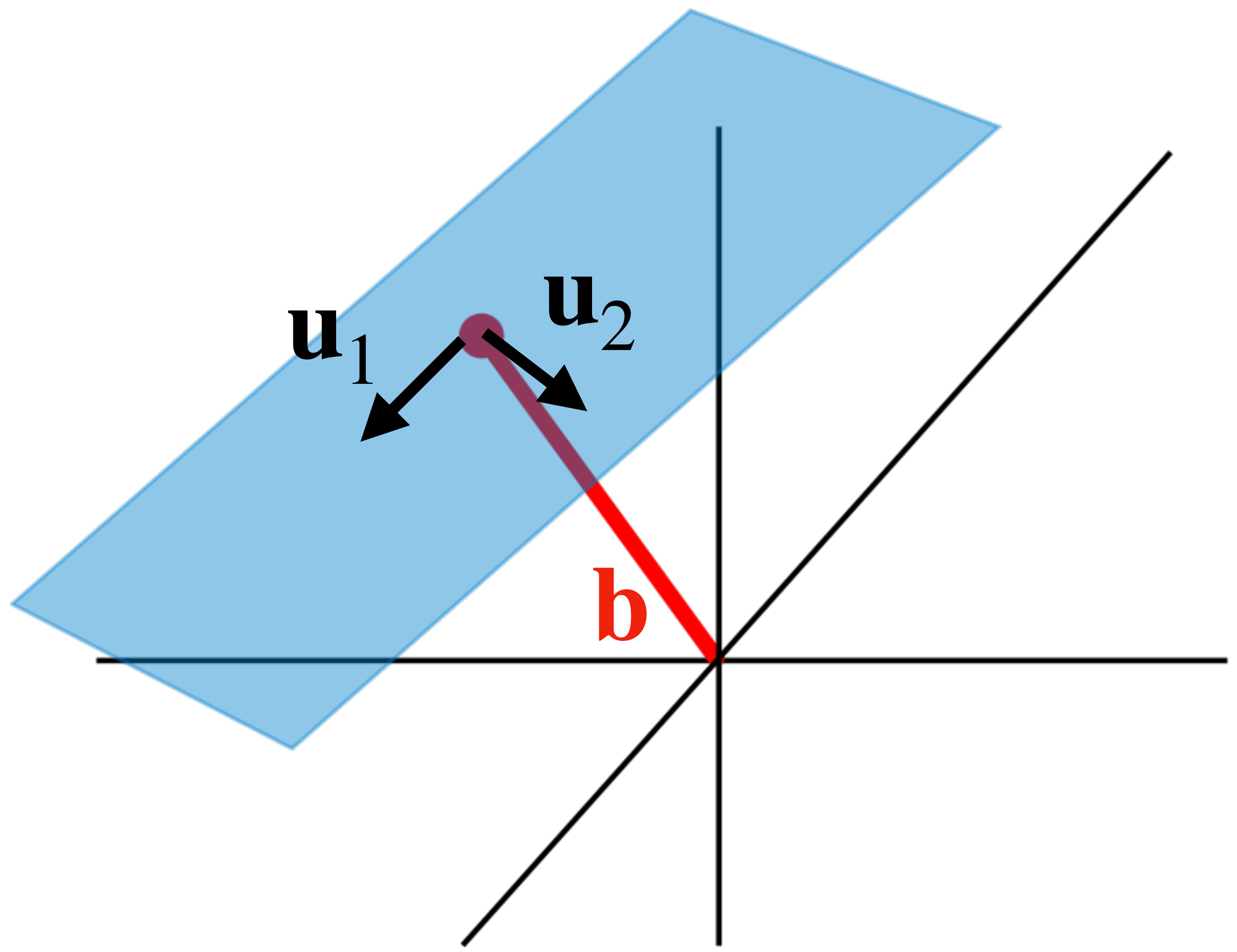
- But first, let's formally define what "projection" is…

Formalisms: Projection

Formalisms

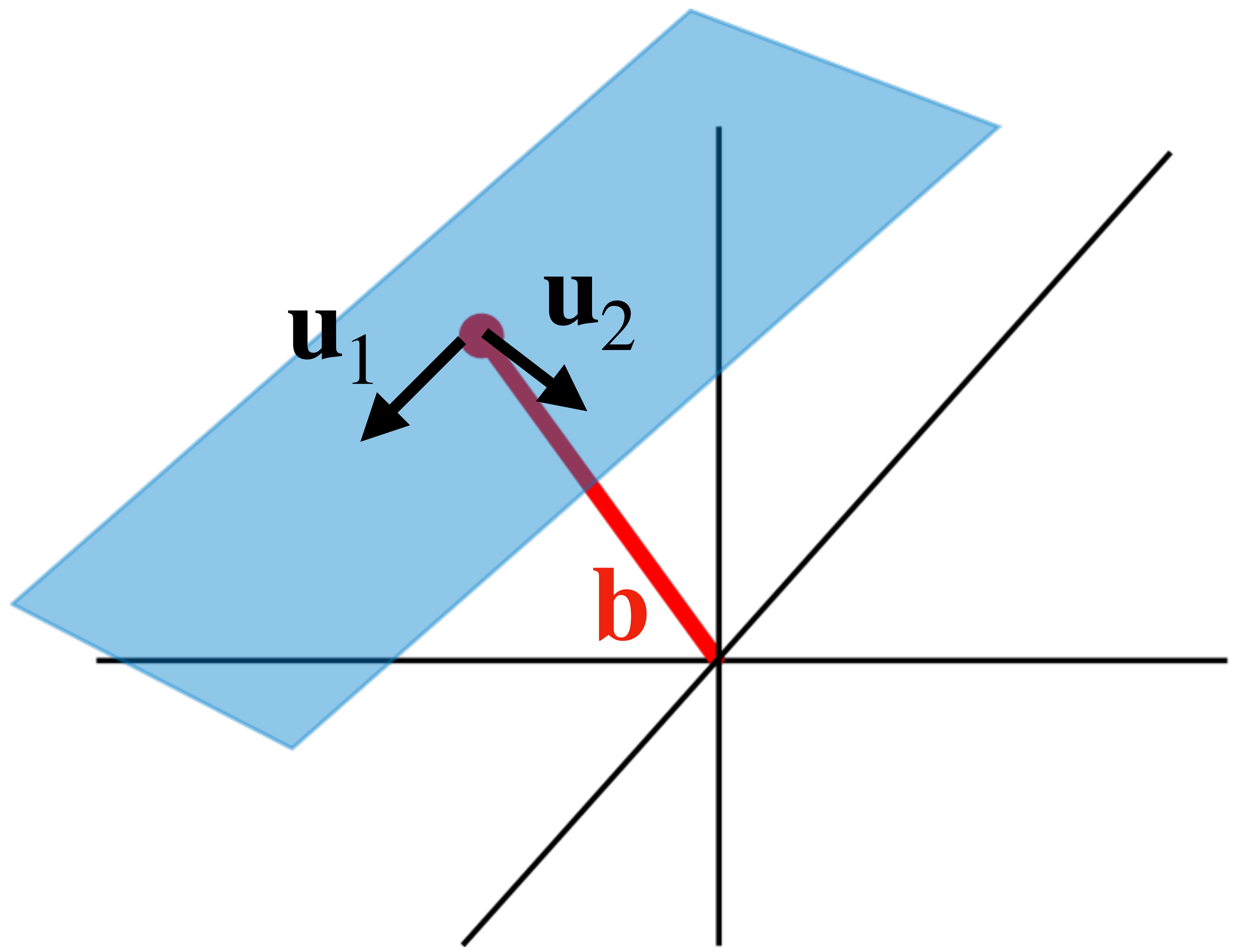
- A k-dimensional affine subspace $\mathsf{U} \subset \mathbb{R}^d$ can be characterized by:
  - Orthonormal basis $\mathbf{u}_1, \ldots, \mathbf{u}_k \in \mathbb{R}^d$
  - Orthogonal bias $\mathbf{b} \in \mathbb{R}^d$

$$\mathsf{U} = \{a_1\mathbf{u}_1 + \cdots + a_k\mathbf{u}_k + \mathbf{b} \;:\; a_i \in \mathbb{R}\}$$



Formalisms

- Any element on U can be represented in two ways:
  - A d-dimensional vector $\mathbf{u} \in$ U
  - A k-dimensional vector $\mathbf{a} = (a_1, a_2, \ldots, a_k)$
    - where $\mathbf{u} = a_1\mathbf{u}_1 + \cdots + a_k\mathbf{u}_k + \mathbf{b}$ holds



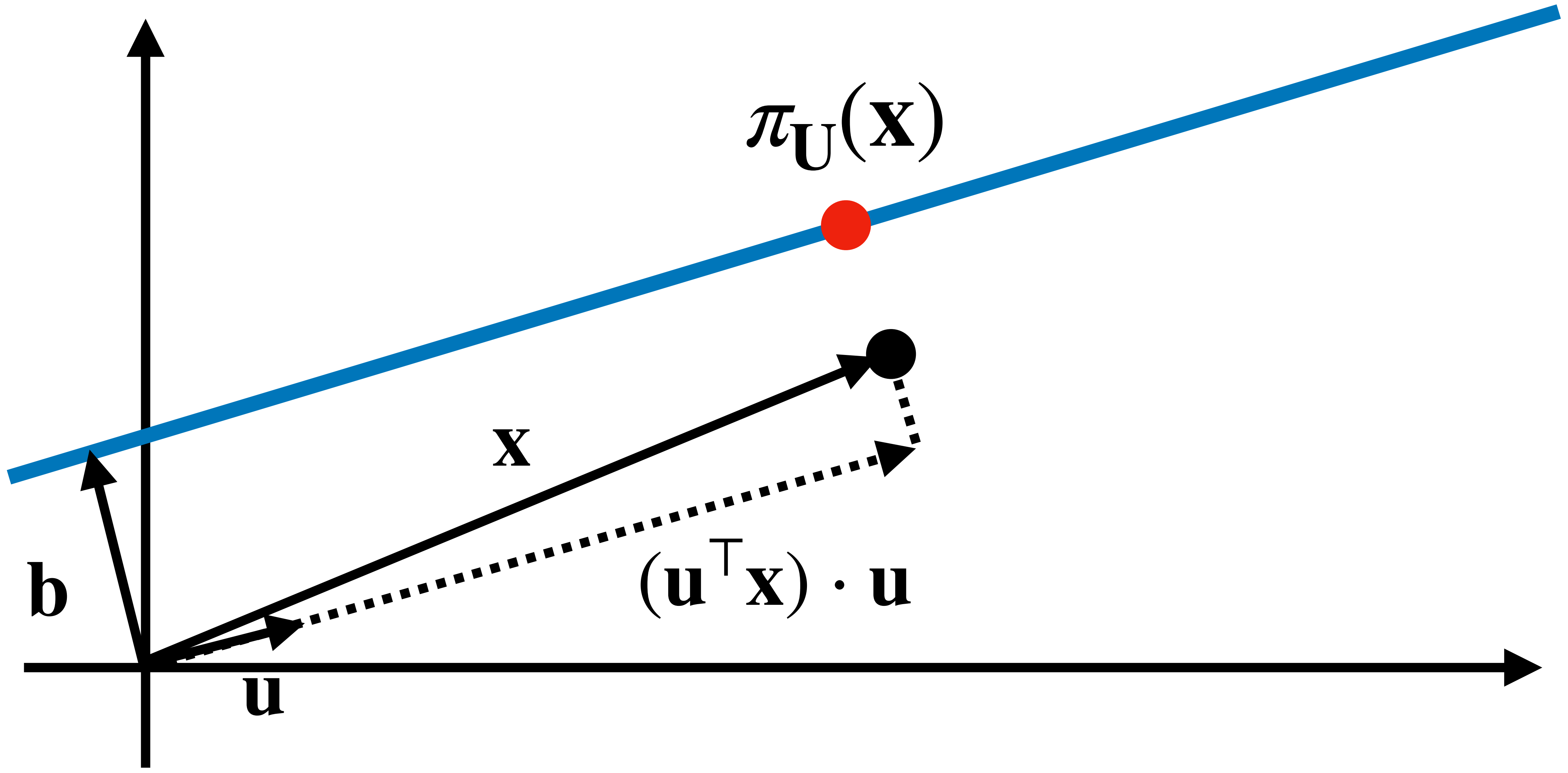
Formalisms

- A **projection** of a vector $\mathbf{x} \in \mathbb{R}^d$ to the affine subspace U is:

$$\pi_{\mathsf{U}}(\mathbf{x}) = \sum_{i=1}^{k} (\mathbf{u}_i^\top \mathbf{x}) \cdot \mathbf{u}_i + \mathbf{b}$$

  - This is a d-dimensional quantity, with an alternative representation:

$$\mathbf{a} = (\mathbf{u}_1^\top \mathbf{x}, \ldots, \mathbf{u}_k^\top \mathbf{x}) \in \mathbb{R}^k$$



Formalisms

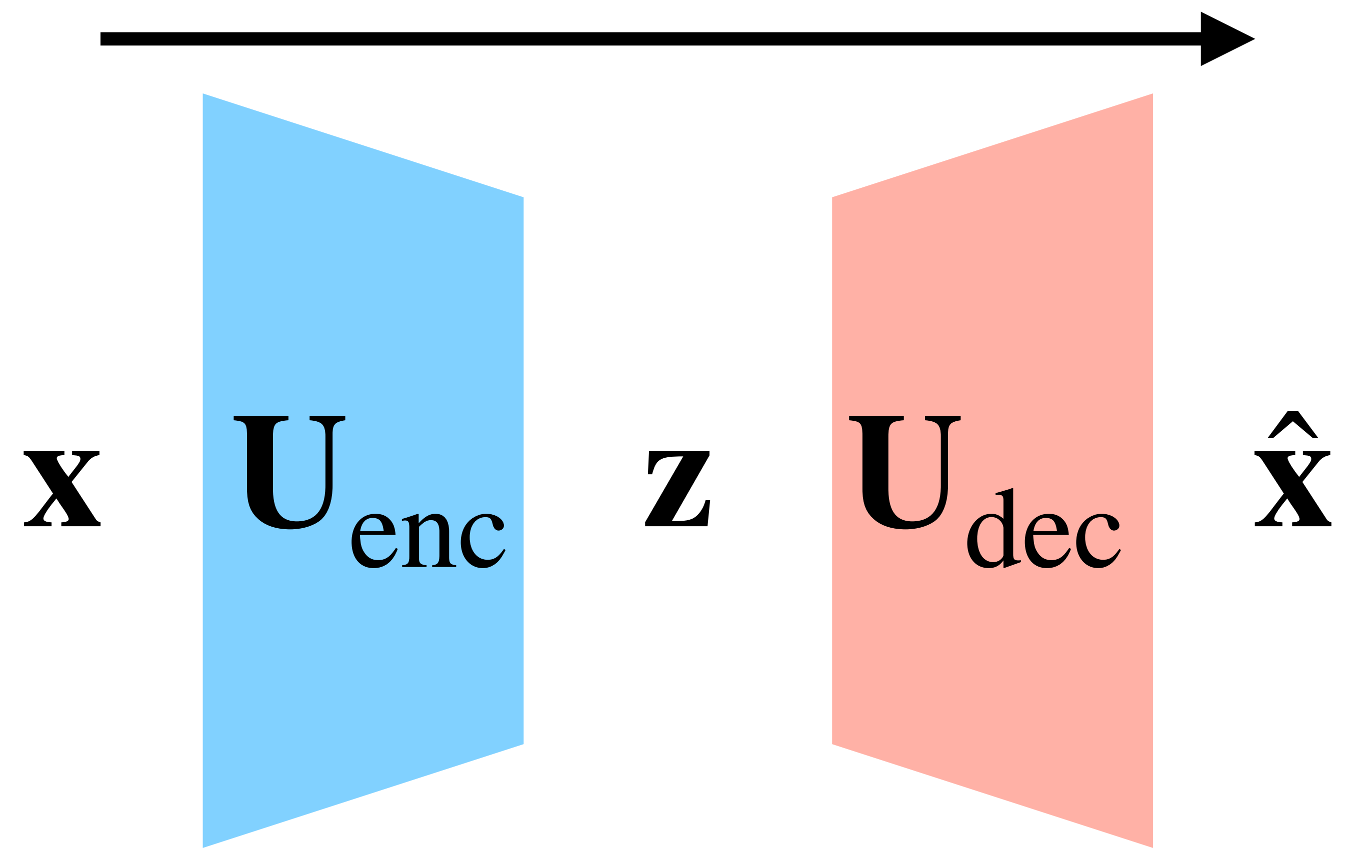
- The projection admits a matrix form:

$$\pi_{\mathsf{U}}(\mathbf{x}) = \left(\sum_{i=1}^{k} \mathbf{u}_i \mathbf{u}_i^\top \right) \mathbf{x} + \mathbf{b}$$

$$=: \mathbf{U}\mathbf{x} + \mathbf{b}$$

- Here, the projection matrix $\mathbf{U}$ is:
  - $d \times d$ matrix with rank k
  - $\mathbf{U}^\top = \mathbf{U}$
  - $\mathbf{U}^\top \mathbf{U} = \mathbf{U}$
- Conversely, called projection matrix if these are satisfied

Formalisms



- In a sense, projection consists of two operations
- **Compression** $\mathbb{R}^d \to \mathbb{R}^k$
  - Also known as "encoding"

$$\mathbf{z} = \mathbf{U}_{\text{enc}}\mathbf{x}, \qquad \text{where} \quad \mathbf{U}_{\text{enc}} = \begin{bmatrix} \leftarrow & \mathbf{u}_1^\top & \rightarrow \\ & \cdots & \\ \leftarrow & \mathbf{u}_k^\top & \rightarrow \end{bmatrix} \in \mathbb{R}^{k \times d}$$

- **Reconstruction** $\mathbb{R}^k \to \mathbb{R}^d$
  - Also known as "decoding"

$$\hat{\mathbf{x}} = \mathbf{U}_{\text{dec}}\mathbf{z} + \mathbf{b}, \qquad \text{where} \quad \mathbf{U}_{\text{dec}} = \mathbf{U}_{\text{enc}}^\top \in \mathbb{R}^{d \times k}$$

PCA: Variance Maximization

Variance Maximization

- In PCA, we want to find a nice $(\mathbf{U}, \mathbf{b})$ which solves

$$\max_{\mathbf{U},\mathbf{b}} \ \mathrm{Var}\left(\mathbf{U}\mathbf{x}_1 + \mathbf{b}, \ldots, \mathbf{U}\mathbf{x}_n + \mathbf{b}\right)$$

- As the constant term does not affect the variance, this is equivalent to

$$\max_{\mathbf{U}} \ \mathrm{Var}\left(\mathbf{U}\mathbf{x}_1, \ldots, \mathbf{U}\mathbf{x}_n\right)$$

Variance Maximization

- Define $\bar{\mathbf{x}}$ as the mean of $\{\mathbf{x}_i\}_{i=1}^n$

- Then, the variance can be written as:

$$\begin{aligned}
\mathrm{Var}\big(\mathbf{U}\mathbf{x}_1, \ldots, \mathbf{U}\mathbf{x}_n\big) &= \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{U}(\mathbf{x}_i - \bar{\mathbf{x}})\|_2^2 \\
&= \frac{1}{n}\sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})^\top \mathbf{U}^\top \mathbf{U}(\mathbf{x}_i - \bar{\mathbf{x}}) \\
&= \frac{1}{n}\sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})^\top \mathbf{U}(\mathbf{x}_i - \bar{\mathbf{x}})
\end{aligned}$$

Variance Maximization

$$\max_{\mathbf{U}} \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})^\top \mathbf{U} (\mathbf{x}_i - \bar{\mathbf{x}})$$

- By the definition of $\mathbf{U}$, we can re-write the above as

$$\max_{\mathbf{U}} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} (\mathbf{x}_i - \bar{\mathbf{x}})^\top \mathbf{u}_j \mathbf{u}_j^\top (\mathbf{x}_i - \bar{\mathbf{x}})$$

$$= \max_{\mathbf{U}} \sum_{j=1}^{k} \mathbf{u}_j^\top \left(\frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top \right) \mathbf{u}_j$$

= sample covariance matrix $\mathbf{S}$
(positive-semidefinite)

Variance Maximization

- Thus, PCA is about solving the **constrained quadratic optimization**

$$\max_{\mathbf{u}_1,\ldots,\mathbf{u}_k} \sum_{j=1}^{k} \mathbf{u}_j^\top \mathbf{S} \mathbf{u}_j, \qquad \text{subject to} \quad \mathbf{u}_i^\top \mathbf{u}_j = \begin{cases} 1 & \cdots & i = j \\ 0 & \cdots & i \neq j \end{cases}$$

- **Question.** How do we solve this?

Solving the quadratic problem

$$\max_{\mathbf{u}_1,\ldots,\mathbf{u}_k} \sum_{j=1}^{k} \mathbf{u}_j^\top \mathbf{S} \mathbf{u}_j, \qquad \text{subject to} \qquad \mathbf{u}_i^\top \mathbf{u}_j = \mathbf{1}\{i = j\}$$

- **Answer.** Of course, the method of Lagrangian multipliers
  - Standard derivation requires complicated matrix derivatives — instead, will give you a simplified proof idea.

- **Strategy.** Conduct a greedy optimization
  - Select a nice $\mathbf{u}_1$ that maximizes $\mathbf{u}_1^\top \mathbf{S} \mathbf{u}_1$ s.t. $\mathbf{u}_1^\top \mathbf{u}_1 = 1$
  - Select a nice $\mathbf{u}_2$ that maximizes $\mathbf{u}_2^\top \mathbf{S} \mathbf{u}_2$ s.t. $\mathbf{u}_2^\top \mathbf{u}_2 = 1, \mathbf{u}_2^\top \mathbf{u}_1 = 0$
  - …

Solving the quadratic problem

- First step is to determine $\mathbf{u}_1$

$$\max_{\mathbf{u}} \mathbf{u}^\top \mathbf{S}\mathbf{u}, \qquad \text{subject to} \quad \mathbf{u}^\top \mathbf{u} = 1$$

- To solve this, consider the Lagrangian relaxation

$$\max_{\mathbf{u}} \mathbf{u}^\top \mathbf{S}\mathbf{u} + \alpha(1 - \mathbf{u}^\top \mathbf{u})$$

  - Critical point is where $\mathbf{S}\mathbf{u} = \alpha\mathbf{u}$ holds
    - i.e., eigenvectors
  - Choose the principal component — i.e., eigenvector w/ maximum eigenvalue — to maximize the value of $\mathbf{u}^\top \mathbf{S}\mathbf{u}$

Solving the quadratic problem

- Next, we determine $\mathbf{u}_2$

$$\max_{\mathbf{u}} \mathbf{u}^\top \mathbf{S}\mathbf{u}, \qquad \text{subject to} \quad \mathbf{u}^\top \mathbf{u} = 1, \mathbf{u}^\top \mathbf{u}_1 = 0$$

- Lagrangian relaxation becomes

$$\mathbf{u}^\top \mathbf{S}\mathbf{u} + \alpha(1 - \mathbf{u}^\top \mathbf{u}) - \beta(\mathbf{u}^\top \mathbf{u}_1)$$

- The critical point condition is:

$$\mathbf{S}\mathbf{u} = \alpha \mathbf{u} + \frac{\beta}{2}\mathbf{u}_1$$

Solving the quadratic problem

$$\mathbf{S}\mathbf{u} = \alpha\mathbf{u} + \frac{\beta}{2}\mathbf{u}_1$$

- Multiplying $\mathbf{u}_1^\top$ on both sides, we get:

$$0 = 0 + \frac{\beta}{2}$$

  - Thus, we have $\beta = 0$
- Then, the Lagrangian becomes

$$\mathbf{u}^\top\mathbf{S}\mathbf{u} + \alpha(1 - \mathbf{u}^\top\mathbf{u})$$

  - Thus the things are the same as in the derivation of $\mathbf{u}_1$
    - Thus, choose the eigenvector for 2nd largest eigenvalue

Solving the quadratic problem

- Repeat this, the solution is to let $\mathbf{u}_1, \ldots, \mathbf{u}_k$ be the top-k principal components of our sample covariance matrix

- This can be done by performing SVD on the **data matrix**

$$\mathbf{X} = [\mathbf{x}_1 - \bar{\mathbf{x}} \mid \cdots \mid \mathbf{x}_n - \bar{\mathbf{x}}] = \mathbf{U}\Sigma\mathbf{V}^\top$$

  and then selecting the columns of $\mathbf{U}$ corresponding to top-k singular values

- **Note.** Did not cover determining $\mathbf{b}$ — will be covered soone

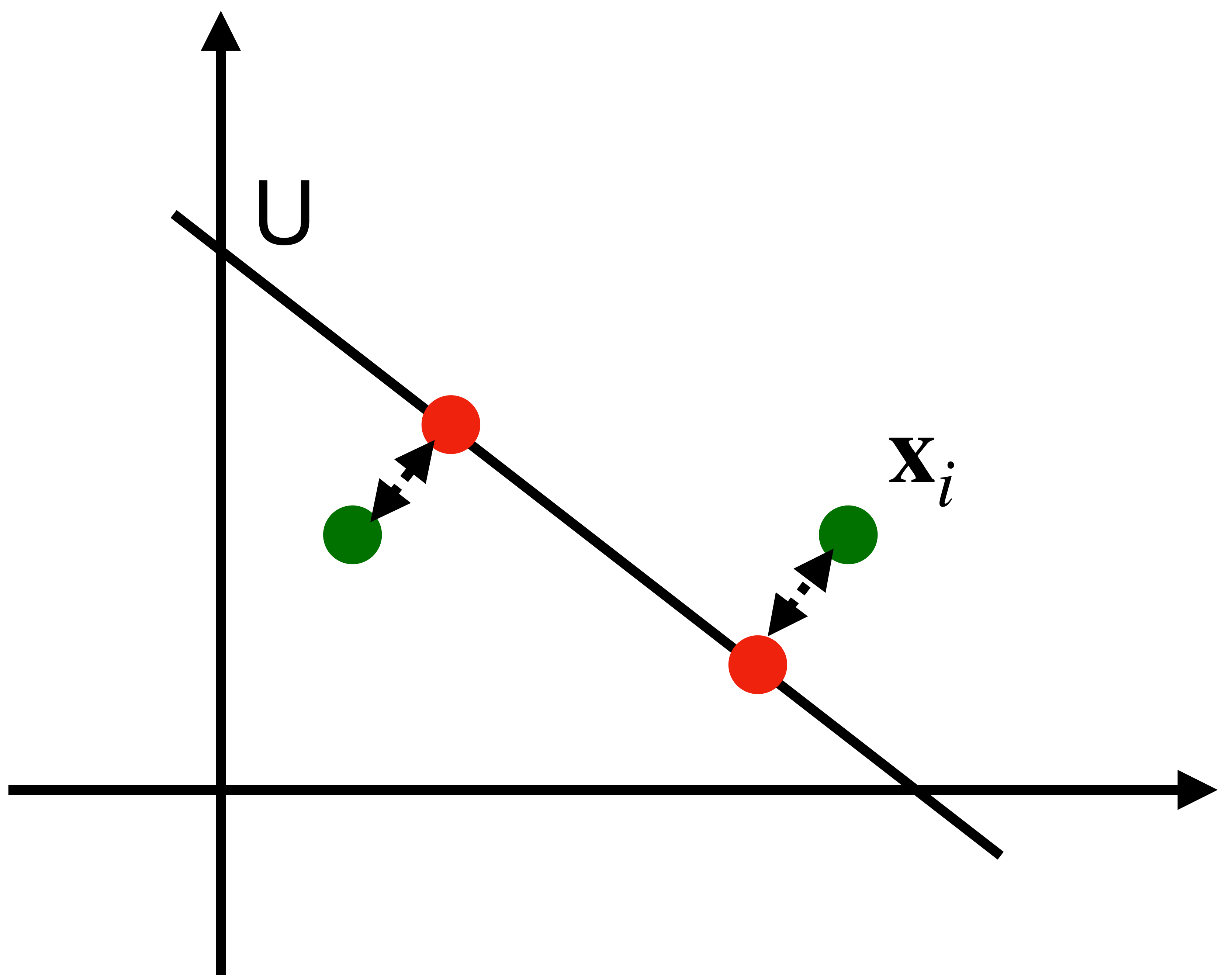
PCA: Distortion Minimization

Distortion Minimization

- Here is the spirit:

"If the projected point is close to the original point, then we did not lose too much information"

- We'll show that this distortion minimization = variance maximization



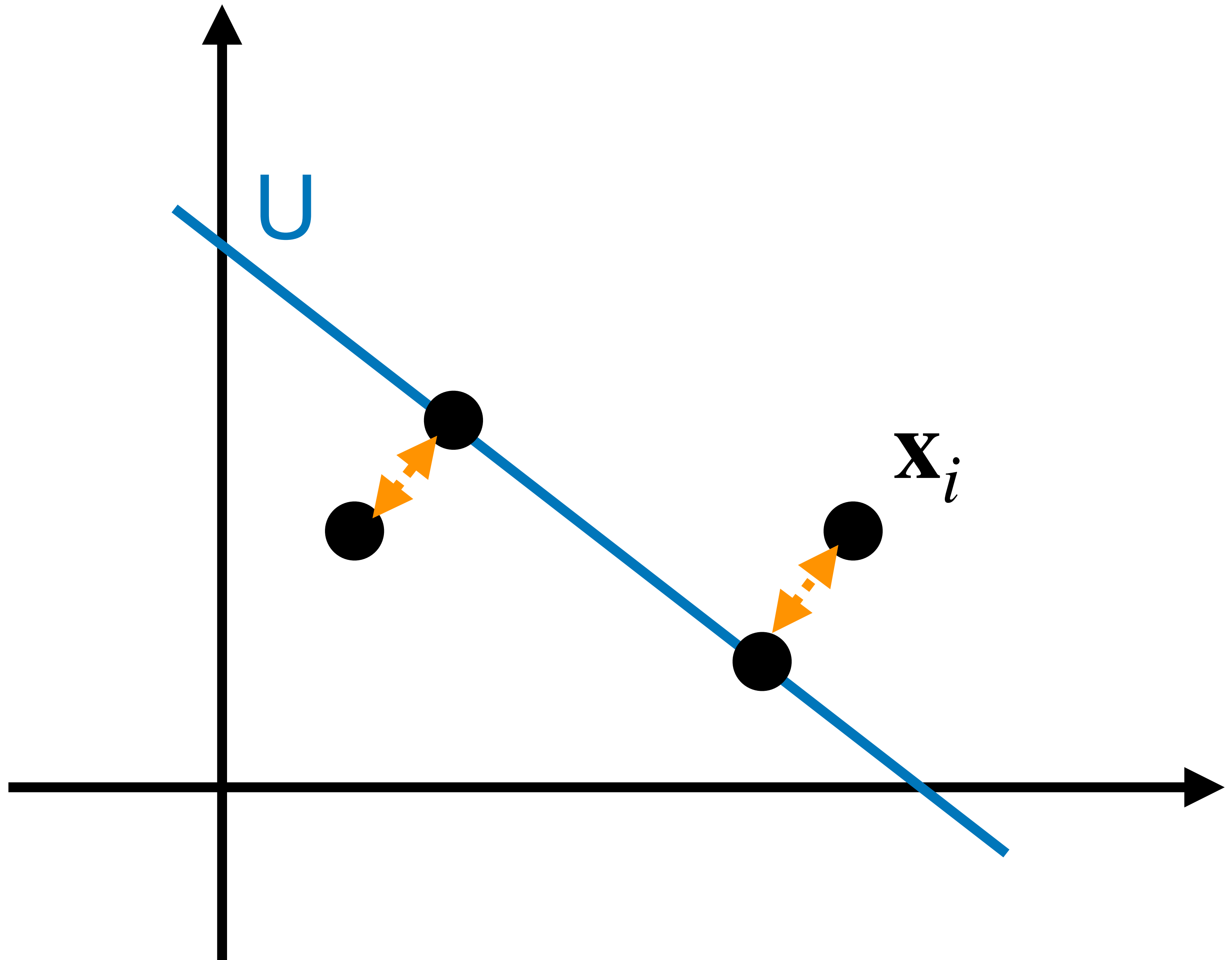
Distortion Minimization

- Formally, we try to find an affine subspace

$$\mathsf{U} = \{a_1\mathbf{u}_1 + \cdots + a_k\mathbf{u}_k + \mathbf{b} \ : \ a_i \in \mathbb{R}\}$$

such that the mean squared error of data from projection is minimized

$$\min_{\mathsf{U}} \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}_i - \pi_{\mathsf{U}}(\mathbf{x}_i)\|^2$$



Distortion Minimization

- Using the definition of projection, we know that

$$\frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}_i - \pi_{\mathsf{U}}(\mathbf{x}_i)\|^2$$

$$= \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{U}\mathbf{x}_i - \mathbf{b}\|^2$$

$$= \frac{1}{n}\sum_{i=1}^{n} \left(\|\mathbf{x}_i\|^2 + \|\mathbf{b}\|^2 - \mathbf{x}_i^\top \mathbf{U}\mathbf{x}_i - 2\mathbf{b}^\top \mathbf{x}_i + 2\mathbf{b}^\top \mathbf{U}\mathbf{x}_i\right)$$

$$= \frac{1}{n}\left(\sum_{i=1}^{n} \|\mathbf{x}_i\|^2\right) + \|\mathbf{b}\|^2 - \frac{1}{n}\left(\sum_{i=1}^{n} \mathbf{x}_i^\top \mathbf{U}\mathbf{x}_i\right) - 2\mathbf{b}^\top \bar{\mathbf{x}} + 2\mathbf{b}^\top \mathbf{U}\bar{\mathbf{x}}$$

Distortion Minimization

- Removing the irrelevant terms, we are solving:

$$\min_{\mathbf{U},\mathbf{b}} \left(\|\mathbf{b}\|^2 - \frac{1}{n} \sum \mathbf{x}_i^\top \mathbf{U} \mathbf{x}_i - 2\mathbf{b}^\top \bar{\mathbf{x}} + 2\mathbf{b}^\top \mathbf{U} \bar{\mathbf{x}} \right)$$

  - For any fixed $\mathbf{U}$, we have

$$\mathbf{b}^* = \bar{\mathbf{x}} - \mathbf{U}\bar{\mathbf{x}}$$

- Plugging in and removing constant terms again, we get:

$$\min_{\mathbf{U}} \left(\bar{\mathbf{x}}^\top \mathbf{U} \bar{\mathbf{x}} - \frac{1}{n} \sum \mathbf{x}_i^\top \mathbf{U} \mathbf{x}_i \right) = -\max_{\mathbf{U}} \left(\sum_{j=1}^{k} \mathbf{u}_j \mathbf{S} \mathbf{u}_j \right)$$

Applications & Limitations

Face Recognition

- Many applications, but here's an interesting one: Eigenface (1991)

- **Goal.** Identify specific person, based on facial image
  - Robust to glass, lightning, …
  - Using 256 x 256 is difficult!

Face Recognition

- **Idea.** Build a PCA database for whole dataset
  - Each $\mathbf{u}_i$ can capture some "feature"
  - Classify based on $(\mathbf{u}_1^\top \mathbf{x}, \ldots, \mathbf{u}_k^\top \mathbf{x})$
    - Rapid recognition
    - Tracking

- **Limitations.**
  - Requires the same size
  - Sensitive to angles
  - Needs "centering"

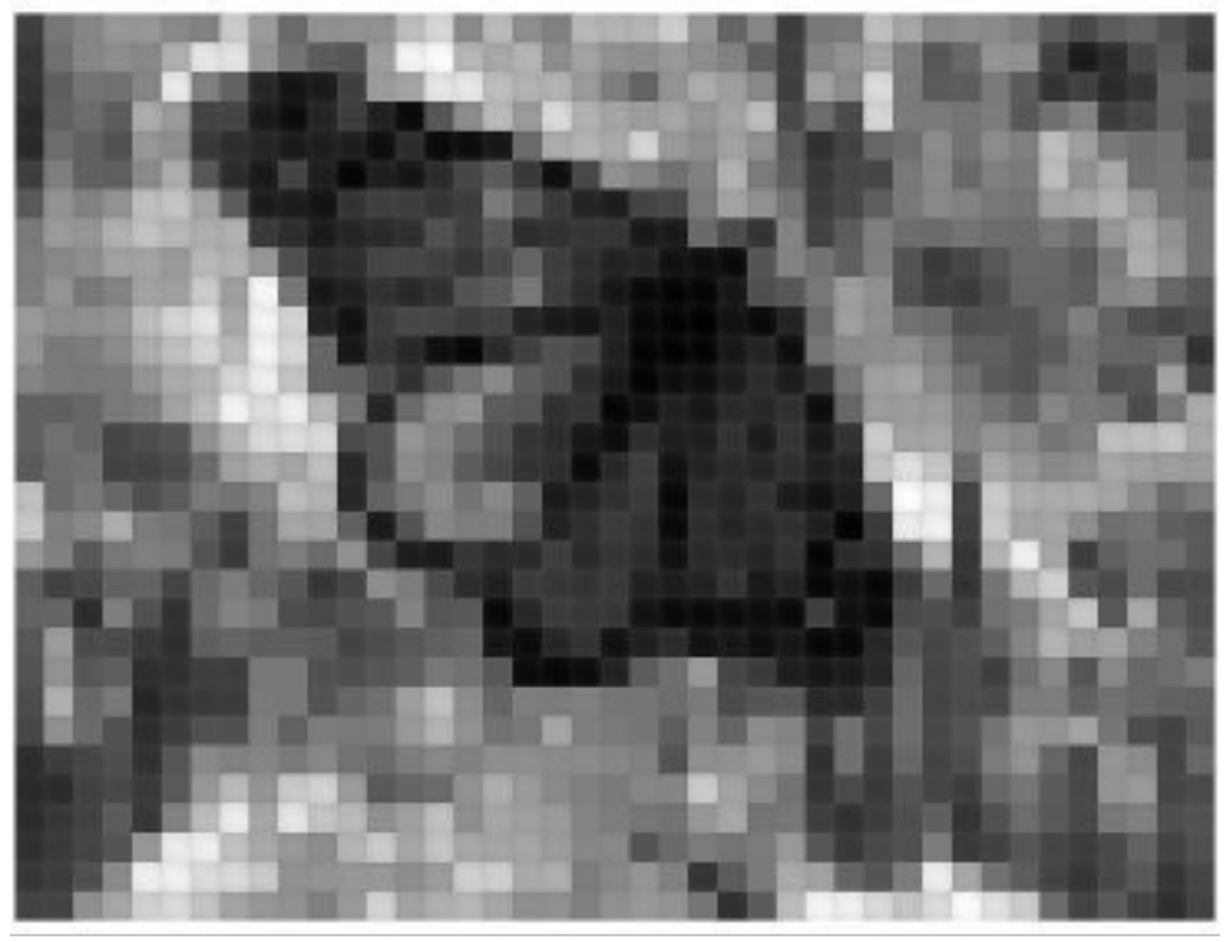
Image Compression

- **Goal.** Represent an image using less dimensions
- **Idea.** Do the following:
  - Divide each image in 12 x 12 patches
  - Conduct PCA
  - For each patch, save K digits $(\mathbf{u}_1^\top \mathbf{x}, \ldots, \mathbf{u}_k^\top \mathbf{x})$

144-dimension
(full)

60-dimension

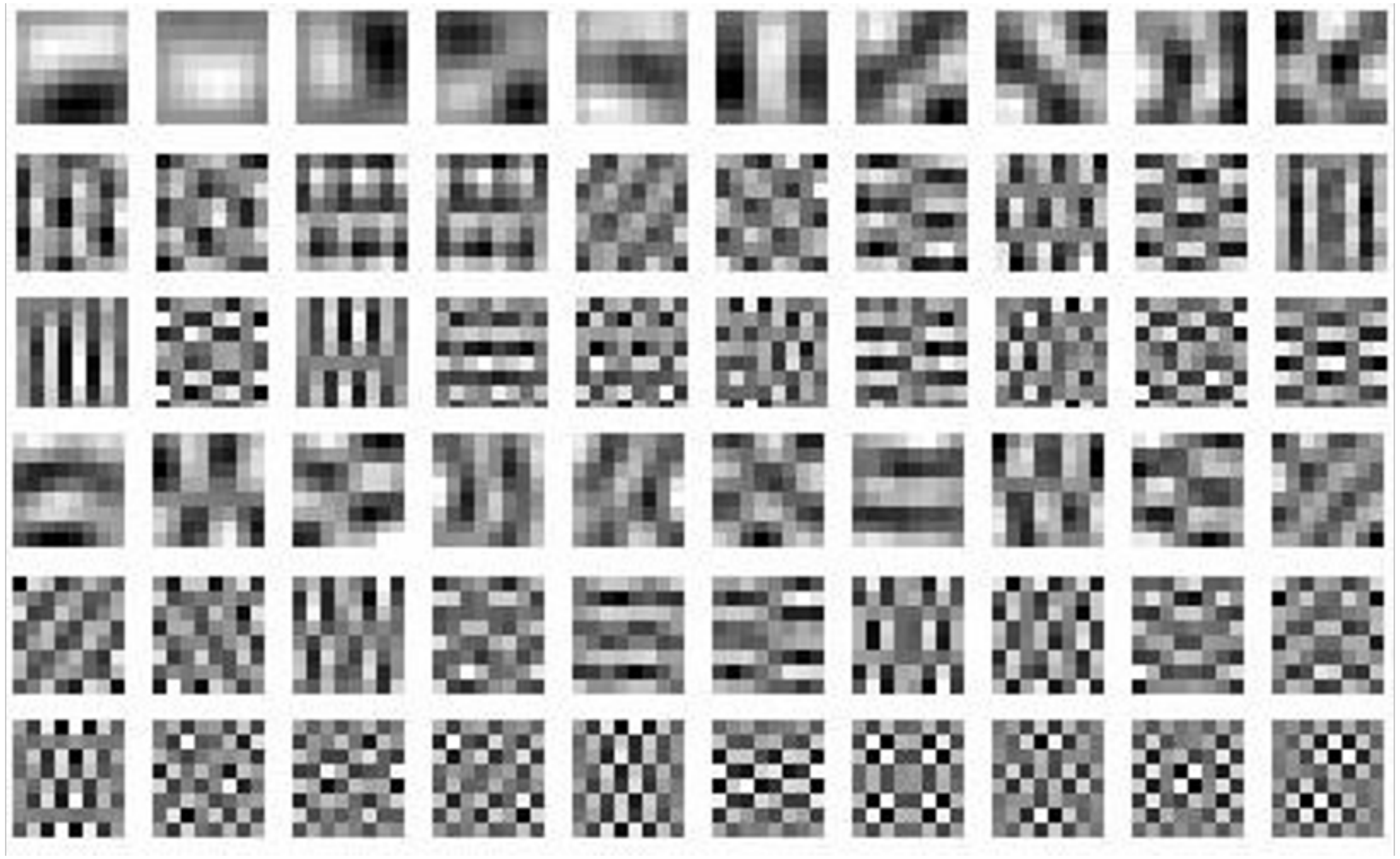
6-dimension

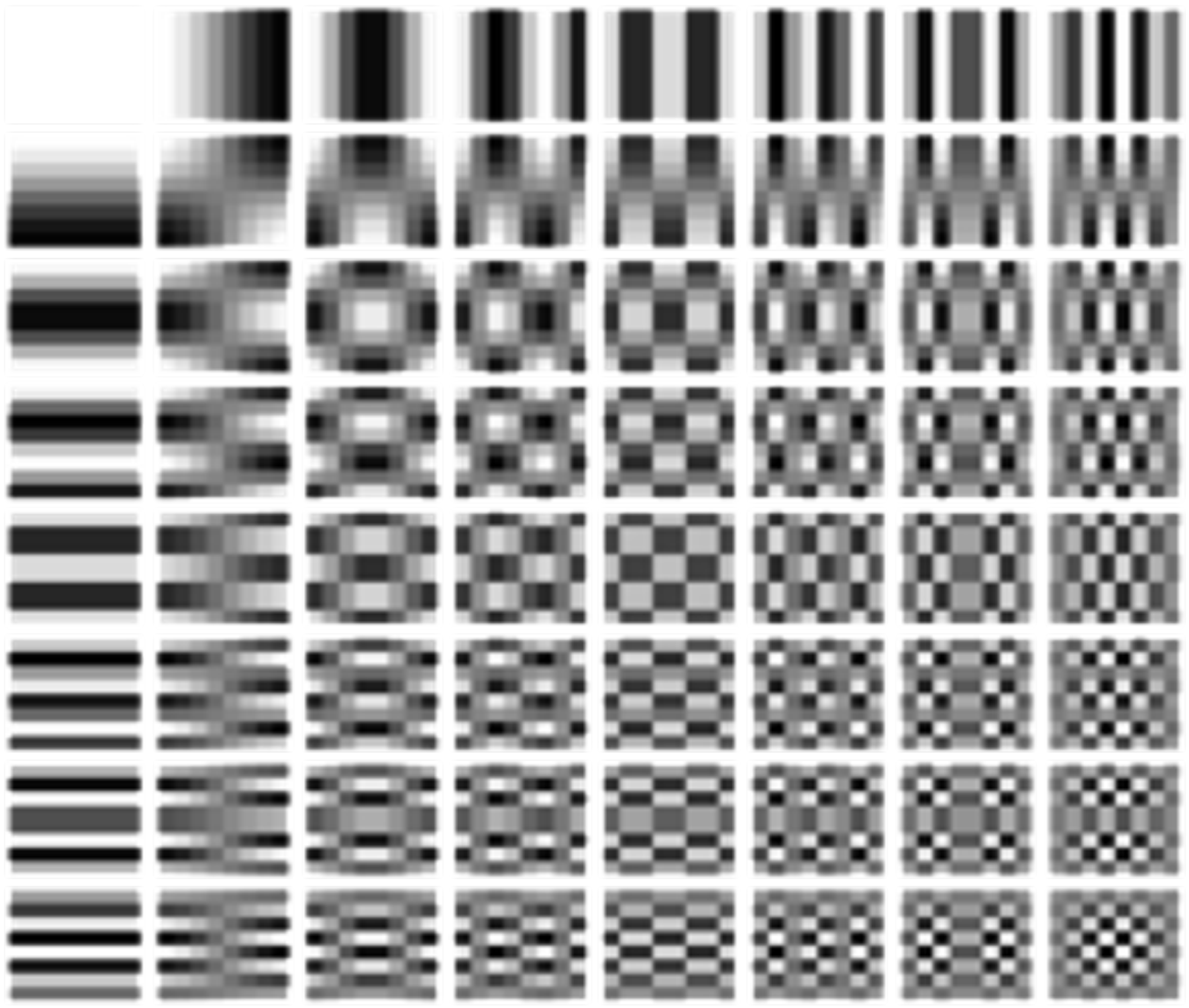


1-dimension

Image Compression

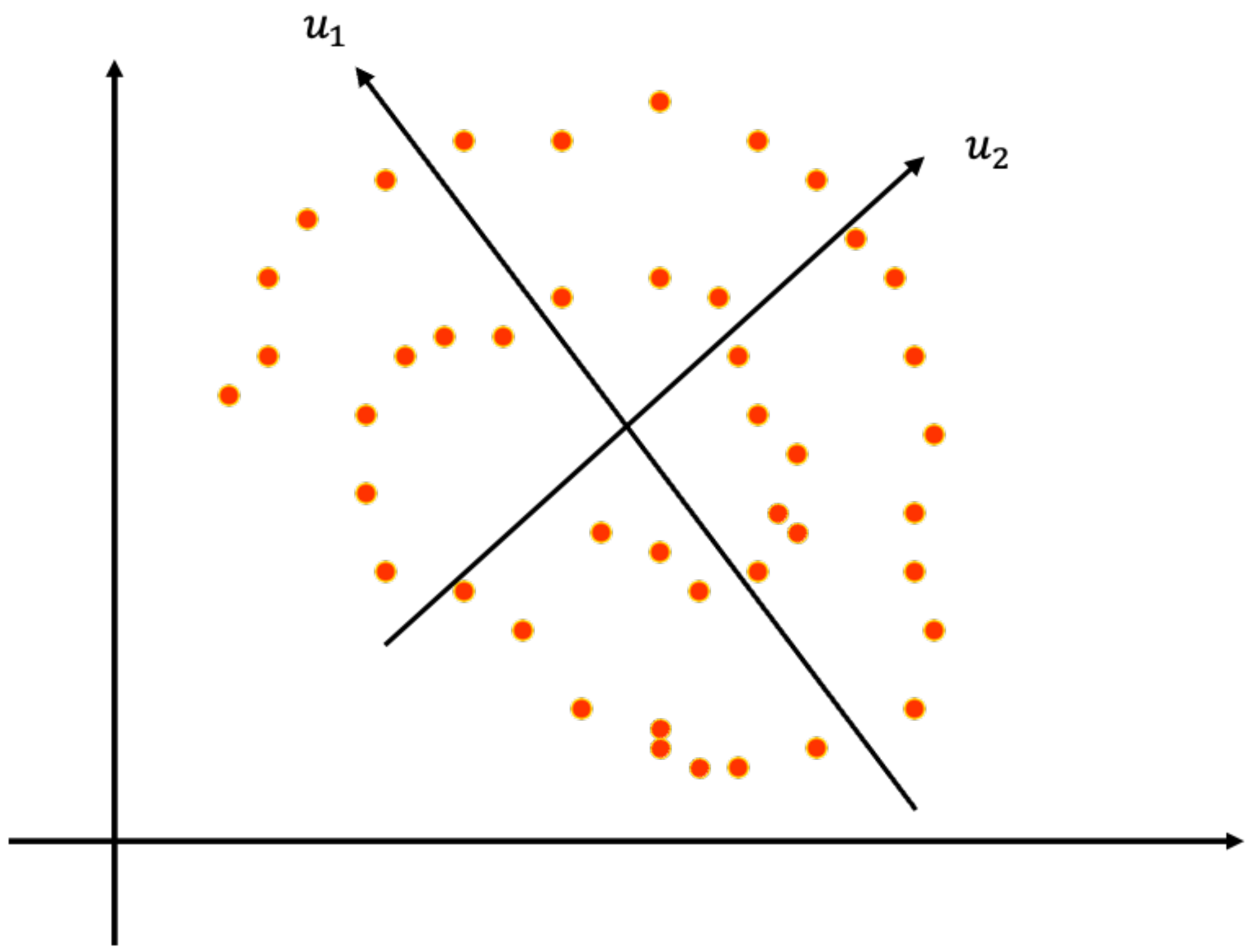
- Interestingly, the eigenvectors look similar to cosine transforms (DCT)
  - A version using DCT is called JPEG

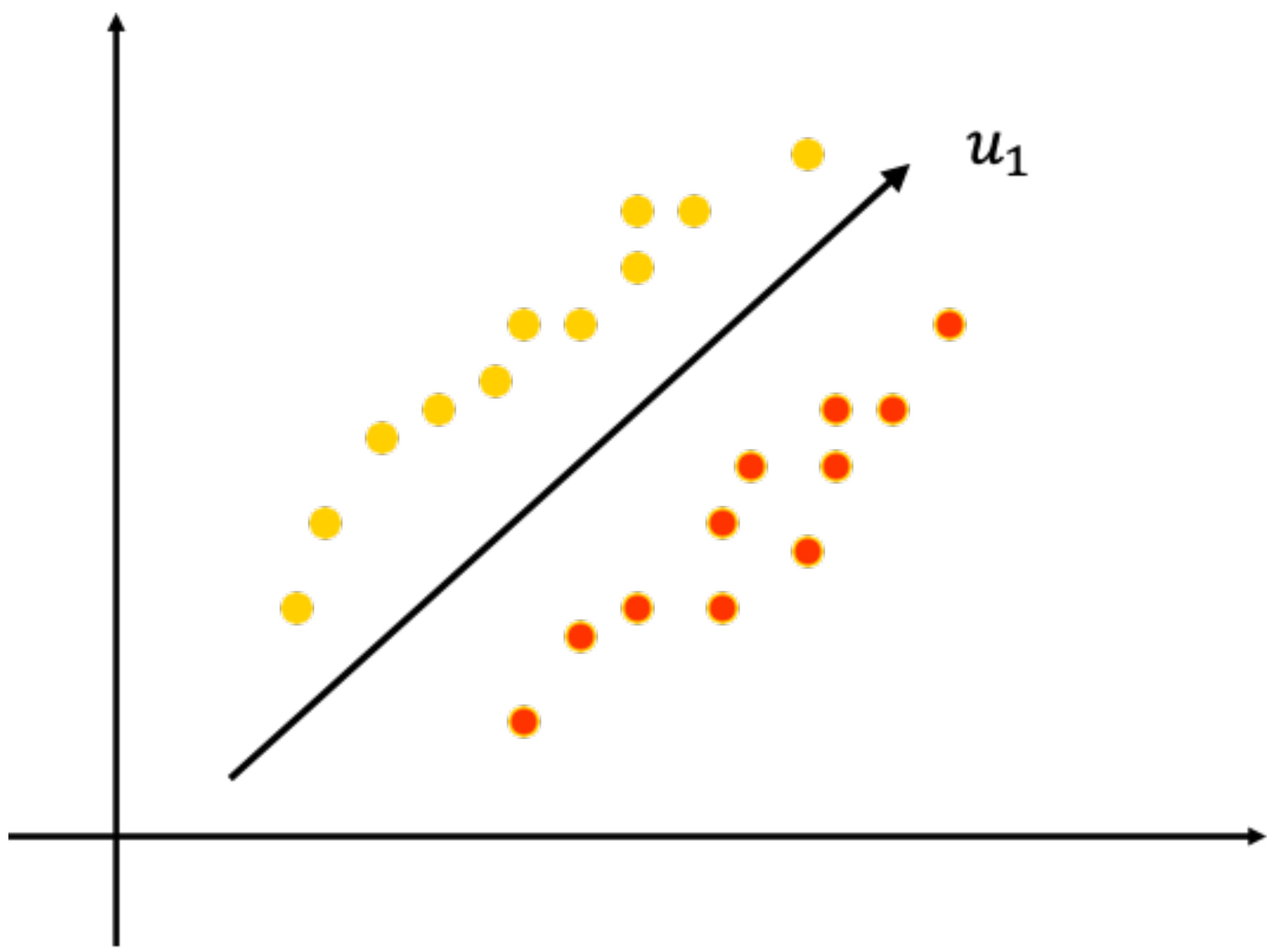

Eigenvectors

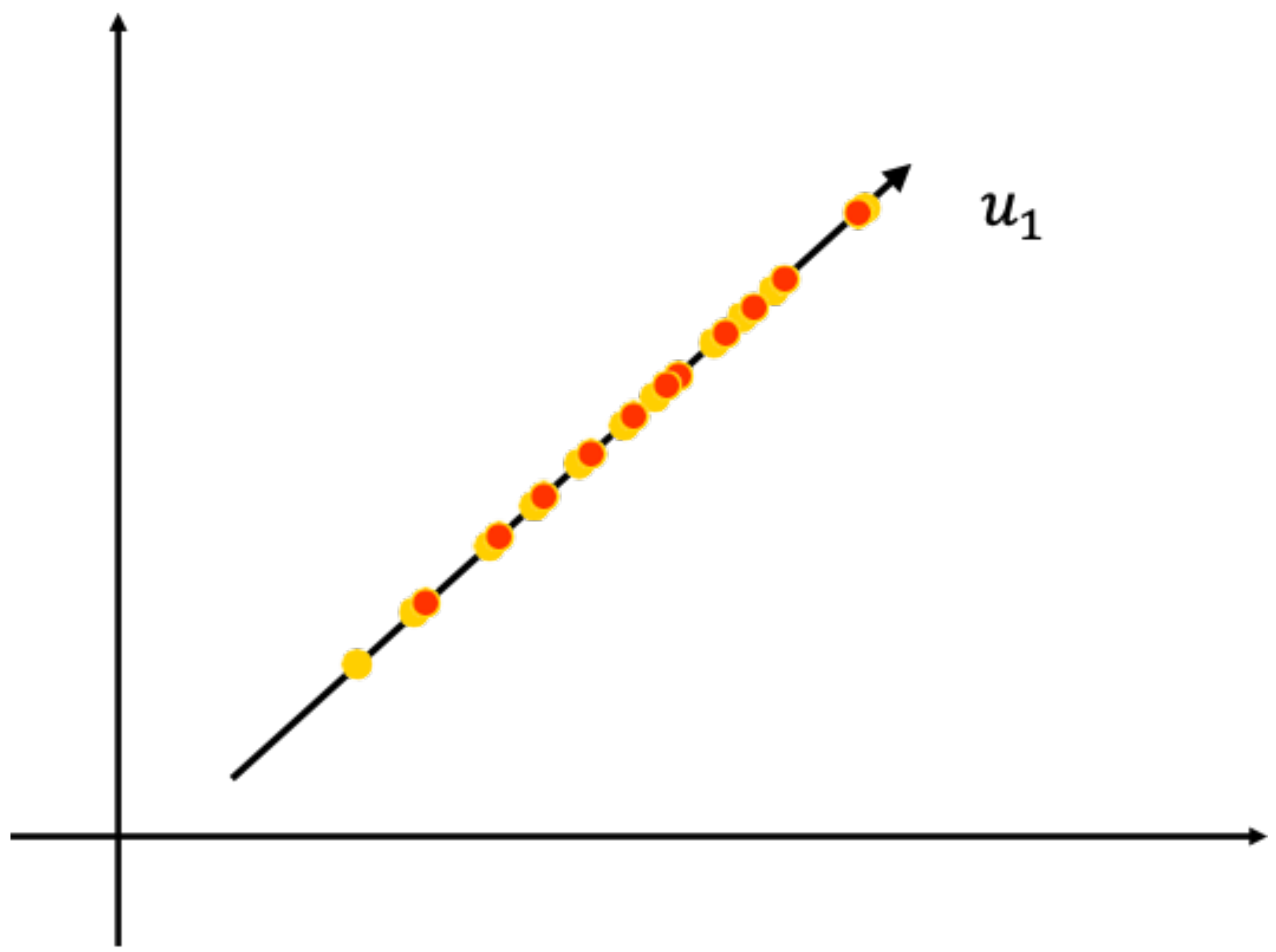

DCT bases

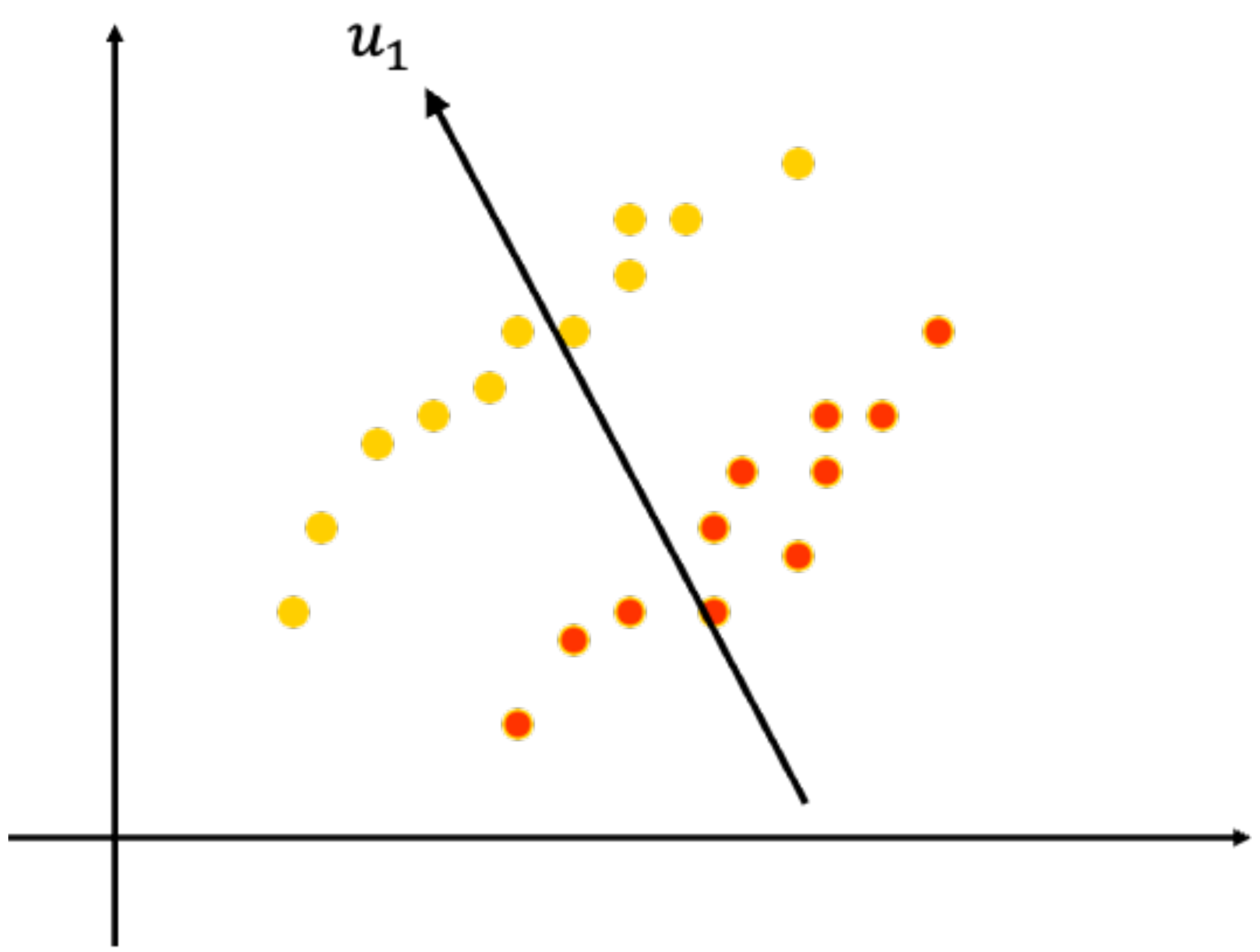
Limitations

- Difficult to capture nonlinear dataset
- Does not account for class labels



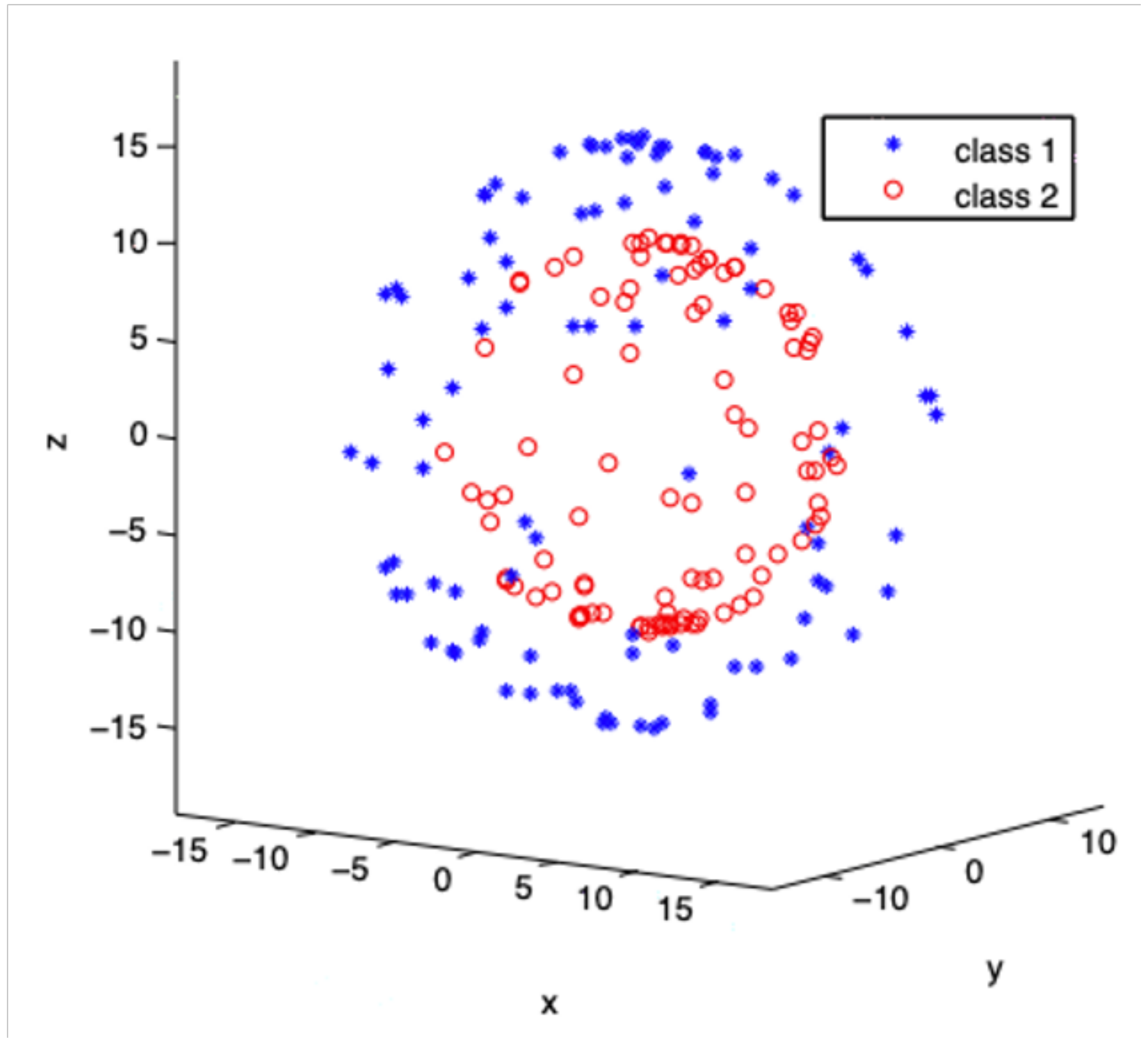




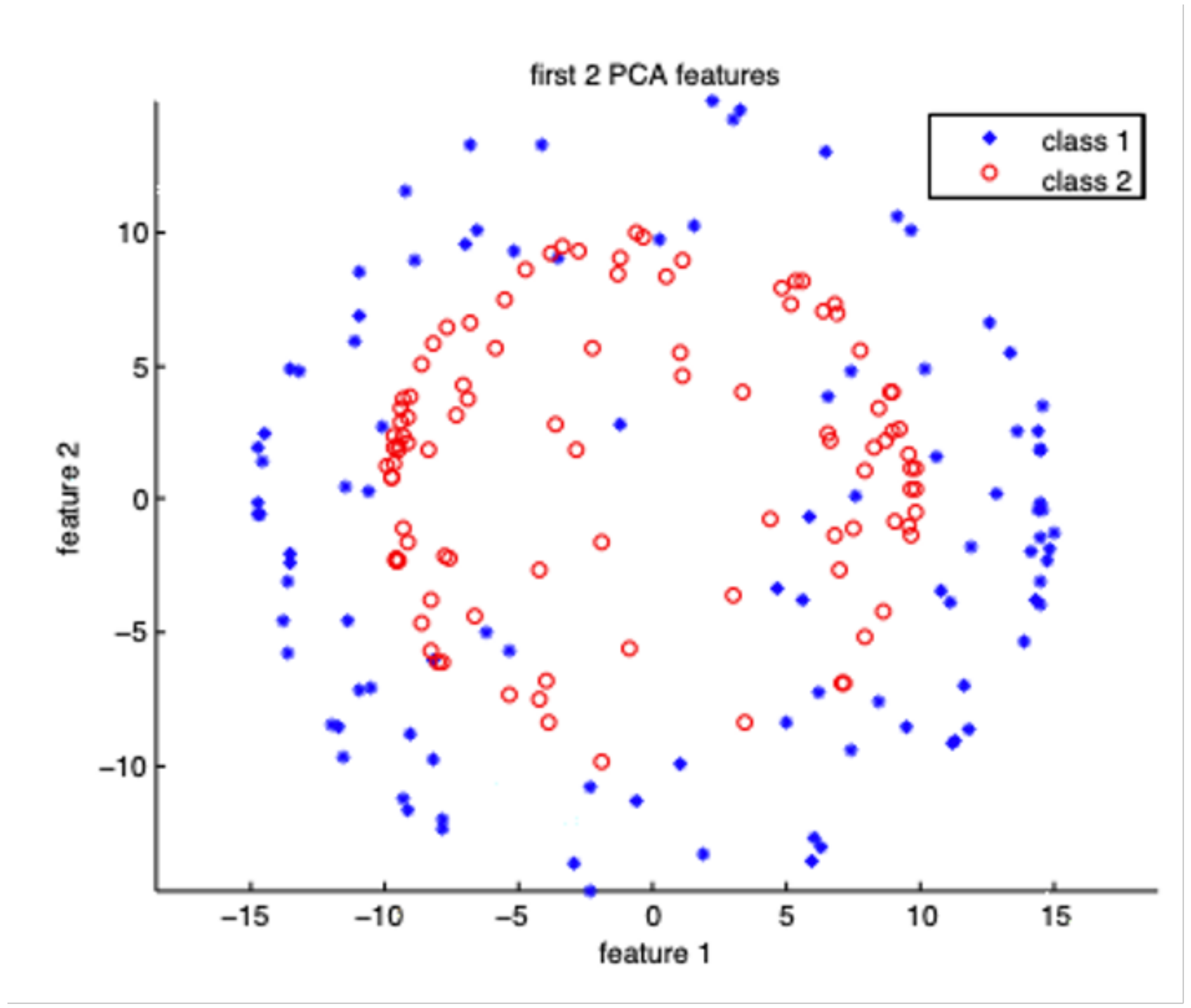


Advanced methods

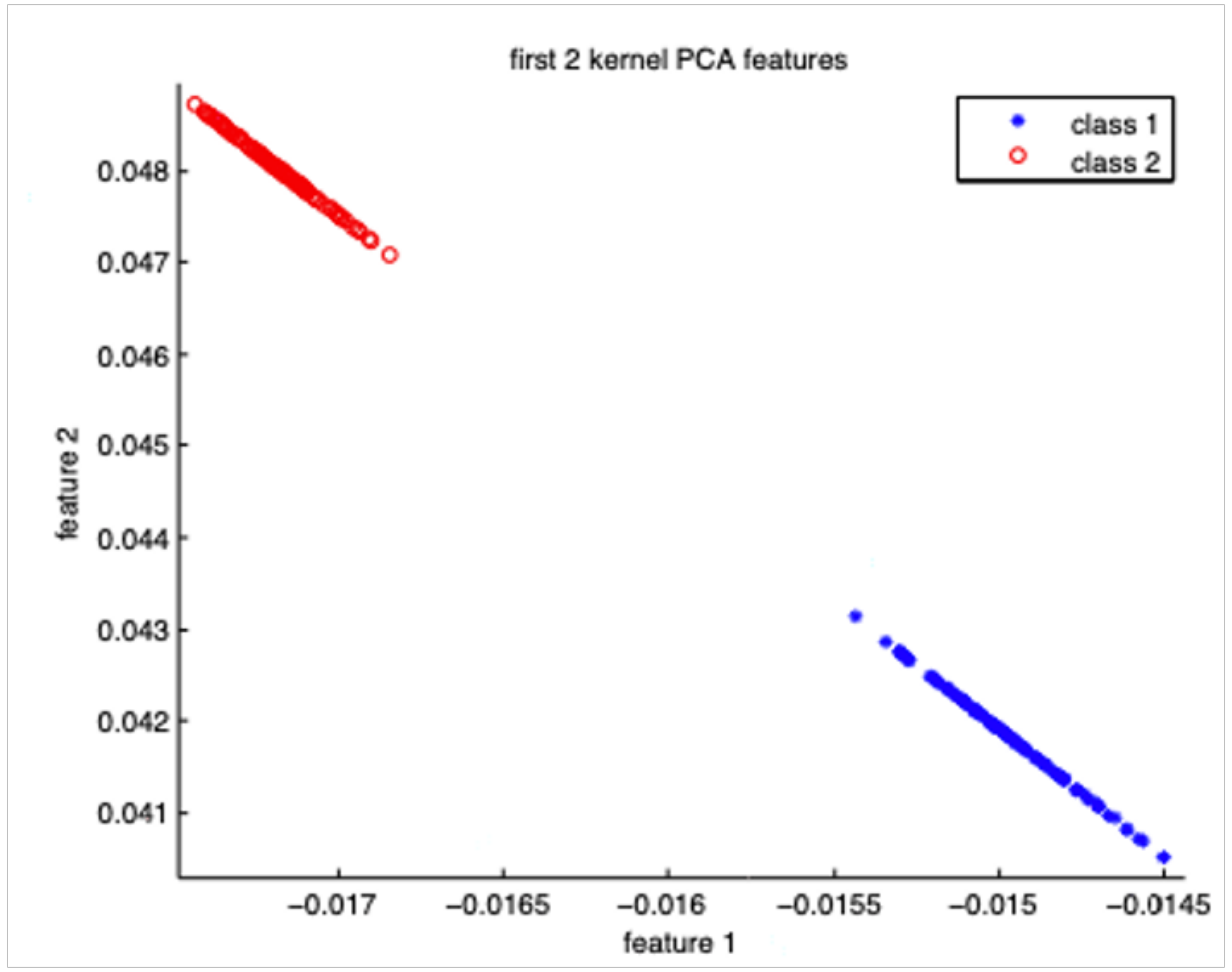
- **Kernel PCA.** Conduct PCA for $\Phi(\mathbf{x})$
  - Requires careful hyperparameter tuning & validation



Spherical Data



No Kernel

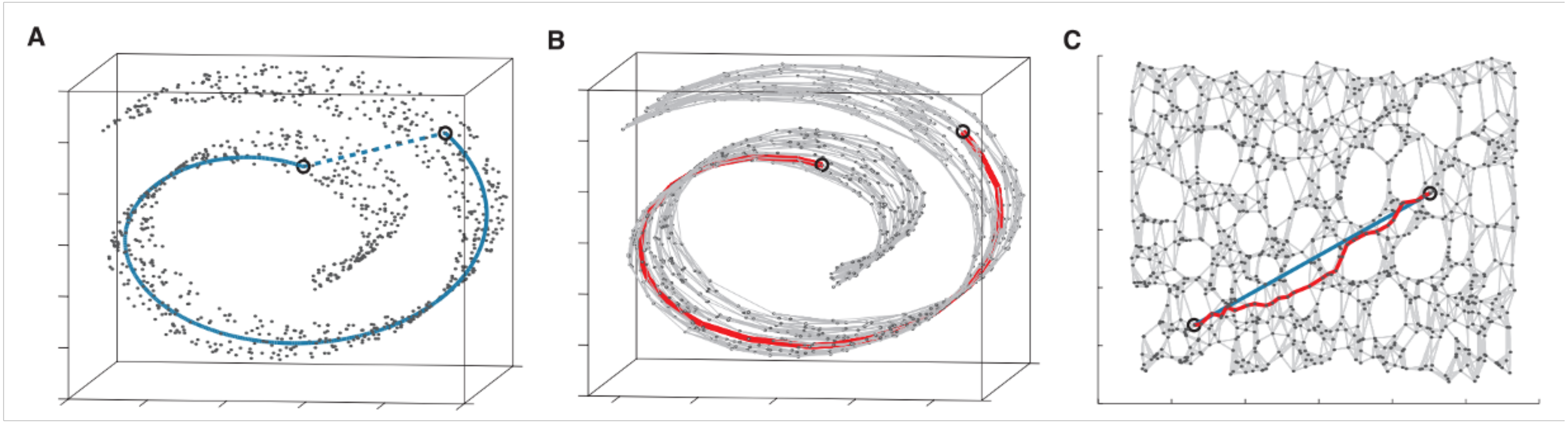


Gaussian Kernel ($\sigma = 20$)

Isomap

- Similarly to spectral clustering, build a graph of points by connecting each point to k-nearest neighbors
- Then, find a mapping to a low-dimensional space such that:

$$\text{distance on graph} \approx \text{distance on embedded space}$$

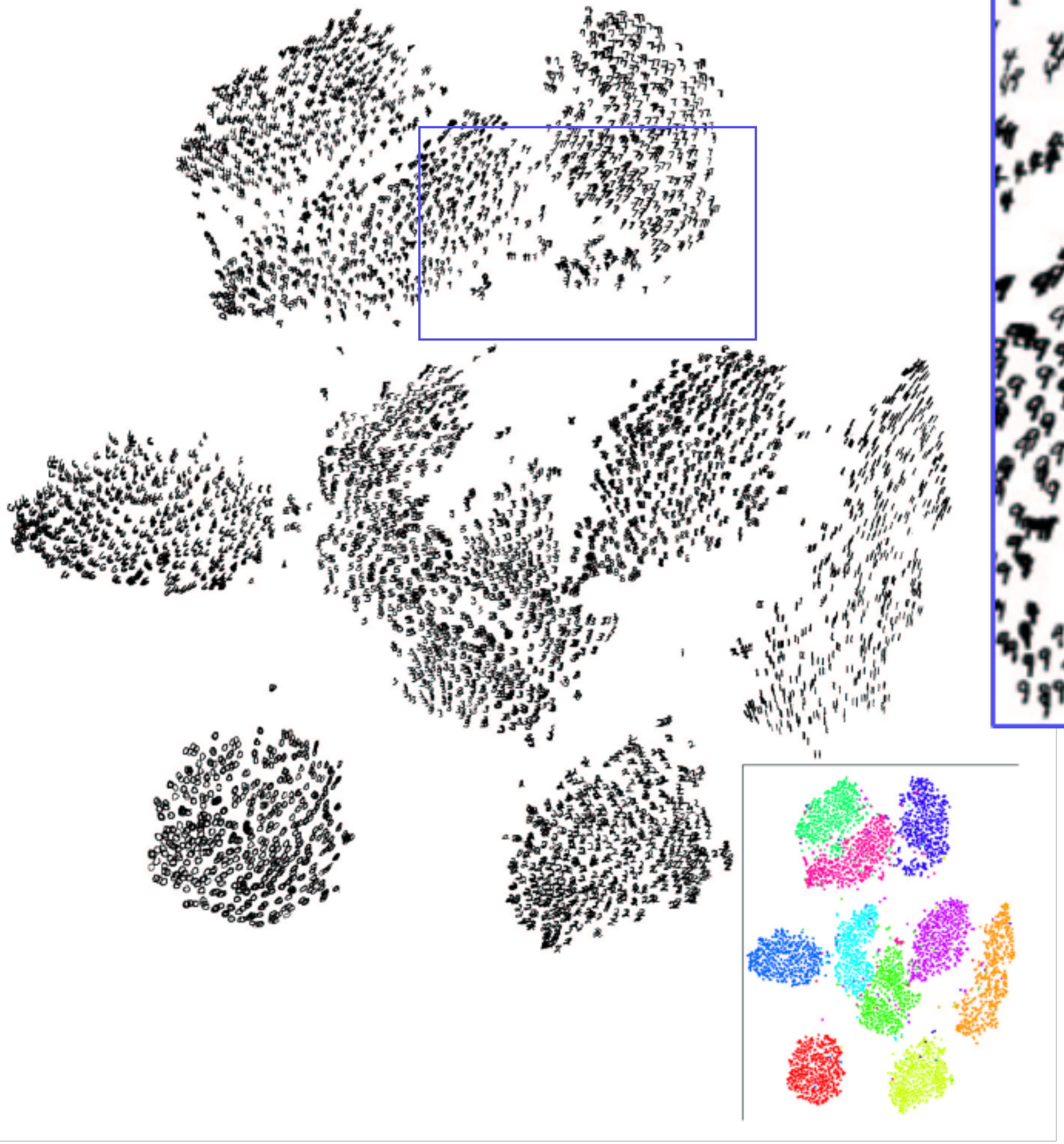


t-SNE

- Similar to Isomap, but use the neighborhood information

$$p_i(j) = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma^2)}$$

- Find a low-dimensional embedding such that $\mathrm{dist}(p_i, p_j) \approx \mathrm{dist}(\mathbf{z}_i, \mathbf{z}_j)$



MNIST embeddings of t-SNE

(requires computing pairwise distances of 60,000 samples)

Next up

- Decision trees

</lecture 9>