Principal Component Analysis

Nishant Mehta

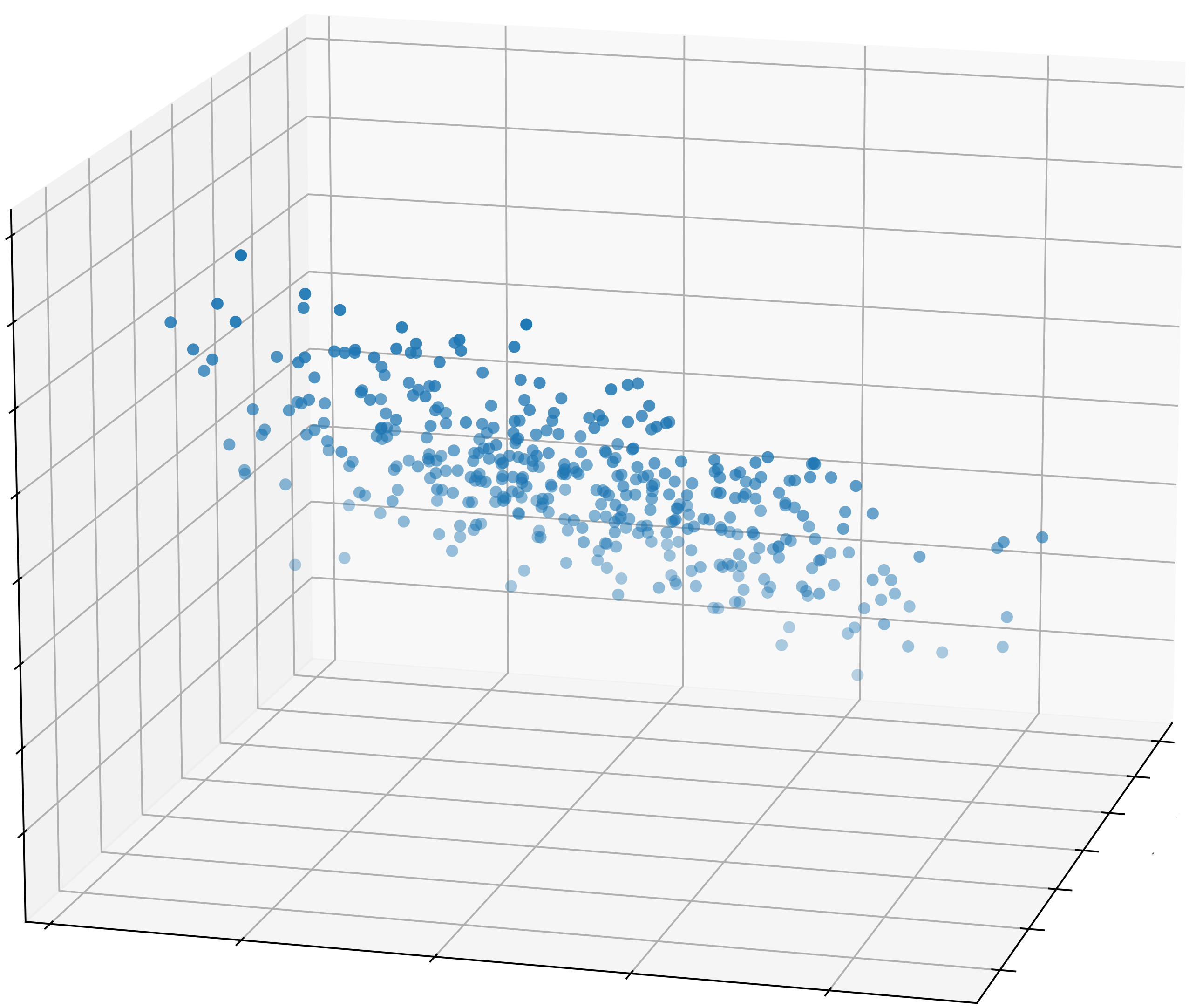
Lecture 20

Dimension reduction

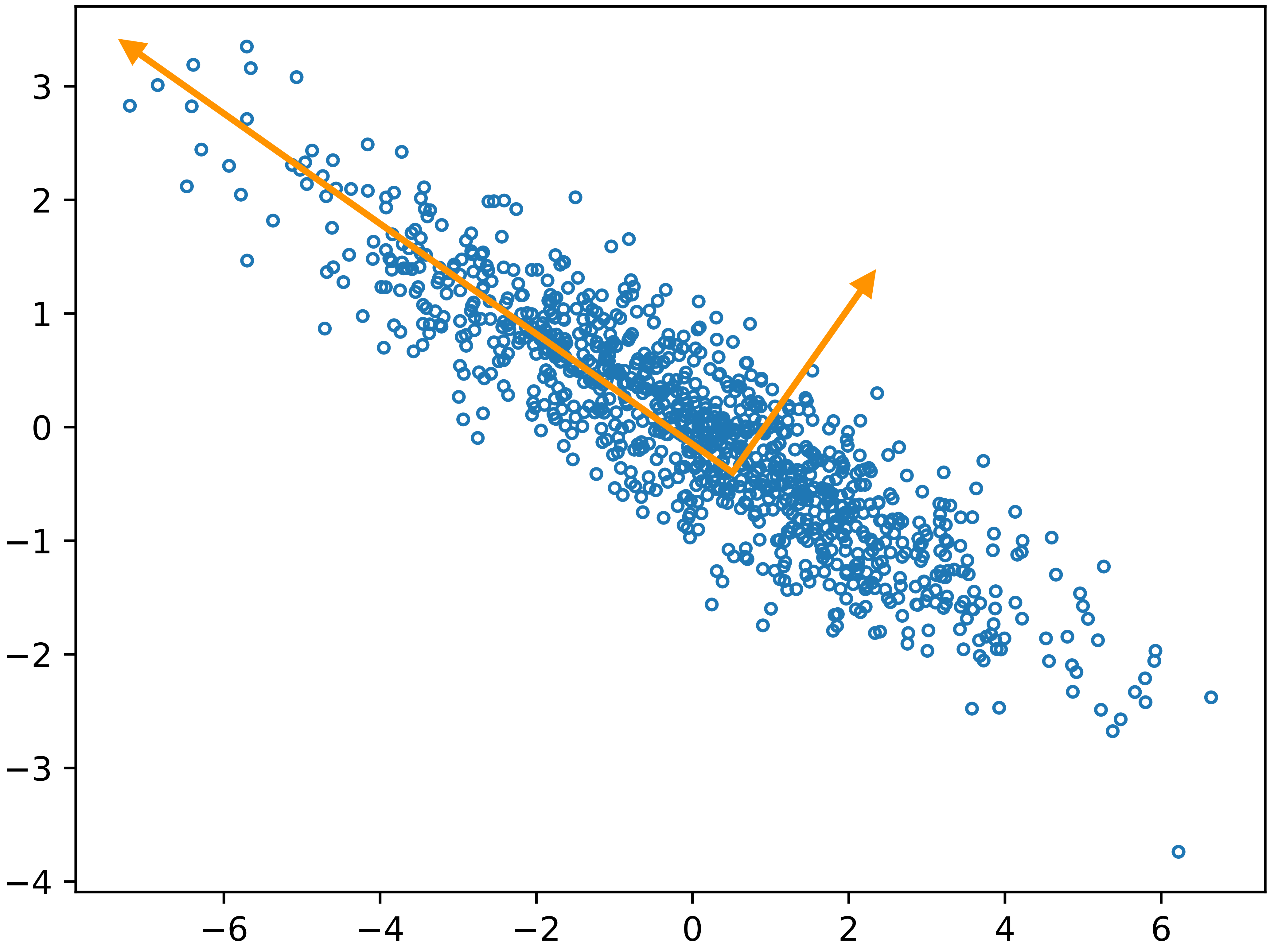
Data is often high-dimensional, but there might be a low-dimensional subspace that captures most of the variability of the data.

How can we find the best-fit low-dimensional subspace?

How can we represent the data in this low-dimensional subspace?

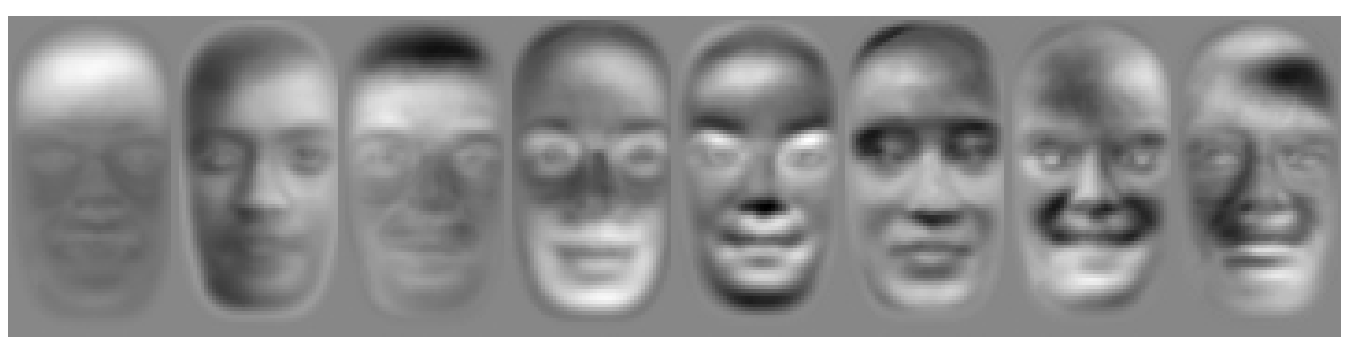


PCA: Finding dominant directions of variation



Teaser Trailer: Eigenfaces

Face dataset



Visualization of principal directions of variation

(Moghaddam, Wahid, and Pentland, 1998)

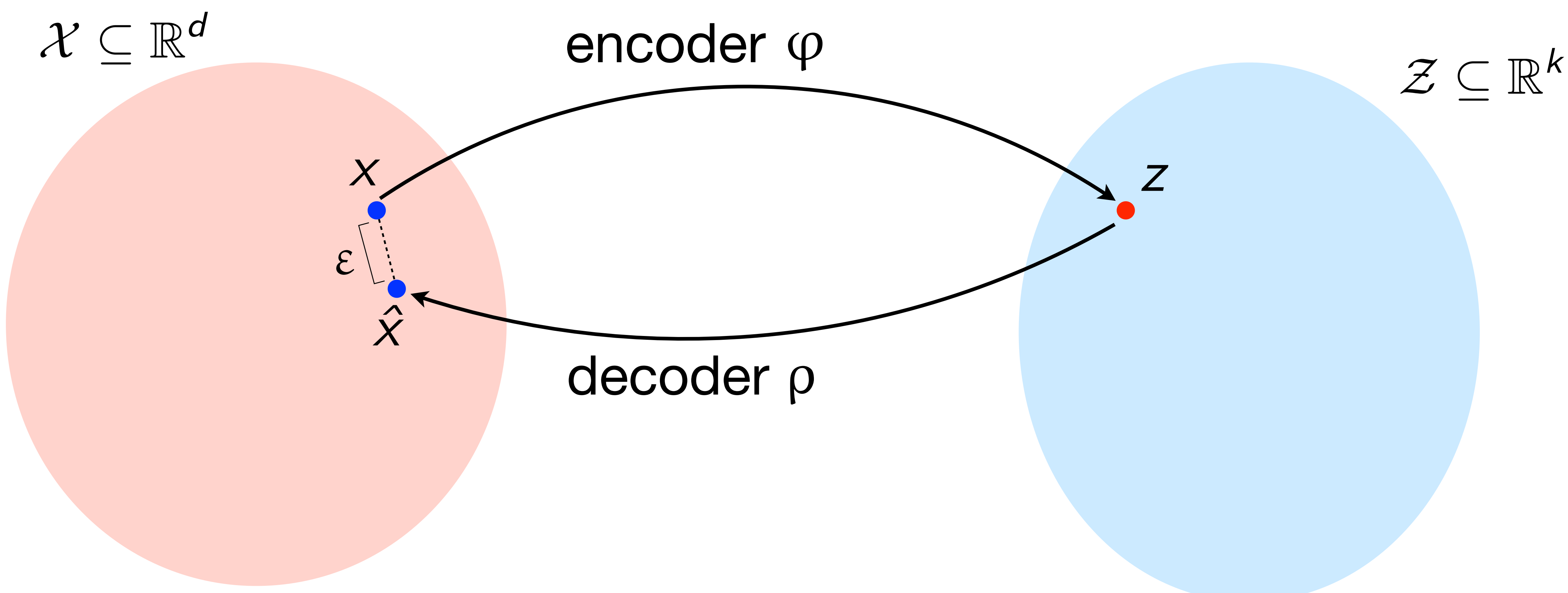
Auto-encoders

An auto-encoder is a way of mapping an input feature vector x to an approximation $\hat{x}$

(also "compressor" or "encoder")

Typically done by compressing via a *feature map* φ and then decompressing via a *reconstruction map* ρ

(also "reconstructor" or "decoder")



Auto-encoders

We often take $k \ll d$ (so, z achieves dimension reduction)

(1) Compressing: $z = \varphi(x)$

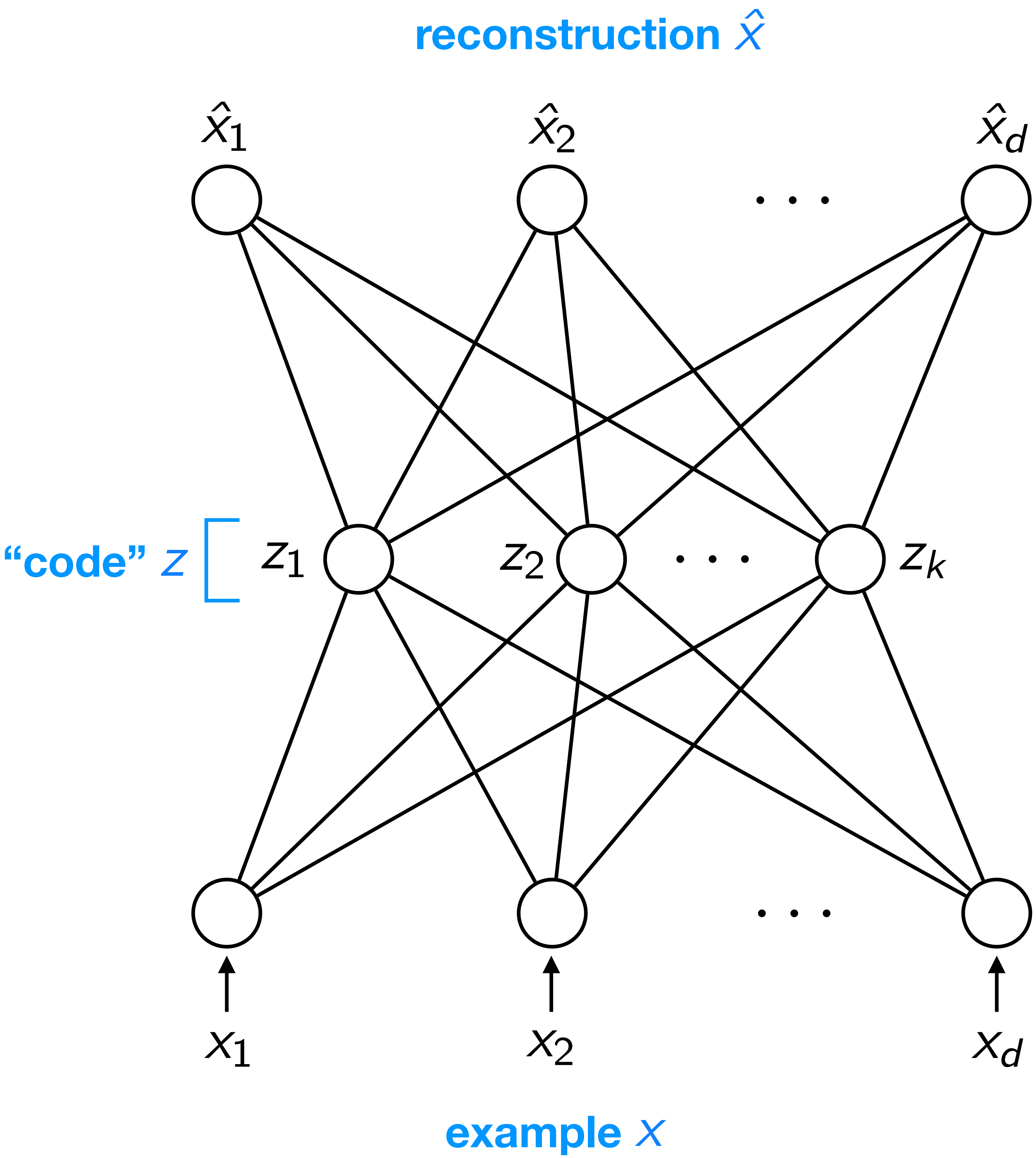
(2) Reconstructing: $\hat{x} = \rho(z) = \rho(\varphi(x))$

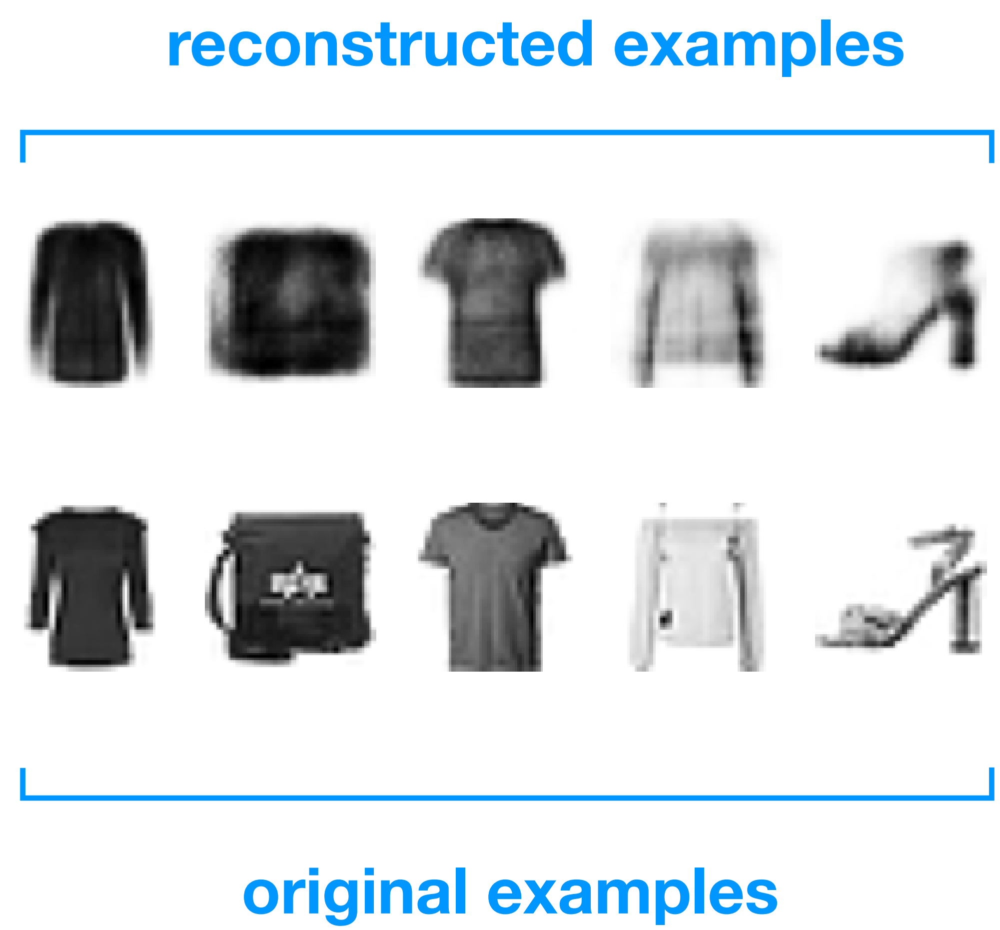
Goal: Achieve low reconstruction error (often take squared error):

So, we seek pair of maps (φ, ρ) such that

$$\|x - \hat{x}\|^2 = \|x - \rho(\varphi(x))\|^2 \text{ is small}$$

Neural network view of auto-encoders



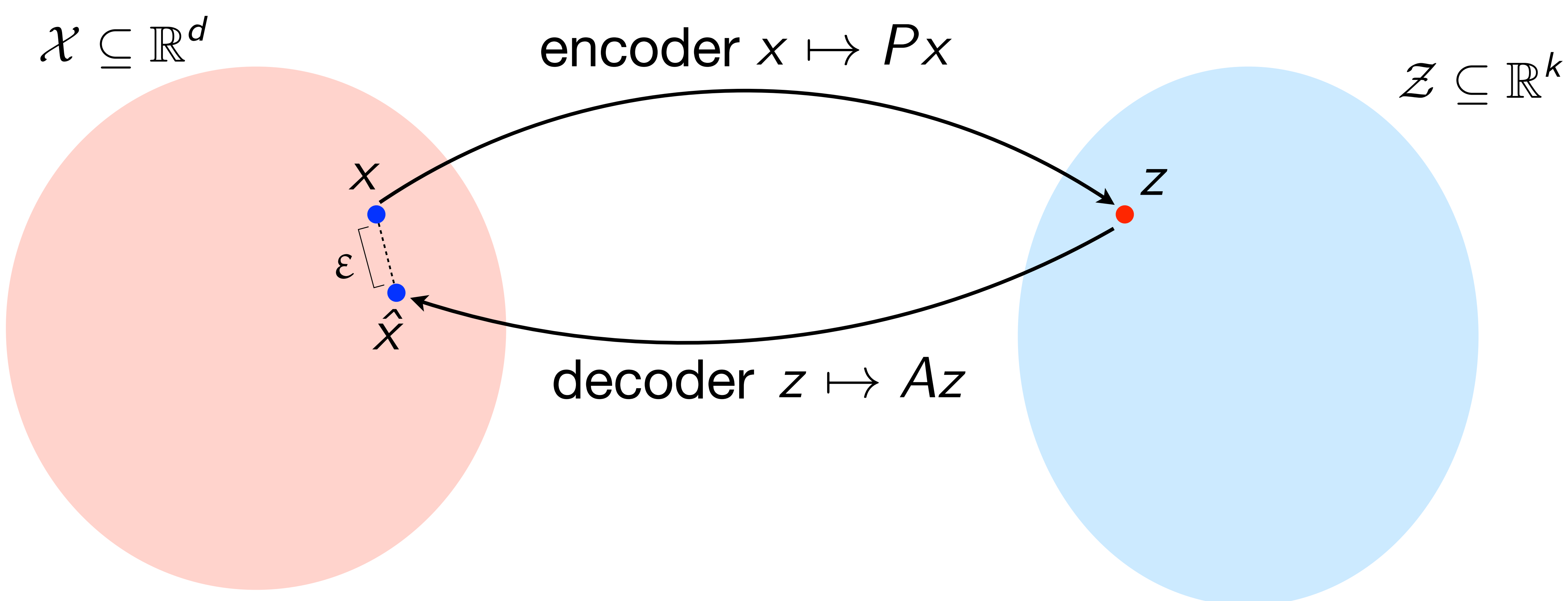


Linear auto-encoders

Among all *linear* auto-encoders that map to a new representation of dimension k, which auto-encoder is the best one?

Linear feature map: $z = \varphi(x) = Px$ for $P \in R^{k \times d}$

Linear reconstruction map: $\hat{x} = \rho(z) = Az$ for $A \in R^{d \times k}$



Linear auto-encoders: Encoding

Let's get familiar with the linear encoding/decoding operations

First, let's write the linear encoder matrix $P \in R^{k \times d}$ as $P = \begin{pmatrix} p_1^T \\ \vdots \\ p_k^T \end{pmatrix}$

Then, x is encoded as $z = Px$, or $\begin{pmatrix} z_1 \\ \vdots \\ z_k \end{pmatrix} = \begin{pmatrix} p_1^T x \\ \vdots \\ p_k^T x \end{pmatrix}$

If each vector p_j is a unit vector, then it represents a direction.

So, the j^{th} new feature z_j measures the activation (or strength) of x in the direction p_j.

Linear auto-encoders: Decoding

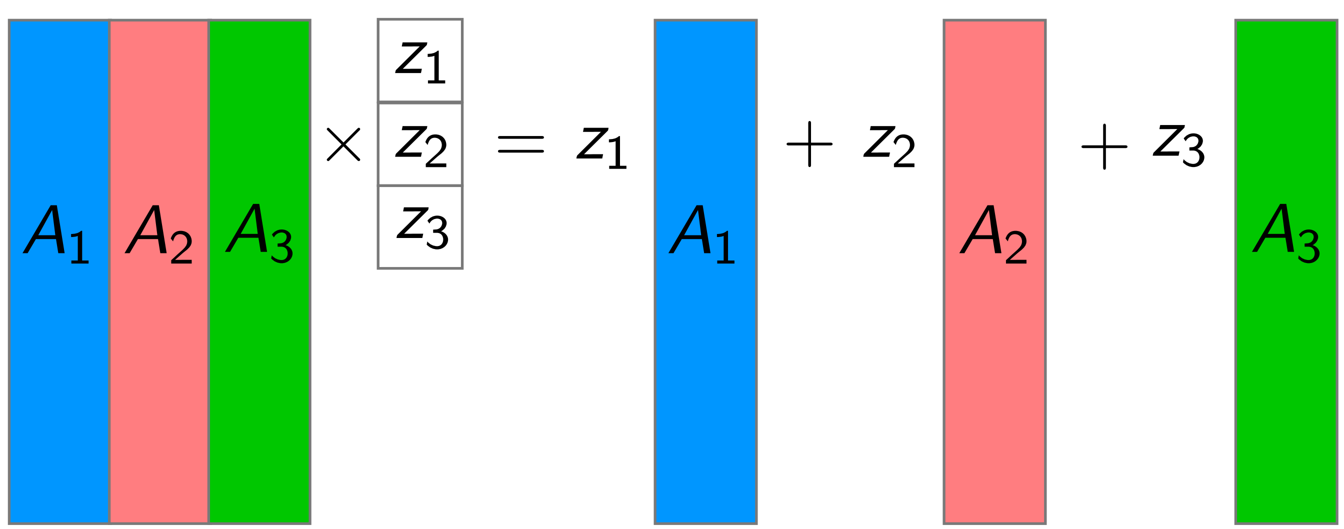
Let's get familiar with the linear encoding/decoding operations

What does decoding look like in terms of linear algebra?

Suppose we have a code (learned representation) z

To decode, use linear decoder matrix $A \in R^{d \times k}$: $Az = \sum_{j=1}^{k} A_j z_j$

j^{th} column of A



Linear auto-encoders: projecting onto a subspace

Let $V = (v_1, \ldots, v_k)$ be a matrix whose columns are orthonormal. Take $P = V^T$ and $A = V$.

Then linear auto-encoding involves applying the matrix VV^T to x. This can also be written as:

$$APx = VV^Tx = \sum_{j=1}^{k} v_j v_j^T x$$

Also, VV^T is a projection matrix that projects any vector x onto the subspace spanned by $v_1, \ldots, v_k$.

Simple exercise: what is VV^T if $k = d$?

Q: Why are we talking about projection matrices? What about PCA?
A: We will see that PCA uses $P = V^T$ and $A = V$ for very special choice of V

Linear auto-encoders

Among all *linear* auto-encoders that map to a new representation of dimension k, which auto-encoder is the best one?

We want $P \in R^{k \times d}$ and $A \in R^{d \times k}$ that minimize reconstruction error:

$$\sum_{i=1}^{n} \|x_i - APx_i\|^2$$

Nice simplification: we may always assume that the columns of A are unit norm (so they are direction vectors). Why? Suppose j^{th} column of A has norm α_j. Then we can form equivalent pair $\tilde{A}$ and $\tilde{P}$:

$$\underbrace{\begin{bmatrix} & A & \end{bmatrix} \begin{bmatrix} \frac{1}{\alpha_1} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{\alpha_k} \end{bmatrix}}_{\tilde{A}} \underbrace{\begin{bmatrix} \alpha_1 & & 0 \\ & \ddots & \\ 0 & & \alpha_k \end{bmatrix} \begin{bmatrix} & P & \end{bmatrix}}_{\tilde{P}}$$

Starting point: k = 1

Let's take the case of $k = 1$ and consider a fixed matrix P

We have $P \in \mathbb{R}^{1 \times d}$, and so $P = p_1^T$ for some vector p_1

Towards PCA

Given: data $X = (x_1 \ x_2 \ \cdots \ x_n) \in \mathbb{R}^{d \times n}$ (*n* examples, *d* dimensions)

First step: Center the data so that $\frac{1}{n}\sum_{i=1}^{n} x_i = 0$

lowest squared error

How can we find the best one-dimensional projection of the data?

Claim. The following are equivalent:

(a) The best 1D projection of the data

(b) The 1D projection of the data
such that $(z_1, \ldots, z_n)$ has maximum variance

Showing the claim in 2 steps

Step 1: Given a reconstructor A, what is the best encoding z of an example x?

Step 2: Which reconstructor A gives the best projection of the data?

Step 1: Given a reconstructor *A*, what is the best projection *z* of an example *x*?

Let $A = a_1$ and z be the encoding (code word) for example x

Then $\hat{x} = A\,z = z\,a_1$

How can we find the best encodings $z_1, \ldots, z_n$ for data $x_1, \ldots, x_n$?

Recall our objective: $$\min_{\substack{a_1 \in \mathbb{R}^d, \|a_1\|=1 \\ z_1, z_2, \ldots, z_n}} \frac{1}{n} \sum_{i=1}^{n} \|x_i - \hat{x}_i\|^2$$

Since a_1 is given, problem decouples: $\min_{z_i} \|x_i - z_i a_1\|^2$

Standard exercise. Set derivative of objective (with respect to z_i) to zero and solve for z_i:

We get $z_i = a_1^T x_i$, so $p_1 = a_1$

Step 2: Which reconstructor A gives best 1D projection?

Recall: $\hat{x}_i = z_i a_1$

$z_i = a_1^T x_i$

Our objective may now be written as:

$$\min_{\substack{a_1 \in \mathbb{R}^d \\ \|a_1\|=1}} \frac{1}{n} \sum_{i=1}^{n} \|x_i - \hat{x}_i\|^2 \equiv \min_{\substack{a_1 \in \mathbb{R}^d \\ \|a_1\|=1}} \frac{1}{n} \left(\|x_i\|^2 - 2 z_i a_1^T x_i + z_i^2 \|a_1\|^2 \right)$$

Step 2: Which reconstructor A gives best 1D projection?

Recall: $\hat{x}_i = z_i a_1$

$z_i = a_1^T x_i$

Our objective may now be written as:

$$\min_{\substack{a_1 \in \mathbb{R}^d \\ \|a_1\|=1}} \frac{1}{n} \sum_{i=1}^{n} \|x_i - \hat{x}_i\|^2 \equiv \min_{\substack{a_1 \in \mathbb{R}^d \\ \|a_1\|=1}} \frac{1}{n} \left(\|x_i\|^2 - 2 z_i a_1^T x_i + z_i^2 \|a_1\|^2 \right)$$

$$\equiv \min_{\substack{a_1 \in \mathbb{R}^d \\ \|a_1\|=1}} -\frac{1}{n} \sum_{i=1}^{n} z_i^2$$

Step 2: Which reconstructor A gives best 1D projection?

Recall: $\hat{x}_i = z_i a_1$

$z_i = a_1^T x_i$

Our objective may now be written as:

$$\min_{\substack{a_1 \in \mathbb{R}^d \\ \|a_1\|=1}} \frac{1}{n} \sum_{i=1}^{n} \|x_i - \hat{x}_i\|^2 \equiv \min_{\substack{a_1 \in \mathbb{R}^d \\ \|a_1\|=1}} \frac{1}{n} \left(\|x_i\|^2 - 2 z_i a_1^T x_i + z_i^2 \|a_1\|^2 \right)$$

$$\equiv \min_{\substack{a_1 \in \mathbb{R}^d \\ \|a_1\|=1}} -\frac{1}{n} \sum_{i=1}^{n} z_i^2$$

$$\equiv \max_{\substack{a_1 \in \mathbb{R}^d \\ \|a_1\|=1}} \frac{1}{n} \sum_{i=1}^{n} z_i^2$$

Step 2: Which reconstructor A gives best 1D projection?

Recall: $\hat{x}_i = z_i a_1$

$z_i = a_1^T x_i$

Our objective may now be written as:

$$\min_{\substack{a_1 \in \mathbb{R}^d \\ \|a_1\|=1}} \frac{1}{n}\sum_{i=1}^{n} \|x_i - \hat{x}_i\|^2 \equiv \min_{\substack{a_1 \in \mathbb{R}^d \\ \|a_1\|=1}} \frac{1}{n}\left(\|x_i\|^2 - 2z_i a_1^T x_i + z_i^2 \|a_1\|^2\right)$$

$$\equiv \min_{\substack{a_1 \in \mathbb{R}^d \\ \|a_1\|=1}} -\frac{1}{n}\sum_{i=1}^{n} z_i^2$$

$$\equiv \max_{\substack{a_1 \in \mathbb{R}^d \\ \|a_1\|=1}} \frac{1}{n}\sum_{i=1}^{n} z_i^2$$

$$\equiv \max_{\substack{a_1 \in \mathbb{R}^d \\ \|a_1\|=1}} \frac{1}{n}\sum_{i=1}^{n} \left(z_i - \frac{1}{n}\sum_{j=1}^{n} z_j\right)^2$$

maximize the sample variance!

$= 0$ since we centered the data

First principal component

Therefore, the first principal direction v_1 is the direction along which the data has the most variance

Given v_1, we can compute the first principal component as

$$v_1^T X = \begin{pmatrix} v_1^T x_1 & v_1^T x_2 & \dots & v_1^T x_n \end{pmatrix} = \begin{pmatrix} z_1 & z_2 & \dots & z_n \end{pmatrix}$$

How can we find v_1? Observe that

(sample) covariance matrix

$$\frac{1}{n}\sum_{i=1}^{n} z_i^2 = \frac{1}{n}\sum_{i=1}^{n} a_1^T x_i x_i^T a_1 = a_1^T \left(\frac{1}{n}\sum_{i=1}^{n} x_i x_i^T \right) a_1 = a_1^T C a_1$$

Question: which unit vector a_1 maximizes $a_1^T C a_1$?

First principal component

Key question: which unit vector a_1 maximizes $a_1^T C a_1$?

Idea: Express C as $C = V \Lambda V^T$ for

$$\begin{pmatrix} v_1 & v_2 & \dots & v_d \end{pmatrix}$$

eigenvectors

$$\begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_d \end{pmatrix}$$

eigenvalues

Since eigenvectors form an orthonormal basis, we can express a_1 in terms of the eigenbasis ($a_1 = \sum_{j=1}^{d} \alpha_j v_j$)

j^{th} coefficient

From there, it is not difficult to show that $a_1 = v_1$.

The best k-dimensional subspace

In general, how can we find the best k-dimensional subspace?

Suppose we have already found the first r principal directions $v_1, v_2, \ldots, v_r$, and now we seek the $(r+1)^{\text{th}}$ principal direction

Observation: The principal directions will be orthogonal. Why?

Hopefully easier question: how can we find the best set of orthogonal unit basis vectors $v_1, \ldots, v_k$?

Objective: $$\min_{v_1, \ldots, v_k} \frac{1}{n} \sum_{i=1}^{n} \|x_i - \hat{x}_i\|^2$$

Reconstruction based on $v_1, \ldots, v_k$

The best *k*-dimensional subspace

original	reconstruction (*k* dimensions)
x_i	$\hat{x}_i$
$\mathrm{I}_{d \times d} x_i$	$\sum_{j=1}^{k} v_j \underbrace{\left(v_j^T x_i\right)}_{z_{i,j}}$
$\left(\sum_{j=1}^{d} v_j v_j^T\right) x_i$	$\left(\sum_{j=1}^{k} v_j v_j^T\right) x_i$

$z_{i,j}$ (j^{th} feature of $\varphi(x_i)$)

The best k-dimensional subspace

original

x_i

$\mathrm{I}_{d\times d}\, x_i$

$$\left(\sum_{j=1}^{d} v_j v_j^T\right) x_i$$

reconstruction (k dimensions)

$\hat{x}_i$

$z_{i,j}$ (j^{th} feature of $\varphi(x_i)$)

$$\sum_{j=1}^{k} v_j \left(v_j^T x_i\right)$$

$$\left(\sum_{j=1}^{k} v_j v_j^T\right) x_i$$

$$\|x_i - \hat{x}_i\|^2 = \left\| \sum_{j=k+1}^{d} v_j v_j^T x_i \right\|^2 = \sum_{j=k+1}^{d} \|v_j v_j^T x_i\|^2 = \sum_{j=k+1}^{d} v_j^T x_i x_i^T v_j$$

$$\Rightarrow \quad \frac{1}{n}\sum_{i=1}^{n} \|x_i - \hat{x}_i\|^2 = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=k+1}^{d} v_j^T x_i x_i^T v_j = \sum_{j=k+1}^{d} v_j^T \left(\frac{1}{n}\sum_{i=1}^{n} x_i x_i^T\right) v_j$$

The best *k*-dimensional subspace

Idea: We still use eigendecomposition of covariance matrix

$$C = V \Lambda V^T$$

$$\begin{pmatrix} v_1 & v_2 & \dots & v_d \end{pmatrix}$$

eigenvectors

$$\begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_d \end{pmatrix}$$

eigenvalues

PCA

How to compute principal components:

(1) Center the data $X \in \mathbb{R}^{d \times n}$ so that $\frac{1}{n}\sum_{i=1}^{n} X_i = 0$

(i^{th} example)
i^{th} column of X

(2) Compute covariance matrix $C = \frac{1}{n} X X^T \in \mathbb{R}^{d \times d}$

(3) Compute eigendecomposition of C

$$C = V \Lambda V^T$$

$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$

$V = \begin{pmatrix} v_1 & v_2 & \ldots & v_d \end{pmatrix}$

$\Lambda = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_d \end{pmatrix}$

(4) The j^{th} principal component is $v_j^T X$
The first k PCs are obtained via $(v_1 \ldots v_k)^T X \in \mathbb{R}^{k \times n}$

Properties of principal components

The principal components have a few fundamental properties

(1) PCs 1 through *d* are in order of decreasing variance

Why? The sample variance of the j^{th} PC is

$$\frac{1}{n}\sum_{i=1}^{n}(v_j^T X_i)^2 = v_j^T\left(\frac{1}{n}\sum_{i=1}^{n}X_i X_i^T\right)v_j = v_j^T \overbrace{C v_j}^{\lambda_j v_j} = \lambda_j \|v_j\|^2 = \lambda_j$$

(2) PCs are uncorrelated. Why? For any distinct *j*, *k*

$$\frac{1}{n}\sum_{i=1}^{n}(v_j^T X)_i (v_k^T X)_i = v_j^T\left(\frac{1}{n}X X^T\right)v_k = v_j^T \overbrace{C v_k}^{\lambda_k v_k} = \lambda_k v_j^T v_k = 0$$

Practical Usage

Let $A = [v_1 \;\; v_2 \;\; \ldots \;\; v_k]$ be matrix of first *k* principal directions

(1) Get principal components for input *x*:

mean of data

$$z = A^T(x - \mu)$$

(2) Reconstruct:

$$\hat{x} = Az + \mu = AA^T(x - \mu) + \mu$$

remember to add the mean back in

Eigenfaces for Recognition

Matthew Turk and Alex Pentland
Vision and Modeling Group
The Media Laboratory
Massachusetts Institute of Technology

Abstract

■ We have developed a near-real-time computer system that can locate and track a subject's head, and then recognize the person by comparing characteristics of the face to those of known individuals. The computational approach taken in this system is motivated by both physiology and information theory, as well as by the practical requirements of near-real-time performance and accuracy. Our approach treats the face recognition problem as an intrinsically two-dimensional (2-D) recognition problem rather than requiring recovery of three-dimensional geometry, taking advantage of the fact that faces are normally upright and thus may be described by a small set of 2-D characteristic views. The system functions by projecting face images onto a feature space that spans the significant variations among known face images. The significant features are known as "eigenfaces," because they are the eigenvectors (principal components) of the set of faces; they do not necessarily correspond to features such as eyes, ears, and noses. The projection operation characterizes an individual face by a weighted sum of the eigenface features, and so to recognize a particular face it is necessary only to compare these weights to those of known individuals. Some particular advantages of our approach are that it provides for the ability to learn and later recognize new faces in an unsupervised manner, and that it is easy to implement using a neural network architecture. ■

Eigenfaces for Recognition

Matthew Turk and Alex Pentland
Vision and Modeling Group
The Media Laboratory
Massachusetts Institute of Technology

that … the … of … this … eory, … per- … cog- … 2-D) … ree- … aces … l set … cting

face images onto a feature space that spans the significant variations among known face images. The significant features are known as "eigenfaces," because they are the eigenvectors (principal components) of the set of faces; they do not necessarily correspond to features such as eyes, ears, and noses. The projection operation characterizes an individual face by a weighted sum of the eigenface features, and so to recognize a particular face it is necessary only to compare these weights to those of known individuals. Some particular advantages of our approach are that it provides for the ability to learn and later recognize new faces in an unsupervised manner, and that it is easy to implement using a neural network architecture. ■

Eigenfaces for Recognition

Matthew Turk and Alex Pentland
Vision and Modeling Group
The Media Laboratory
Massachusetts Institute of Technology

mean

face images onto a feature space that spans the significant

top 7 eigenfaces