CS-C3240 - Machine Learning

Feature Learning

Alexander Jung

Learning Goals

- understand challenges with long raw feature vectors
- basic idea of feature learning
- using feature learning to visualize data and protect privacy
- principal component analysis
- random projections

What Are Three Components of ML ?

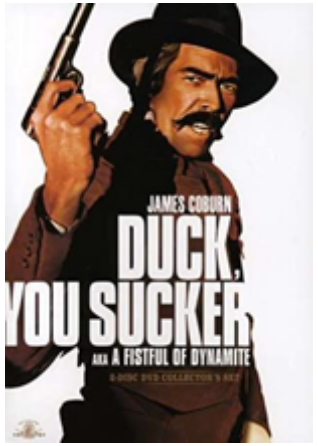
A Recent Movie Night

- Partner A: “What movie should we watch?”
- Partner B: “Let’s watch ”The Campaign”!!!”
- Partner A: “No F*** Way!”
- Partner B: “Let’s explore using ML ..”

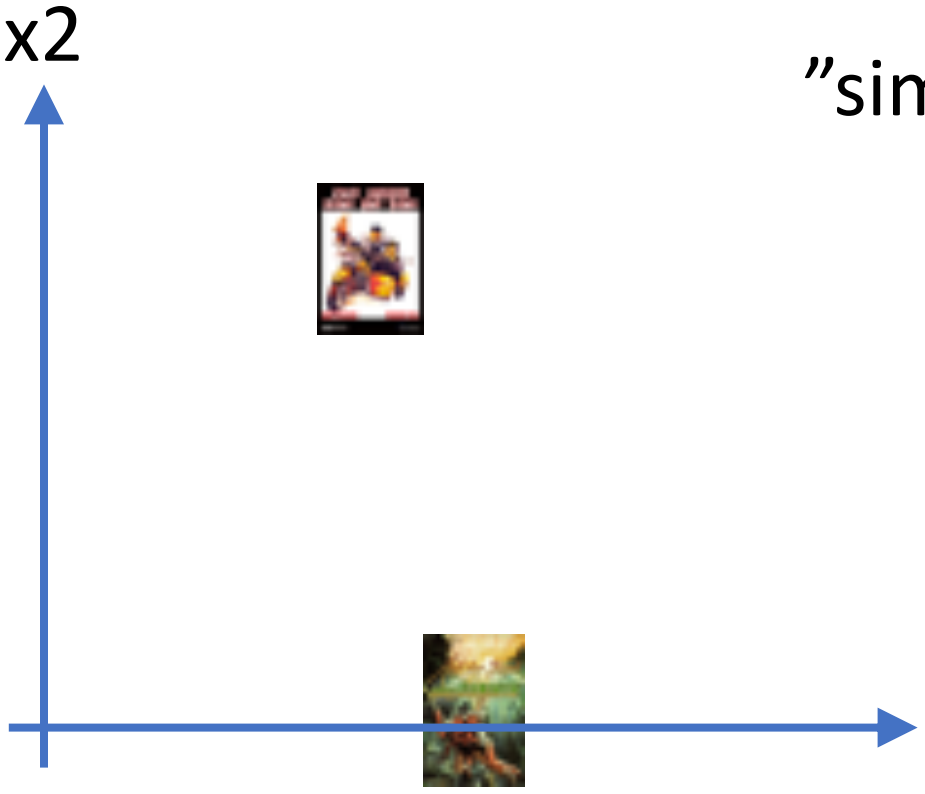
Data Point = “Some Movie”

several Gigabytes of raw feature bits!

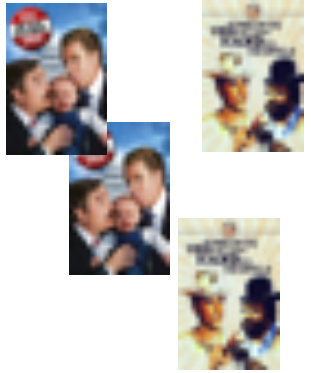
Dataset = ”Bunch of Movies”

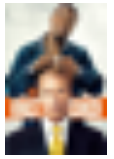


Scatterplot of Movies?



”similar” movies should be placed nearby!





movie represented by two features x1,x2 !

Curse of Big Data

Overwhelmed by Tons of Features!

- consider data point representing a person
- digital footprint can be used for constructing features
- health-records (including genetic fingerprint)
- credit-card transactions
- social media posts
- media collections
- travelling profile over last 20 years
- ….

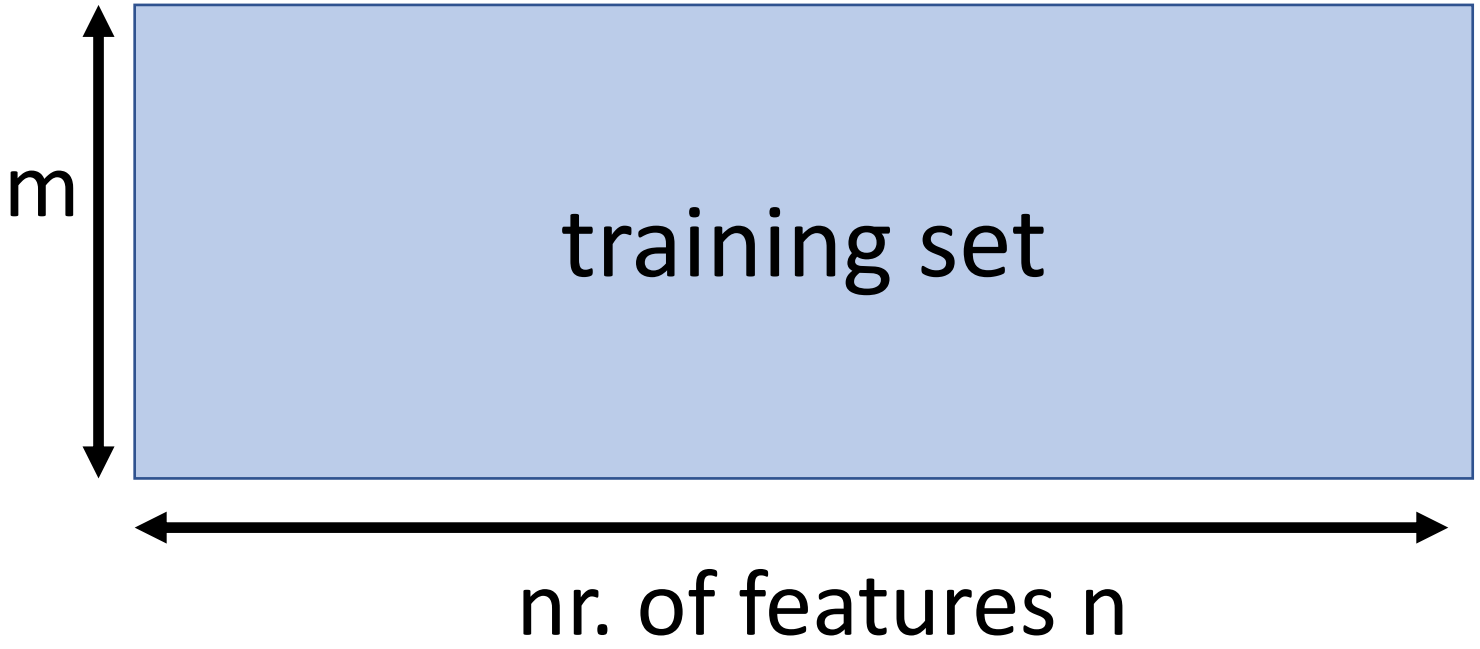
Statistical Challenge

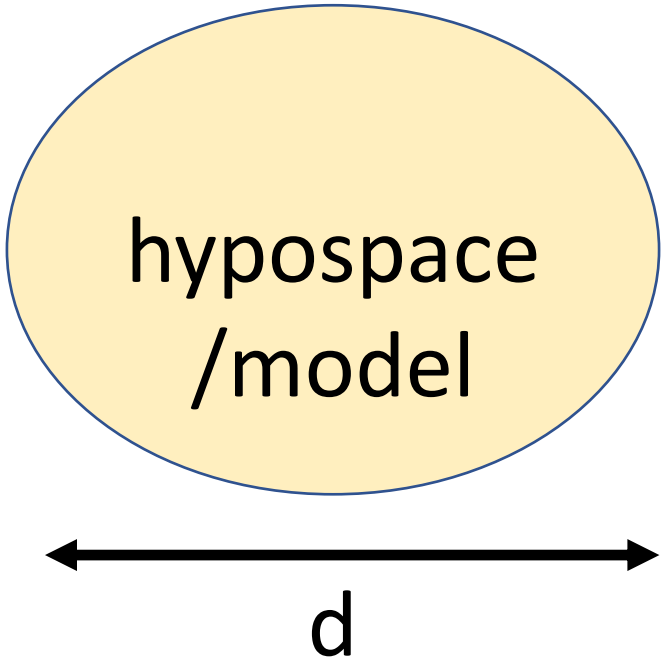
- effective dimension d of hypothesis space
- d typically increases with larger nr. of features
- overfitting is likely when d > sample size m

Linear Regression

- consider data points with n features
- have m labeled data points
- for m < n, plain linear regression will overfit
- n might be billions (e.g., HD-movies)
- would need billions of labeled data points

Data and Model Size





overfitting as soon as
$d/m > 1$!

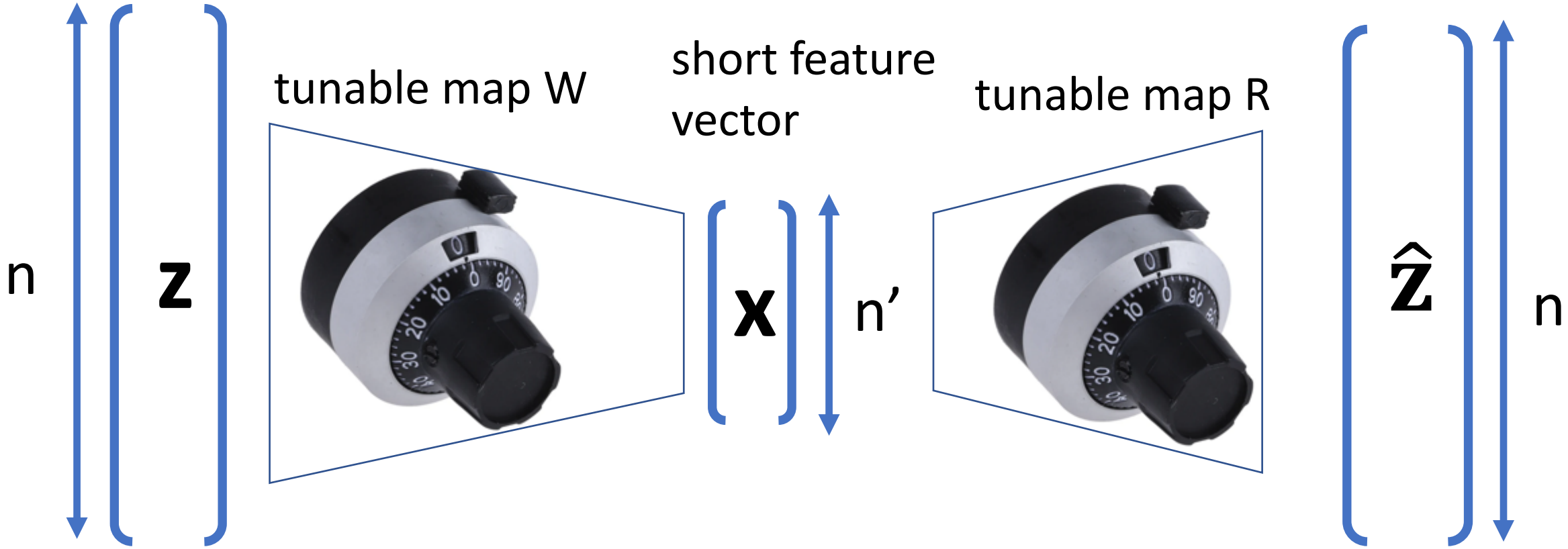
Computational Challenge

nr. of arithmetic operations required to find a good hypothesis (by minimizing training error) scales with number of features !

Basic Idea of Feature Learning

raw feature vector

reconstruction

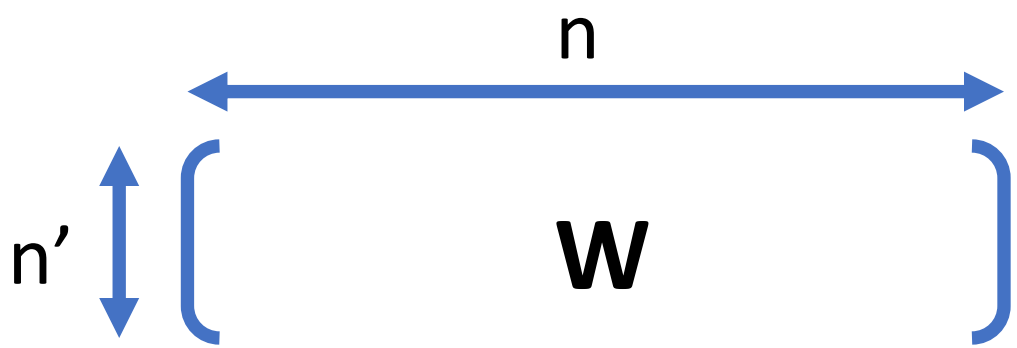


choose (learn) W and R to minimize reconstruction error $\mathbf{z} - \hat{\mathbf{z}}$

Linear Feature Learning

- use linear maps for compression and reconstruction

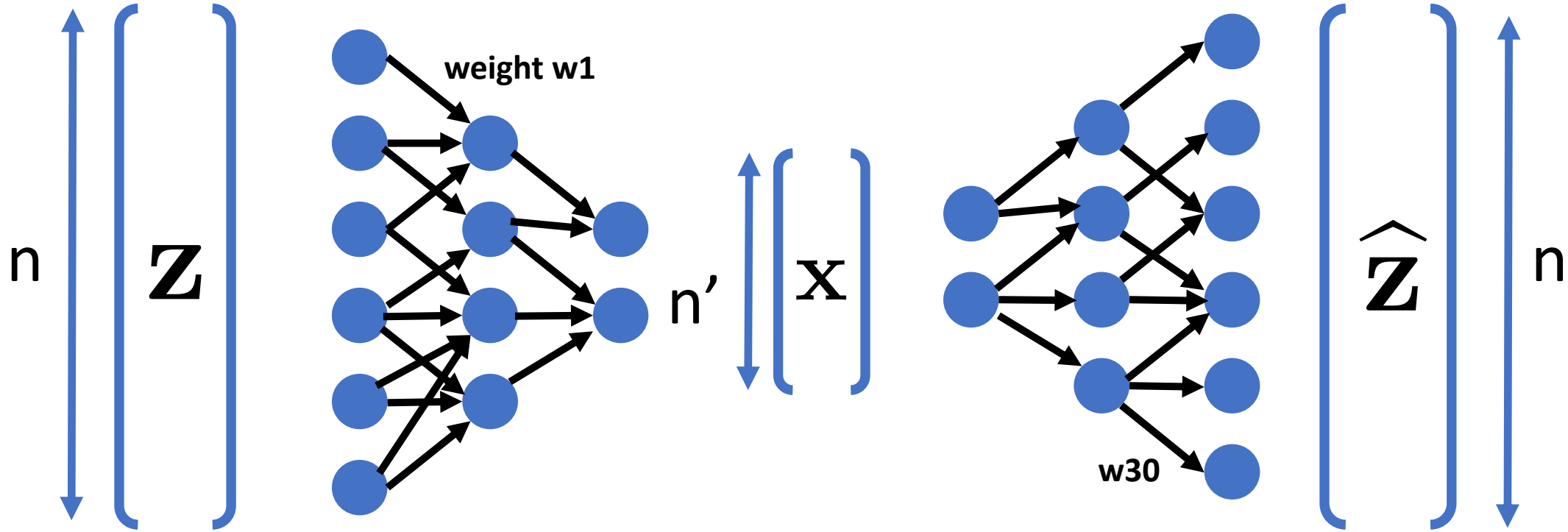
$$\mathbf{x} = \mathbf{W}\mathbf{z} \qquad \hat{\mathbf{z}} = \mathbf{R}\mathbf{x}$$



choose matrices W and R to minimize $\mathbf{z} - \hat{\mathbf{z}} = \left(\mathbf{I} - \mathbf{R}\mathbf{W}\right)\mathbf{z}$

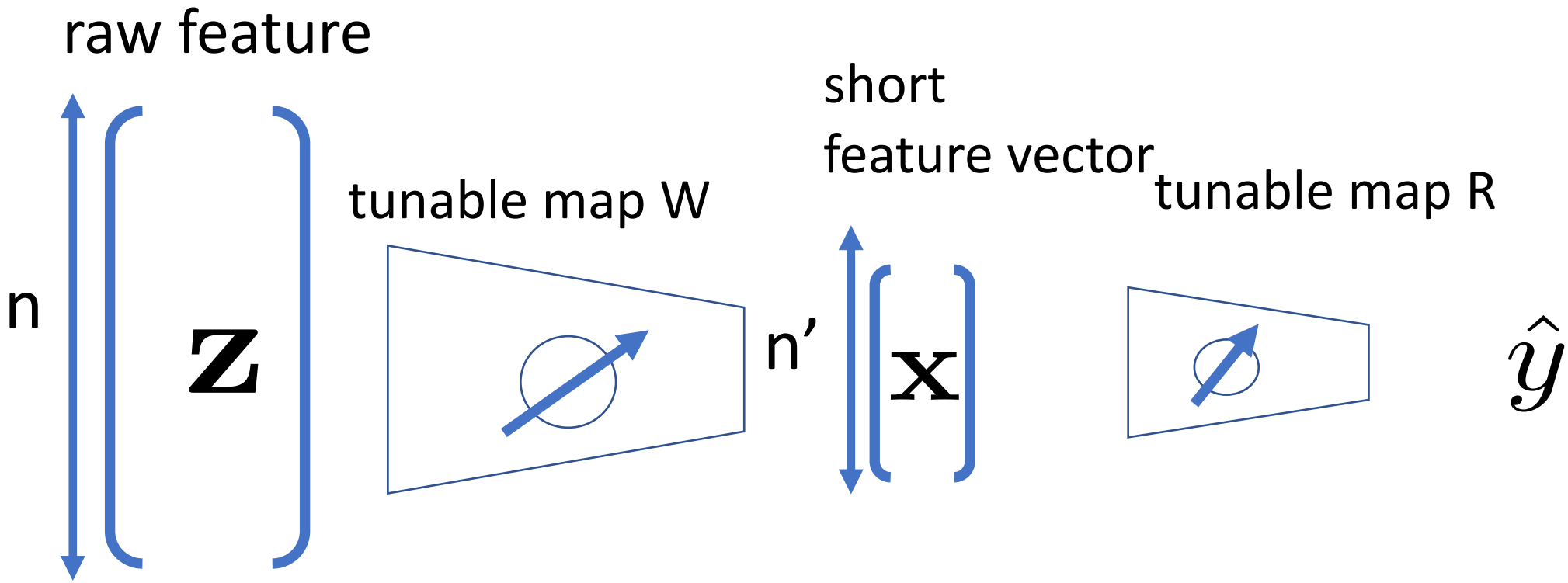
Non-Linear Feature Learning (“Autoencoder”)

use artificial neural networks for compression and reconstruction



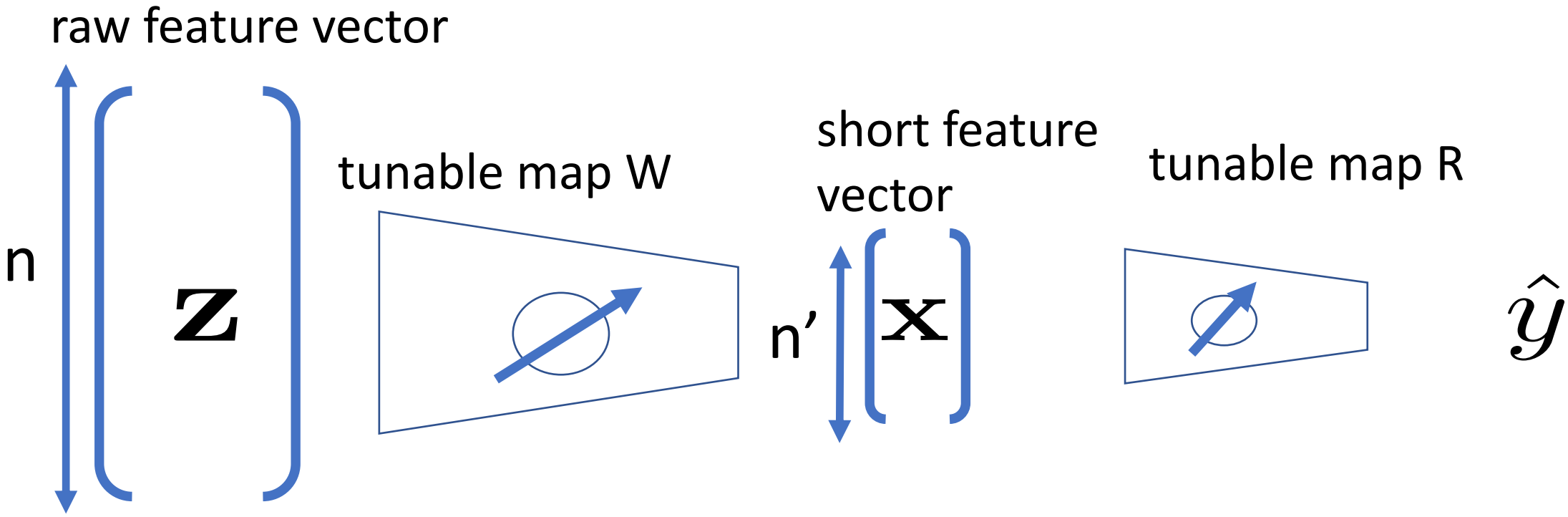
much like linear maps, ANNs are just **parametrized maps**!

Feature Learning for Labeled Data



choose W such that we can predict (using some map R) the label y from z with maximum accuracy

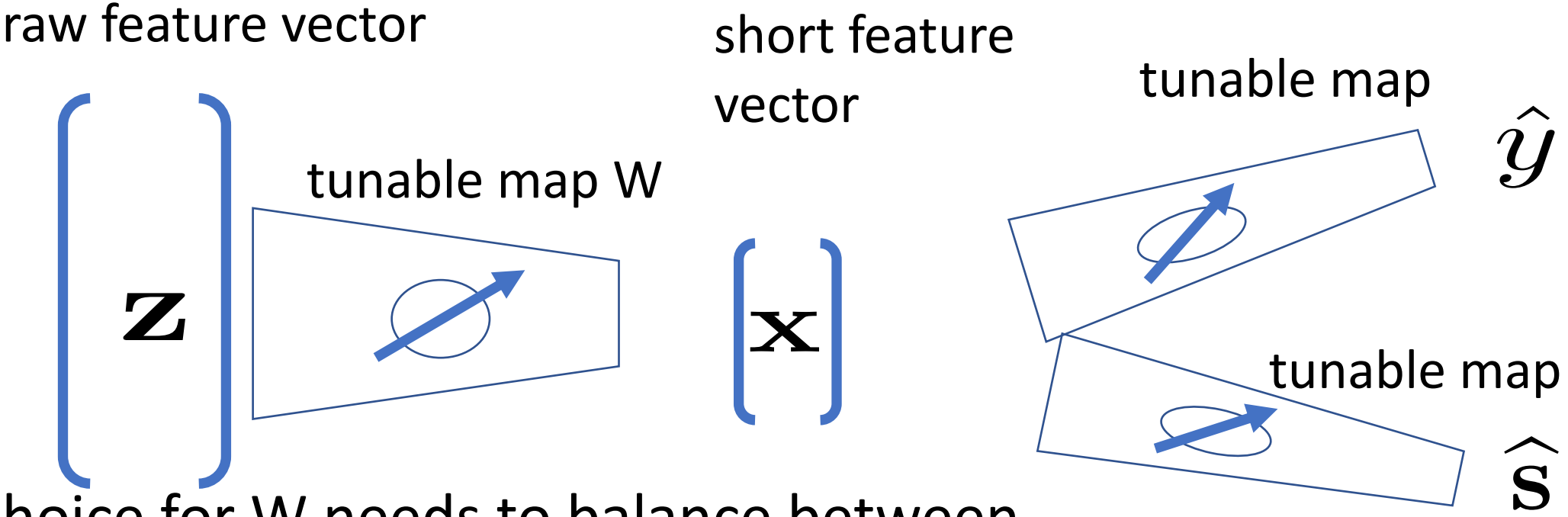
Feature Learning for Labeled Data



choice for W needs to balance between

- compressing raw feature vector as much as possible
- keep parts of raw features that are relevant for predicting y

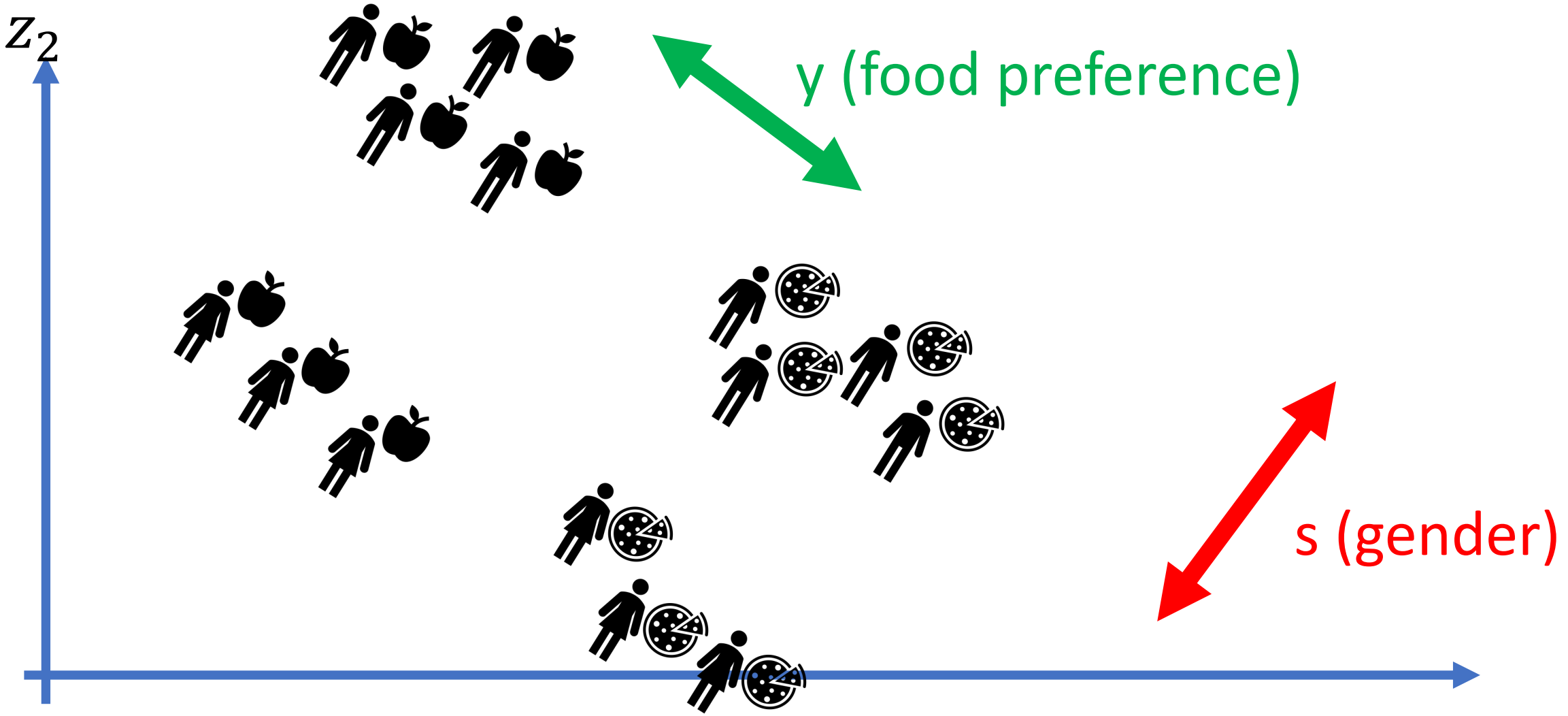
Privacy-Preserving Feature Learning



choice for W needs to balance between

- compressing raw feature vector as much as possible
- keep parts of raw features that are relevant for predicting y
- predicting private variable ”s” is not predictable from x

Privacy-Preserving Feature Learning

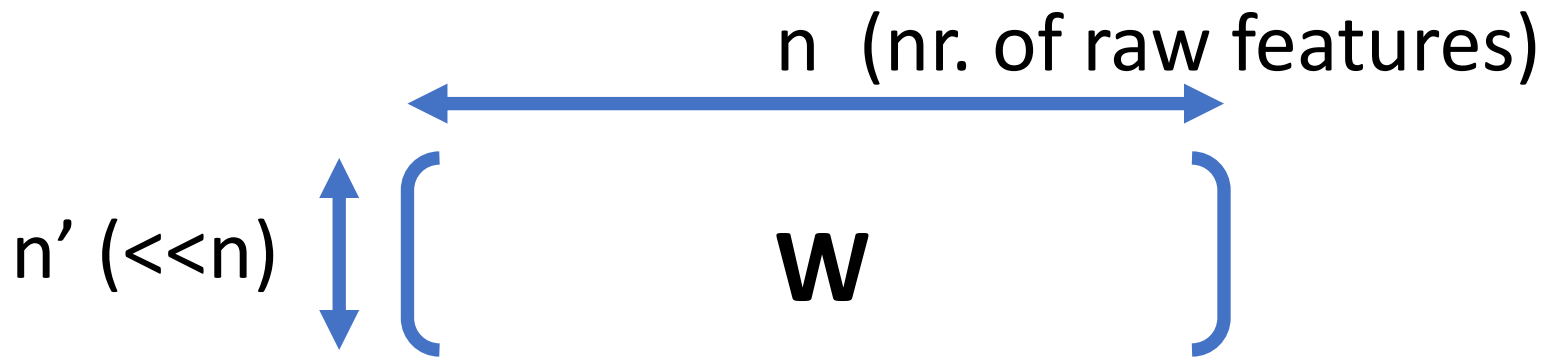


Principal Component Analysis (PCA)

Linear Feature Learning

- use linear maps for compression and reconstruction

$$\mathbf{x} = \mathbf{W}\mathbf{z} \qquad \hat{\mathbf{z}} = \mathbf{R}\mathbf{x}$$



choose matrices W and R to minimize $\mathbf{z} - \hat{\mathbf{z}} = \left(\mathbf{I} - \mathbf{R}\mathbf{W}\right)\mathbf{z}$

PCA as Risk Minimization

- m datapoints with raw feature vecs $\mathbf{z}^{(1)}, \cdots, \mathbf{z}^{(m)}$

  choose matrices **W, R** to minimize reconstruction error

$$\mathcal{E}(\mathbf{W}, \mathbf{R}) = \frac{\mathbf{1}}{\mathbf{m}} \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}} \left\| \mathbf{z}^{(\mathrm{i})} - \mathbf{R}\mathbf{W}\mathbf{z}^{(\mathrm{i})} \right\|^{\mathbf{2}}$$

Principal Component Analysis (PCA)

- optimal compression matrix $\mathbf{W} = \left(\mathbf{u}^{(1)}, \cdots, \mathbf{u}^{(\mathrm{n}')}\right)$

- using "top" eigenvectors $\boldsymbol{u}^{(i)}$ of sample covariance matrix

$$\widehat{\mathbf{C}} = (1/m) \sum_{i=1}^{m} \mathbf{z}^{(i)} \left(\mathbf{z}^{(i)}\right)^T$$

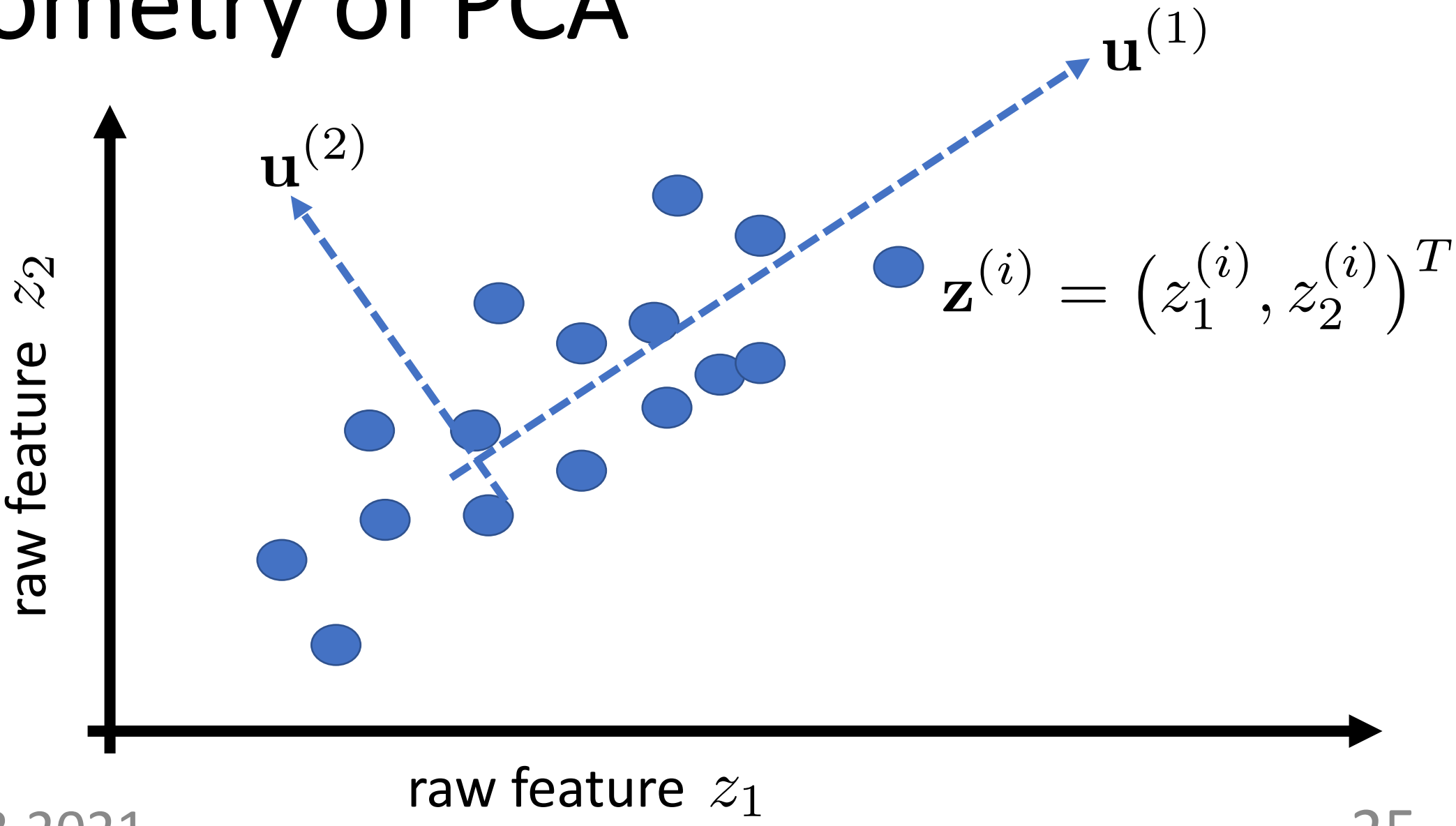
- eigenvalue decomposition of psd sample cov. matrix:

$$\widehat{\mathbf{C}} = \left(\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(n)}\right) \mathrm{diag}\left(\lambda_1, \ldots, \lambda_n\right) \left(\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(n)}\right)^T$$

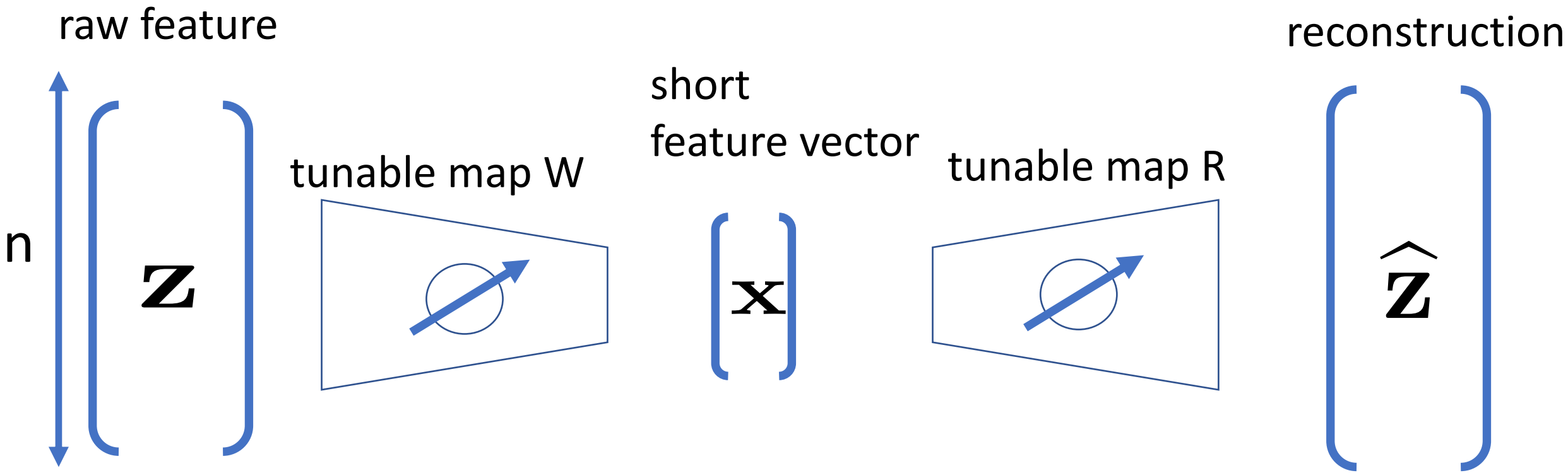
non-negative eigenvalues

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0.$$

Geometry of PCA



Computational Complexity



choose (learn) W and R to minimize reconstruction error $\mathbf{z} - \hat{\mathbf{z}}$

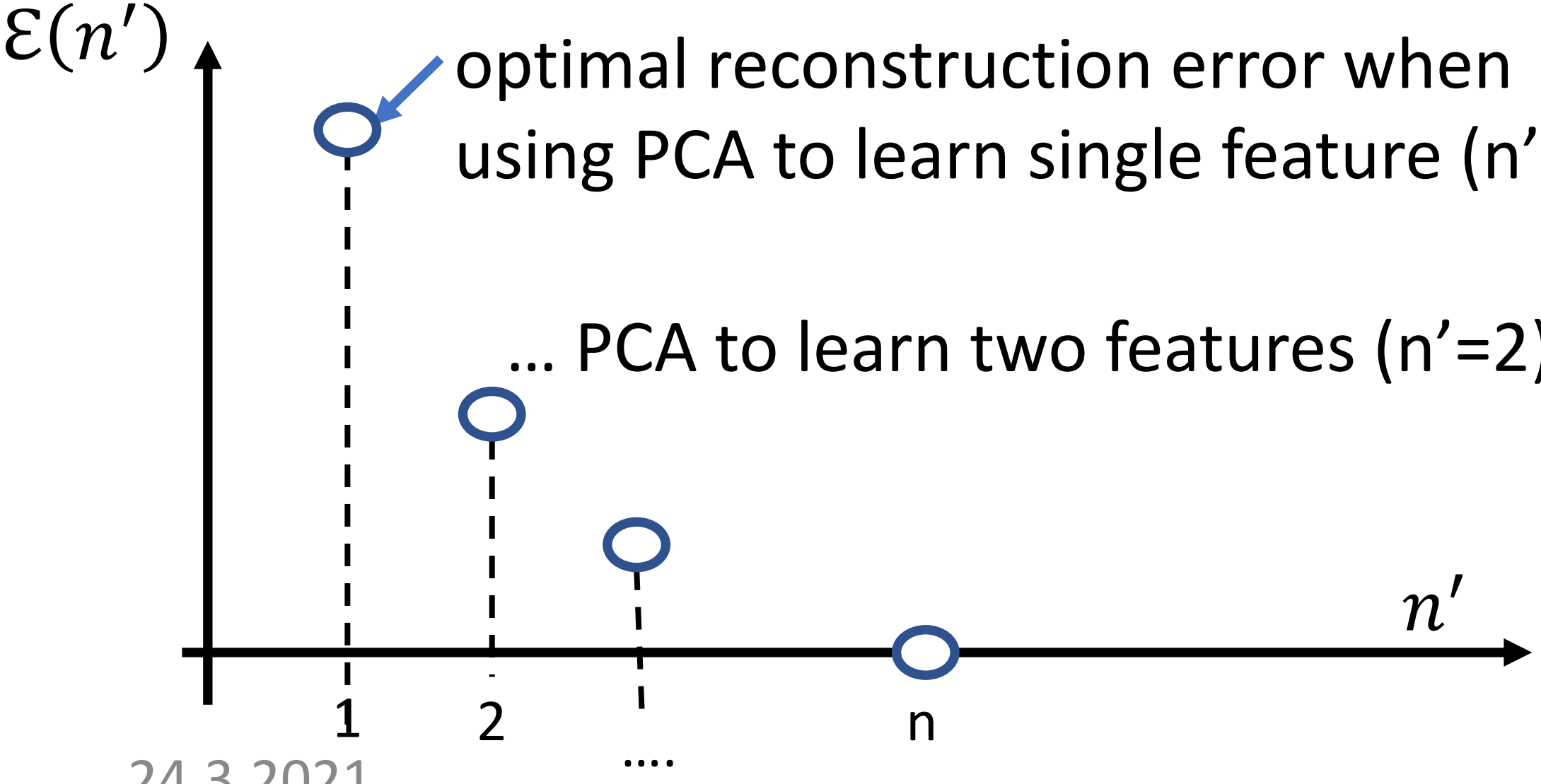
PCA requires eigenvalue decomposition of “n x n” matrix!

How to choose n’ ?

- PCA requires number n’ of learned features as input
- n’=2 for visualization (scatter plot)
- chose n’ to balance compression with optimal reconstruction error

$$\mathcal{E}(n') = \min_{\substack{\mathbf{W}\in\mathbb{R}^{n'\times n} \\ \mathbf{R}\in\mathbb{R}^{n\times n'}}} \mathcal{E}(\mathbf{W},\mathbf{R})$$

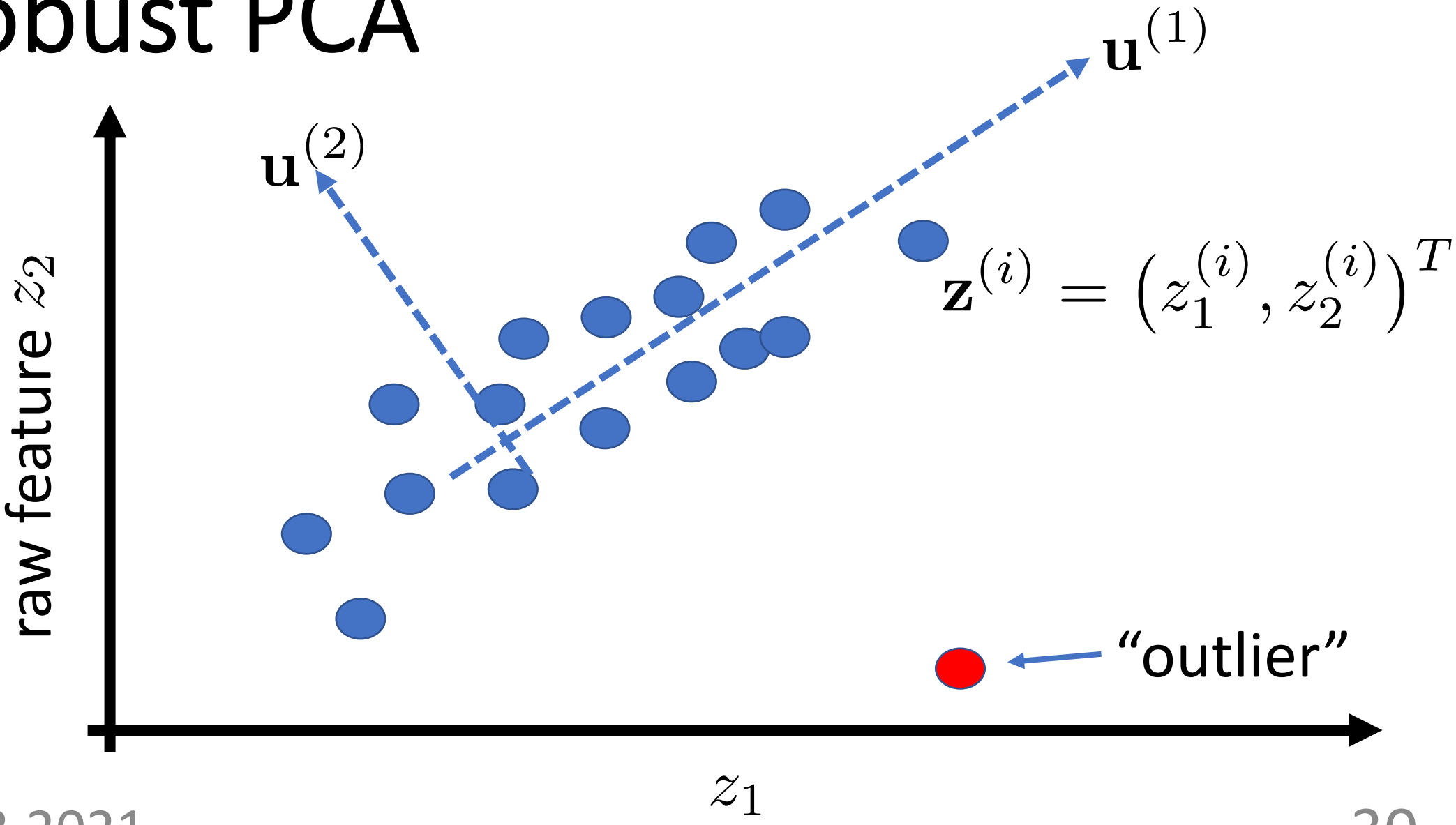
Elbow Method



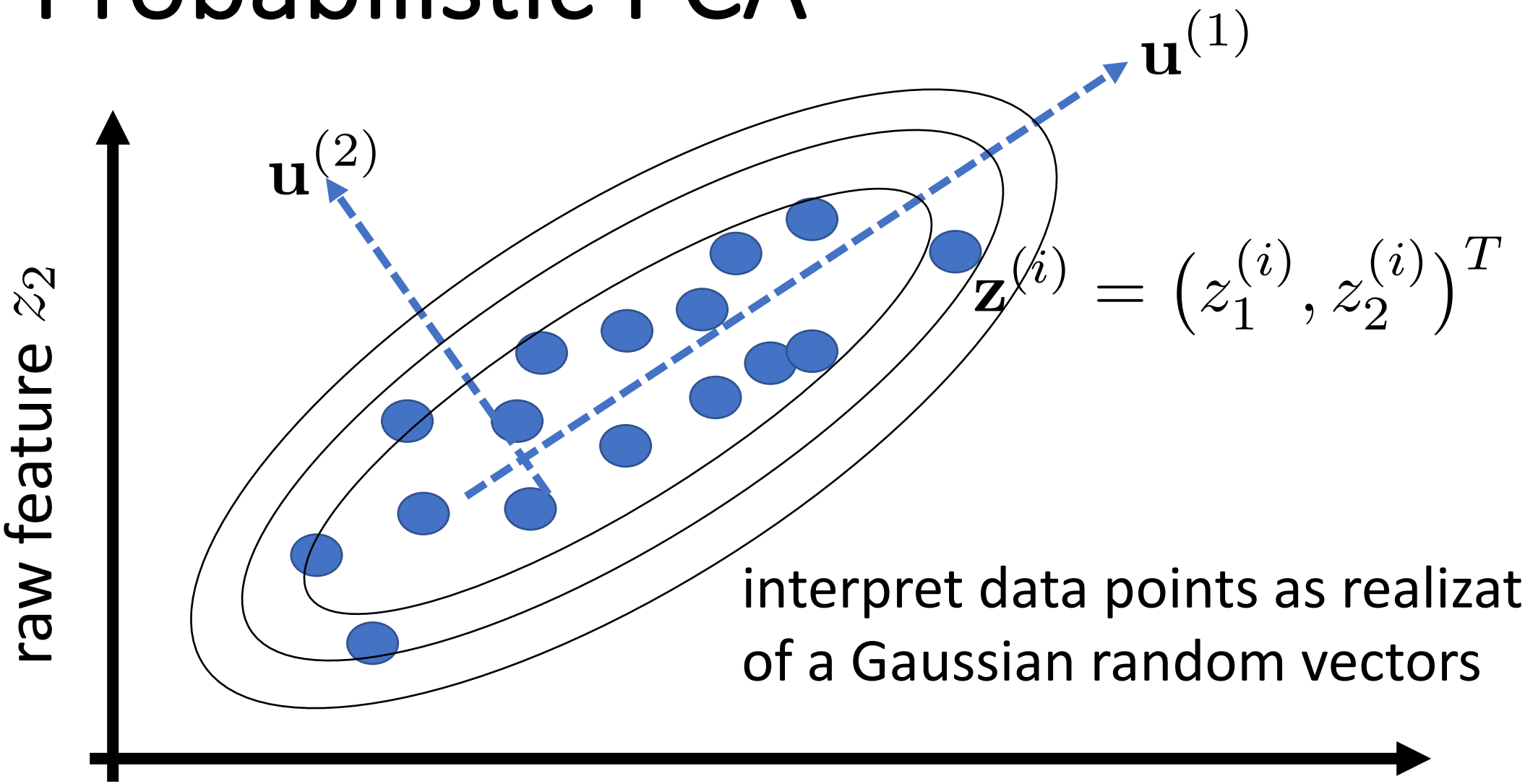
Variations of PCA

- robust PCA: uses a different measure of reconstruction error
- probabilistic PCA: uses a statistical model for data points
- sparse PCA: new features depend on few raw features

Robust PCA

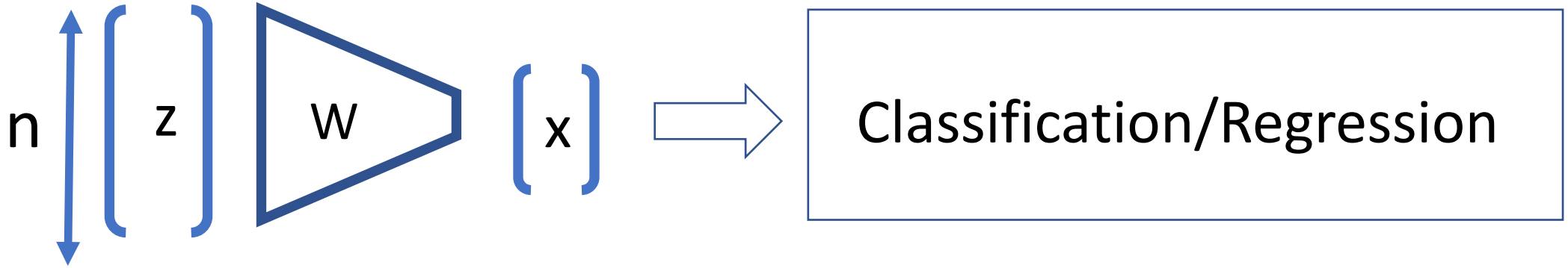

$\mathbf{u}^{(1)}$
$\mathbf{u}^{(2)}$
$\mathbf{z}^{(i)} = \left(z_1^{(i)}, z_2^{(i)}\right)^T$
raw feature z_2
"outlier"
z_1

Probabilistic PCA



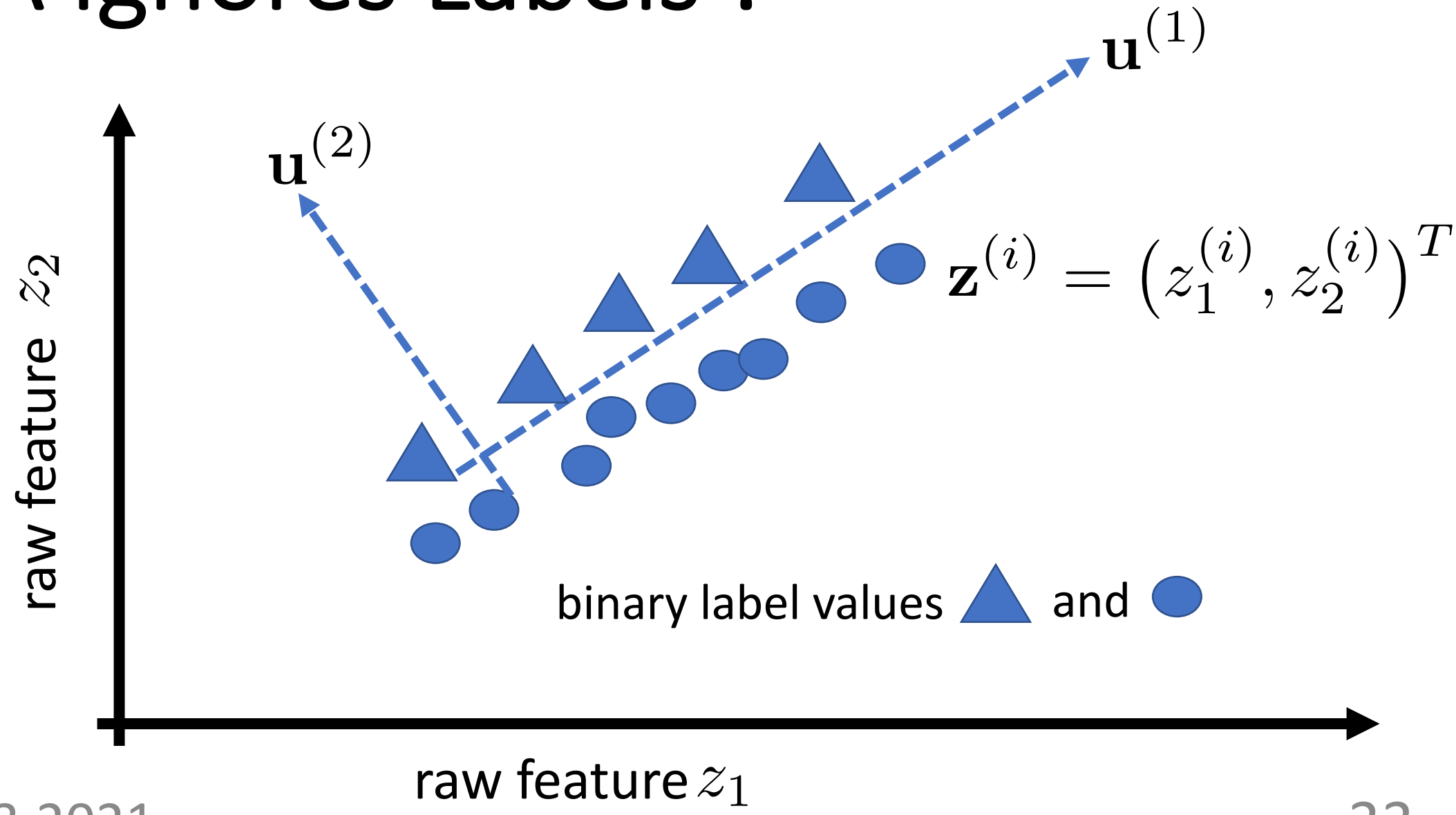
interpret data points as realizations of a Gaussian random vectors

PCA as Pre-Processing

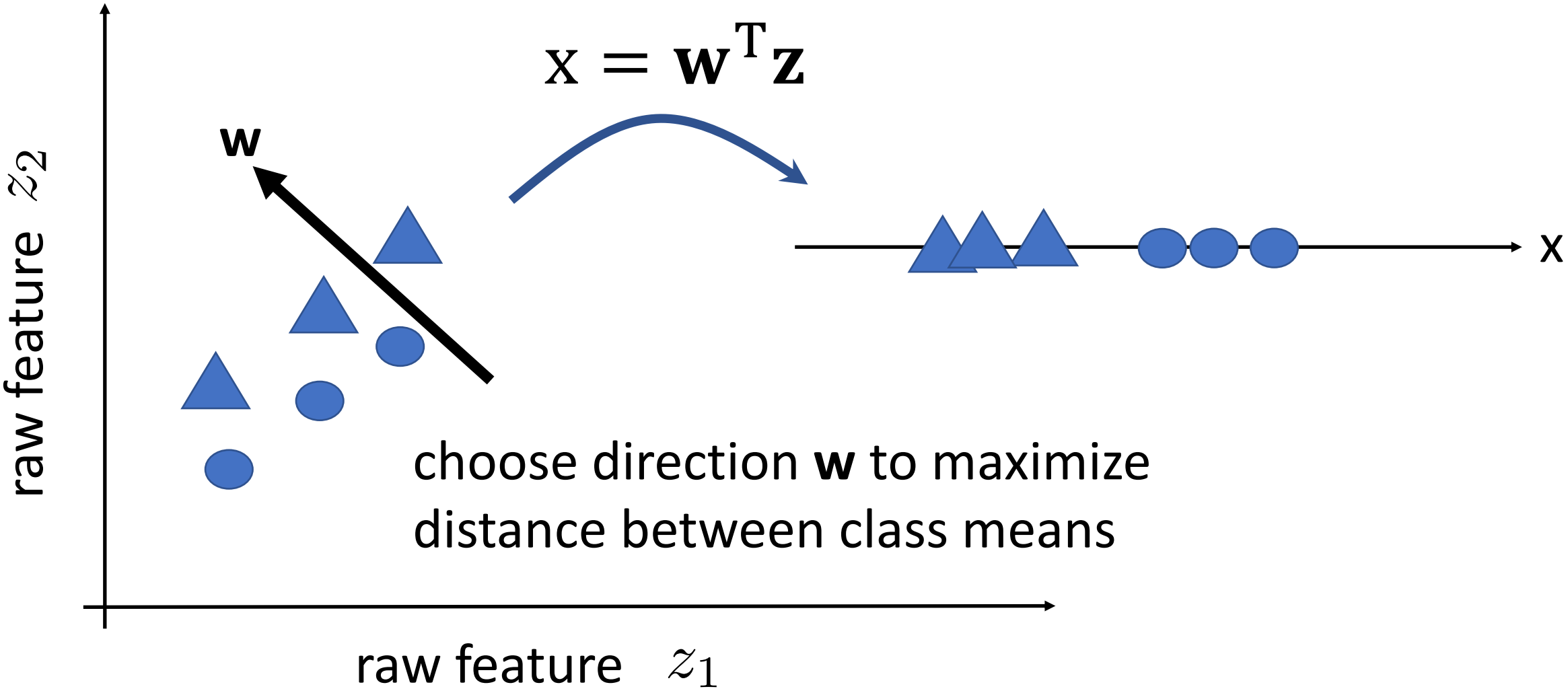


- PCA delivers a compression matrix W
- replace (long) raw features z with shorter features x =Wz
- apply regression/classification methods to new features x
- CAUTION: PCA ignores label information!

PCA Ignores Labels !



Fisher's Linear Discriminant



Random Projections

- consider random projection **x**= **W z**
- entries of matrix **W** are randomly chosen
- no learning/tuning of **W** required!
- in many settings, works surprisingly well
- known as compressed sensing

So What?

- feature learning methods determine relevant features
- learning two features allows to scatter plot !
- optimal linear feature learning = PCA
- PCA ignores label information!
- random projections as computationally light alternative

Questions?