AI in Healthcare

Feature Learning

Alex Jung

Assistant Professor for Machine Learning

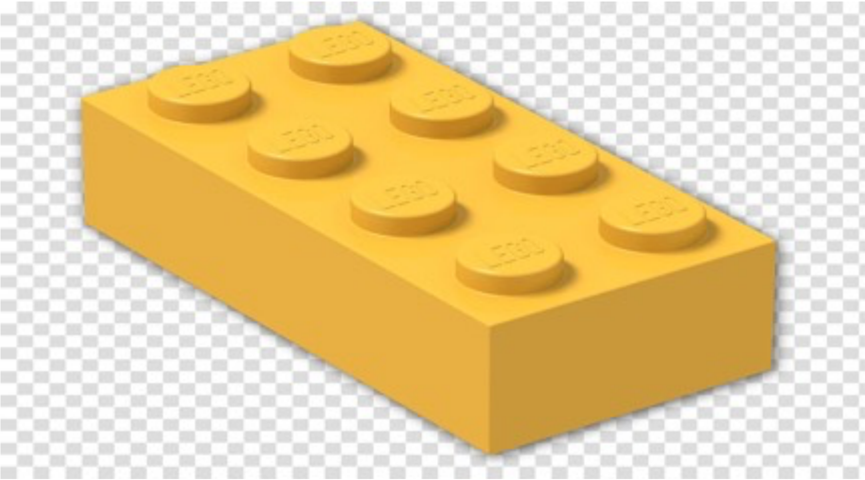
Dept. Computer Science, Aalto University, Finland

Learning Goals of This Lecture

- 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

Quick Refresher

Components
of
Machine Learning

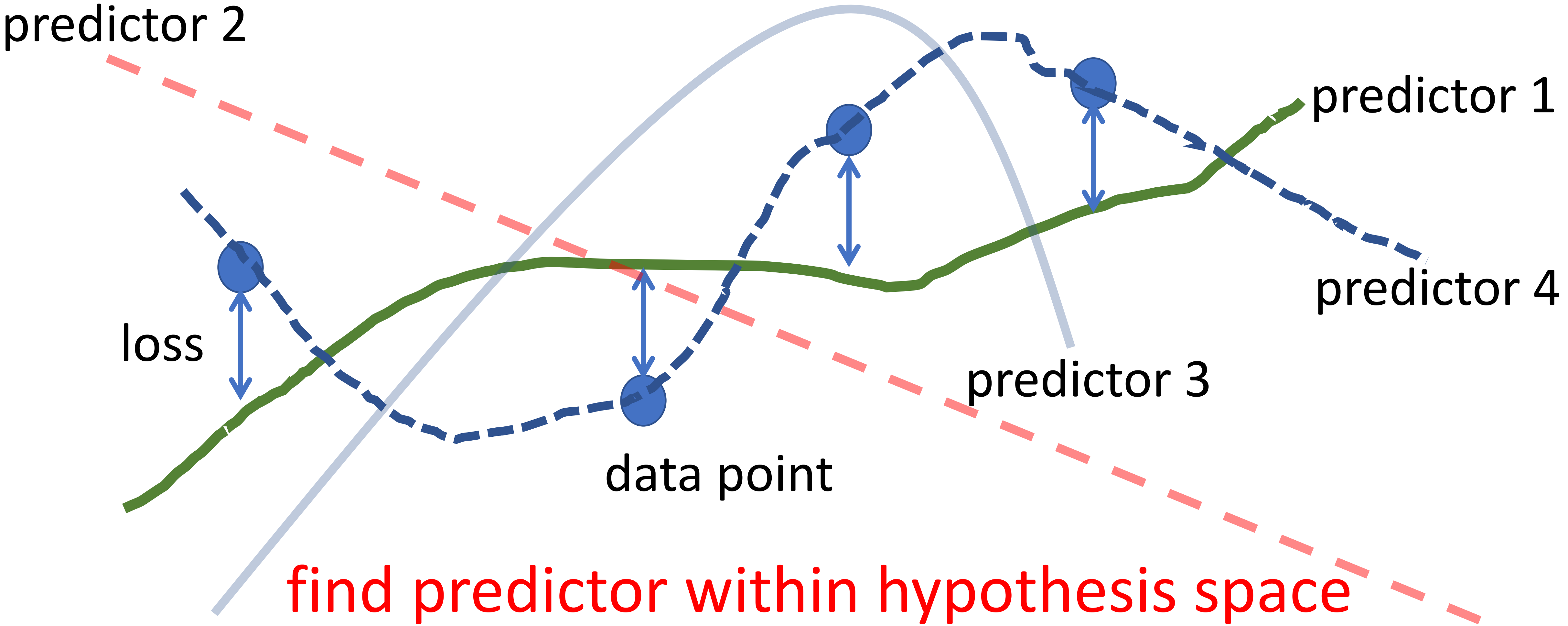


data: features, labels

loss function

hypothesis space

Machine Learning ≈ Fitting Models to Data

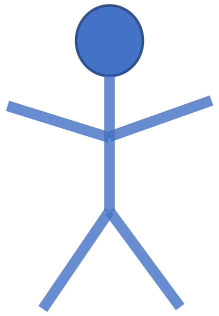


find predictor within hypothesis space such that average loss is smallest

Predicting Risk of Needing ICU

- data point = “some person/patient”
- features = health records, mobile phone data, social media activity, photos showing the person, …. TONS OF FEATURES!
- label = risk (percent) of needing ICU during next week

ICU Need Predictor



stack all features of person into a vector

$$\mathbf{x} = \left(x_1, \ldots, x_n\right)^T \in \mathbb{R}^n$$

label (risk of needing ICU during next week) y

try a linear predictor (hypothesis space given by linear functions):

$$\hat{y} = \mathbf{w}^T \mathbf{x}$$

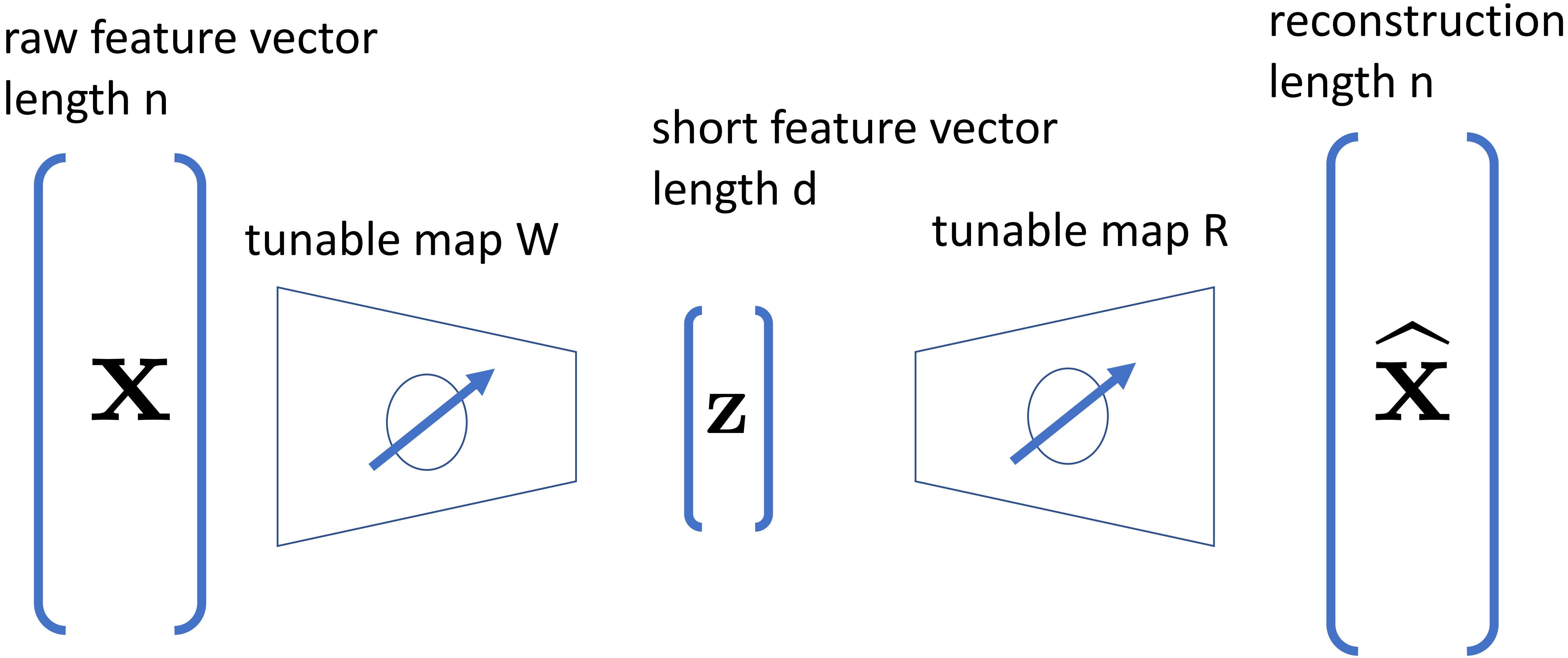
note that this predictor is parametrized by the weight vector:

$$\mathbf{w} = \left(w_1, \ldots, w_n\right)^T \in \mathbb{R}^n$$

ICU Predictor

- assume we have n=1000000 features of a patient
- how many training examples do we need ?
- how can we visualize patients based on features ?
- what are most relevant features ?
- which features do not violate privacy protection?

Basic Idea of Feature Learning

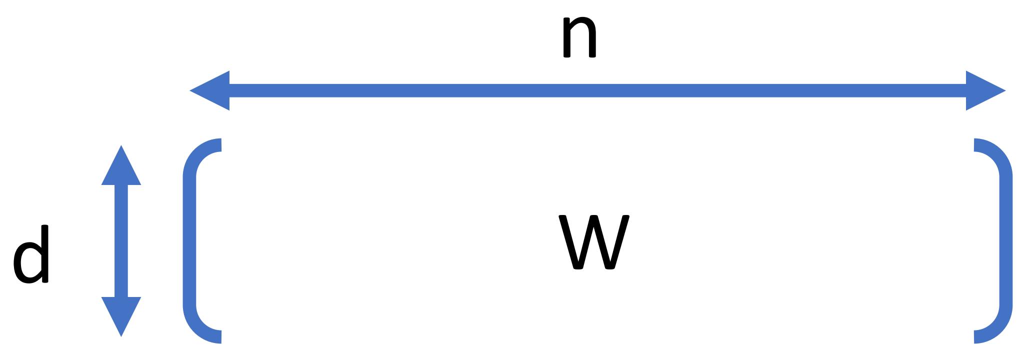


choose (learn) W and R to minimize reconstruction error $\mathbf{x} - \hat{\mathbf{x}}$

Linear Feature Learning

- use linear maps for compression and reconstruction

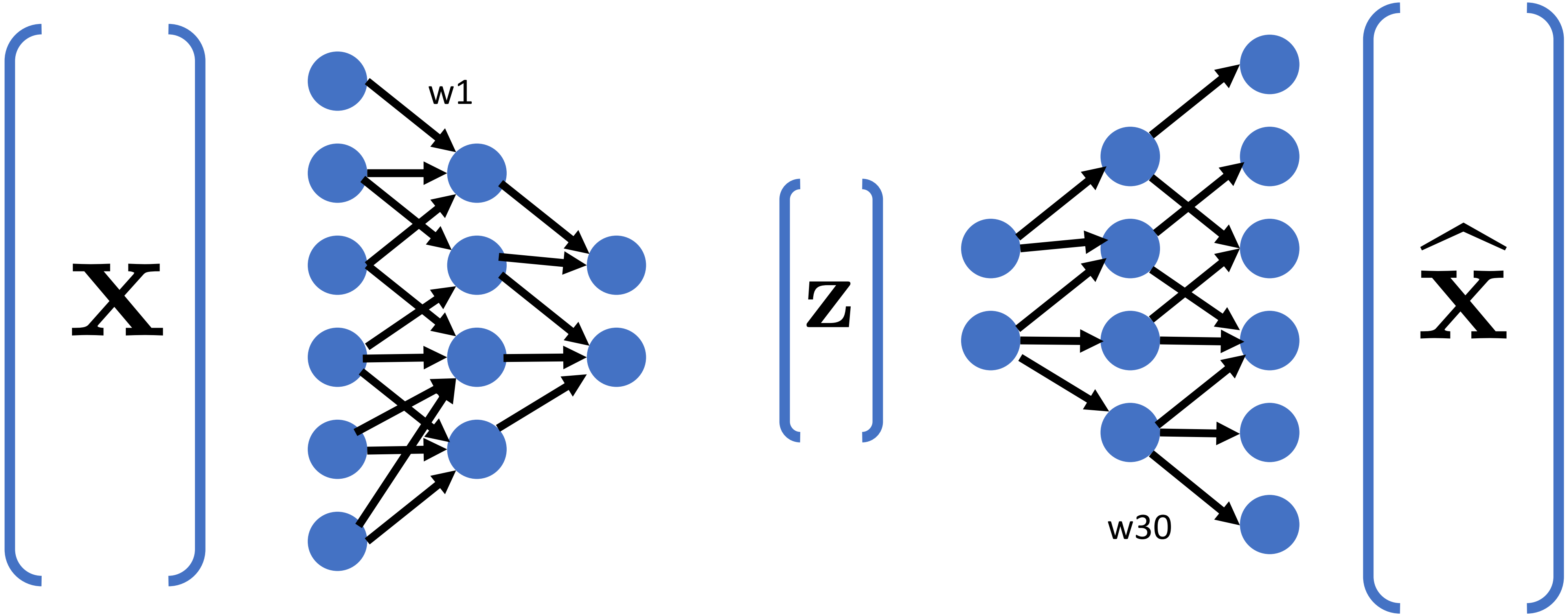
$$\mathbf{z} = \mathbf{W}\mathbf{x} \qquad \widehat{\mathbf{x}} = \mathbf{R}\mathbf{z}$$



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

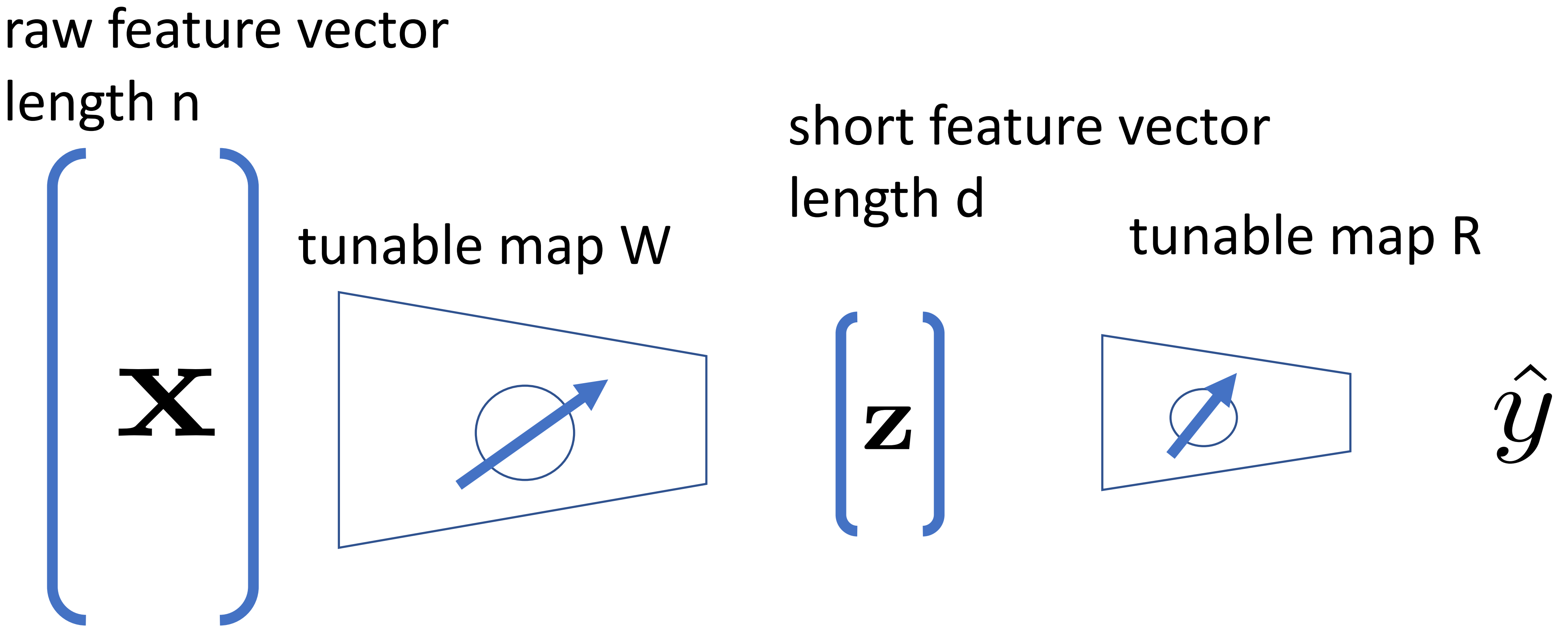
Autoencoder

use artificial neural networks for compression and reconstruction



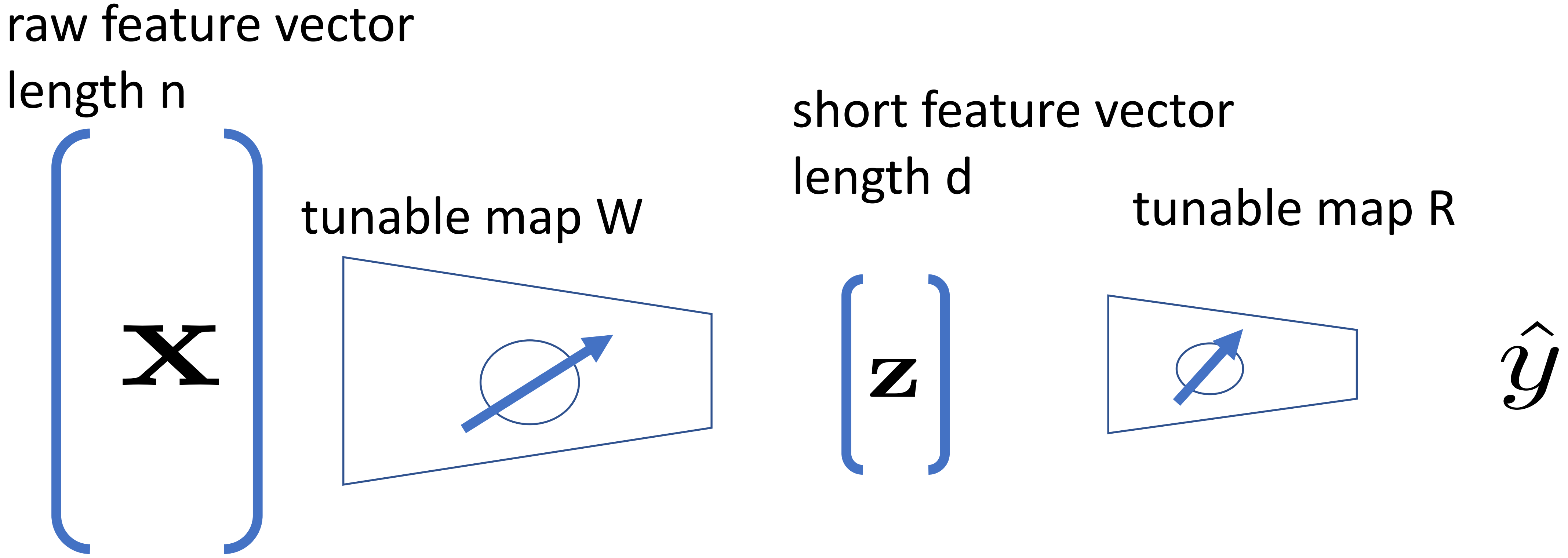
ANNs are just **parametrized maps** (like linear maps)!

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

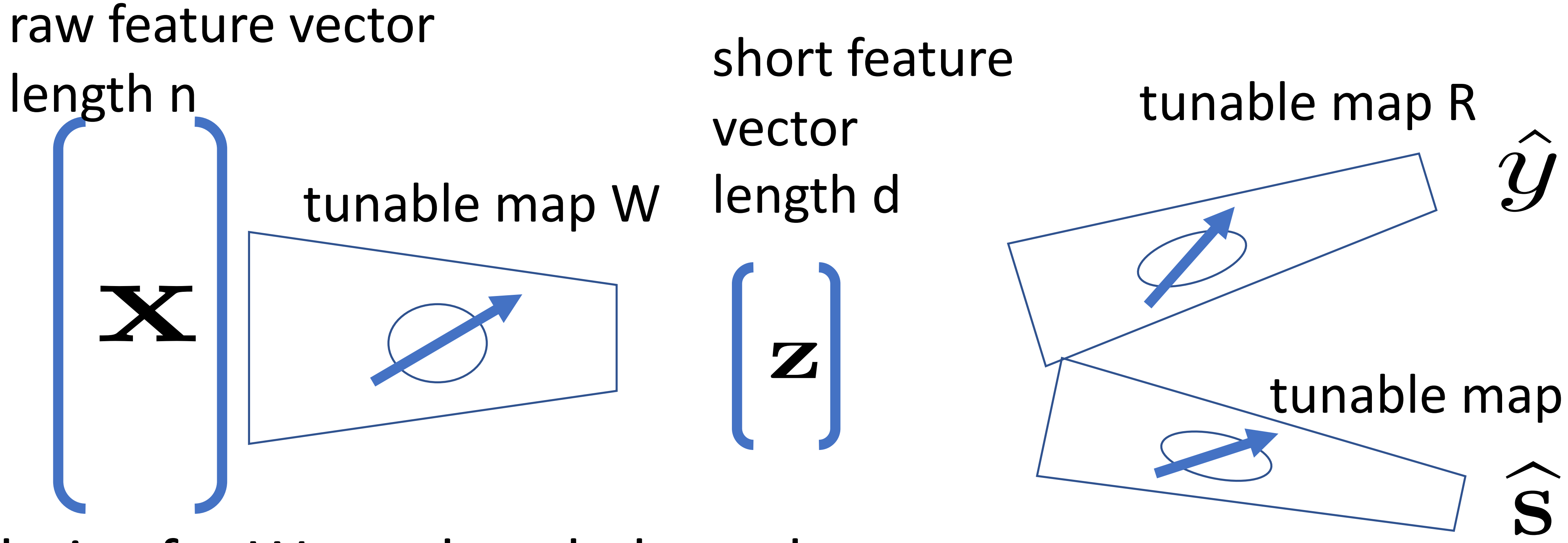
Information Theoretic Approach (advanced!)

- assume probabilistic model for data
- quantify compression by mutual information I(x;z)
- quantify relevance of z for label y by I(z;y)
- choose map W from x to z via information bottleneck

$$\min_W I(z;x) - \beta I(z;y)$$

tuning parameter β trades compression against prediction accuracy

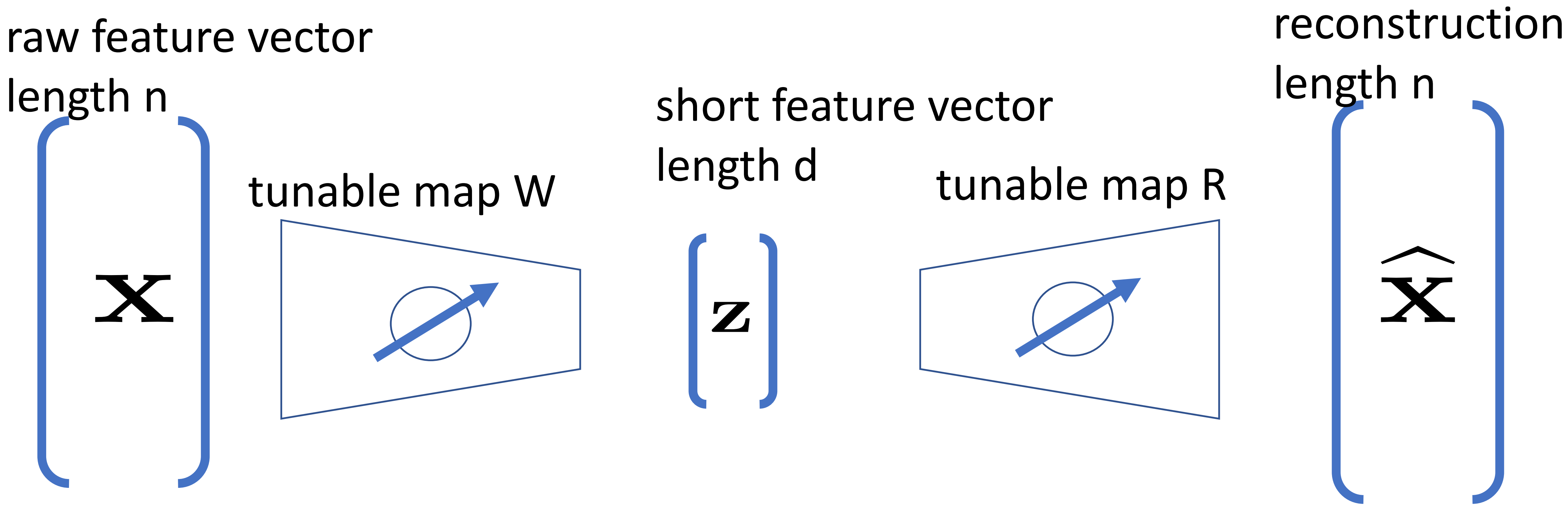
Privacy Preserving Feature Learning



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 z

Computational Complexity



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

amounts to a (typically difficult) optimization problem !

Random Projections

- consider random projection $z = W x$
- entries of matrix W are i.i.d. random variables (Gaussian,...)
- no learning/tuning of W involved (computationally cheap)
- in surprisingly many settings, this works quite well
- amounts to compressed sensing

Further Reading

- Chapter 9 of https://arxiv.org/pdf/1805.05052.pdf
- Chapter 14.5 of https://web.stanford.edu/~hastie/Papers/ESLII.pdf
- J.P. Cunningham, Z. Ghahramani "**Linear Dimensionality Reduction: Survey, Insights, and Generalizations".** http://www.jmlr.org/papers/v16/cunningham15a.html
- G. Chechik et.al., "**Information Bottleneck for Gaussian Variables**", https://papers.nips.cc/paper/2457-information-bottleneck-for-gaussian-variables

Principal Component Analysis

Linear Feature Learning

- consider data set with raw features $\mathbf{x}^{(i)}, i = 1, \ldots, m$
- compress raw vectors to short vectors $\mathbf{z}^{(i)} = \mathbf{W}\mathbf{x}^{(i)}$
- how well can we reconstruct x from z ?
- optimal (minimal) reconstruction error

$$\mathcal{E}(\mathbf{W}) = \min_{\mathbf{R} \in \mathbb{R}^{n \times d}} \sum_{i=1}^{m} \|\mathbf{R}\mathbf{z}^{(i)} - \mathbf{x}^{(i)}\|_2^2$$

Massaging the Cost Function

- optimal (minimal) reconstruction error

$$\mathcal{E}(\mathbf{W}) = \min_{\mathbf{R}} \sum_{i=1}^{m} \left\| \mathbf{R}\mathbf{z}^{(i)} - \mathbf{x}^{(i)} \right\|_2^2$$

$$= \min_{\mathbf{R}} \sum_{i=1}^{m} \left\| \mathbf{R}\mathbf{W}\mathbf{x}^{(i)} - \mathbf{x}^{(i)} \right\|_2^2$$

$$= \min_{\mathbf{R}} \sum_{i=1}^{m} \left\| \left(\mathbf{R}\mathbf{W} - \mathbf{I}\right)\mathbf{x}^{(i)} \right\|_2^2$$

Principal Component Analysis

- optimal compression matrix $\mathbf{W} = \left(\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(d)}\right)^T$

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

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

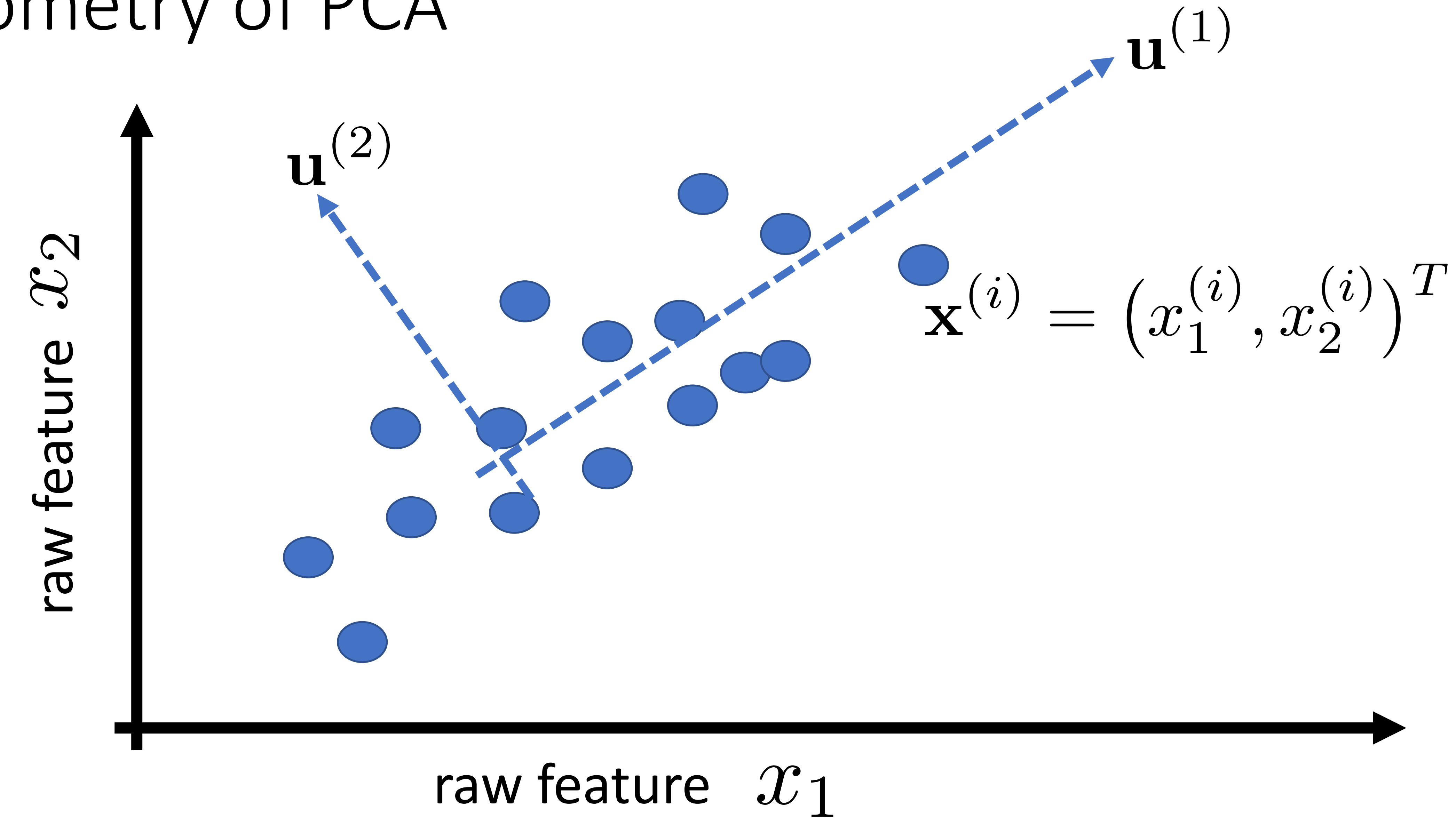
- eigenvalue decomposition of psd sample cov. matrix:

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

non-negative eigenvalues

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

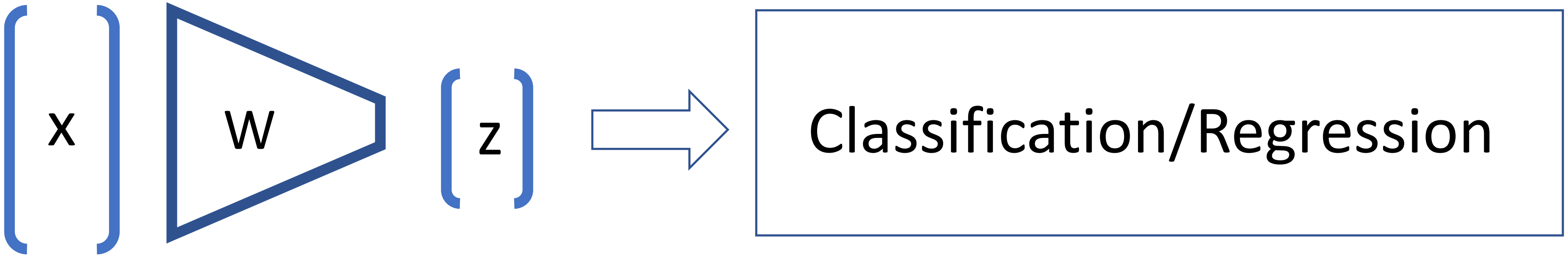
Geometry of PCA



Variations of PCA

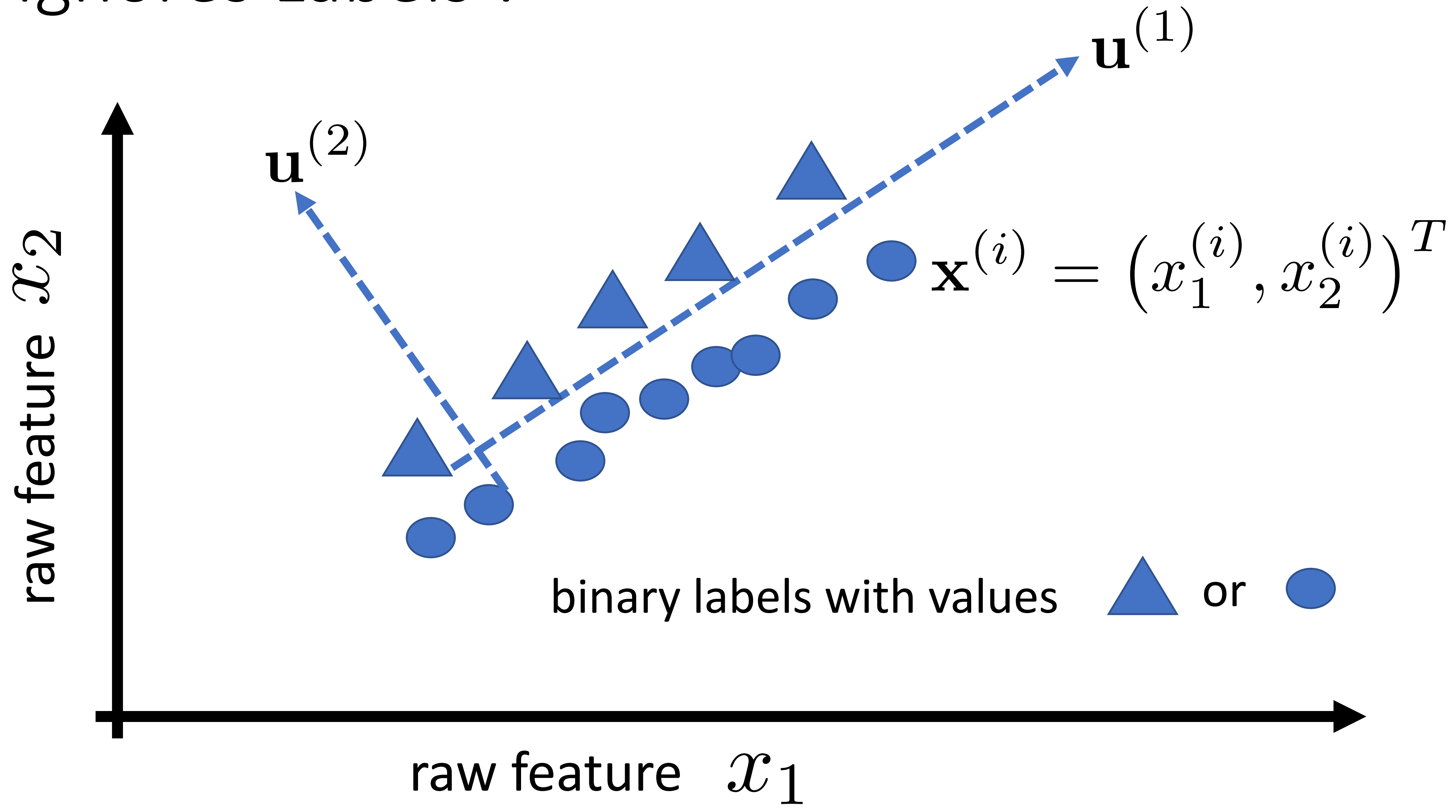
- robust PCA uses a different measure of reconstruction error
- probabilistic PCA uses a statistical model for data points
- sparse PCA requires that new features depend on few raw features

PCA as Pre-Processing for Regression/Classification



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

PCA Ignores Labels !



Thank You!