Day 9: Unsupervised learning, dimensionality reduction

Introduction to Machine Learning Summer School
June 18, 2018 - June 29, 2018, Chicago

Instructor: Suriya Gunasekar, TTI Chicago

28 June 2018

TOYOTA TECHNOLOGICAL INSTITUTE AT CHICAGO

THE UNIVERSITY OF CHICAGO

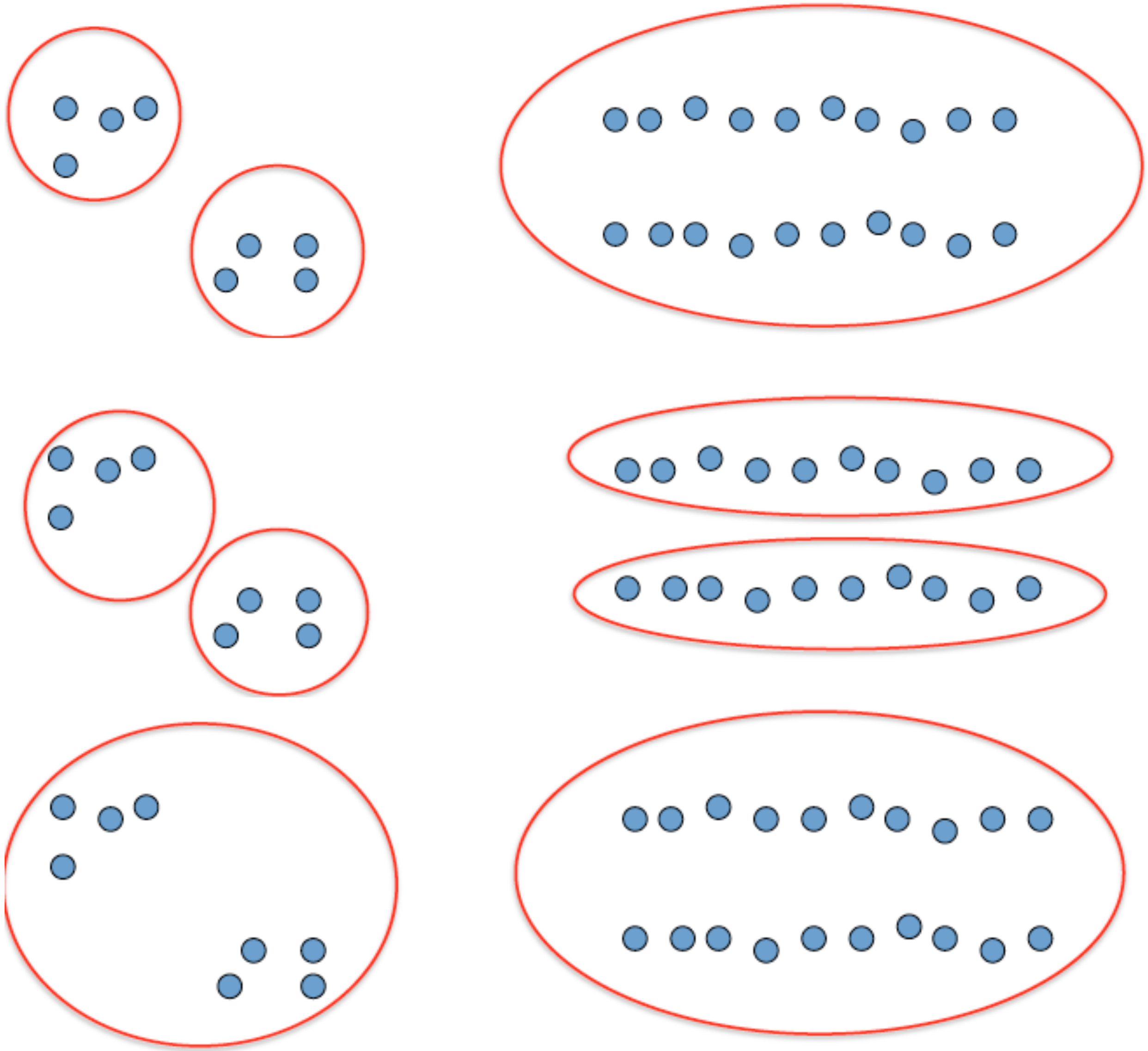
Topics so far

- Linear regression
- Classification
  - Logistic regression
  - Maximum margin classifiers, kernel trick
  - Generative models
  - Neural networks
  - Ensemble methods
- Today and Tomorrow
  - Unsupervised learning – dimensionality reduction, clustering
  - Review

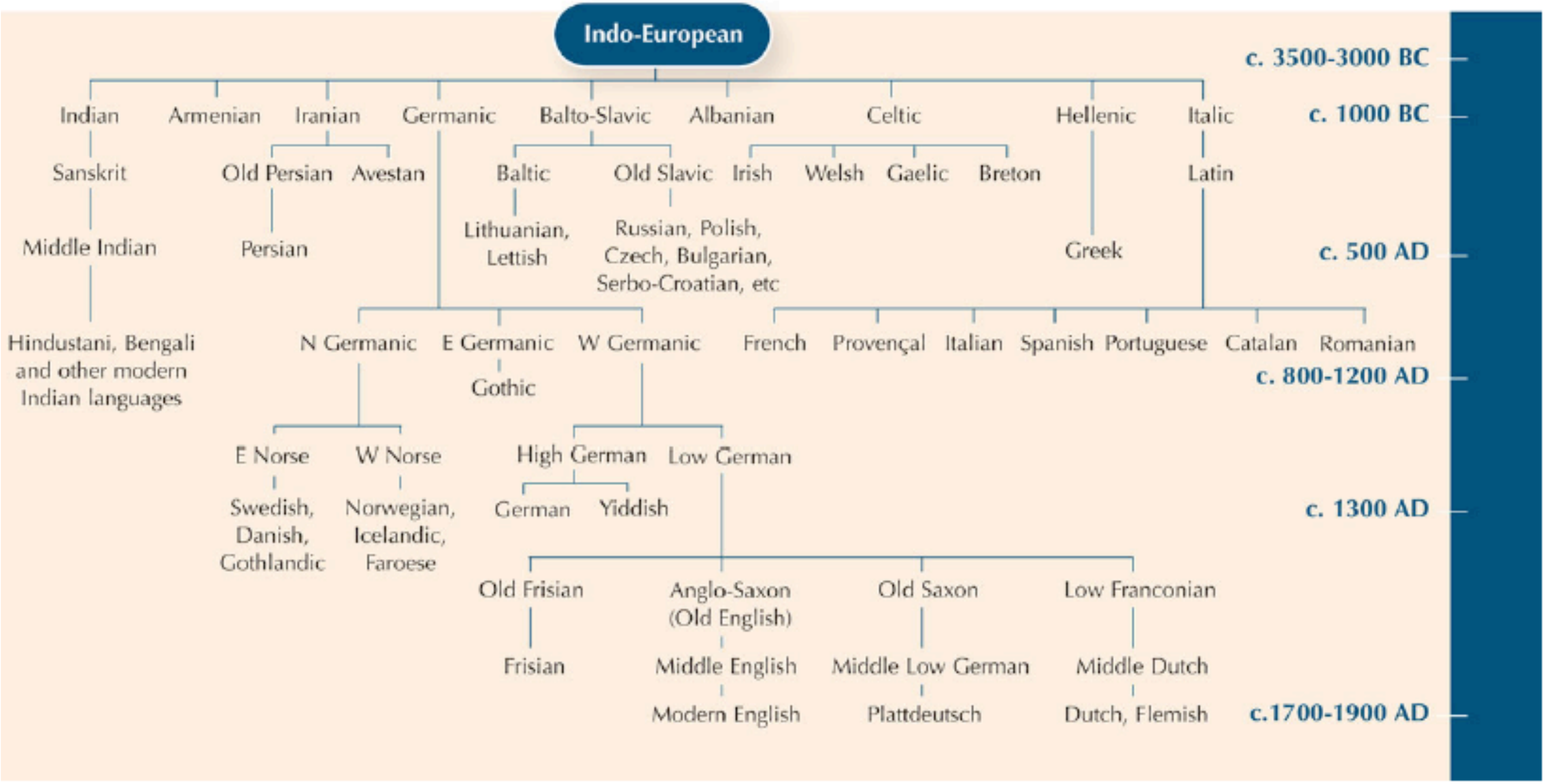
Unsupervised learning

- Unsupervised learning:
  Requires data $x \in \mathcal{X}$, but no labels
- Goal?: Compact representation of the data by detecting patterns
  - e.g. Group emails by topic
- Useful when we don't know what we are looking for
  - makes evaluation tricky
- Applications in visualization, exploratory data analysis, semi-supervised learning

Clustering



Clustering languages



[Image from scienceinschool.org]

Clustering species (phylogeny)

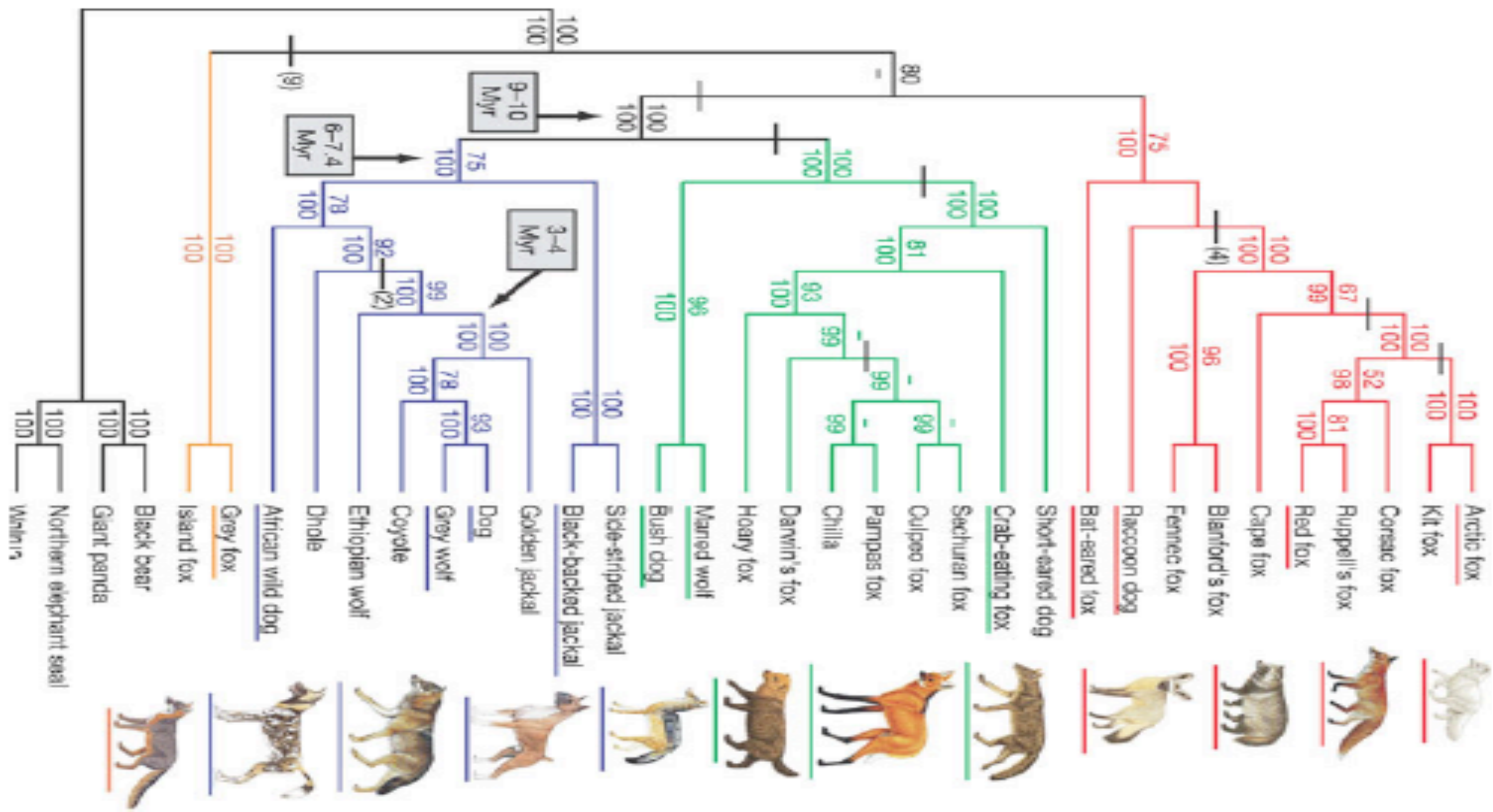
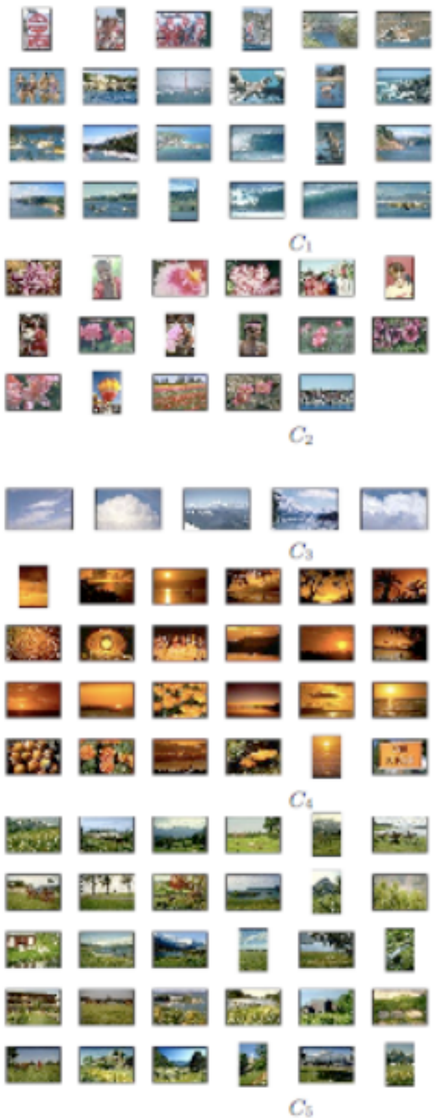


Image clustering/segmentation



[Goldberger et al.]

Goal: Break up the image into meaningful or perceptually similar regions

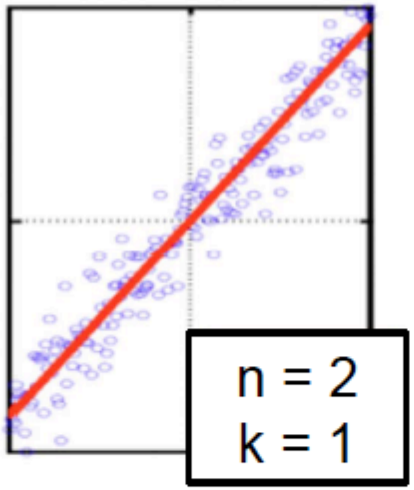
[Slide from James Hayes]

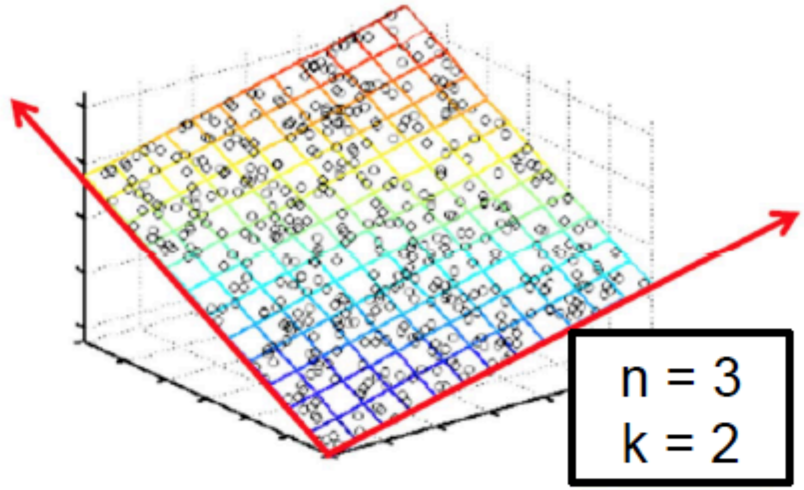
Current trend is to use datasets with labels for such task e.g., MS COCO

Dimensionality reduction

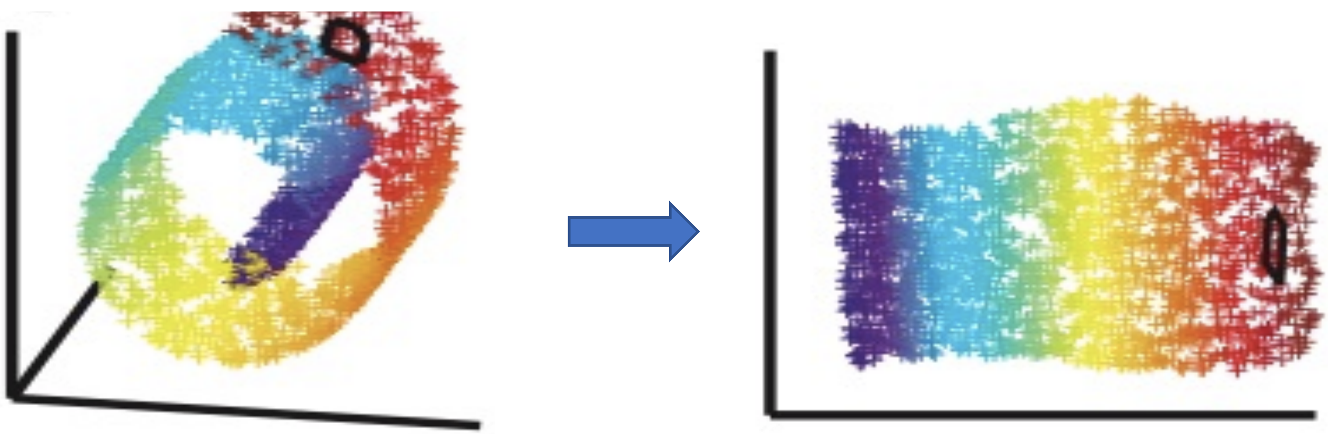
- Input data $x \in \mathcal{X}$ may have thousands or millions of dimensions!
  - e.g., text data represented as bag or words
  - e.g., video stream of images
  - e.g., fMRI data #voxels x #timesteps
- Dimensionality reduction: represent data with fewer dimensions
  - easier learning in subsequent tasks (preprocessing)
  - visualization
  - discover intrinsic patterns in the data

Manifolds

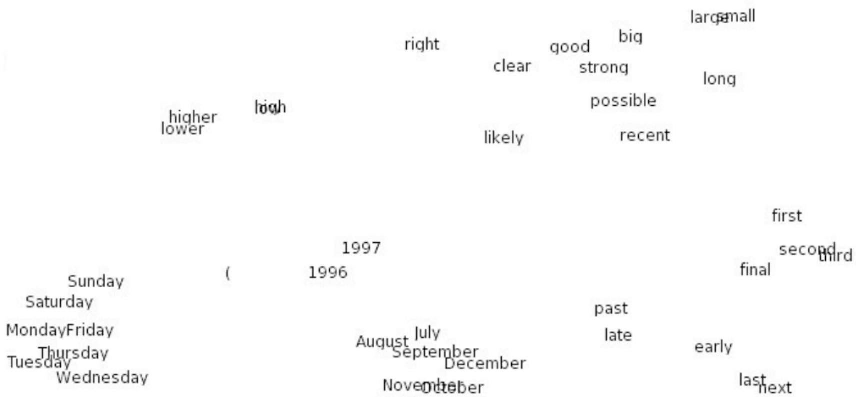




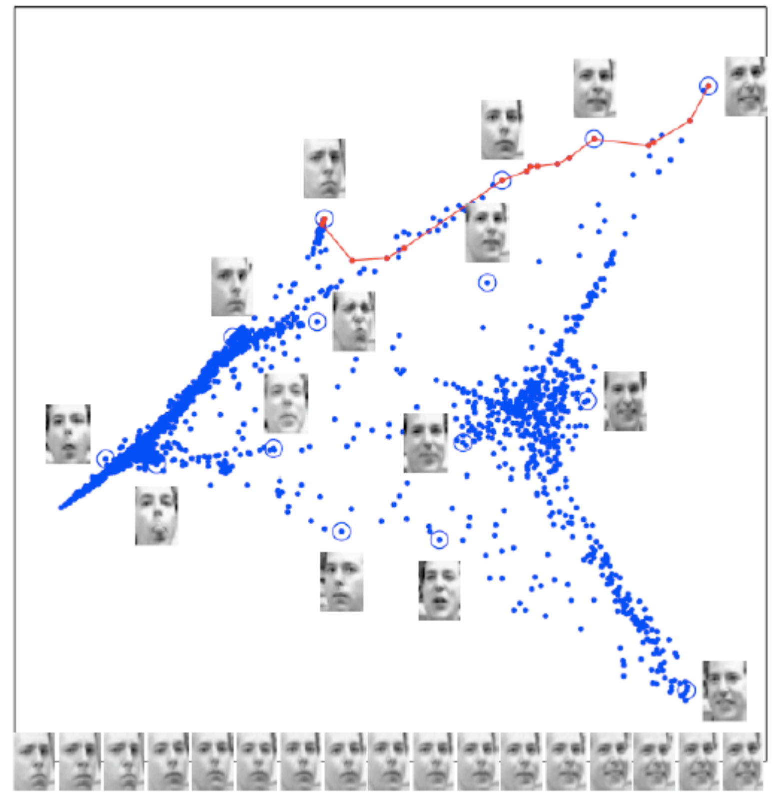
Slide from Yi Zhang



Embeddings



t-SNE visualization from Turian et al. (2010)



[Saul & Roweis '03]

Low dimensional embedding

- Given high dimensional feature
$$\boldsymbol{x} = [x_1, x_2, \dots, x_d]$$
find transformations
$$z(\boldsymbol{x}) = [z_1(\boldsymbol{x}), z_2(\boldsymbol{x}), \dots, z_k(\boldsymbol{x})]$$
so that "almost all useful information" about $\boldsymbol{x}$ is retained in $z(\boldsymbol{x})$
- In general $k \ll d$, and $z(\boldsymbol{x})$ is not invertible
- Transformation learned from a dataset of examples of x
$$S = \{\boldsymbol{x}^{(\boldsymbol{i})} \in \mathbb{R}^d : i = 1,2, \dots, N\}$$
    - Note: typically no labels y

Linear dimensionality reduction

- Given high dimensional feature
  $$\boldsymbol{x} = [x_1, x_2, \dots, x_d]$$
  find transformations
  $$\boldsymbol{z} = z(\boldsymbol{x}) = [z_1(\boldsymbol{x}), z_2(\boldsymbol{x}), \dots, z_k(\boldsymbol{x})]$$
- Restrict $\mathrm{z}(\boldsymbol{x})$ to be a linear function of $\boldsymbol{x}$
  $$z_1 = \boldsymbol{w_1}.\boldsymbol{x}$$
  $$z_2 = \boldsymbol{w_2}.\boldsymbol{x}$$
  $$\vdots$$
  $$z_k = \boldsymbol{w_k}.\boldsymbol{x}$$

$$\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_k \end{bmatrix} = \begin{bmatrix} \leftarrow \boldsymbol{w}^{(1)} \rightarrow \\ \leftarrow \boldsymbol{w}^{(2)} \rightarrow \\ \vdots \\ \leftarrow \boldsymbol{w}^{(k)} \rightarrow \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_d \end{bmatrix}$$

$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x}$$

where

$\boldsymbol{z} \in \mathbb{R}^k$,

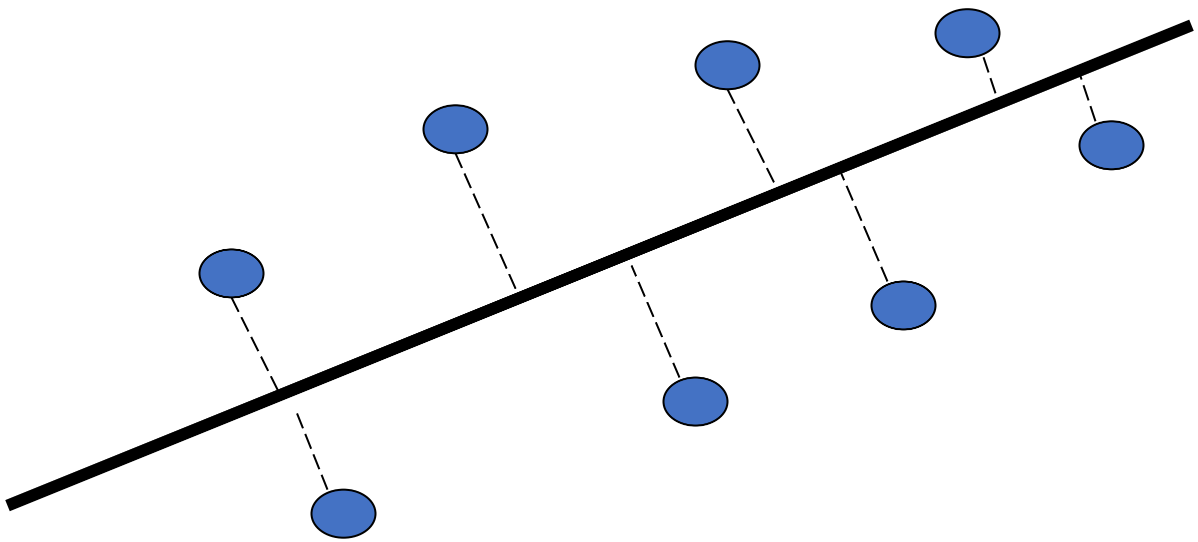
$\boldsymbol{W} \in \mathbb{R}^{k \times d}$,

$\boldsymbol{x} \in \mathbb{R}^d$

only question is which $\boldsymbol{W}$?

Linear dimensionality 2D example

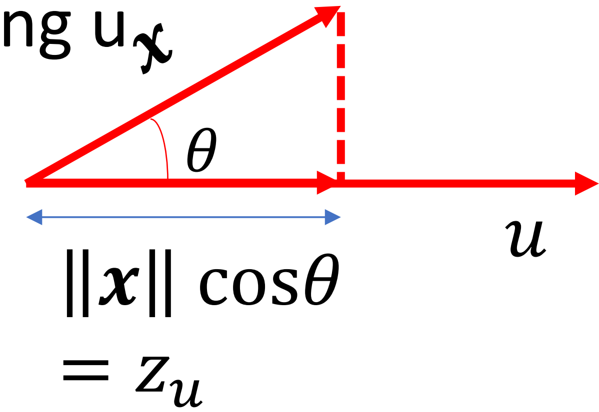
- Given points $S = \{\boldsymbol{x}^{(\boldsymbol{i})}: i = 1,2,\ldots,N\}$ in 2D, we want a 1D representation
  - project $\left\{\boldsymbol{x}^{(\boldsymbol{i})}\right\}$ onto a line $\boldsymbol{w}.\boldsymbol{x} = 0$



  - Find $\boldsymbol{w}$ to minimizes the sum of squared distances to the line

Vector projections

- $\boldsymbol{x}.\boldsymbol{u} = \|\boldsymbol{x}\|\|\boldsymbol{u}\| \cos\theta$
- Assuming $\|\boldsymbol{u}\| = \mathbf{1}$,
- $\boldsymbol{x}.\boldsymbol{u} = \|\boldsymbol{x}\|\cos\theta = z_u$ → value of x along u



- distance of $\boldsymbol{x}$ to projection is $\|z_u\,\boldsymbol{u} - \boldsymbol{x}\| = \|(\boldsymbol{x}.\boldsymbol{u})\boldsymbol{u} - \boldsymbol{x}\|$

Principal component analysis

- For a 1D embedding along direction $\boldsymbol{u}$, distance of $\boldsymbol{x}$ to the projection along $\boldsymbol{u}$ is given by

$$\|z_u\,\boldsymbol{u} - \boldsymbol{x}\| = \|(\boldsymbol{x}.\boldsymbol{u})\boldsymbol{u} - \boldsymbol{x}\|$$

- More generally for k dimensional embedding:
  - find orthonormal basis of the k dimensional subspace $\boldsymbol{u_1}, \boldsymbol{u_2}, \ldots, \boldsymbol{u_k} \in \mathbb{R}^d$, i.e., $\boldsymbol{u_i}.\boldsymbol{u_j} = 1$ if $i = j$, and 0 otherwise
  - let $\boldsymbol{U} \in \mathbb{R}^{k\times d}$ be the matrix with $\boldsymbol{u_1}, \boldsymbol{u_2}, \ldots, \boldsymbol{u_k}$ along rows
  - distance of projection of $\boldsymbol{x}$ to span$\{\boldsymbol{u_1}, \boldsymbol{u_2}, \ldots, \boldsymbol{u_k}\}$

$$\|\boldsymbol{U}^\top \boldsymbol{U}\boldsymbol{x} - \boldsymbol{x}\|$$

  - also from orthonormality of $\boldsymbol{u_1}, \boldsymbol{u_2}, \ldots, \boldsymbol{u_k}$, check $\boldsymbol{U}\boldsymbol{U}^\top = \boldsymbol{I}$
- PCA objective

$$\min_{\boldsymbol{U}\in\mathbb{R}^{k\times d}} \sum_{i=1}^{N} \left\|\boldsymbol{U}^\top \boldsymbol{U}\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(i)}\right\|^2 \; s.t.\;\; \boldsymbol{U}\boldsymbol{U}^\top = I$$

PCA

- PCA objective

$$\min_{\boldsymbol{U}\in\mathbb{R}^{k\times d}} \frac{1}{N}\sum_{i=1}^{N}\left\|\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{x}^{(\boldsymbol{i})}-\boldsymbol{x}^{(\boldsymbol{i})}\right\|^2 \; s.t. \;\; \boldsymbol{U}\boldsymbol{U}^\top = I$$

- Also, for all $\boldsymbol{U}\boldsymbol{U}^\top = I$

$$\|\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{x}-\boldsymbol{x}\|^2 = \|\boldsymbol{x}\|^2 + x^\top\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{x} - 2\boldsymbol{x}^\top\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{x}$$

$$= \|\boldsymbol{x}\|^{\mathbf{2}} - \boldsymbol{x}^\top\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{x} = \|\boldsymbol{x}\|^{\mathbf{2}} - \|\boldsymbol{U}\boldsymbol{x}\|^{\mathbf{2}}$$

- Equivalent PCA objective

$$\max_{\boldsymbol{U}} \frac{1}{N}\sum_{i=1}^{N}\left\|\boldsymbol{U}\boldsymbol{x}^{(\boldsymbol{i})}\right\|^2 = \sum_{j\in[k]} u_j^\top\hat{\Sigma}_{xx}u_j \;\; s.t. \;\; \boldsymbol{U}\boldsymbol{U}^\top = I$$

where $\hat{\Sigma}_{xx} = \frac{1}{N}\sum_{i=1}^{N} x^{(i)}x^{(i)^\top}$ (derivation in board)

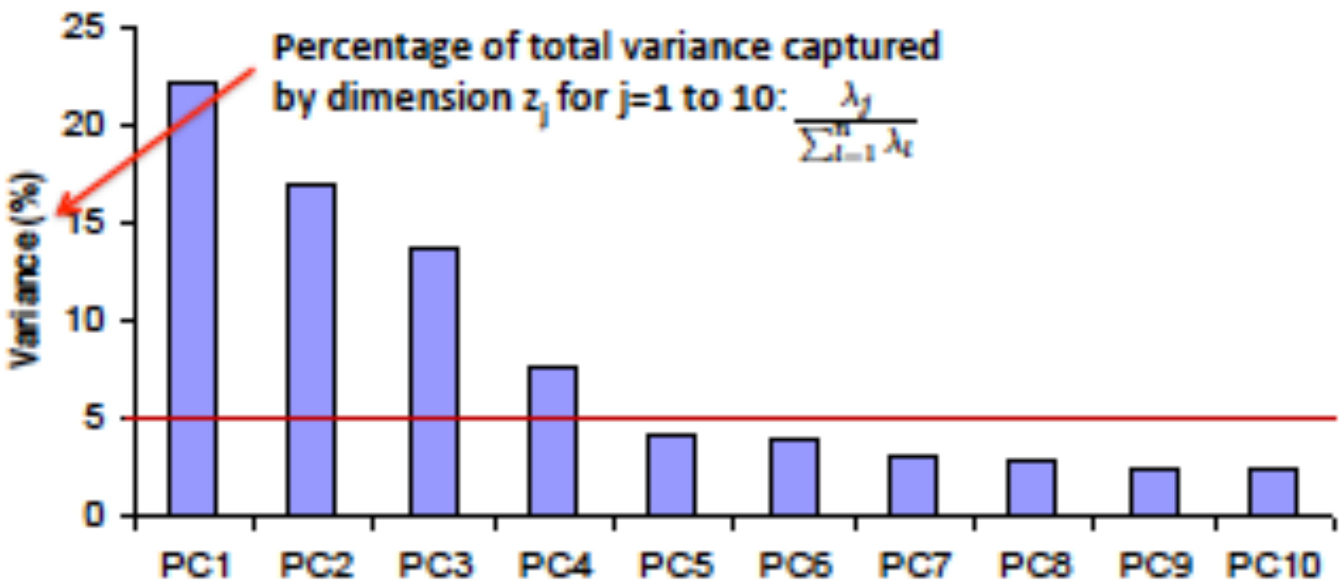
- This is the same as finding top k eigenvectors of $\hat{\Sigma}_{xx}$

PCA algorithm

- Given $S = \{\boldsymbol{x}^{(\boldsymbol{i})} \in \mathbb{R}^d : i = 1,2,\ldots,N\}$
- Let $\boldsymbol{X} \in \mathbb{R}^{N\times d}$ be data matrix
  - make sure X is re-centered so that column mean is 0
- $\hat{\Sigma}_{xx} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}^{(\boldsymbol{i})}\boldsymbol{x}^{(\boldsymbol{i})^\top} = \frac{1}{N}\boldsymbol{X}^\top\boldsymbol{X} \in \mathbb{R}^{d\times d}$
- $\boldsymbol{u_1}, \boldsymbol{u_2}, \ldots, \boldsymbol{u_k} \in \mathbb{R}^d$ are top k eigenvectors of $\hat{\Sigma}_{xx}$

How to pick k?

- Data assumed to be low dimensional projection + noise
- Only keep projections onto components with large eigenvalues and ignore the rest

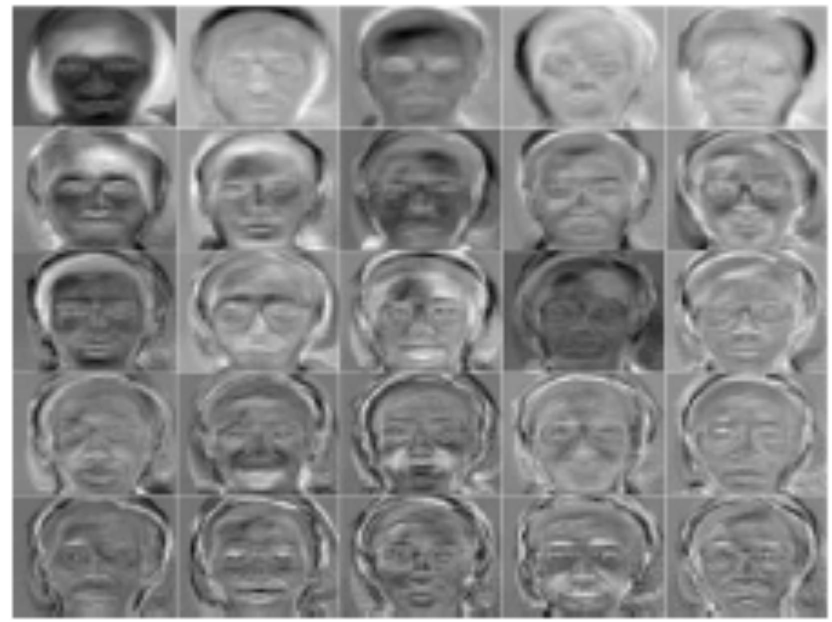


Slide credit: Arti Singh

Eigenfaces

- Input images:

- Principal components:



- Turk and Pentland ’91

SVD version

- Given $S = \{\boldsymbol{x}^{(i)} \in \mathbb{R}^d : i = 1,2,\dots,N\}$
- Let $\boldsymbol{X} \in \mathbb{R}^{N\times d}$ be data matrix
  - make sure $\boldsymbol{X}$ is re-centered so that column mean is 0
- $\boldsymbol{X} = \overline{\boldsymbol{V}}\overline{\boldsymbol{S}}\overline{\boldsymbol{U}}^\top$ be the Singular Value Decomposition (SVD) of $\boldsymbol{X}$, where
  - $\overline{\boldsymbol{V}} \in \mathbb{R}^{N\times d}$ have orthonormal columns, i.e., $\overline{\boldsymbol{V}}^\top\overline{\boldsymbol{V}} = \boldsymbol{I}$
    - columns of $\overline{\boldsymbol{V}}$ are called left singular vectors
  - $\overline{\boldsymbol{U}} \in \mathbb{R}^{d\times d}$ also has orthonormal columns, i.e., $\overline{\boldsymbol{U}}^\top\overline{\boldsymbol{U}} = \boldsymbol{I}$
    - columns of $\overline{\boldsymbol{U}}$ are called right singular vectors
  - $\overline{\boldsymbol{S}} = diagonal(\sigma_1, \sigma_2, \dots, \sigma_d) \in \mathbb{R}^{d\times d}$
    - $\sigma_1, \sigma_2, \dots, \sigma_d$ are called the singular values
- First k columns of $\overline{\boldsymbol{U}}$ are the $\boldsymbol{u_1}, \boldsymbol{u_2}, \dots, \boldsymbol{u_k}$ we want.

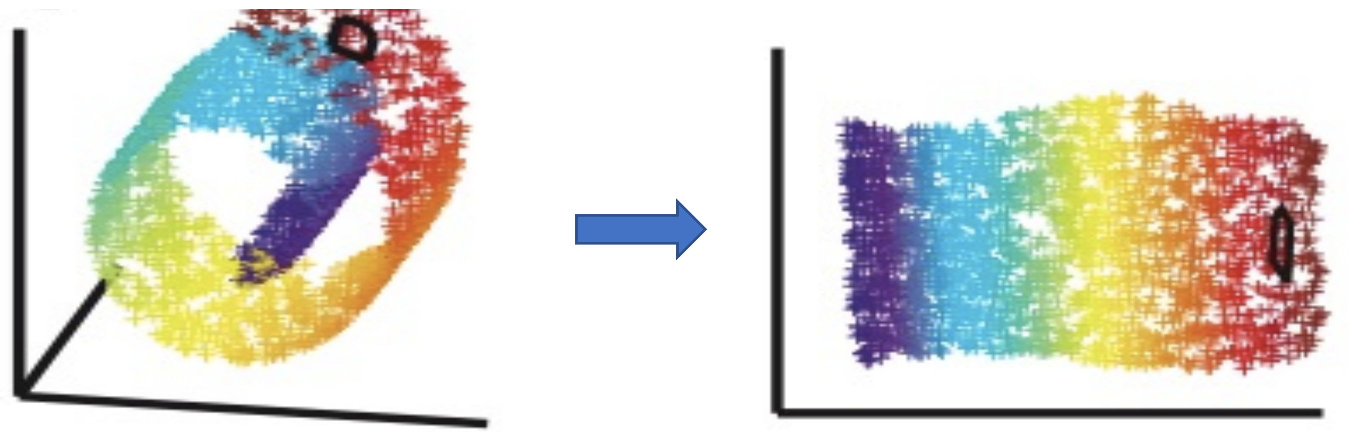
- Representation of $\boldsymbol{x} \in \mathbb{R}^d$ as $z(\boldsymbol{x}) \in \mathbb{R}^k$ is given by

$$z(\boldsymbol{x})_j = \sigma_j\, \boldsymbol{u_j}.\boldsymbol{x} \quad \text{for } j = 1,2,\dots,k$$

Other linear dimensionality reduction

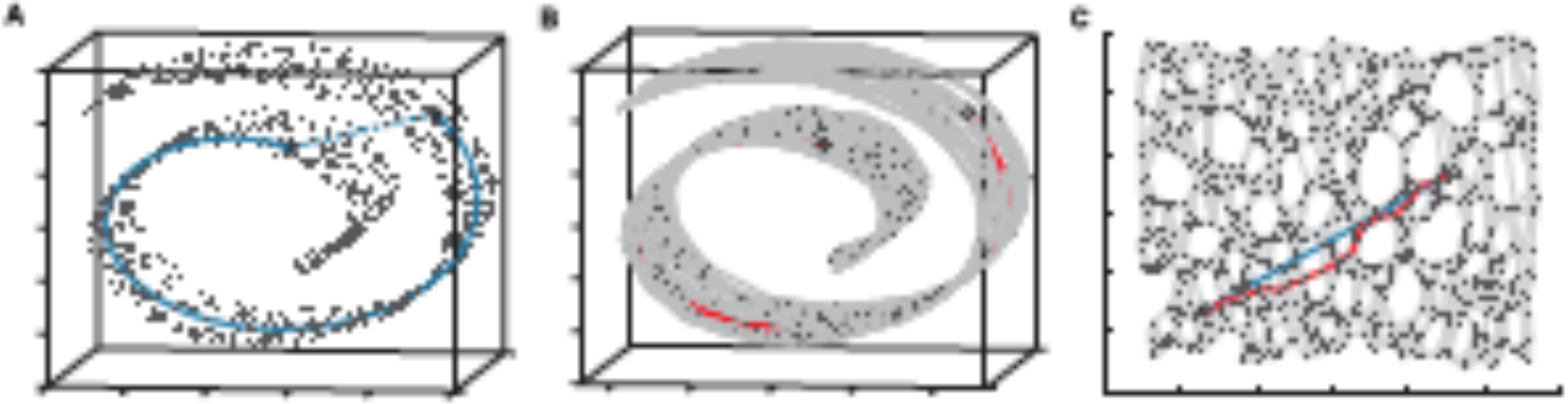
- PCA: given data $x \in \mathbb{R}^d$, find $\mathrm{U} \in \mathbb{R}^{k \times d}$ to minimize
$$\min_{\mathrm{U}} \|U^\top U x - x\|_2^2 \ \ s.t.\ \ UU^\top = I$$
- Canonical correlation analysis: given two "views" of data $x \in \mathbb{R}^d$ and $x' \in \mathbb{R}^{d'}$, find $\mathrm{U} \in \mathbb{R}^{k \times d}, U' \in \mathbb{R}^{k \times d'}$ to minimize
$$\|Ux - U'x'\|_2^2 \ \ s.t.\ \ UU^\top = U'U'^\top = I$$
- Sparse dictionary learning: learn a sparse representation of x as a linear combination of over-complete dictionary
$$x \to Dz \text{ where } \mathrm{D} \in \mathbb{R}^{d \times m}, z \in \mathbb{R}^m$$
    - unlike PCA, here $m \gg d$ so z is higher dimensional, but learned to be sparse!
- Independent component analysis
- Factor analysis
- Linear discriminant analysis

Non linear dimensionality reduction



- **Isomap**
- **Autoencoders**
- Kernel PCA
- Local linear embedding
- Check out t-SNE for 2D visualization
- …

Isomap



[Tenenbaum, Silva, Langford. Science 2000]

Isomap – algorithm

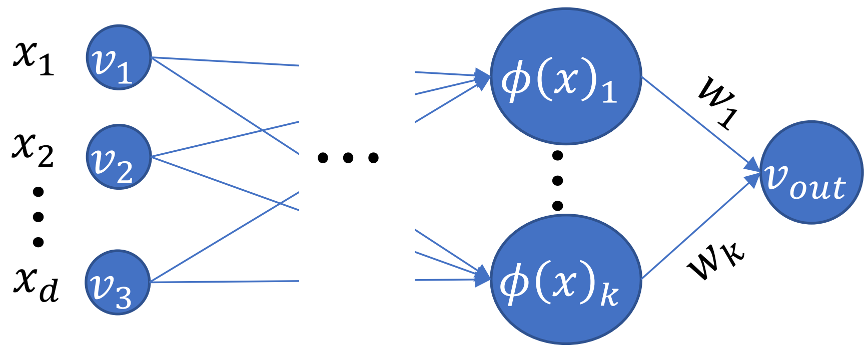
- Dataset of N points $S = \{\boldsymbol{x}^{(\boldsymbol{i})} \in \mathbb{R}^d : i = 1,2, \dots, N\}$
- Represent the points as a kNN-graph with weights proportional to distance between the points
- The geodesic distance $d(x, x')$ between points in the manifold is the length of shortest path in the graph
- Use any shortest path algorithm can be used to construct a matrix $M \in \mathbb{R}^{N \times N}$ of $d\left(x^{(i)}, x^{(j)}\right)$ for all $x^{(i)}, x^{(j)} \in S$
- MDS: Find a (low dimensional) embedding $z(x)$ of x so that distances are preserved

$$\min_{\mathrm{z}} \sum_{i,j \in [N]} \left(\left\|z\left(x^{(i)}\right) - z\left(x^{(j)}\right)\right\| - M_{ij}\right)^2$$

  - sometimes $\min_{\mathrm{z}} \sum_{i,j \in [N]} \frac{\left(\left\|z\left(x^{(i)}\right) - z\left(x^{(j)}\right)\right\| - M_{ij}\right)^2}{M_{ij}^2}$

Autoencoders

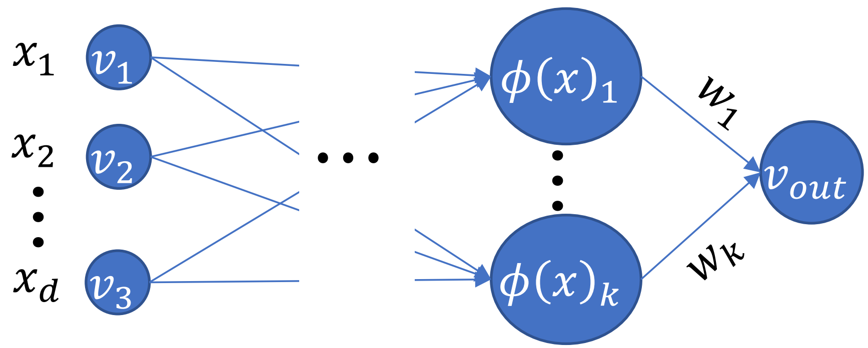
- Recall neural networks as feature learning



  - was learned for some supervised learning task
  - weights learned by minimizing $\ell(v_{out}, y)$
  - but we don't have y anymore!

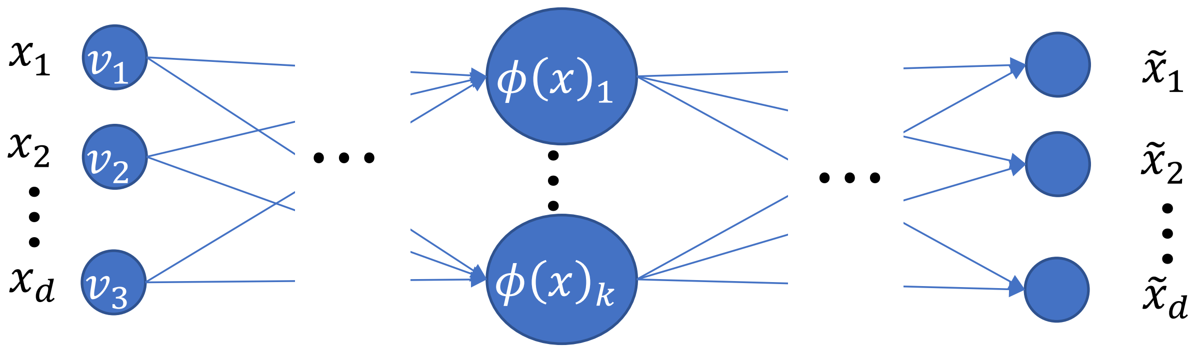
Autoencoders

- Recall neural networks as feature learning



  - was learned for some supervised learning task
  - weights learned by minimizing $\ell(v_{out}, y)$
  - but we don't have y anymore!
  - instead use another "decoder" network to reconstruct x

Autoencoders



- $\phi(x) = f_{\boldsymbol{W_1}}(\boldsymbol{x})$
- $\tilde{\boldsymbol{x}} = f_{\boldsymbol{W_2}}(\phi(\boldsymbol{x}))$
- some loss $\ell(\tilde{x}, x)$

$$\widehat{W}_1, \widehat{W}_2 = \min_{W_1, W_2} \sum_{i=1}^{N} \ell\left(f_{\boldsymbol{W_2}}\left(f_{\boldsymbol{W_1}}\left(\boldsymbol{x}^{(i)}\right)\right), \boldsymbol{x}^{(i)}\right)$$

- learn using SGD with backpropagation