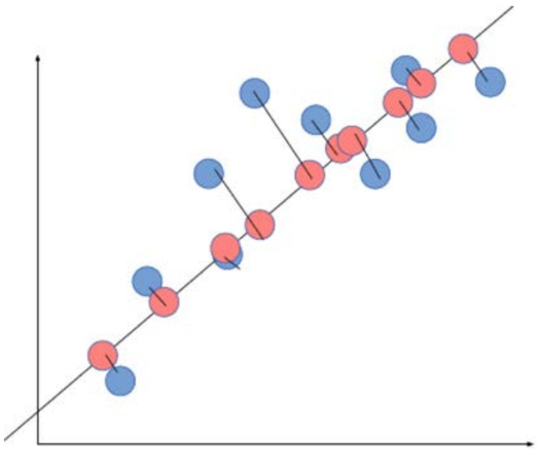
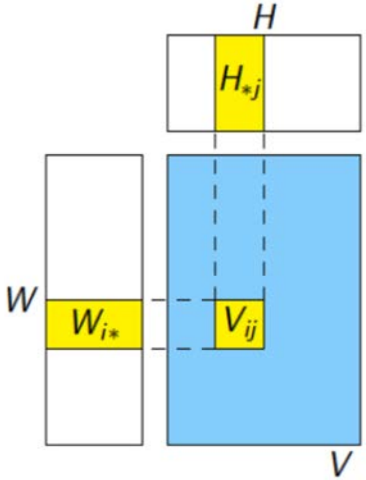
Tufts COMP 135: Introduction to Machine Learning

https://www.cs.tufts.edu/comp/135/2019s/

Dimensionality Reduction & Embedding (part 2/2)





Prof. Mike Hughes

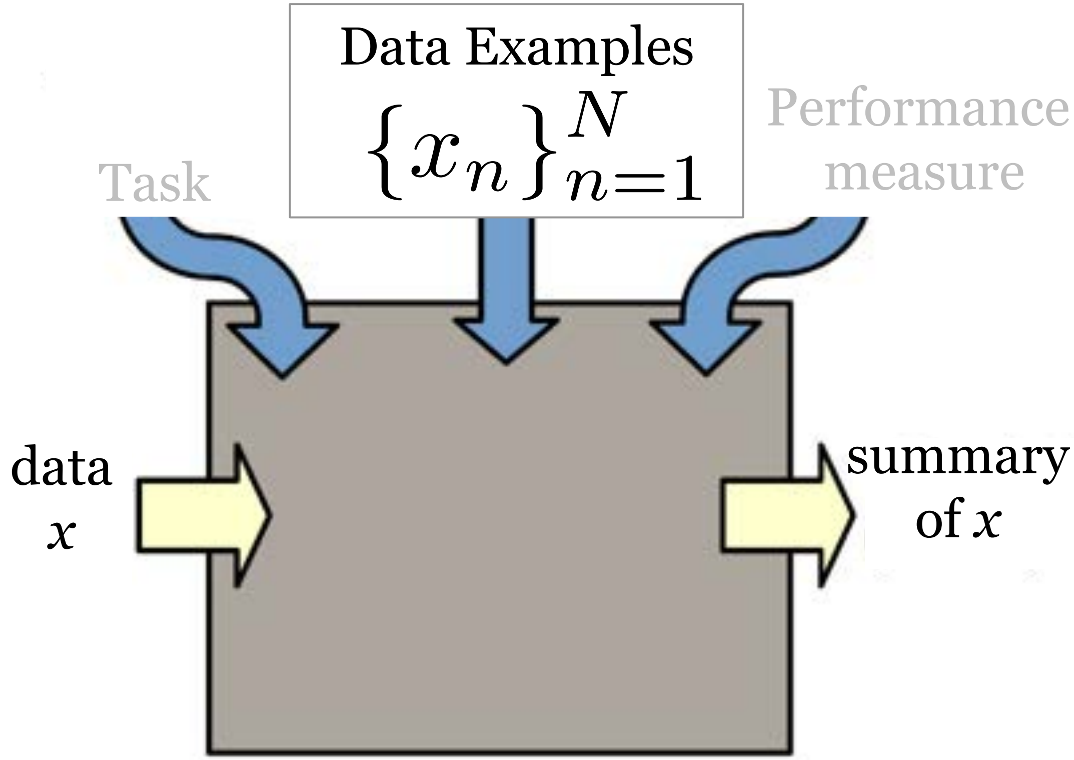
Many ideas/slides attributable to:
Emily Fox (UW), Erik Sudderth (UCI)

What will we learn?

Supervised Learning

Unsupervised Learning

Reinforcement Learning



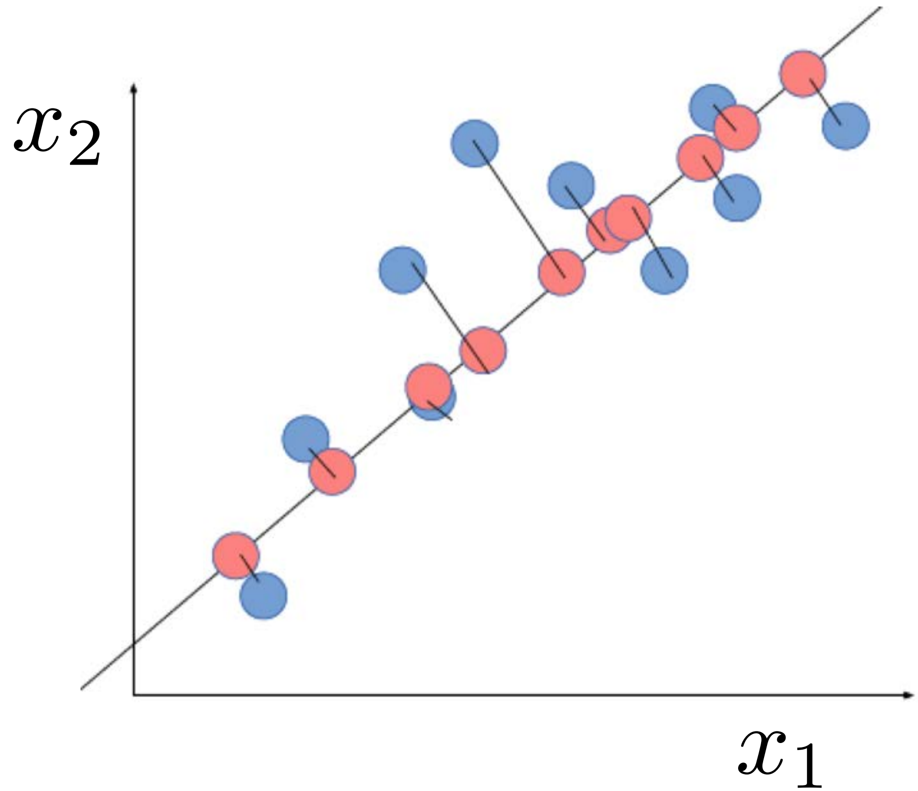
Task: Embedding

Supervised Learning

Unsupervised Learning

embedding

Reinforcement Learning



Dim. Reduction/Embedding
Unit Objectives

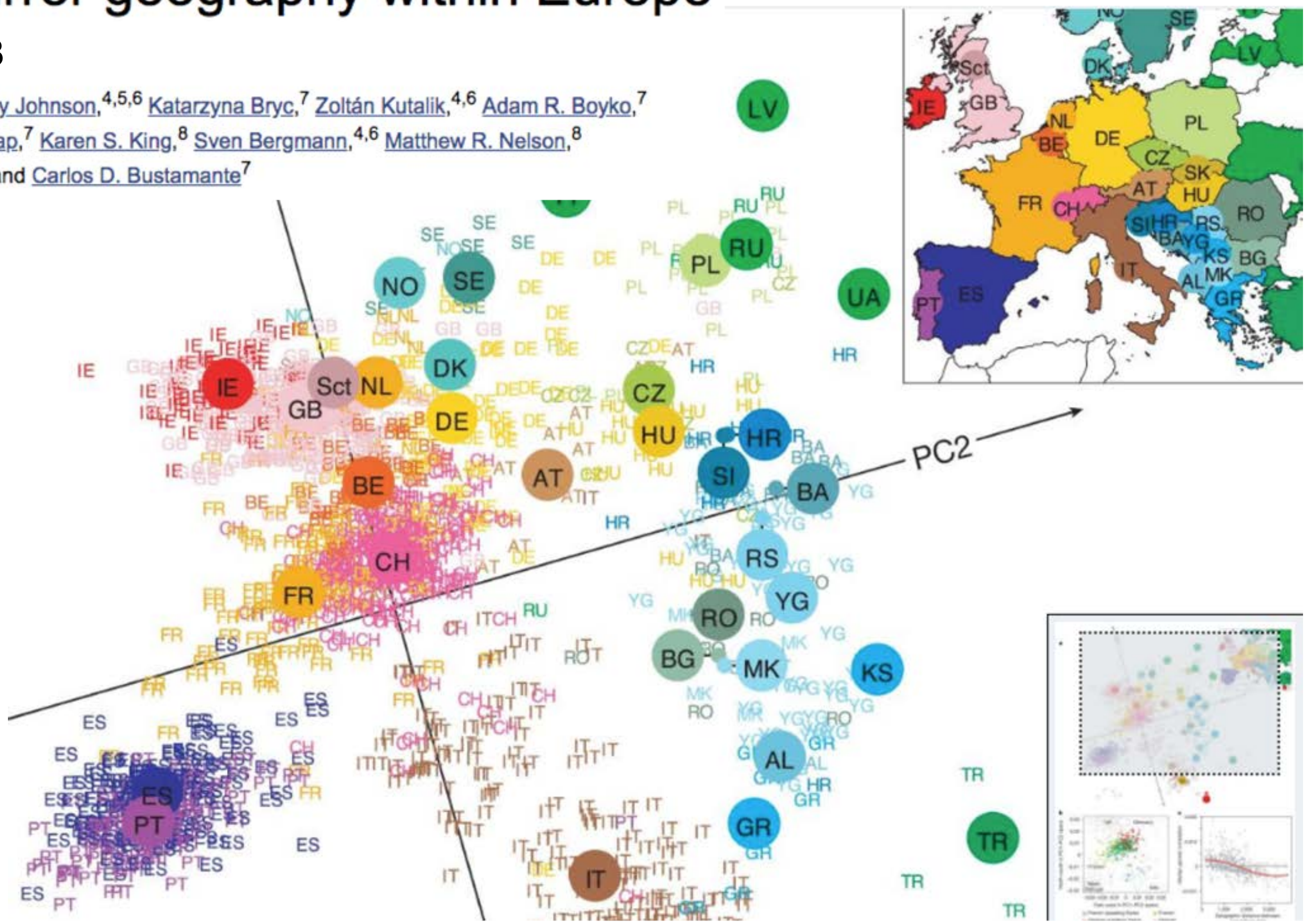
- Goals of dimensionality reduction
  - Reduce feature vector size (keep signal, discard noise)
  - "Interpret" features: visualize/explore/understand
- Common approaches
  - Principal Component Analysis (PCA) + Factor Analysis
  - t-SNE ("tee-snee")
  - word2vec and other neural embeddings
- Evaluation Metrics
  - Storage size
  - "Interpretability"
  - Reconstruction error
  - Prediction error

Example: Genes vs. geography

Genes mirror geography within Europe

Nature, 2008

John Novembre,[1,2] Toby Johnson,[4,5,6] Katarzyna Bryc,[7] Zoltán Kutalik,[4,6] Adam R. Boyko,[7] Adam Auton,[7] Amit Indap,[7] Karen S. King,[8] Sven Bergmann,[4,6] Matthew R. Nelson,[8] Matthew Stephens,[2,3] and Carlos D. Bustamante[7]

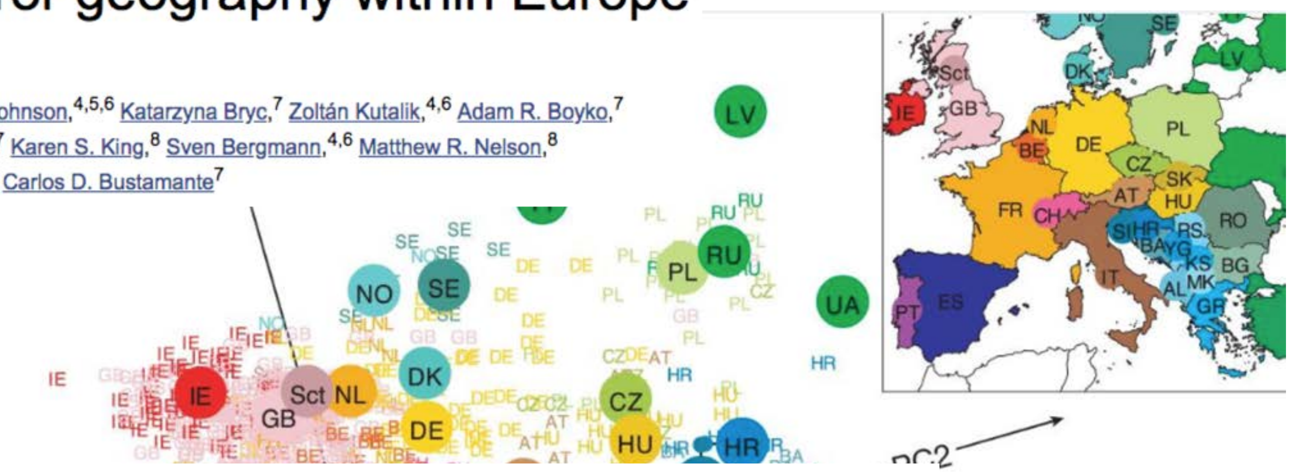


Example: Genes vs. geography

Genes mirror geography within Europe

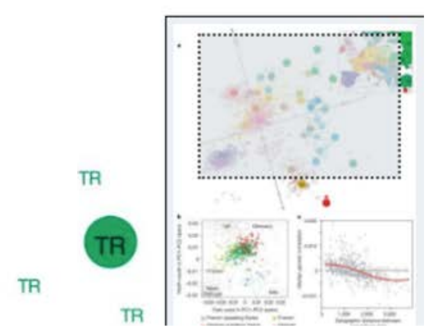
Nature, 2008

John Novembre,[1,2] Toby Johnson,[4,5,6] Katarzyna Bryc,[7] Zoltán Kutalik,[4,6] Adam R. Boyko,[7] Adam Auton,[7] Amit Indap,[7] Karen S. King,[8] Sven Bergmann,[4,6] Matthew R. Nelson,[8] Matthew Stephens,[2,3] and Carlos D. Bustamante[7]



Where possible, we based the geographic origin on the observed country data for grandparents. We used a ‘strict consensus’ approach: if all observed grandparents originated from a single country, we used that country as the origin. If an individual’s observed grandparents originated from different countries, we excluded the individual. Where grandparental data were unavailable, we used the individual’s country of birth.

Total sample size after exclusion: 1,387 subjects
Features: over half a million variable DNA sites in the human genome



Eigenvectors and Eigenvalues

Eigenvalues and Eigenvectors

Here is the most important definition in this text.

⚠ **Definition.** Let A be an $n \times n$ matrix.

1. An ***eigenvector*** of A is a *nonzero* vector v in $\mathbf{R}^n$ such that $Av = \lambda v$, for some scalar λ.
2. An ***eigenvalue*** of A is a scalar λ such that the equation $Av = \lambda v$ has a *nontrivial* solution.

If $Av = \lambda v$ for $v \neq 0$, we say that λ is the ***eigenvalue for*** v, and that v is an ***eigenvector for*** λ.

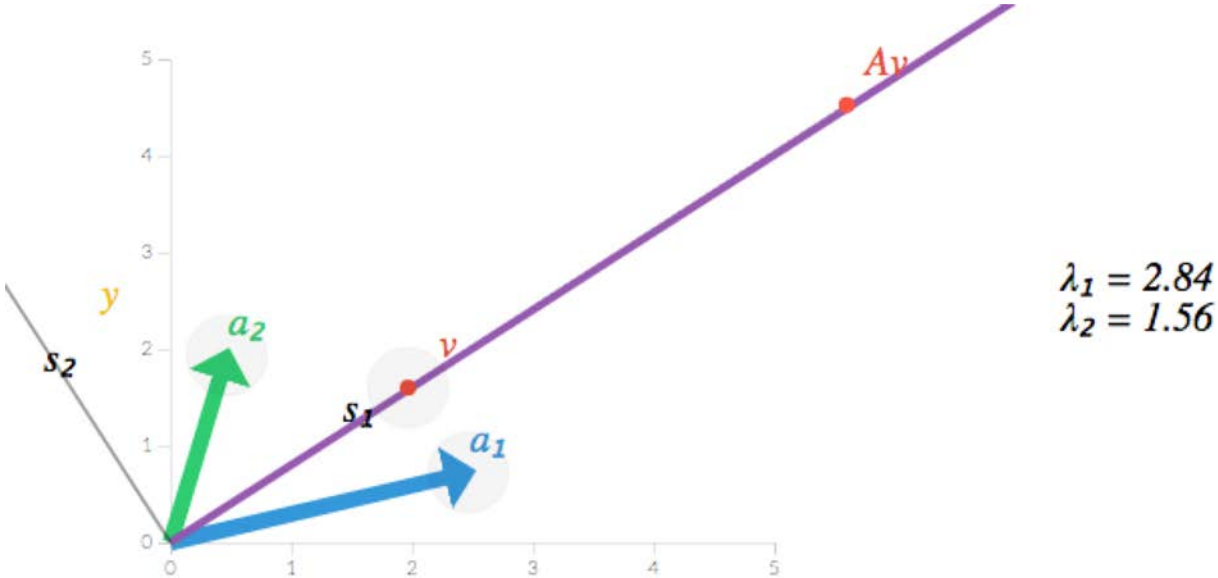
The German prefix "eigen" roughly translates to "self" or "own". An eigenvector of A is a vector that is taken to a multiple of itself, which partially explains the terminology.

Note. Eigenvalues and eigenvectors are only for square matrices.

Source: https://textbooks.math.gatech.edu/ila/eigenvectors.html

Demo: What is an Eigenvector?

- http://setosa.io/ev/eigenvectors-and-eigenvalues/



Centering the Data
Goal: each feature's mean = 0.0

Why center?

- Think of mean vector as simplest possible "reconstruction" of a dataset
- No example specific parameters, just one F-dim vector

$$\min_{m \in \mathbb{R}^F} \quad \sum_{n=1}^{N} (x_n - m)^T (x_n - m)$$

$$m^* = \mathrm{mean}(x_1, \ldots x_N)$$

Principal Component Analysis

Reconstruction with PCA

$$x_i = W z_i + m$$

F vector	F x K	K vector	F vector
High-dim. data	Basis	Low-dim vector	mean

Principal Component Analysis

Training step: .fit()

- Input:
  - X : training data, N x F
    - N high-dim. example vectors
  - K : int, number of components
    - Satisfies 1 <= K <= F

- Output:
  - m : mean vector, size F
  - W : learned basis of eigenvectors, F x K
    - One F-dim. vector (magnitude 1) for each component
    - Each of the K vectors is orthogonal to every other

Principal Component Analysis

Transformation step: .transform()

- Input:
  - X : training data, N x F
    - N high-dim. example vectors
  - Trained PCA "model"
    - m : mean vector, size F
    - W : learned basis of eigenvectors, F x K
      - One F-dim. vector (magnitude 1) for each component
      - Each of the K vectors is orthogonal to every other
- Output:
  - Z : projected data, N x K

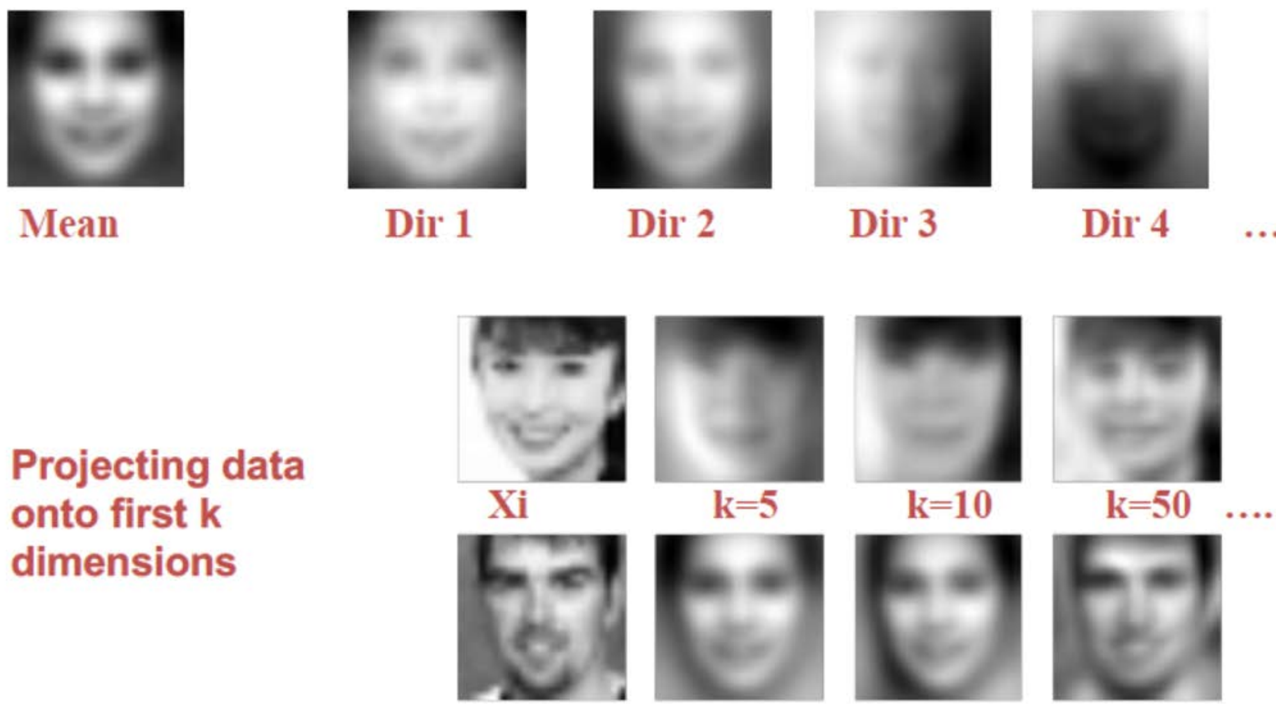
PCA Demo

- http://setosa.io/ev/principal-component-analysis/

Example: EigenFaces

Ex: Viola Jones data set

- 24x24 images of faces = 576 dimensional measurements
- Take first K PCA components



Credit: Erik Sudderth

PCA Principles

- Minimize **reconstruction error**
  - Should be able to recreate x from z

- Equivalent to **maximizing variance**
  - Want reconstructions to retain maximum information

PCA: How to Select K?

- 1) Use downstream supervised task metric
  - Regression error
- 2) Use memory constraints of task
  - Can't store more than 50 dims for 1M examples? Take K=50
- 3) Plot cumulative "variance explained"
  - Take K that seems to capture most or all variance

Empirical Variance of Data X

$$\mathrm{Var}(X) = \frac{1}{N}\sum_{n=1}^{N}\sum_{f=1}^{F} x_{nf}^2$$

$$= \frac{1}{N}\sum_{n=1}^{N} x_n^T x_n$$

- (Assumes each feature is centered)

Variance of reconstructions

$$= \frac{1}{N} \sum_{n=1}^{N} x_n^T x_n$$

$$= \frac{1}{N} \sum_{n=1}^{N} (z_{n1} w_1 + \ldots + z_{nK} w_K)^T (z_{n1} w_1 + \ldots + z_{nK} w_K)$$

$$= \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk}^2$$

$$= \sum_{k=1}^{K} \lambda_k$$

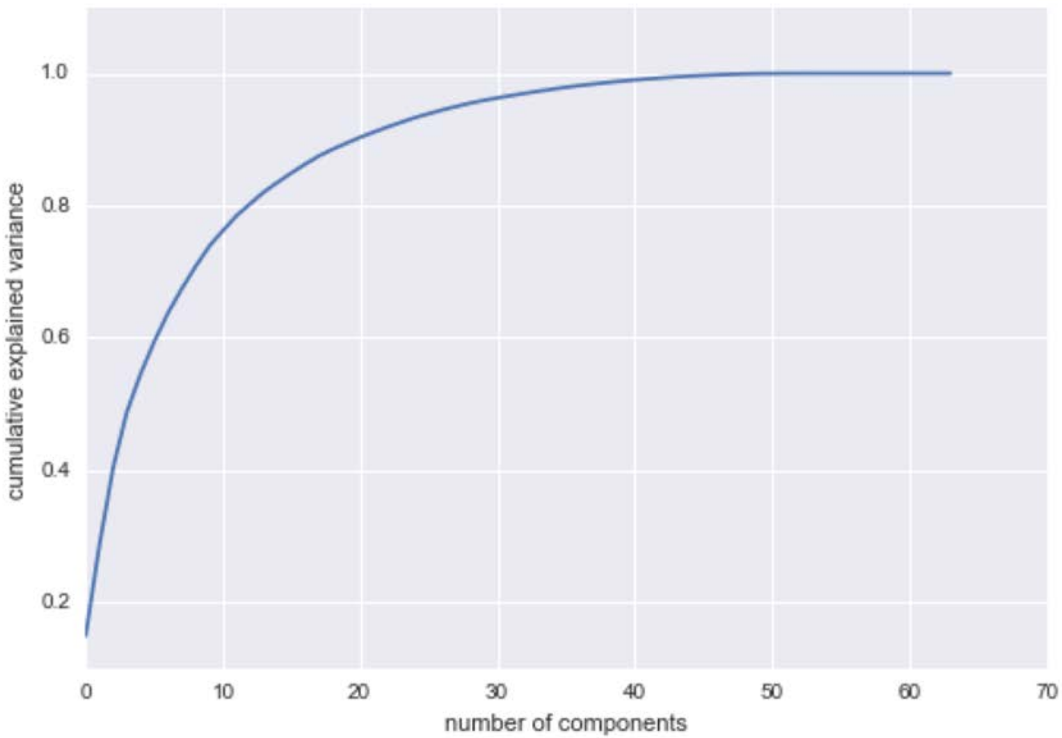
Just sum up the top K eigenvalues!

Proportion of Variance Explained by first K components

$$\mathrm{PVE}(K) = \frac{\sum_{k=1}^{K} \lambda_k}{\sum_{f=1}^{F} \lambda_f}$$

Variance explained curve

```
pca = PCA().fit(digits.data)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance');
```



```
from sklearn.datasets import load_digits
digits = load_digits()
digits.data.shape
```

```
(1797, 64)
```

PCA Summary

PRO

- Usually, fast to train, fast to test
  - Slowest step: finding K eigenvectors of an F x F matrix
- Nested model
  - PCA with K=5 overlaps with PCA with K=4

CON

- Sensitive to rescaling of input data features
- Learned basis known only up to +/- scaling
- Not often best for supervised tasks

PCA: Best Practices

- If features all have different units
  - Try rescaling to all be within (-1, +1) or have variance 1

- If features have same units, may not need to do this

Beyond PCA: Factor Analysis

A Probabilistic Model

$$x_i = W z_i + m + \epsilon_i$$

F vector	F x K	K vector	F vector	F vector
High-dim. data	Basis	Low-dim vector	mean	noise

$$\epsilon_i \sim \mathcal{N}(0, I_F)$$

A Probabilistic Model

$$x_i = W z_i + m + \epsilon_i$$

In terms of matrix math:

$$X = W Z + M + E$$

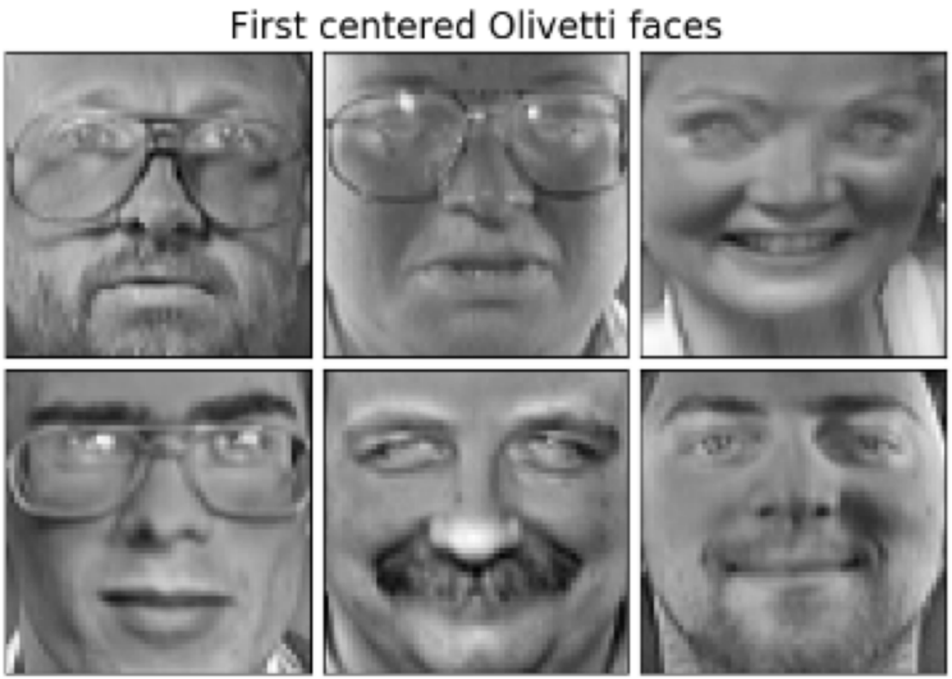
A Probabilistic Model

$$x_i = W z_i + m + \epsilon_i$$

x_i	W	z_i	m	ϵ_i
F vector	F x K	K vector	F vector	F vector
High-dim. data	Basis	Low-dim vector	mean	noise

$$\epsilon_i \sim \mathcal{N}(0, \begin{bmatrix} \sigma^2 & 0 & 0 \\ 0 & \sigma^2 & 0 \\ 0 & 0 & \sigma^2 \end{bmatrix})$$

Face Dataset



$$\epsilon_i \sim \mathcal{N}(0, \begin{bmatrix} \sigma^2 & 0 & 0 \\ 0 & \sigma^2 & 0 \\ 0 & 0 & \sigma^2 \end{bmatrix})$$

Is this noise model realistic?

Each pixel might need own variance!

$$\epsilon_i \sim \mathcal{N}(0, \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{bmatrix})$$

Factor Analysis

sklearn.decomposition.FactorAnalysis

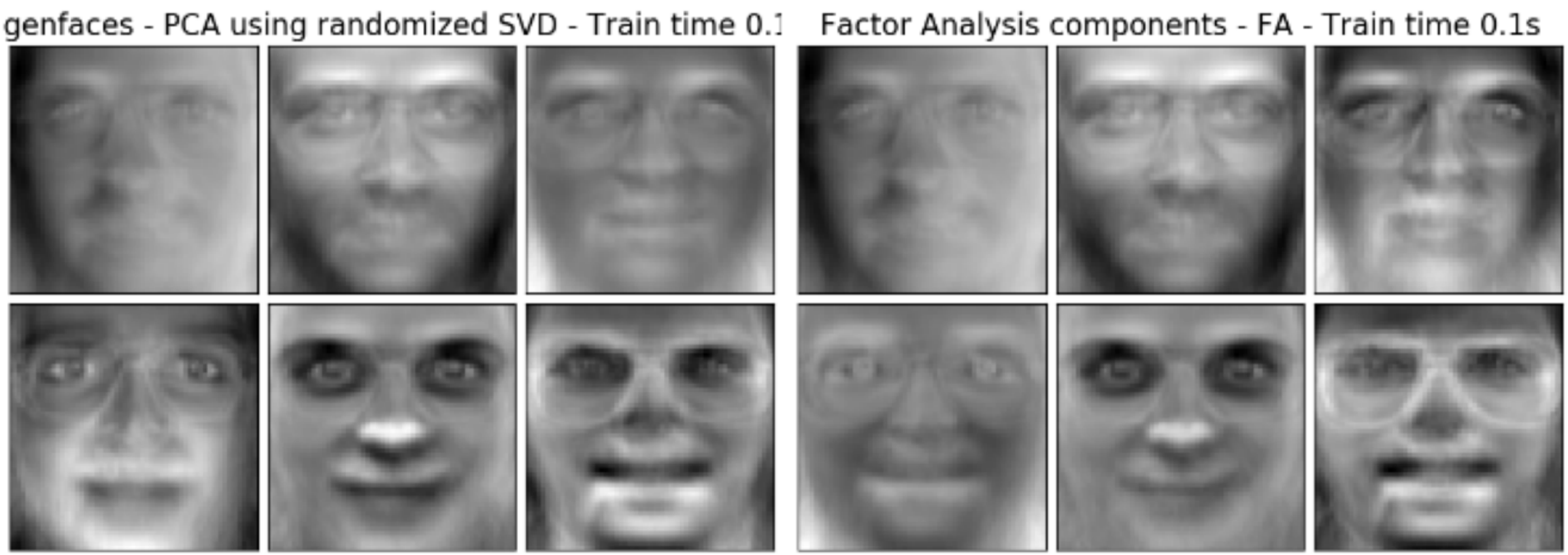
class `sklearn.decomposition.` **`FactorAnalysis`** (*n_components=None, tol=0.01, copy=True, max_iter=1000, noise_variance_init=None, svd_method='randomized', iterated_power=3, random_state=0*) ¶ [source]

- Finds a linear basis like PCA, but allows ***per-feature*** estimation of variance

$$\epsilon_i \sim \mathcal{N}(0, \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{bmatrix})$$

- Small detail: columns of estimated basis may not be orthogonal

PCA vs Factor Analysis



Matrix Factorization and Singular Value Decomposition

Matrix Factorization (MF)

- User i represented by vector $\boldsymbol{z}_i \in R^k$
- Item j represented by vector $w_j \in R^k$
- Inner product $\boldsymbol{z}_i^\top \boldsymbol{w}_j$ approximates the utility X_{ij}
- Intuition:
    - Two items with similar vectors get similar utility scores from the same user;
    - Two users with similar vectors give similar utility scores to the same item

Example Factors

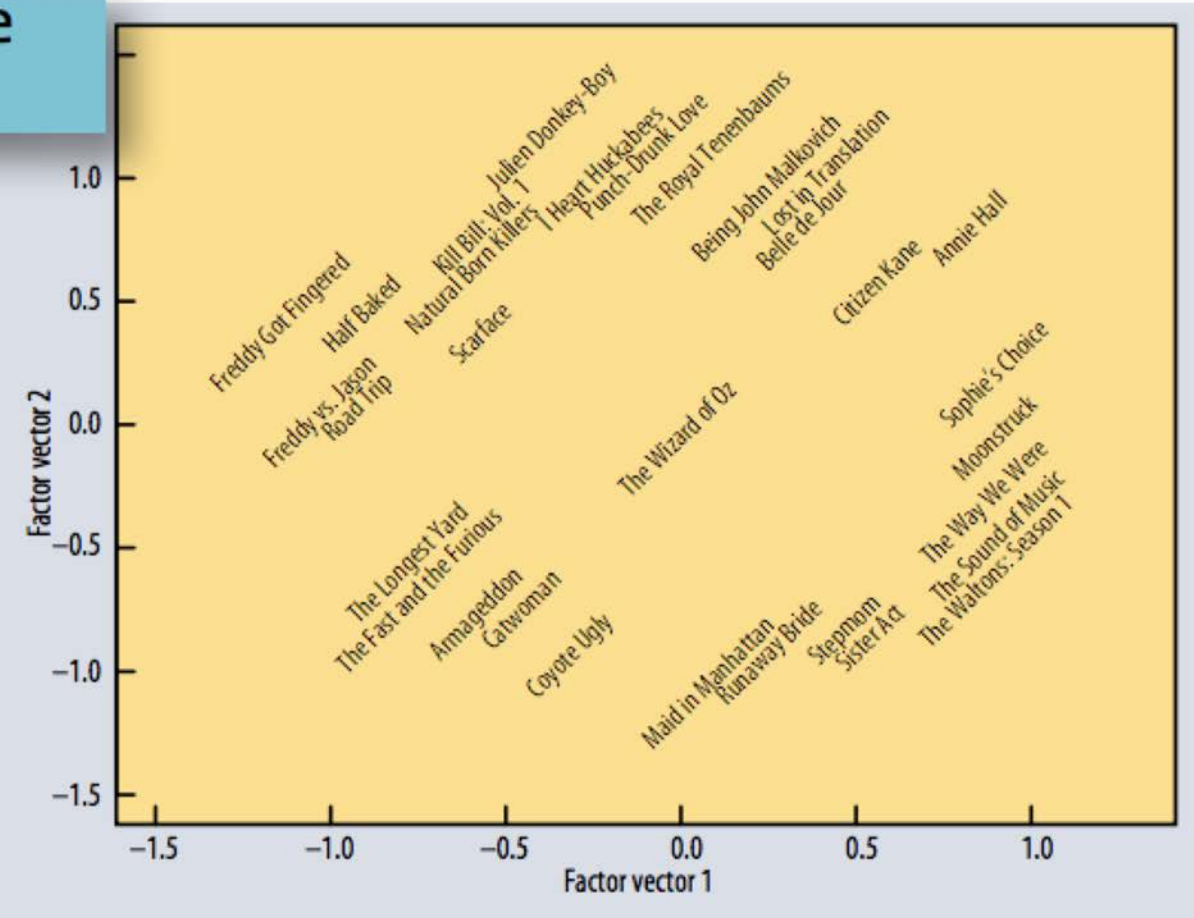
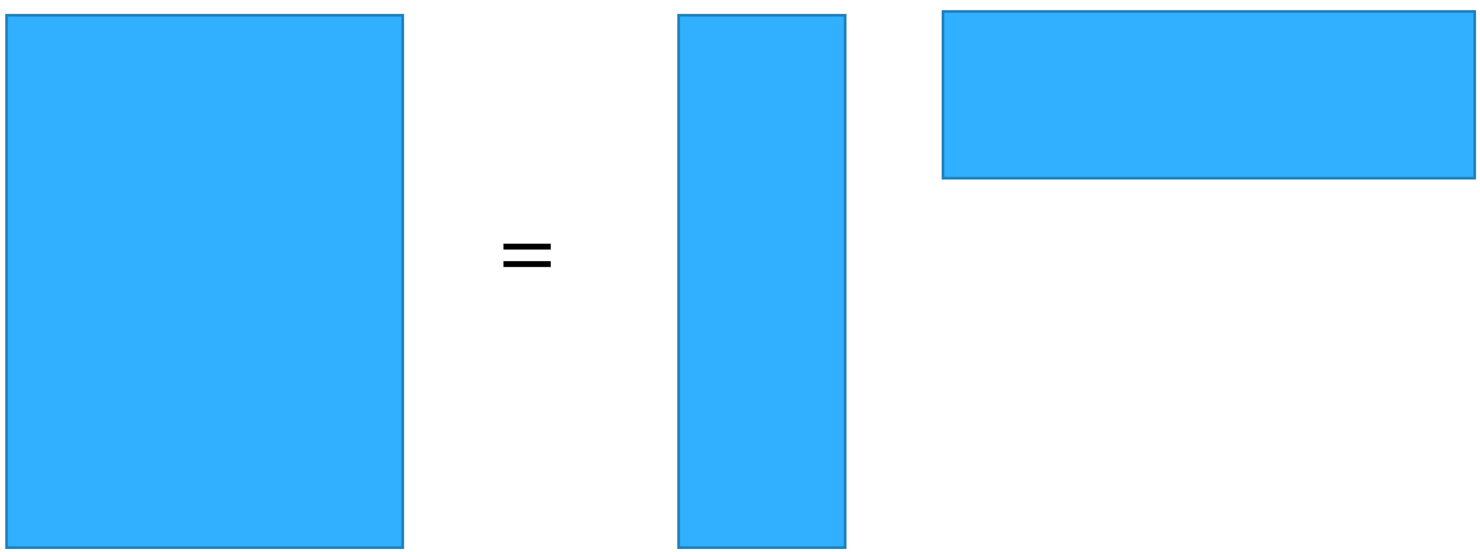


Figure 3. The first two vectors from a matrix decomposition of the Netflix Prize data. Selected movies are placed at the appropriate spot based on their factor vectors in two dimensions. The plot reveals distinct genres, including clusters of movies with strong female leads, fraternity humor, and quirky independent films.

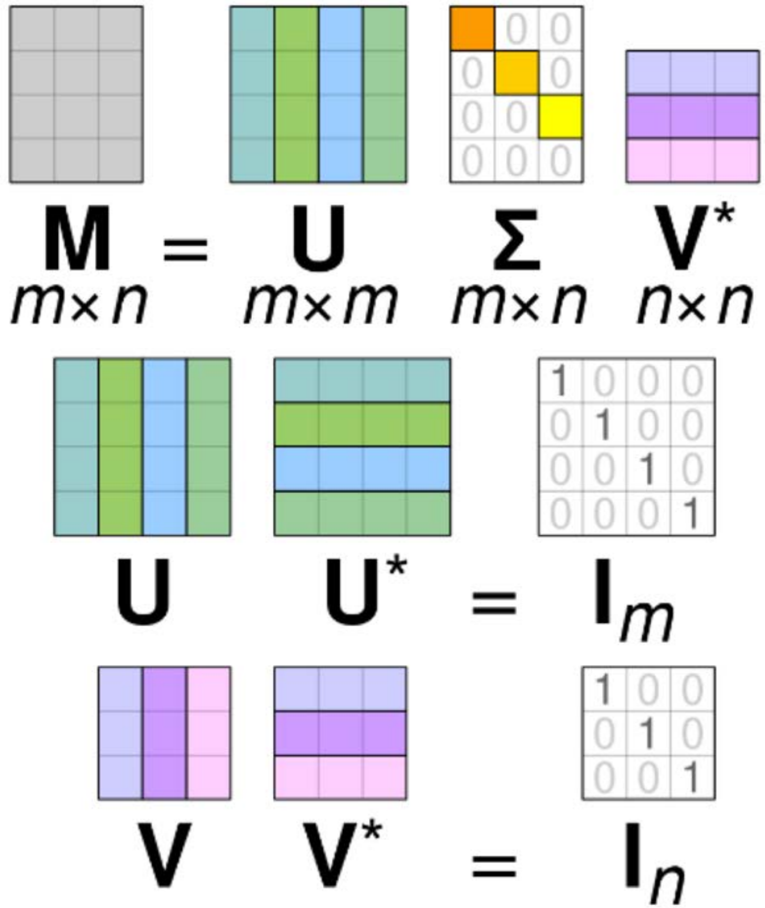
Figure from Koren et al. (2009)

General Matrix Factorization

$$X = ZW$$



SVD: Singular Value Decomposition



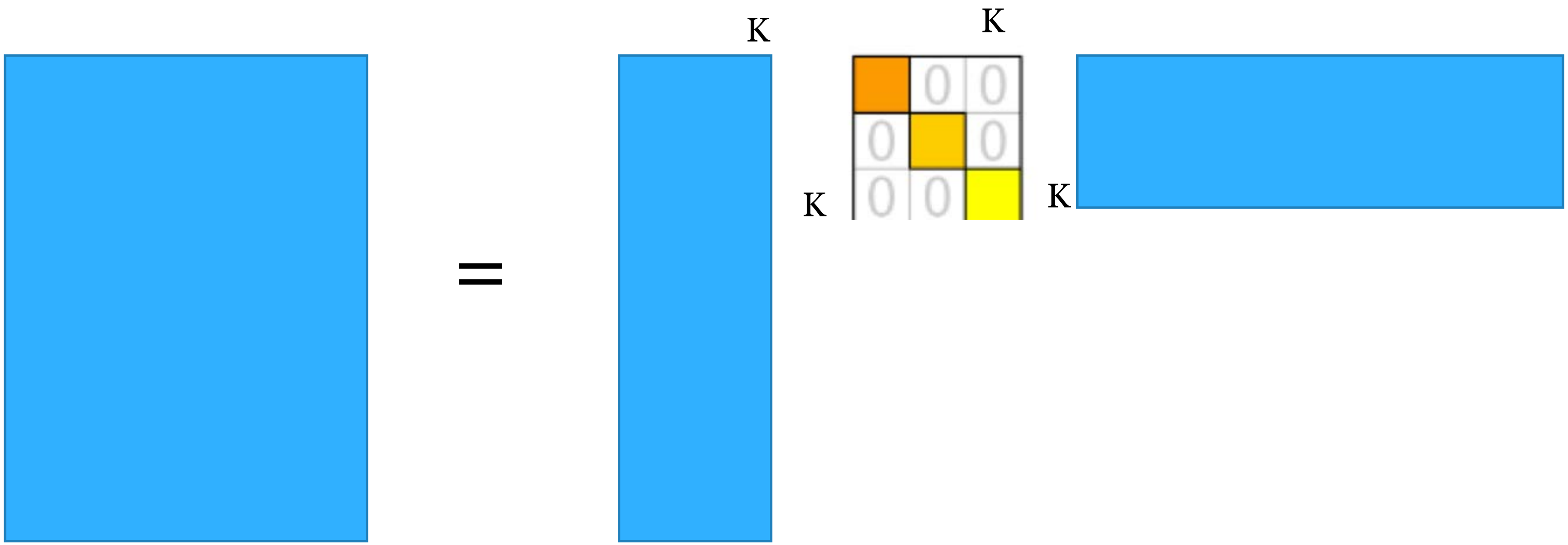
Credit: Wikipedia

Truncated SVD

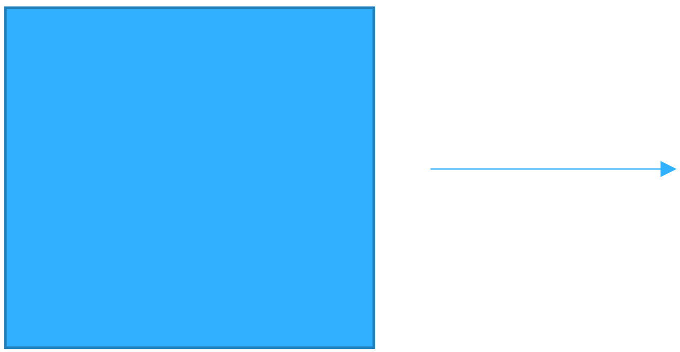
sklearn.decomposition.TruncatedSVD

class sklearn.decomposition. **TruncatedSVD** (*n_components=2, algorithm='randomized', n_iter=5, random_state=None, tol=0.0*)

$$X = UDV^T$$



Recall: Eigen Decomposition



$$\lambda_1, \lambda_2, \ldots \lambda_K$$

$$w_1, w_2, \ldots w_K$$

Two ways to "fit" PCA

- First, apply "centering" to X
- Then, do one of these two options:

- 1) Compute SVD of X
  - Eigenvalues are rescaled entries of the diagonal D
  - Basis = first K columns of V

- 2) Compute covariance Cov(X)
  - Eigenvalues = largest eigenvalues of Cov(X)
  - Basis = corresponding eigenvectors of Cov(X)

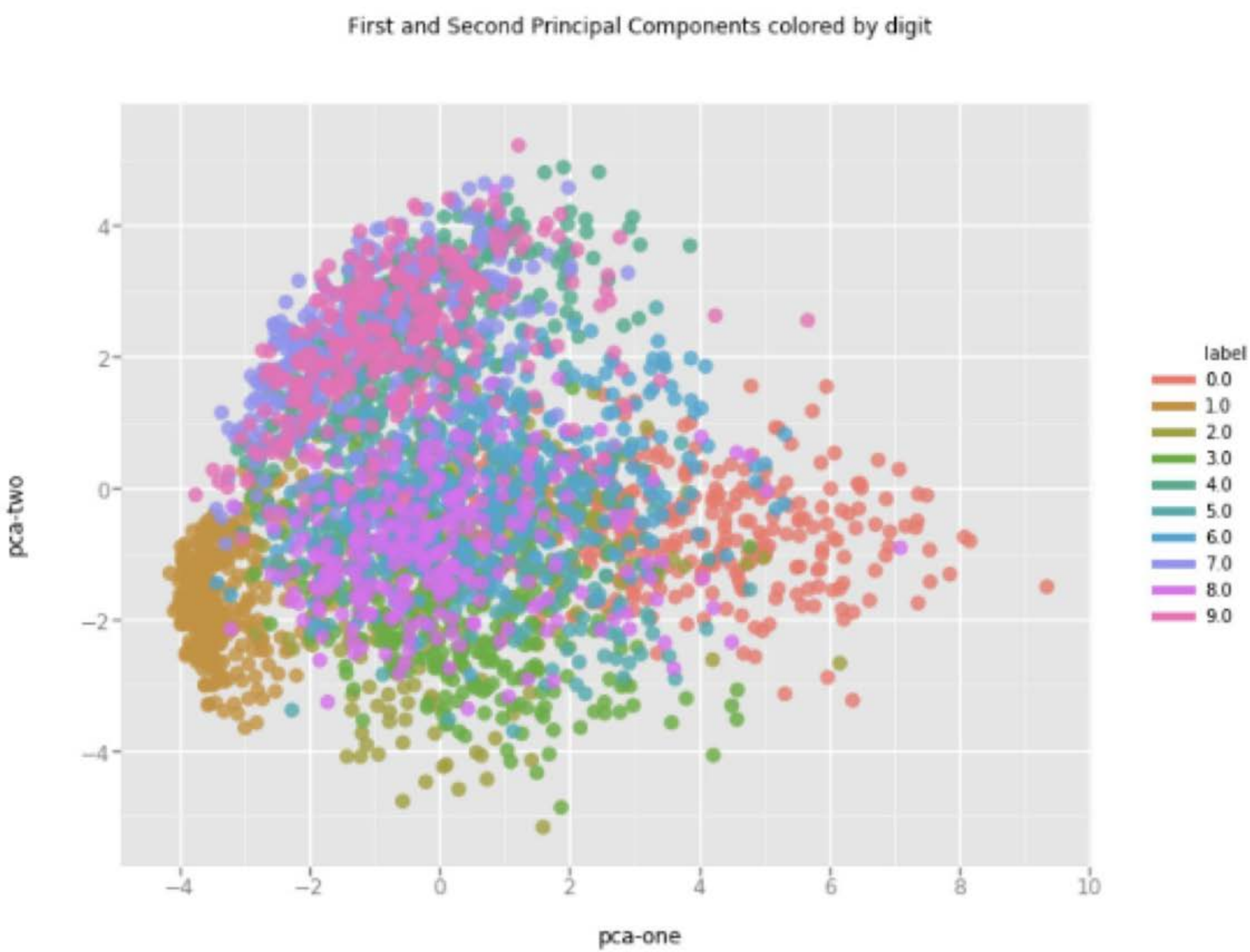
Visualization with t-SNE

Reducing Dimensionality of Digit Images

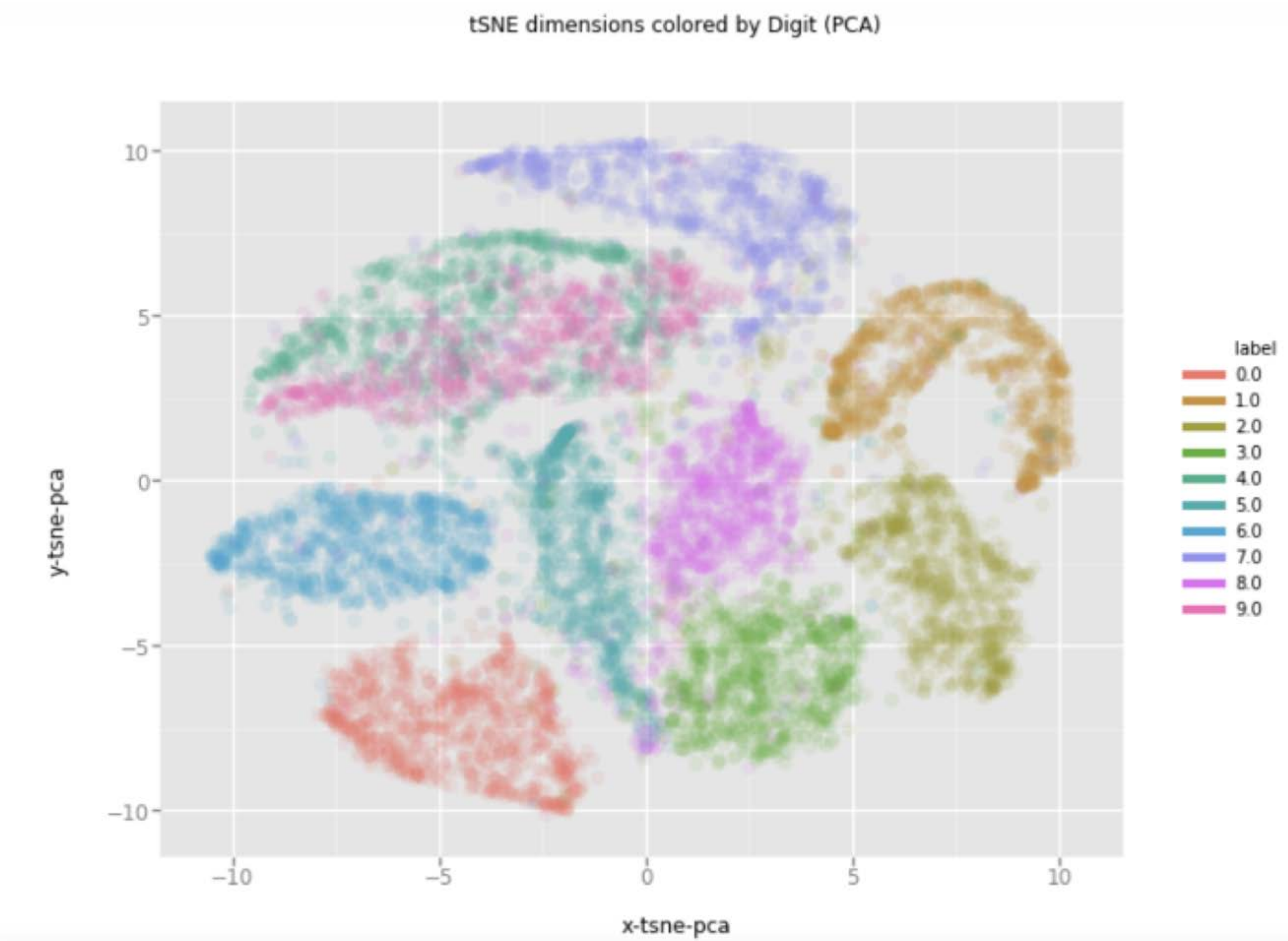
INPUT: Each image represented by 784-dimensional vector

Apply PCA transformation with K=2

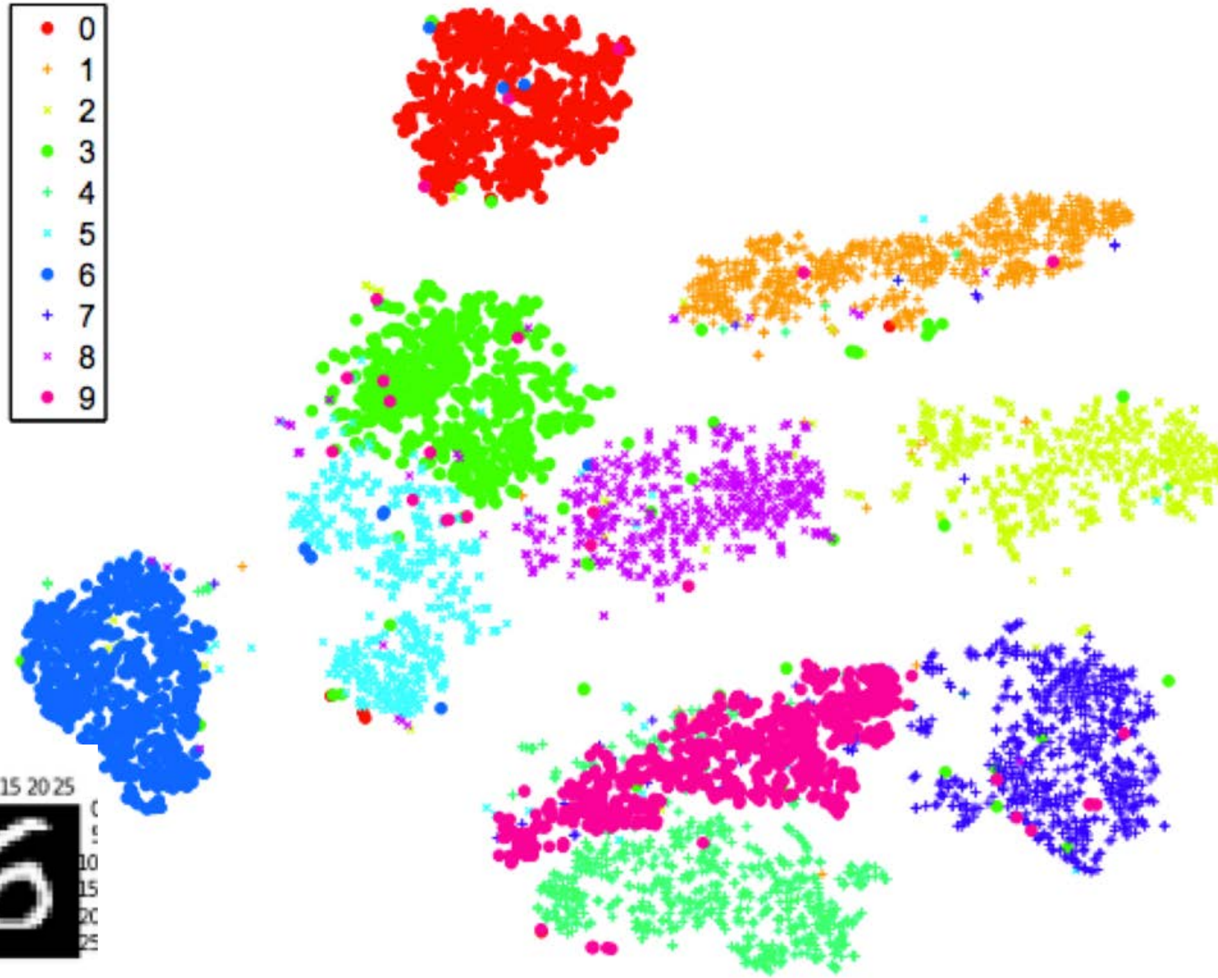
OUTPUT: Each image is a 2-dimensional vector



Credit: Luuk Derksen (https://medium.com/@luckylwk/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b)



Credit: Luuk Derksen (https://medium.com/@luckylwk/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b)



(a) Visualization by t-SNE.

Practical Tips for t-SNE

- If dim is very high, preprocess with PCA to ~30 dims, then apply t-SNE
- Beware: Non-convex cost function

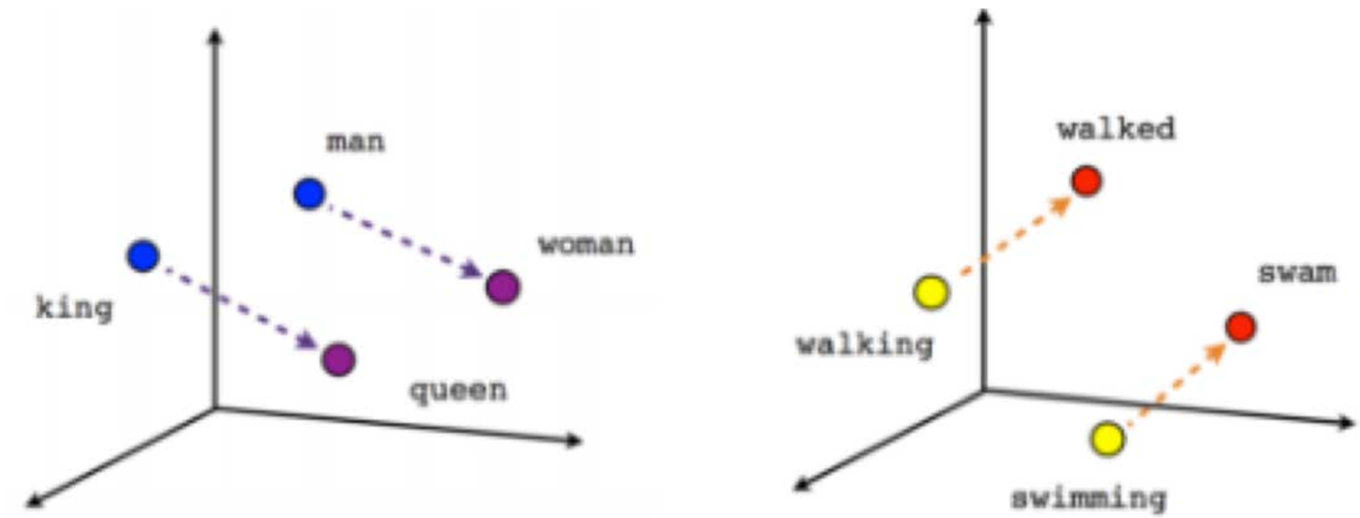
How to Use t-SNE Effectively

https://distill.pub/2016/misread-tsne/

Word Embeddings

Word Embeddings (word2vec)

Goal: map each word in vocabulary to an embedding vector
- Preserve semantic meaning in this new vector space



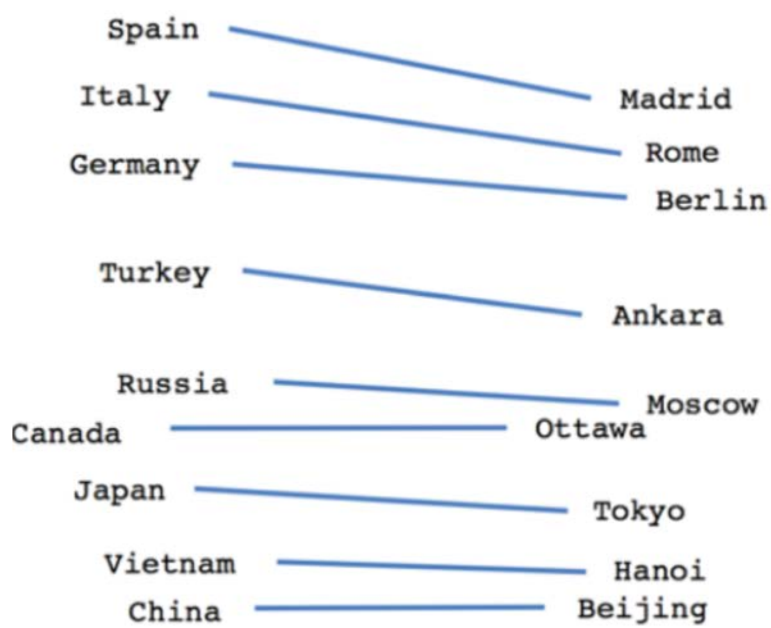
Male-Female

Verb tense

vec(swimming) – vec(swim) + vec(walk) = vec(walking)

Word Embeddings (word2vec)

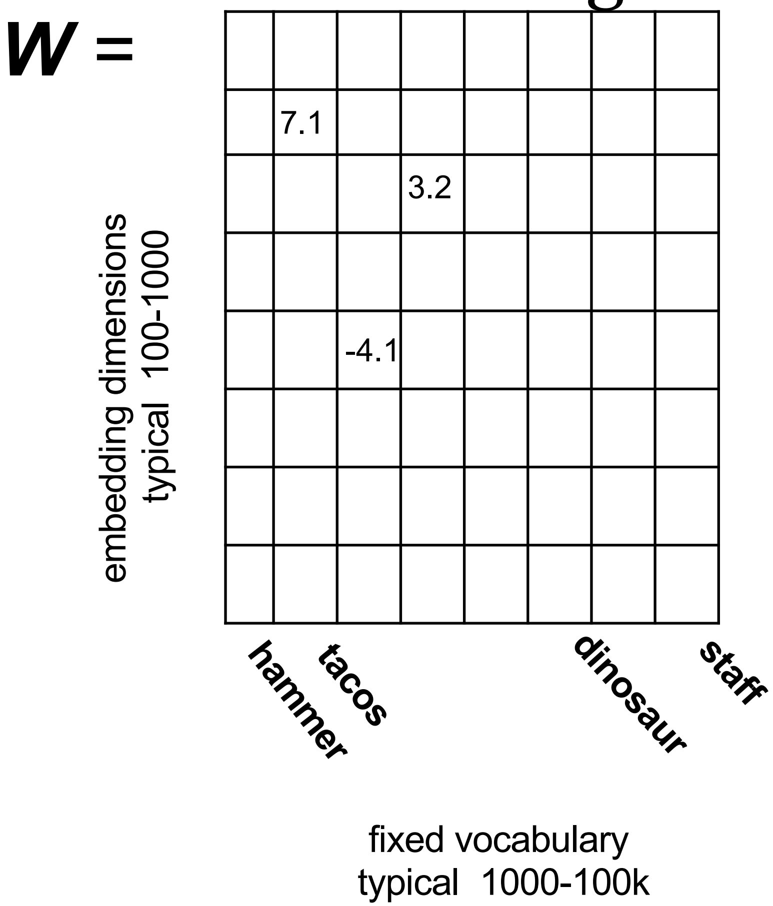
Goal: map each word in vocabulary to an embedding vector
- Preserve semantic meaning in this new vector space



Country-Capital

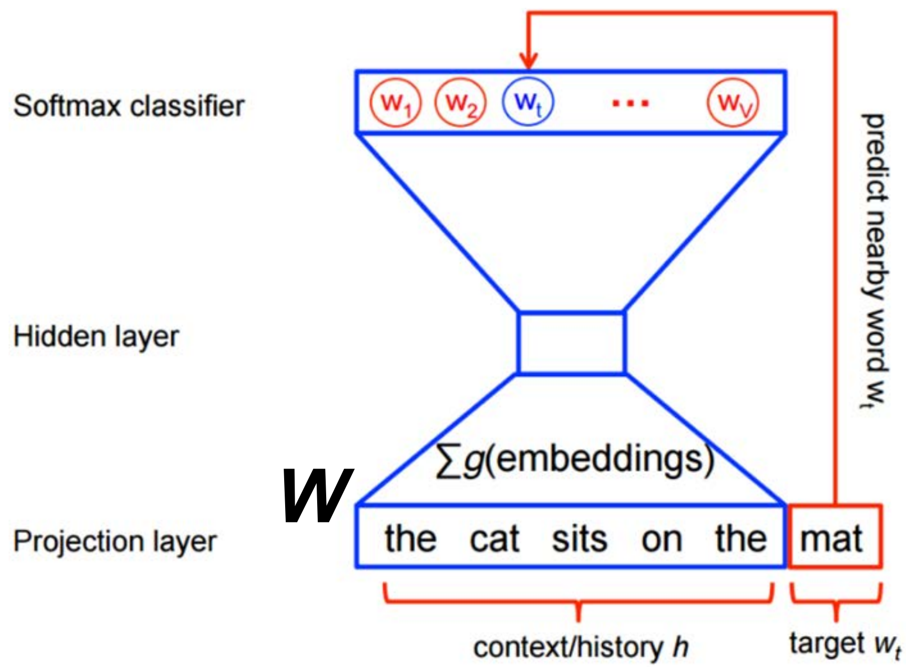
How to embed?

Goal: learn weights



Training

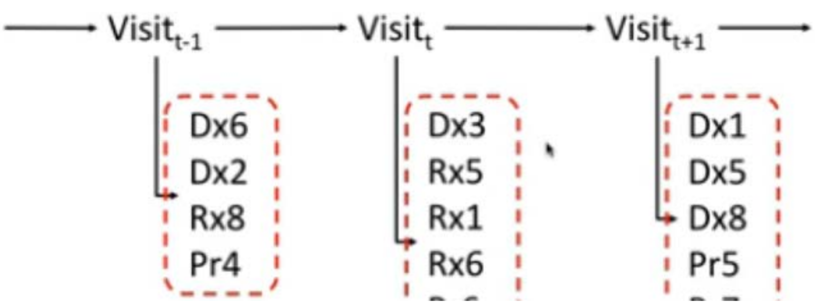
Reward embeddings that predict nearby words in the sentence.



Credit:
https://www.tensorflow.org/tutorials/representation/word2vec

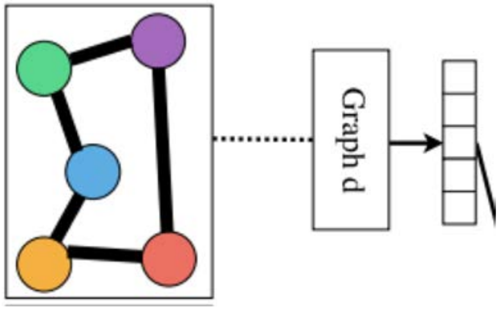
Embeddings Everywhere

- seq2vec
- med2vec



Credit:
[Choi et al. KDD 2016]

- graph2vec
  - https://arxiv.org/abs/1707.05005
  - https://arxiv.org/abs/1805.11921



Credit:
Ivanov & Burnaev ICML 2018