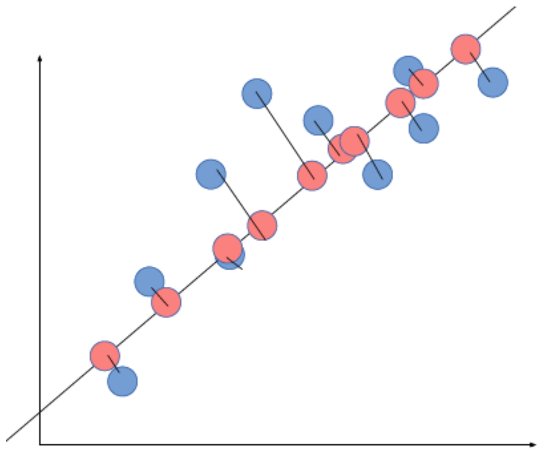
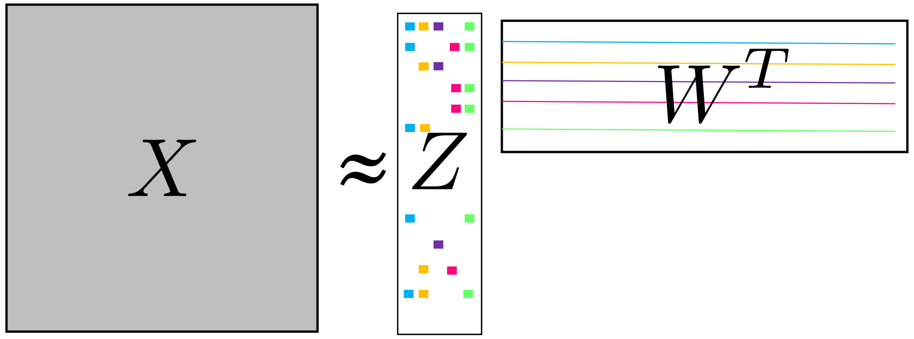
Tufts CS 135: Intro to Machine Learning
https://www.cs.tufts.edu/cs/135/2023f/

Principal Components Analysis (PCA)





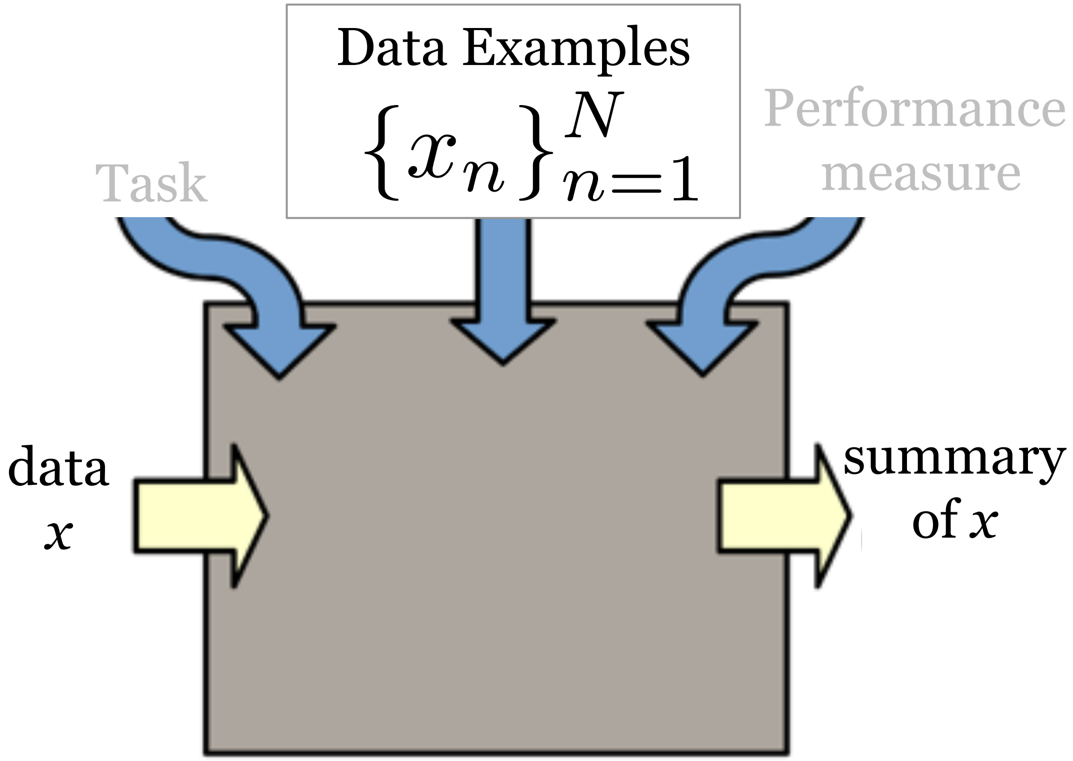
Prof. Mike Hughes

Many ideas/slides attributable to:
Liping Liu (Tufts), Emily Fox (UW)
Matt Gormley (CMU)

What will we learn?

Supervised Learning

Unsupervised Learning

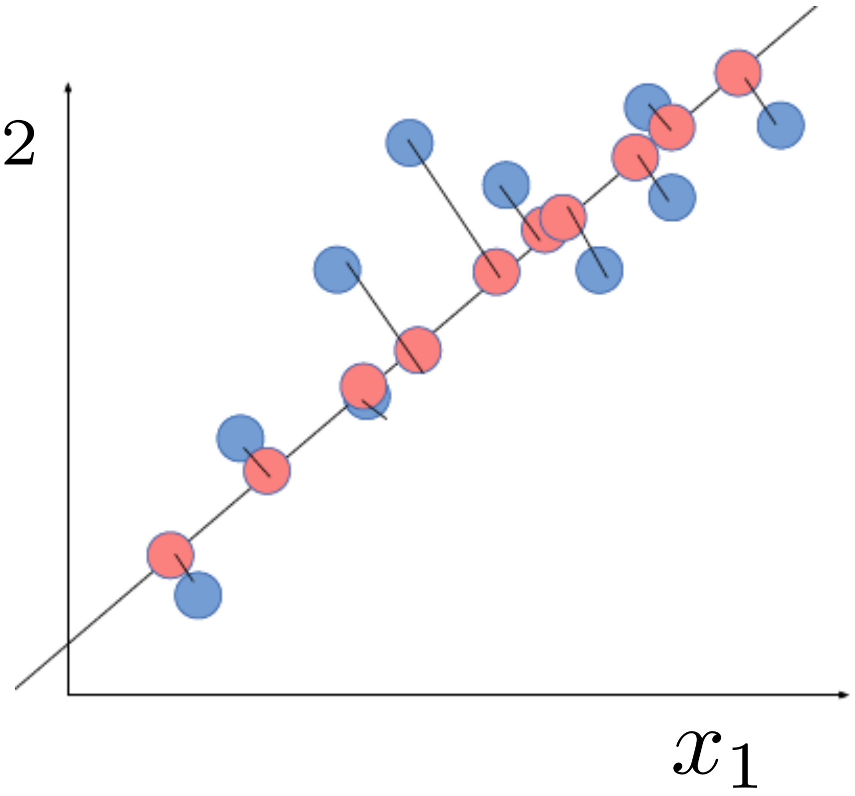


Task: Embedding

Supervised Learning

Unsupervised Learning

embedding



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)
- Evaluation Metrics
  - Storage size - Reconstruction error
  - "Interpretability"

Example Factors

Example: 2D viz. of movies

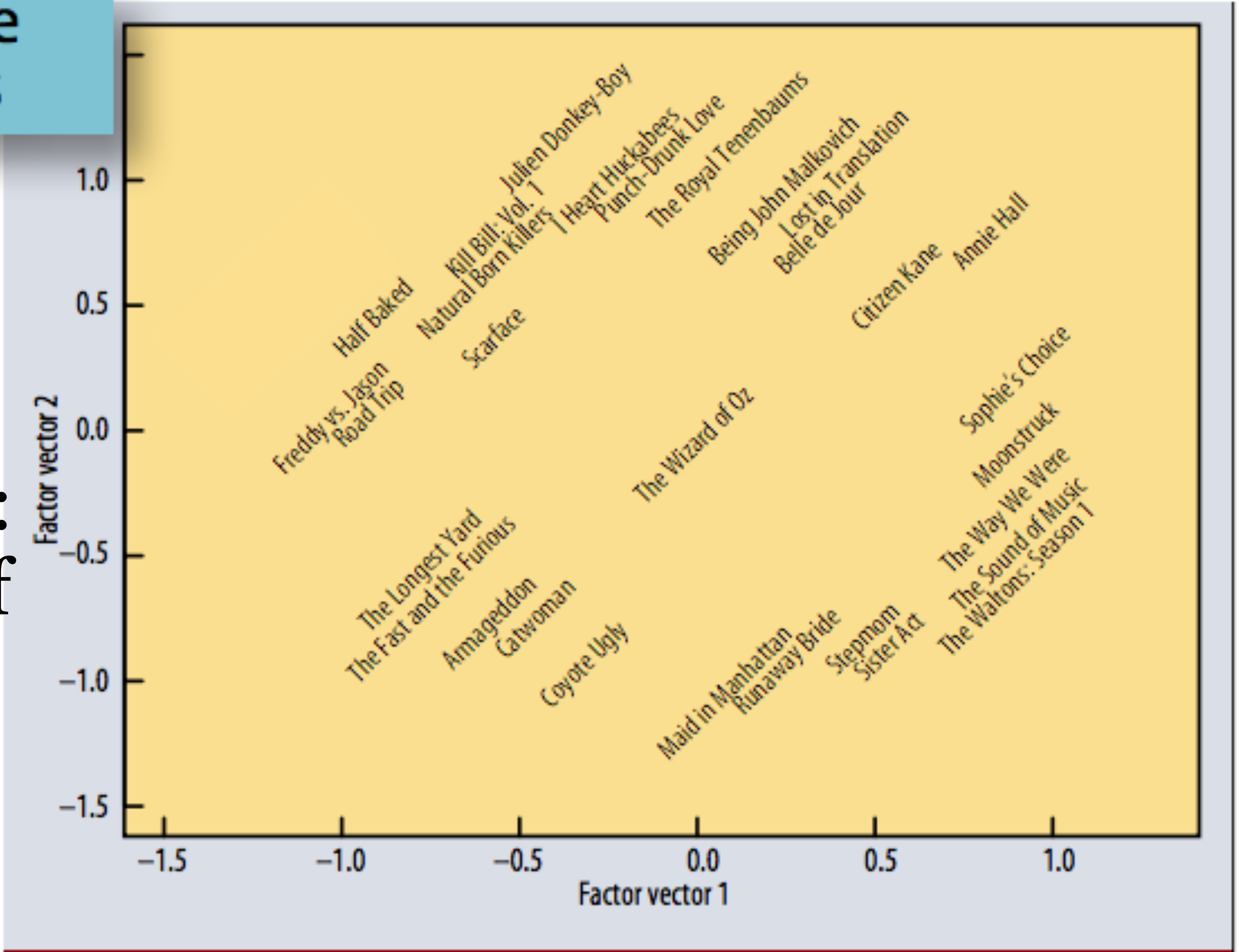
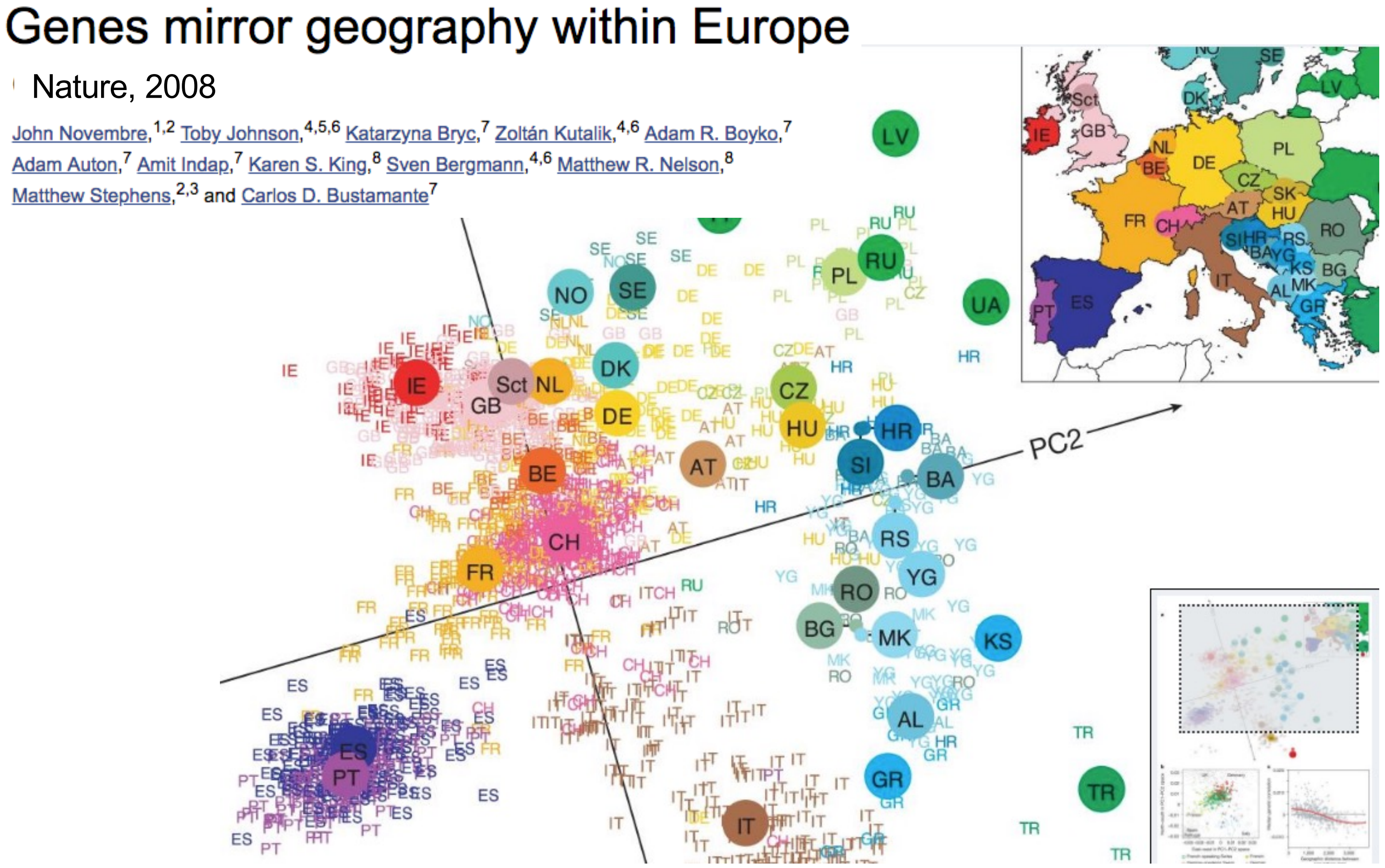


Figure 3. The first two vectors from a matrix decomposition of the Netflix Prize data. Selected movies are placed at the appropriate spot based on their factor vectors in two dimensions. The plot reveals distinct genres, including clusters of movies with strong female leads, fraternity humor, and quirky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

Figure from Koren et al. (2009)

Example: Genes vs. geography



Genes mirror geography within Europe

Nature, 2008

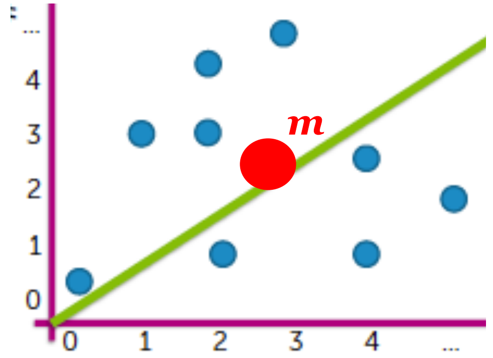
John Novembre,[1,2] Toby Johnson,[4,5,6] Katarzyna Bryc,[7] Zoltán Kutalik,[4,6] Adam R. Boyko,[7] Adam Auton,[7] Amit Indap,[7] Karen S. King,[8] Sven Bergmann,[4,6] Matthew R. Nelson,[8] Matthew Stephens,[2,3] and Carlos D. Bustamante[7]

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

Constant Reconstruction model

$$\hat{\mathbf{x}}_i = m$$

Parameters: m, an F-dim vector



Training problem: Minimize reconstruction error

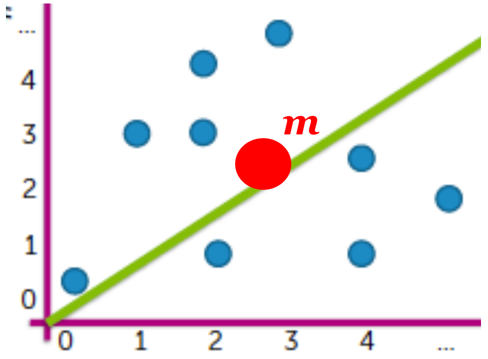
$$\min_{m \in \mathbb{R}^F} \sum_{n=1}^{N} (x_n - m)^T (x_n - m)$$

squared error between two vectors

Constant Reconstruction model

$$\hat{\mathbf{x}}_i = m$$

Parameters: m, an F-dim vector



Training problem: Minimize reconstruction error

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

squared error between two vectors

Optimal parameters:

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

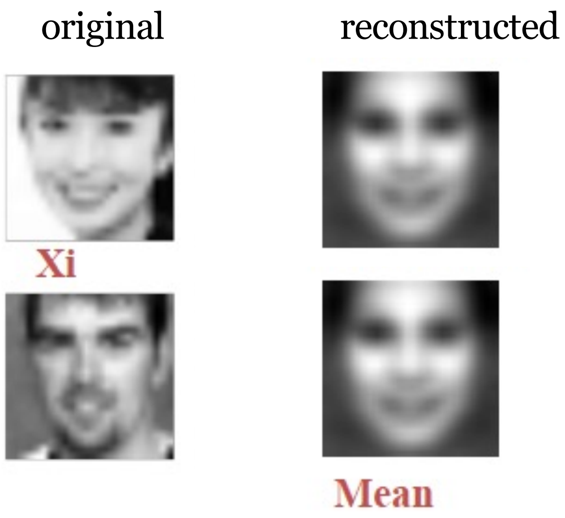
Think of mean vector as optimal "reconstruction" of a dataset if you must use a single vector

Mean reconstruction

Ex: Viola Jones data set

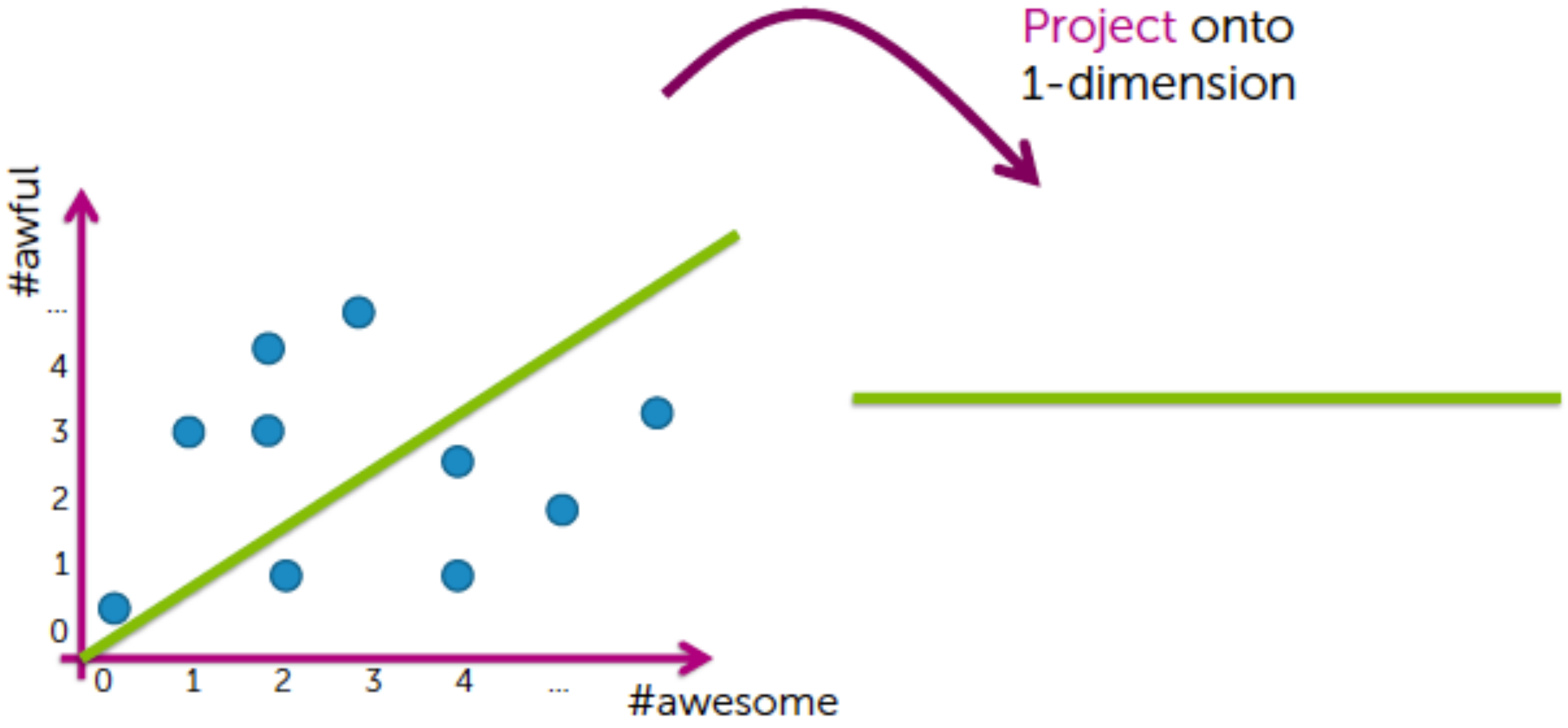
– 24x24 images of faces = 576 dimensional measurements

Mean

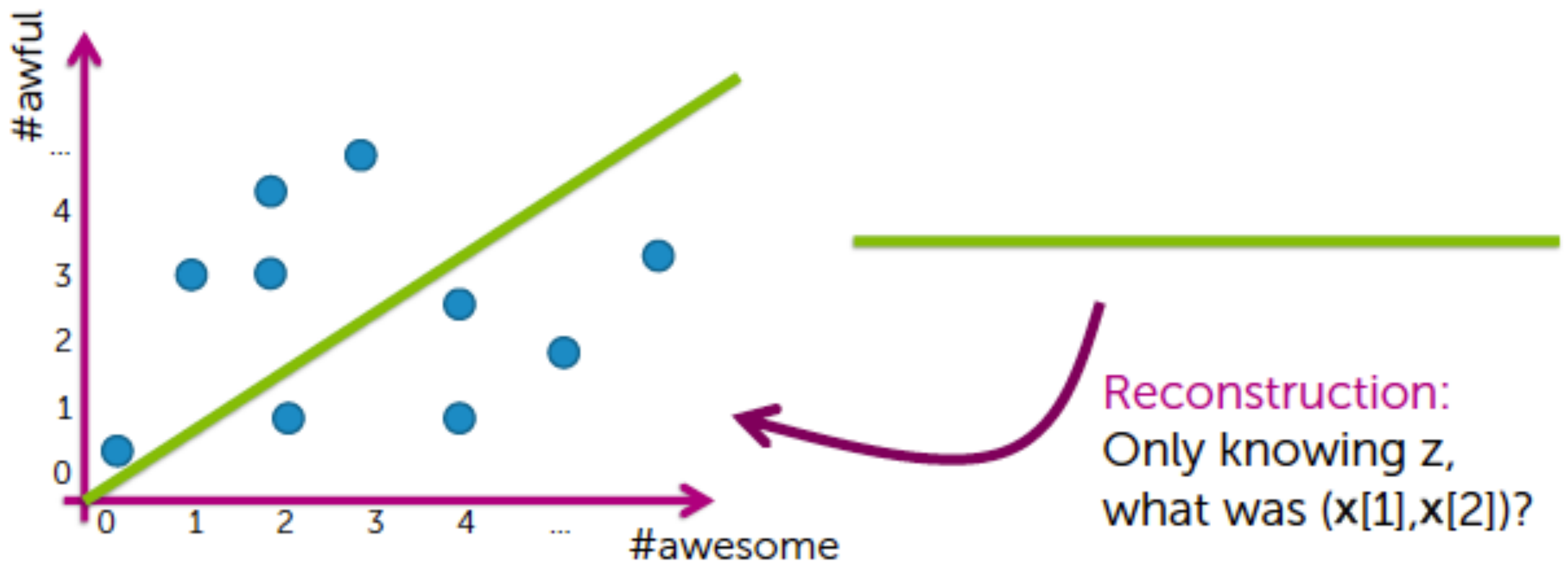


Linear Reconstruction and Principal Component Analysis

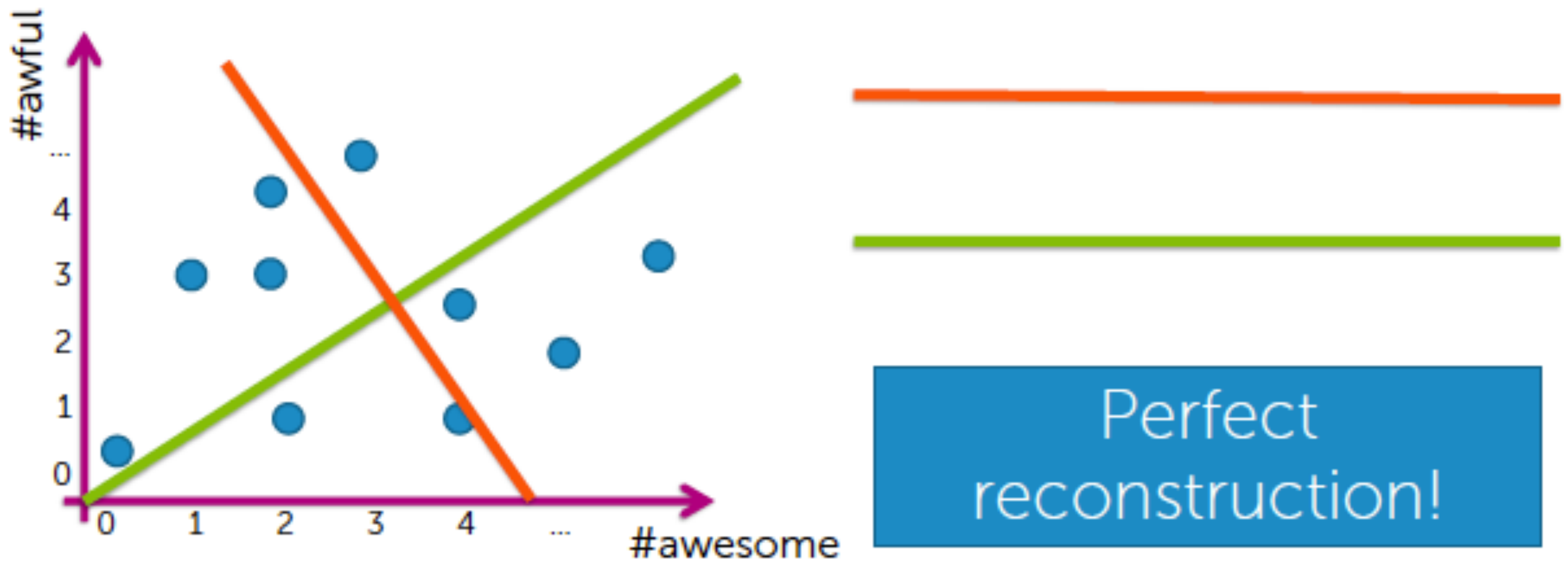
Linear Projection to 1D



Reconstruction from 1D to 2D

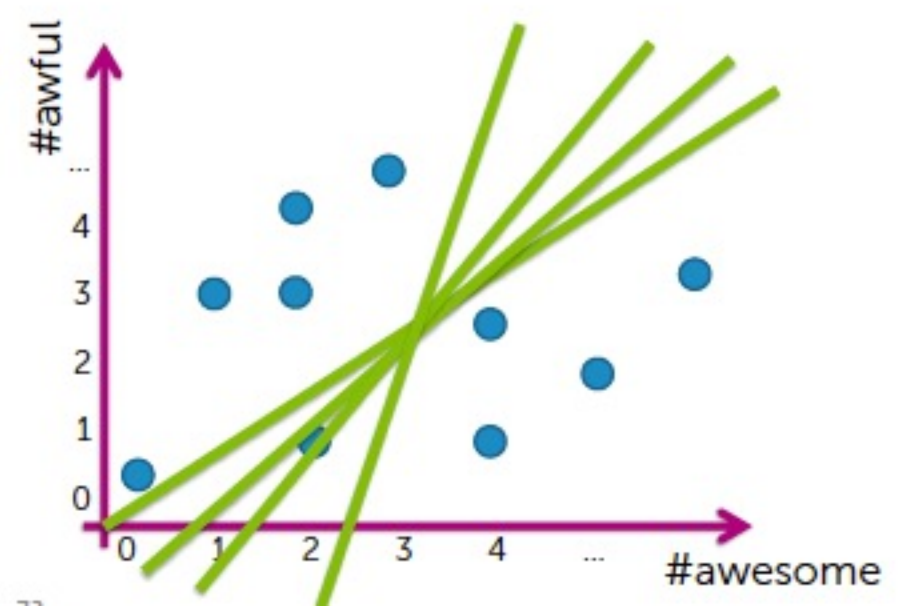


2D Orthogonal Basis



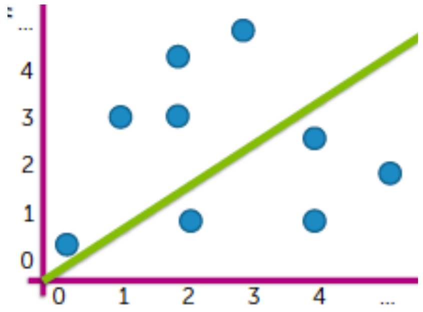
If we could project into 2 dims (same as F), we can perfectly reconstruct

Which 1D projection is best?



Idea: Minimize reconstruction error

Linear Reconstruction Model with 1 components

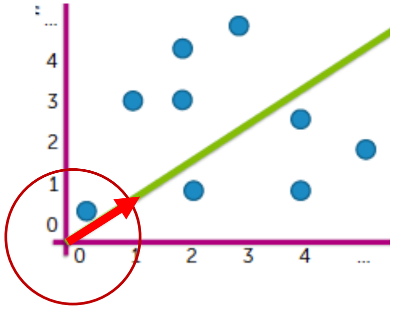


$$\hat{\mathbf{x}}_i = \mathbf{w} z_i + \mathbf{m}$$

$\hat{\mathbf{x}}_i$	$\mathbf{w}$	z_i	$\mathbf{m}$
Fx1	F x 1	1 x 1	F x 1
High-dim. data	Weights	Low-dim embedding or "score"	"mean" vector

Linear Reconstruction Model with 1 components

$$\hat{\mathbf{x}}_i = \mathbf{w} z_i + \mathbf{m}$$



W is a vector on unit circle. Magnitude is always 1.

Problem: "Over-parameterized". Too many possible solutions!

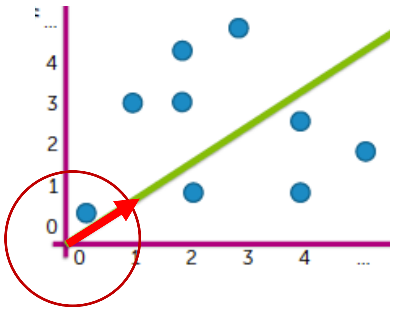
Suppose we have an alternate model with weights w' and embedding z'
We would get equivalent reconstructions if we set:

- w' = w * 2
- z' = z / 2

Solution: Constrain magnitude of w.
w is a unit vector. We care about direction, not scale.

$$\sum_{f=1}^{F} w_f^2 = 1$$

Linear Reconstruction Model with 1 components



$$\hat{\mathbf{x}}_i = \mathbf{w} z_i + \mathbf{m}$$

Fx1 F x 1 1 x 1 F x 1

W is a vector on unit circle. Magnitude is always 1.

Given fixed weights w and a specific x, what is the optimal scalar z value?

Minimize reconstruction error!

$$\min_{z \in \mathbb{R}} \quad (\mathbf{x} - (\mathbf{w}z + \mathbf{m}))^2$$

Exact analytical solution (take gradient, set to zero, solve for z) gives:

$$z = w^T(x - m)$$

Projection of feature vector x onto vector w after "centering" (removing the mean)

Linear Reconstruction Model with K components

$$\hat{\mathbf{x}}_i = \mathbf{W}\mathbf{z}_i + \mathbf{m}$$

F x 1	F x K	K x 1	F x 1
High-dim. data	Weights	Low-dim vector	Mean of data vector

Each of the K weight vectors $\boldsymbol{w}_{\mathrm{k}}$ is one "component".

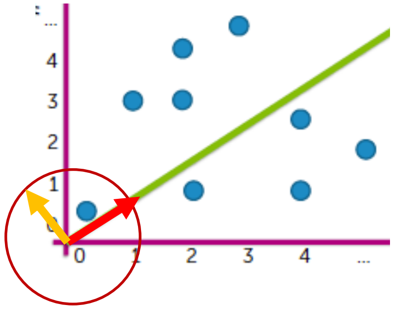
$$W = \begin{bmatrix} | & | & & | \\ \mathbf{w}_1 & \mathbf{w}_2 & \ldots & \mathbf{w}_{\mathbf{K}} \\ | & | & & | \end{bmatrix}$$

Our goal is to find the K weight vectors that best reconstruct our training dataset

$$\min_{W \in \mathbb{R}^{F \times K}} \quad \sum_{n=1}^{N} \sum_{f=1}^{F} (x_{nf} - \hat{x}_{nf}(W))^2$$

Solving this squared error reconstruction objective is known as principal components analysis (PCA)

Linear Reconstruction Model with K components



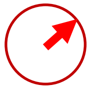
$$\hat{\mathbf{x}}_i = \mathbf{W}\mathbf{z}_i + \mathbf{m}$$

F x 1	F x K	K x 1	F x 1
High-dim. data	Weights	Low-dim vector	Mean of data vector

$$W = \begin{bmatrix} | & | & & | \\ \mathbf{w}_1 & \mathbf{w}_2 & \dots & \mathbf{w}_\mathbf{K} \\ | & | & & | \end{bmatrix}$$

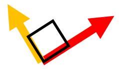
We will ***require*** that:

- (1) All weight vectors are unit vectors
  - This fixes scale and avoid several W with same error



$$\mathbf{w}_k^T \mathbf{w}_k = 1 \quad \rightarrow \sum_{f=1}^F W_{fk}^2 = 1$$

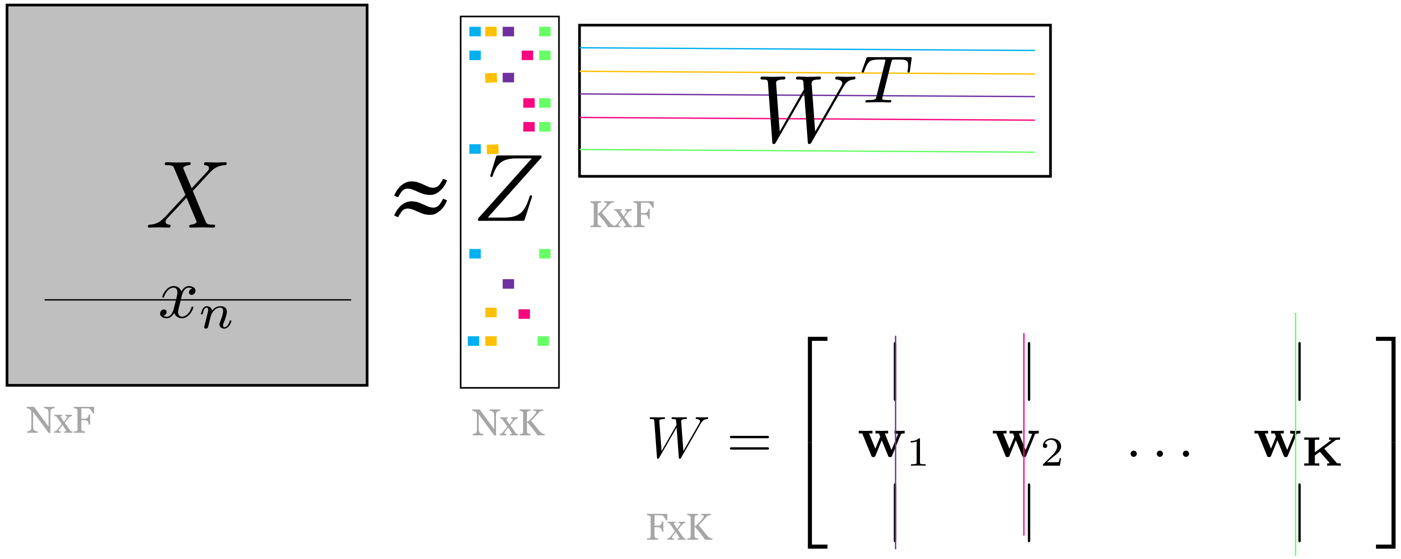
- (2) Component directions are *orthogonal (perpendicular)*
  - Avoids information redundancy in W's components



$$\mathbf{w}_j^T \mathbf{w}_k = 0 \quad \rightarrow \sum_{f=1}^F W_{fj} W_{fk} = 0 \quad \forall j \neq k$$

Weights that satisfy (1) and (2) form an "orthonormal basis"

View: PCA as Matrix Factorization

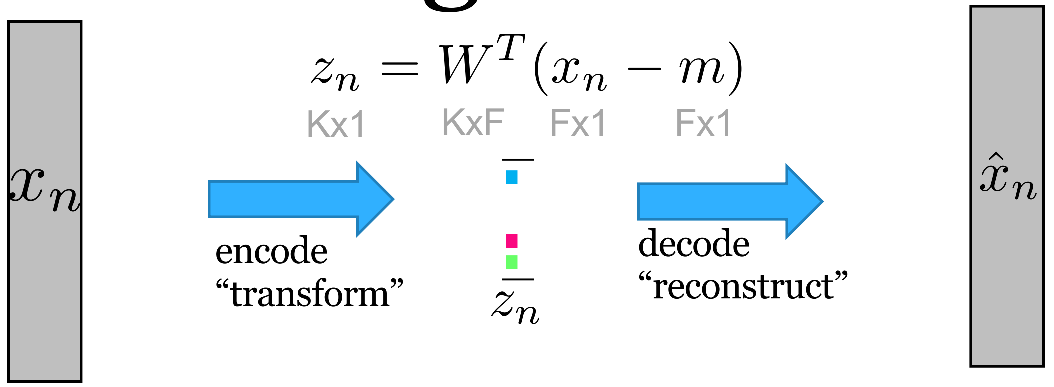


X
x_n
NxF
$\approx Z$
NxK
W^T
KxF
$W = \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \ldots & \mathbf{w}_\mathbf{K} \end{bmatrix}$
FxK

View: Encoding and Decoding

$$z_n = W^T (x_n - m)$$

Kx1 KxF Fx1 Fx1



$$\hat{\mathbf{x}}_i = \mathbf{W}\mathbf{z}_i + \mathbf{m}$$

Principal Component Analysis

Transformation step

What happens when you call `pca.transform(x_QF)`

Input:

- X : query data, Q x F
  - Q examples of high-dim. feature vectors
- Trained PCA parameters (contained inside `pca`)
  - m : mean vector, size F
  - W : learned basis of weight vectors, F x K

Output:

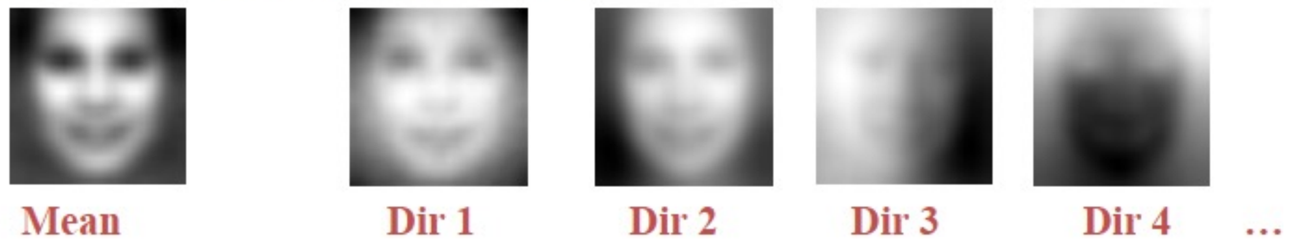
- Z : projections, N x K
  - Each row Z[n] is a low-dim. “embedding” (size K) of X[n]

$$z_n = W^T(x_n - m)$$

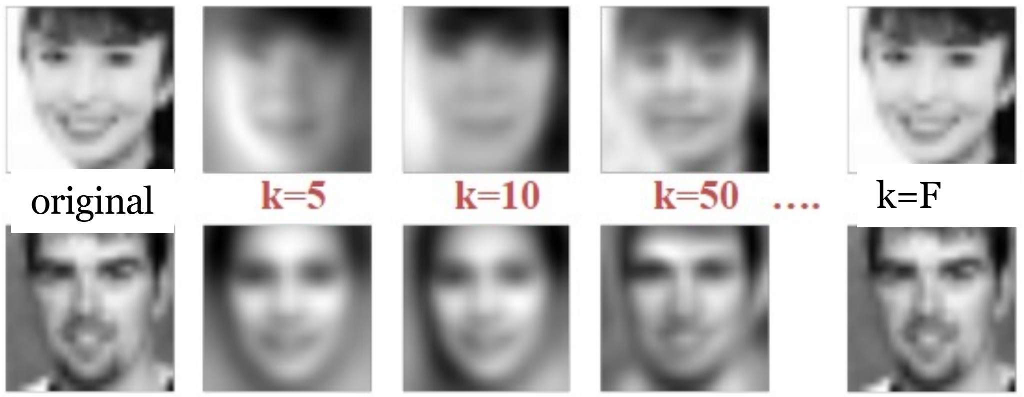
Example: PCA on Faces

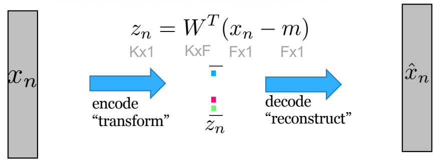
Ex: Viola Jones data set

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



Projecting data onto first k dimensions





If we use all possible components, we *perfectly reconstruct* original data

Principal Component Analysis

Training step : What happens when we call `pca.fit(x_NF)`

Input:

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

$$\min_{m \in \mathbb{R}^F, W \in \mathbb{R}^{F \times K}} \sum_{n=1}^{N} \sum_{f=1}^{F} (x_{nf} - \hat{x}_{nf}(m, W))^2$$

$$\text{subject to: } \quad W^T W = I_K$$

Orthonormal Constraint

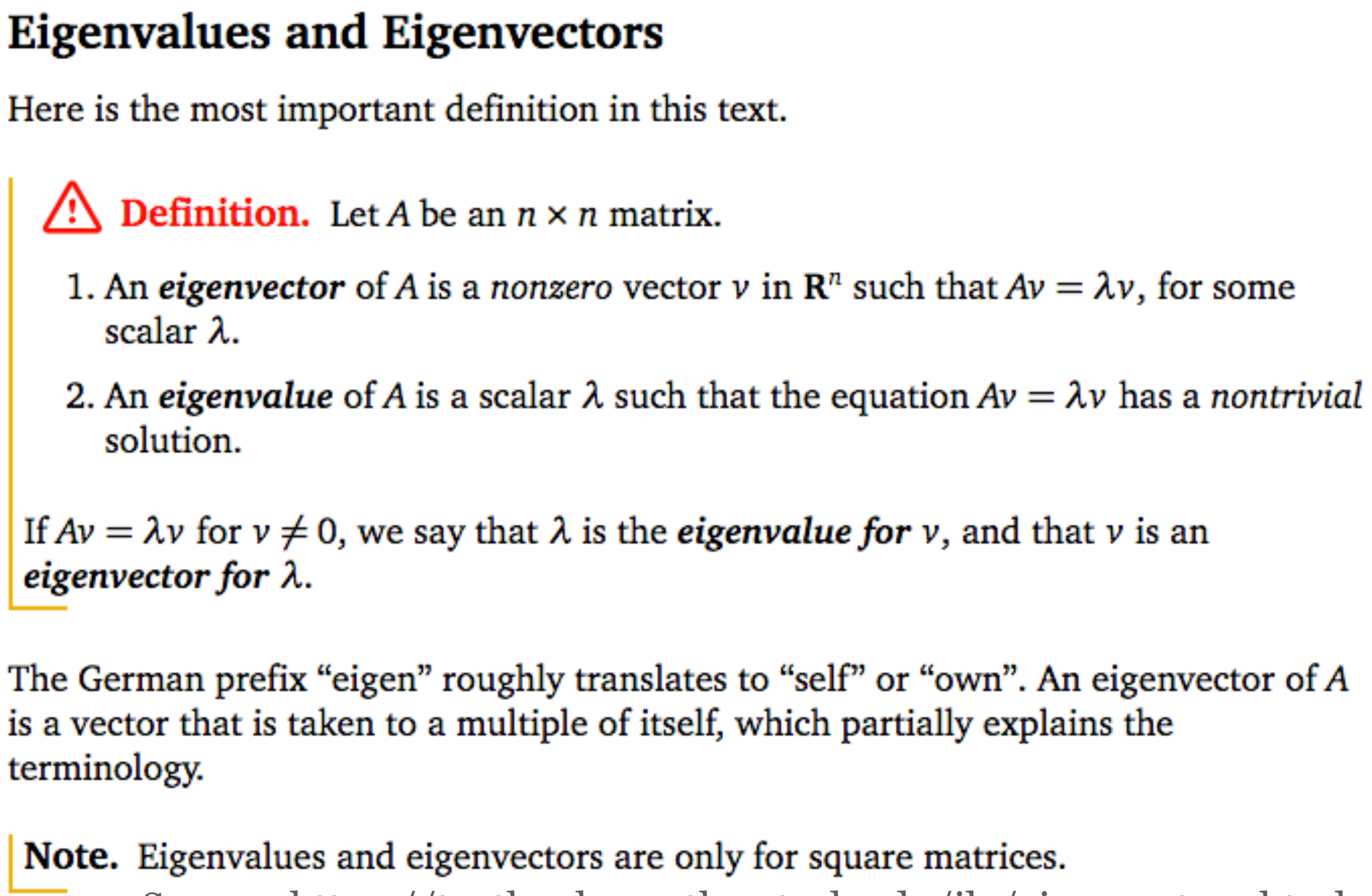
- Each vec has magnitude 1

- Vectors are orthogonal

Output: *Trained parameters for PCA*

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

Recall from Linear Algebra



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

The weight component vectors are the eigenvectors of the covariance matrix of the centered dataset

$$S = \frac{1}{N} \sum_{n=1}^{N} (x_n - m)(x_n - m)^T$$

Every principal component vector w_k satisfies this equation:

$$S w_k = \lambda_k w_k$$

$$W = \begin{bmatrix} | & | & & | \\ \mathbf{w}_1 & \mathbf{w}_2 & \dots & \mathbf{w}_\mathbf{K} \\ | & | & & | \end{bmatrix}$$

When we fit K principal components to a dataset, the optimal ones (that minimize reconstruction error) are those with the K largest eigenvalues.

Can use standard linalg libraries to compute the eigenvalues/vectors!

PCA Principles

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

$$\min_{m \in \mathbb{R}^F, W \in \mathbb{R}^{F \times K}} \quad \sum_{n=1}^{N} \sum_{f=1}^{F} (x_{nf} - \hat{x}_{nf}(m, W))^2$$

$$\text{subject to:} \quad W^T W = I_K$$

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

PCA: How to Select K?

- 1) Use downstream supervised task metric
  - e.g. regression error, classifier AUROC
- 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

Assume we've computed the empirical mean vector: $m \triangleq \frac{1}{N} \sum_{n=1}^{N} x_n$

Empirical variance is defined as averaged squared error from the empirical mean:

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

$$= \frac{1}{N} \sum_{n=1}^{N} (x_n - m)^T (x_n - m)$$

Empirical Variance of reconstructions

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

Assume we have already zero-centered the features

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

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

Simplify with lots of linear algebra

$$\frac{1}{N}\sum_{n} \mathrm{tr}(x_n^T W W^T x_n)$$
$$= \mathrm{tr}(W^T S W)$$
$$= \sum_{k} w_k^T S w_k$$
$$= \sum_{k} \lambda_k w_k^T w_k$$

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

Just sum up the top K eigenvalues!

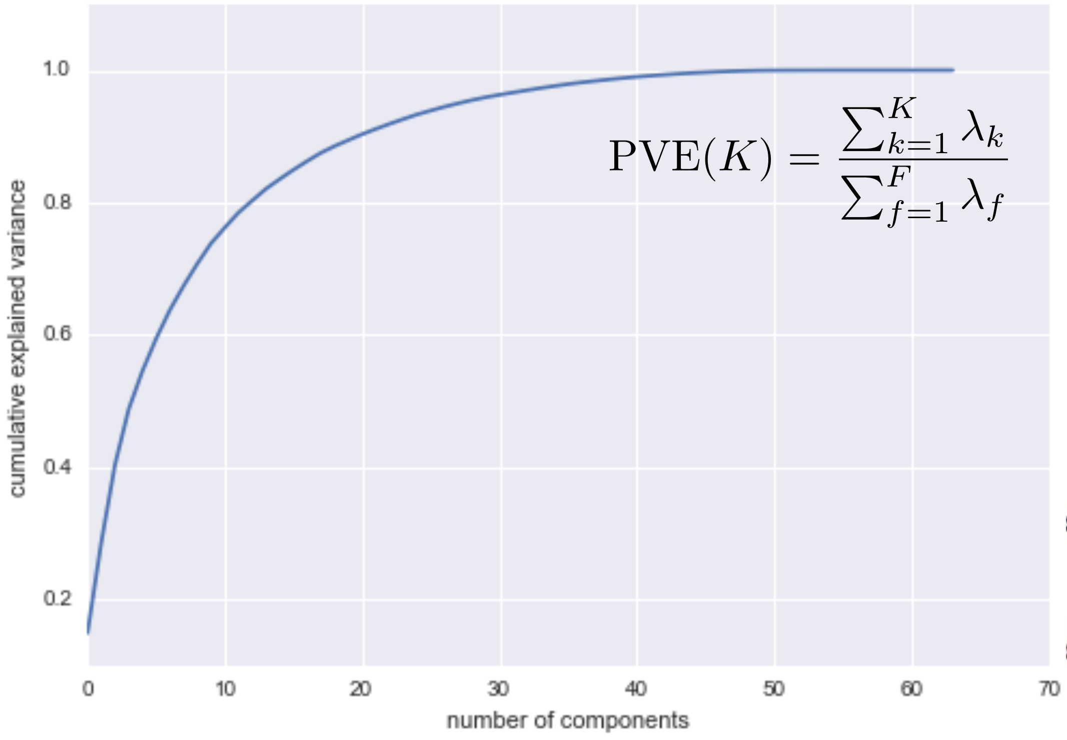
Proportion of Variance Explained by first K components

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

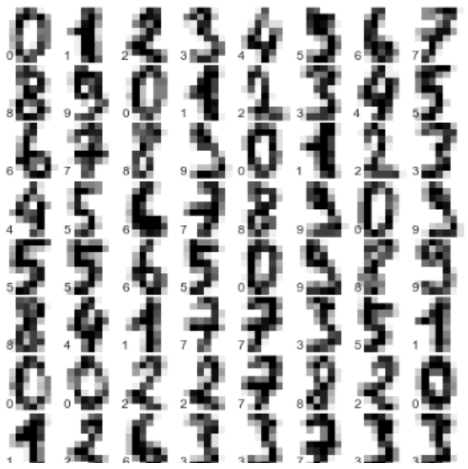
Goal: Want K value where proportion of variance explained is large.
Indicates good reconstruction ability on our training set.

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');
```



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



```
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

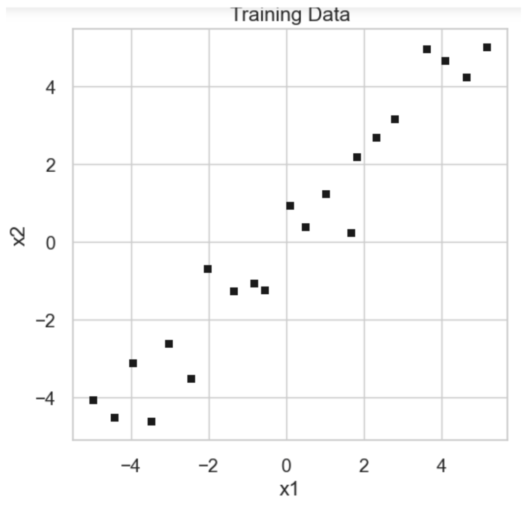
- If features have same units, may not need to do this

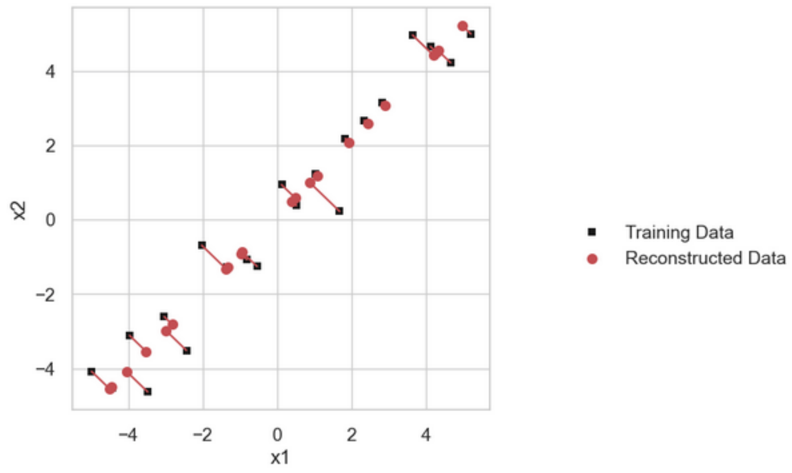
Dim. Reduction/Embedding
Unit Objectives

- Goals of dimensionality reduction
  - Reduce feature vector size (keep signal, discard noise)
  - “Interpret” features: visualize/explore/understand
- Common approaches
  - Principal Component Analysis (PCA)
  - word2vec and other non-linear embeddings
- Evaluation Metrics
  - Storage size - Reconstruction error
  - “Interpretability”

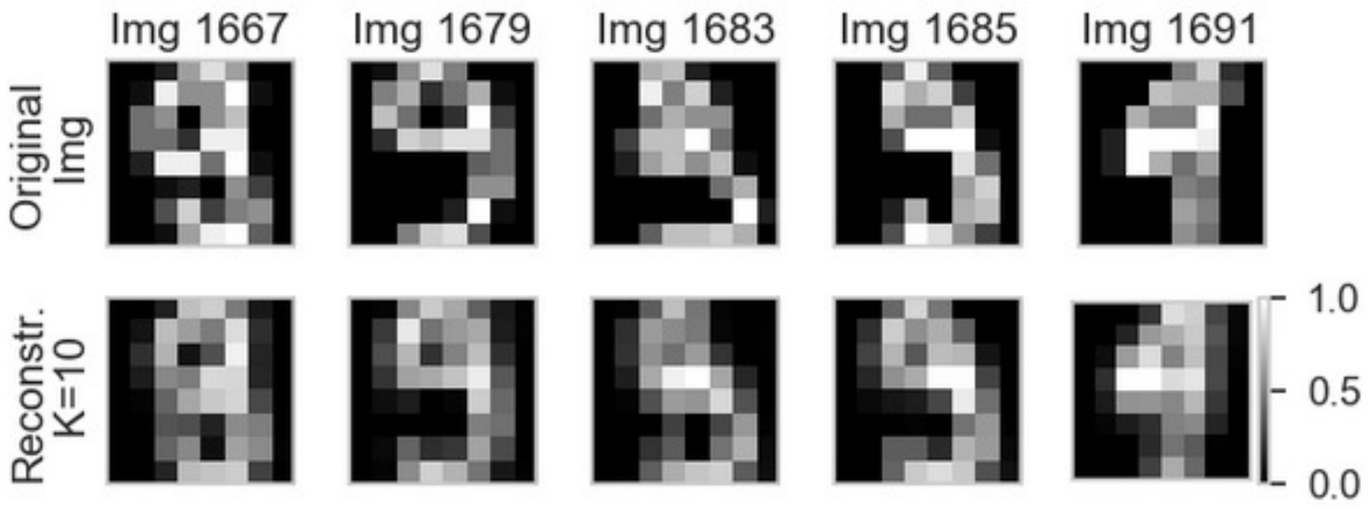
Lab

- Part 1:





- Part 2


Img 1667
Img 1679
Img 1683
Img 1685
Img 1691
Original Img
Reconstr. K=10