Lecture 11
Principal Component Analysis

EE-UY 4563 / EL-GY 9123: INTRODUCTION TO MACHINE LEARNING

PROF. SUNDEEP RANGAN, WITH MODIFICATION BY YAO WANG AND YUAN WANG

Goal: find a “good ” representation

- Sufficient (for the task): I(data; task) = I(representation; task)
- Minimal (information): I(data; representation) is minimal
- Invariant (to nuisances): I(nuisances; task) = 0

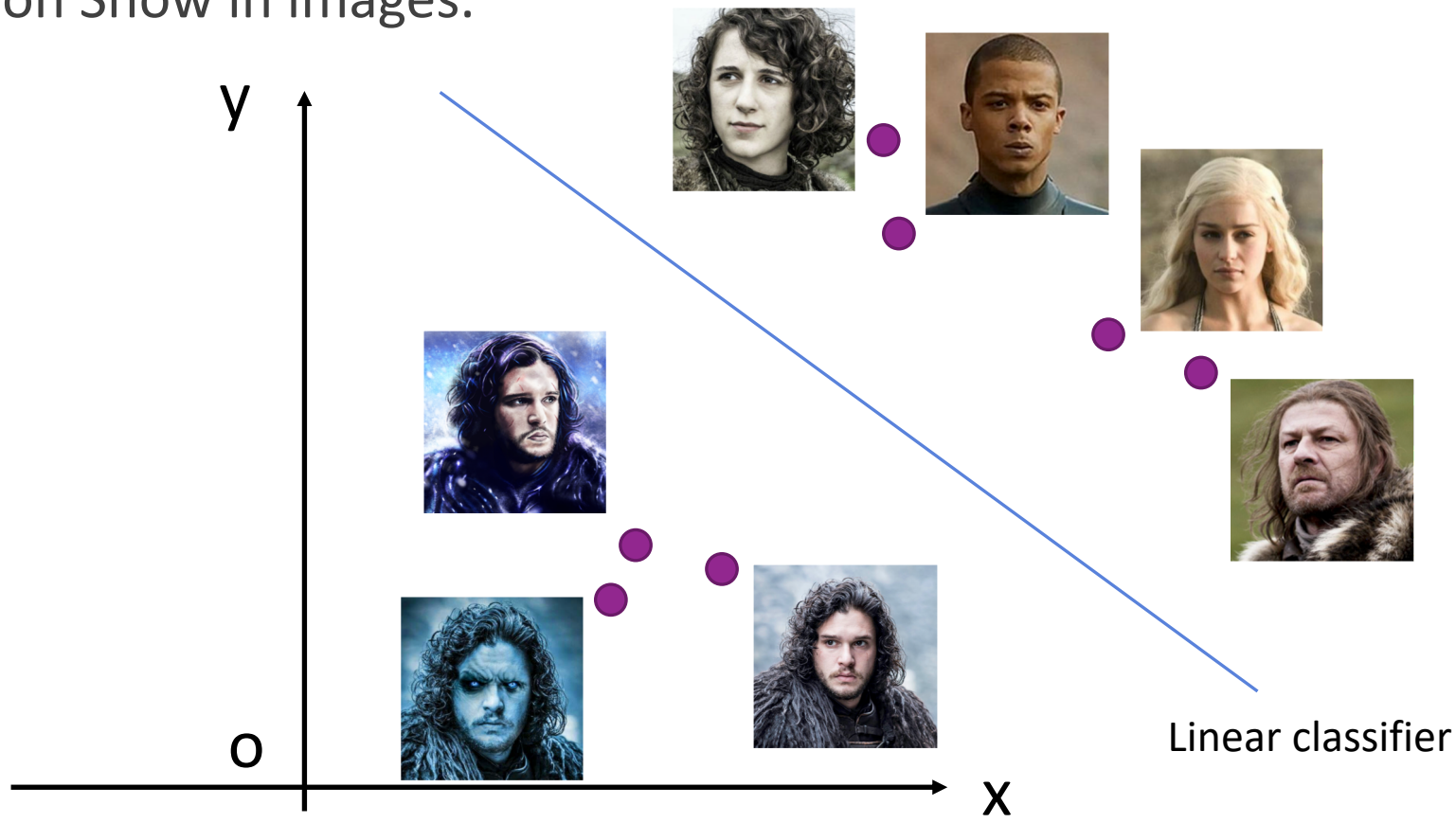
Goal: find a “good ” representation

- Example task: identify Jon Snow in images.

Images from the Internet

Goal: find a “good ” representation

- Example task: identify Jon Snow in images.
  - Sufficient
  - Minimal
  - Invariant



Goal: find a “good ” representation

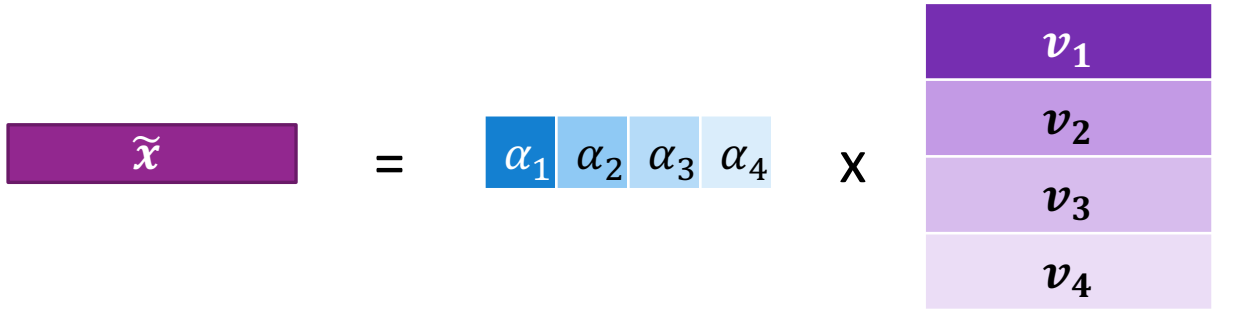
- Today we focus on the simplest linear case:

sample coefficient Dictionary / bases

Goal: find a “good ” representation

- Today we focus on the simplest linear case:

$$\tilde{\boldsymbol{x}} = \boldsymbol{\alpha} V = \sum_i \alpha_i \boldsymbol{v}_i$$

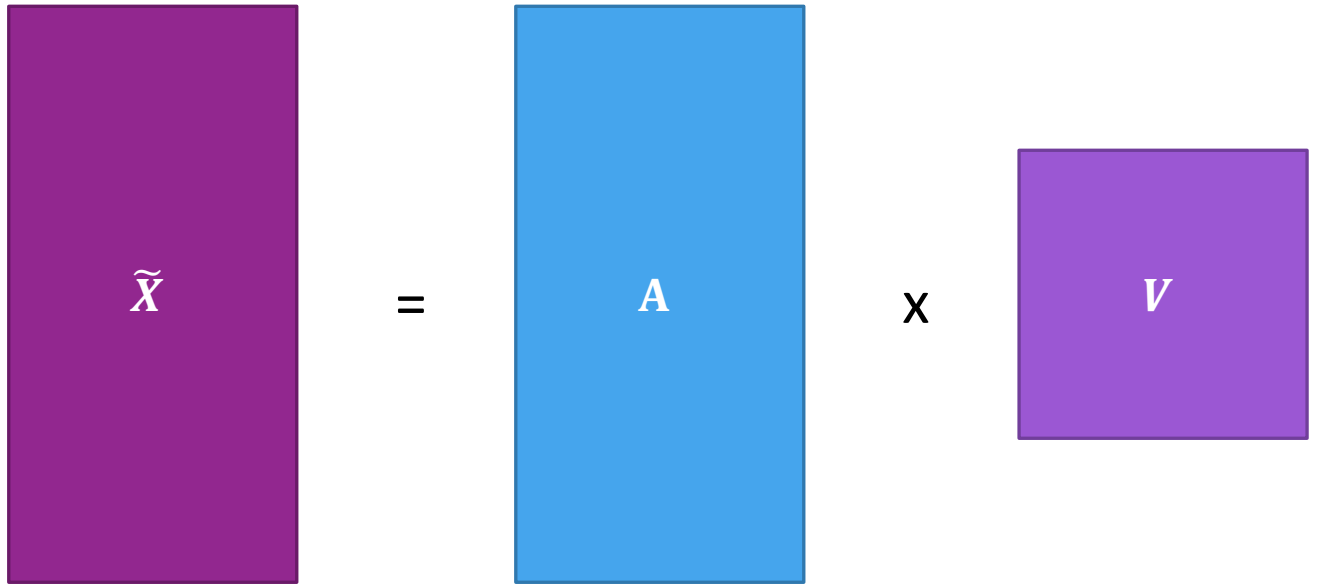


sample	coefficient	Dictionary / bases
$p(\tilde{\boldsymbol{x}})$	$p(\alpha)$	$V(p)$

Goal: find a “good ” representation

- Today we focus on the simplest linear case:

$$\tilde{X} = AV$$



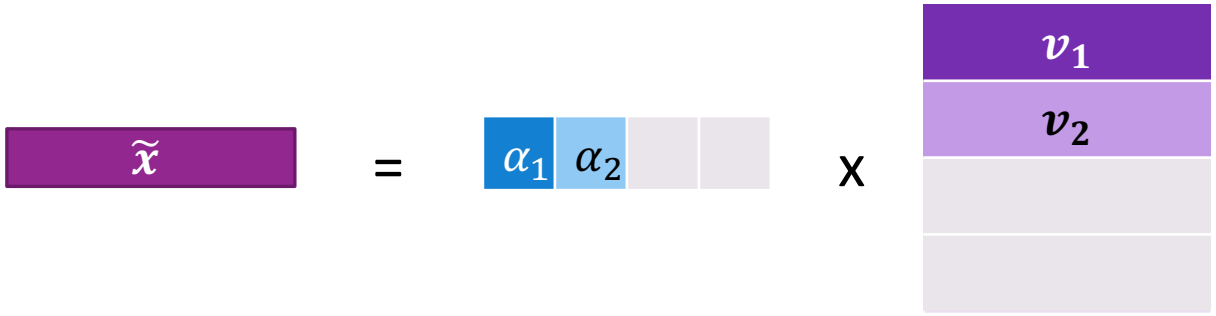
Data(many samples in rows) coefficients Dictionary / bases

Goal: find a “good ” representation

- Today we focus on the simplest linear case:
- Wanted properties:
  - $\tilde{\boldsymbol{x}} = \boldsymbol{\alpha} V$
  - $\boldsymbol{\alpha}$ to be short
    - $\boldsymbol{v}$ to be typical patterns of $\tilde{\boldsymbol{x}}$
  - $\boldsymbol{\alpha}$ is easy to compute
    - $\boldsymbol{\alpha} = \boldsymbol{x}\text{“}V^{-1}\text{”}$
    - $\boldsymbol{\alpha} = \boldsymbol{x}V^T \Leftrightarrow VV^T = I$

$$\underset{V}{\operatorname{argmin}} \min_{\boldsymbol{\alpha}} \mathbb{E}(\|\tilde{\boldsymbol{x}} - \boldsymbol{\alpha} V\|_2)$$

$$s.t. VV^T = I, |\alpha| = d \leq |\tilde{\boldsymbol{x}}| = p$$



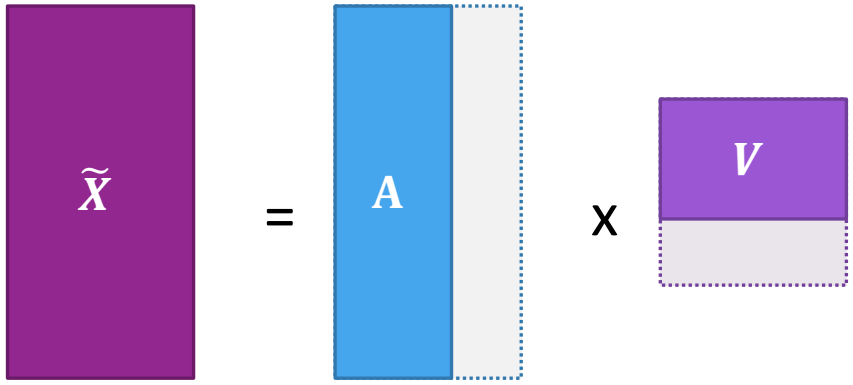
Goal: find a “good ” representation

- Today we focus on the simplest linear case:
- Wanted properties:
  - $\tilde{\boldsymbol{x}} = \boldsymbol{\alpha} V$
  - $\boldsymbol{\alpha}$ to be short
    - $\boldsymbol{v}$ to be typical patterns of $\tilde{\boldsymbol{x}}$
  - $\boldsymbol{\alpha}$ is easy to compute
    - $\boldsymbol{\alpha} = \boldsymbol{x}$“$V^{-1}$”
    - $\boldsymbol{\alpha} = \boldsymbol{x}V^T \Leftrightarrow VV^T = I$

$$\underset{V}{\operatorname{argmin}} \min_{A} \left\| \tilde{X} - AV \right\|_F$$

$$s.t. VV^T = I$$

$$\tilde{X} \in \mathbb{R}^{N \times p}, A \in \mathbb{R}^{N \times d}, V \in \mathbb{R}^{d \times p}, d \leq p$$



Outline

- ❑ Dimensionality reduction
- ❑ Principal components and directions of variance (a geometric perspective)
- ❑ Computing PCs via the SVD
- ❑ Approximation with PCs
- ❑ Face example in python
- ❑ Extensions

Dimensionality Reduction

- Many modern data sets have very high dimension
- Want to reduce dimension:
  - The Curse of Dimensionality (task “difficulty” ~ exp(d))
  - Visualize data
  - Find underlying commonalities in data

Outline

- ❑ Dimensionality reduction
- ❑ Principal components and directions of variance (a geometric perspective)
- ❑ Computing PCs via the SVD
- ❑ Approximation with PCs
- ❑ Face example in python
- ❑ Extensions

Data Definitions

- $\boldsymbol{x}$ is a random variable $\boldsymbol{x} = [x_1, x_2, \dots, x_p]$
- Data points $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\}$ are independent & random samples drawn from a distribution
- Data can be represented as an $N \times p$ matrix X
- No label: unsupervised learning

- $\tilde{\boldsymbol{x}}$ is a demeaned version of $\boldsymbol{x}$

Simple properties of $\tilde{\boldsymbol{x}}$

- $\tilde{\boldsymbol{x}}$ is a demeaned version of $\boldsymbol{x}$
- Expectation $\boldsymbol{\mu}$:
  - $\boldsymbol{\mu} = \mathbb{E}(\tilde{\boldsymbol{x}}) = \mathbb{E}(\boldsymbol{x} - \overline{\boldsymbol{x}}) = \mathbb{E}(\boldsymbol{x}) - \mathbb{E}(\overline{\boldsymbol{x}}) = 0$
- Covariance Σ:
  - $\Sigma = cov(\tilde{\boldsymbol{x}}) = \mathbb{E}\big((\tilde{\boldsymbol{x}} - \boldsymbol{\mu})^T(\tilde{\boldsymbol{x}} - \boldsymbol{\mu})\big) = \mathbb{E}(\tilde{\boldsymbol{x}}^T\tilde{\boldsymbol{x}})$
  - $\Sigma \approx \frac{1}{N}\Sigma_{n=1}^{N} \tilde{\boldsymbol{x}}_n{}^T \tilde{\boldsymbol{x}}_n = \frac{1}{N}\tilde{X}^T\tilde{X} \equiv Q$

Sample covariance

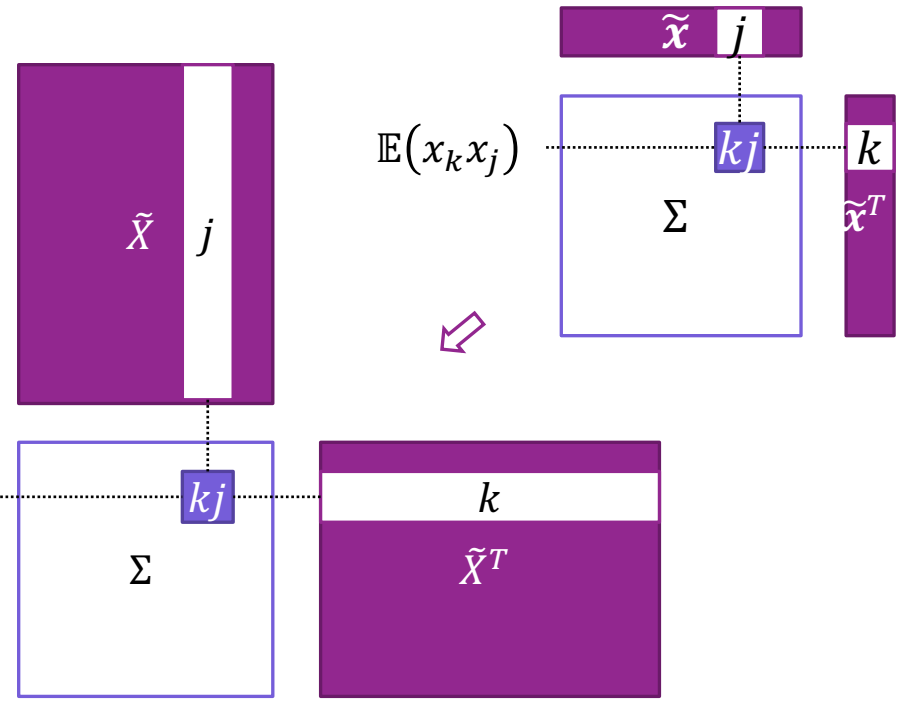
Simple manipulation

$\frac{1}{N}\Sigma_{n=1}^{N} x_{nk}x_{nj} = \frac{1}{N}\tilde{X}_k \cdot \tilde{X}_j$

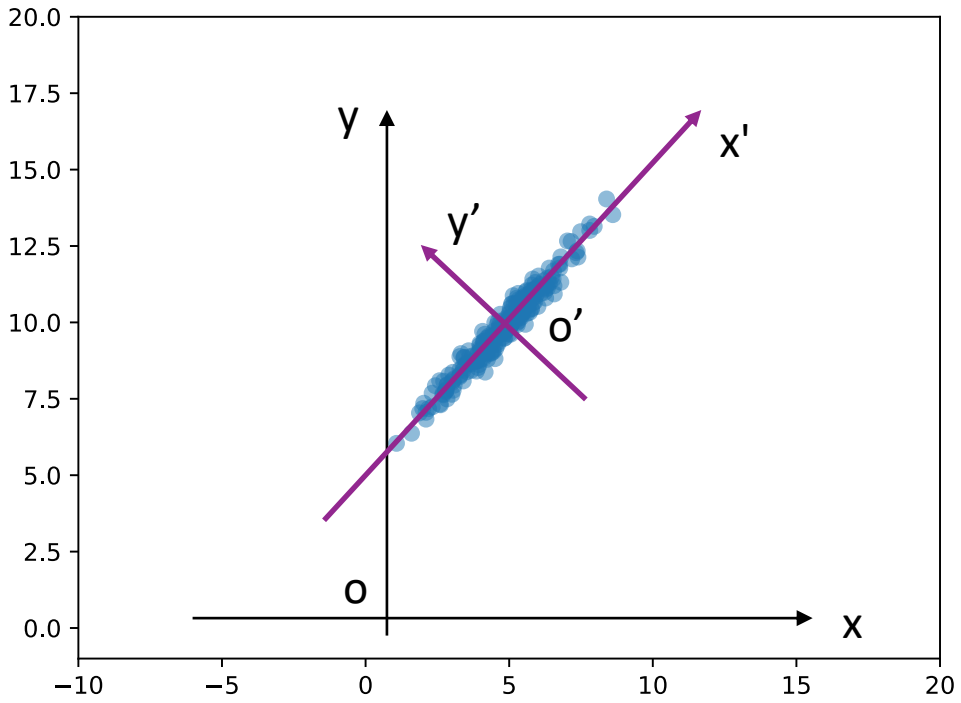
Sampling approximation:

$\mathbb{E}\big(f(\boldsymbol{x})\big) = \int p(\boldsymbol{x}) f(\boldsymbol{x}) d\boldsymbol{x} \approx \frac{1}{N}\Sigma_n f(\boldsymbol{x}_n)$

Here $f(\tilde{\boldsymbol{x}}) = \tilde{\boldsymbol{x}}^T\tilde{\boldsymbol{x}}$

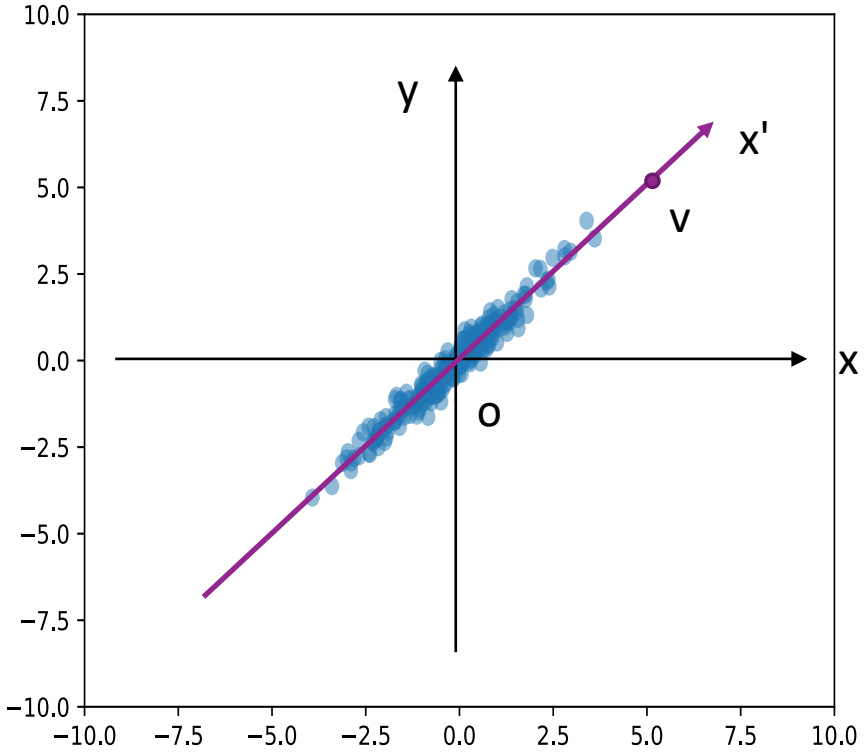


Geometric intuition



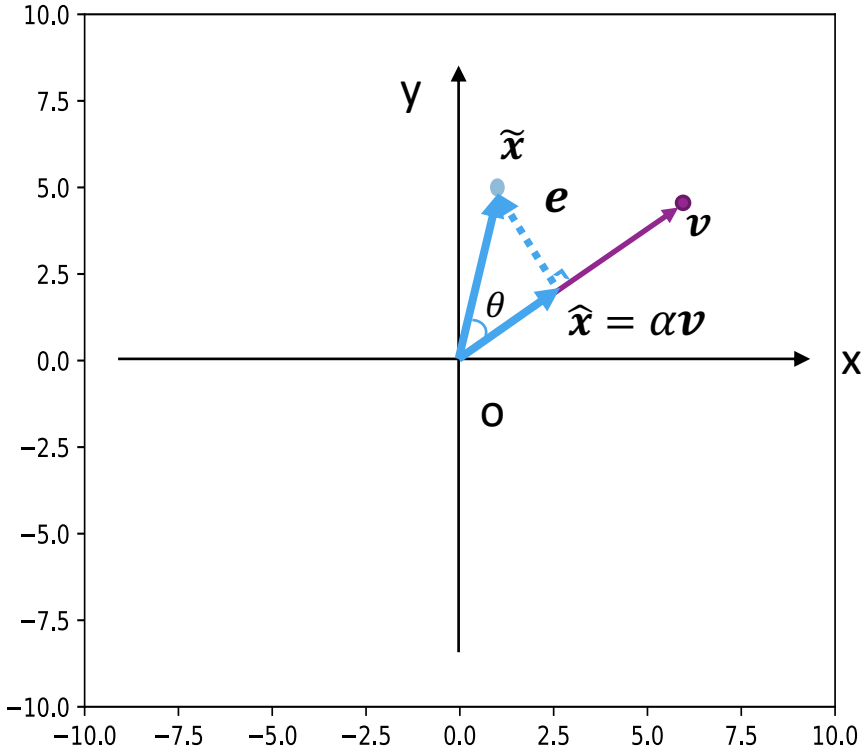
- How to find the new coordinate system?
- How specify a axis?
  - 2 points in the space in general
  - 1 point with common start point
  - Data mean serves a good candidate

Geometric intuition



- Demean, so that the origin is 0.
- What is the key property of the new axis?

Geometric intuition



- Demean, so that the origin is 0.
- What is the key property of the new axis?
  - For a fixed v, chose α to approximate $\tilde{x}$ with $\hat{x} = \alpha v$

$$\tilde{x} = \hat{x} + e$$

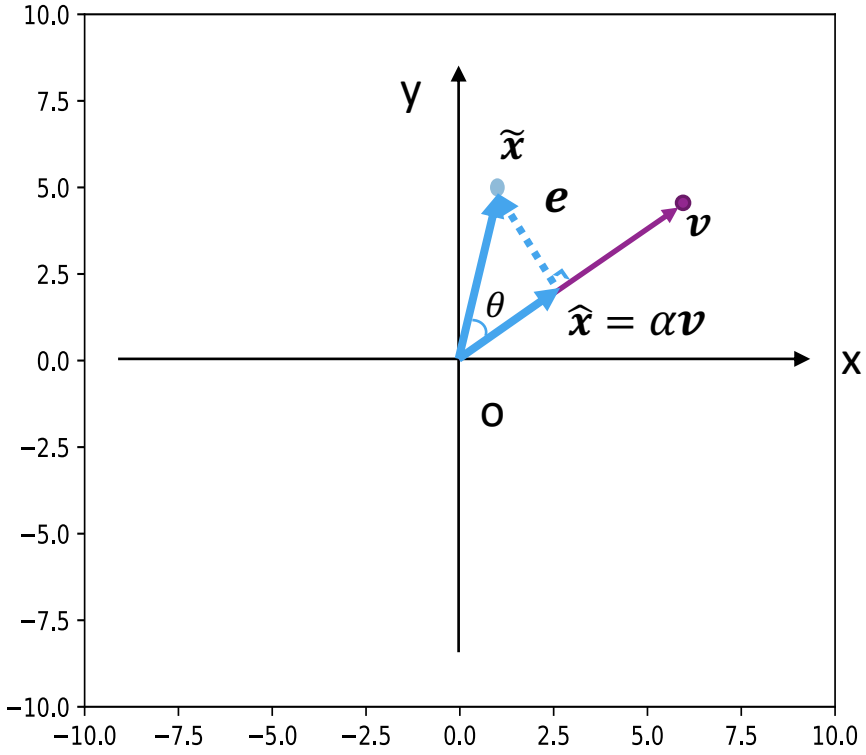
  - The approximation error $\|e\|^2$ is minimized when $e \perp v$:

$$\|e\|^2 = (\|\tilde{x}\| \sin\theta)^2$$
$$\|\hat{x}\|^2 = (\|\tilde{x}\| \cos\theta)^2$$
$$\|e\|^2 + \|\hat{x}\|^2 = \|\tilde{x}\|^2$$

  - Minimizing $\|e\|^2$ is equivalent to maximizing $\|\hat{x}\|^2$

$$\max_{v} \mathbb{E}(\|\hat{x}\|^2)$$

Geometric intuition

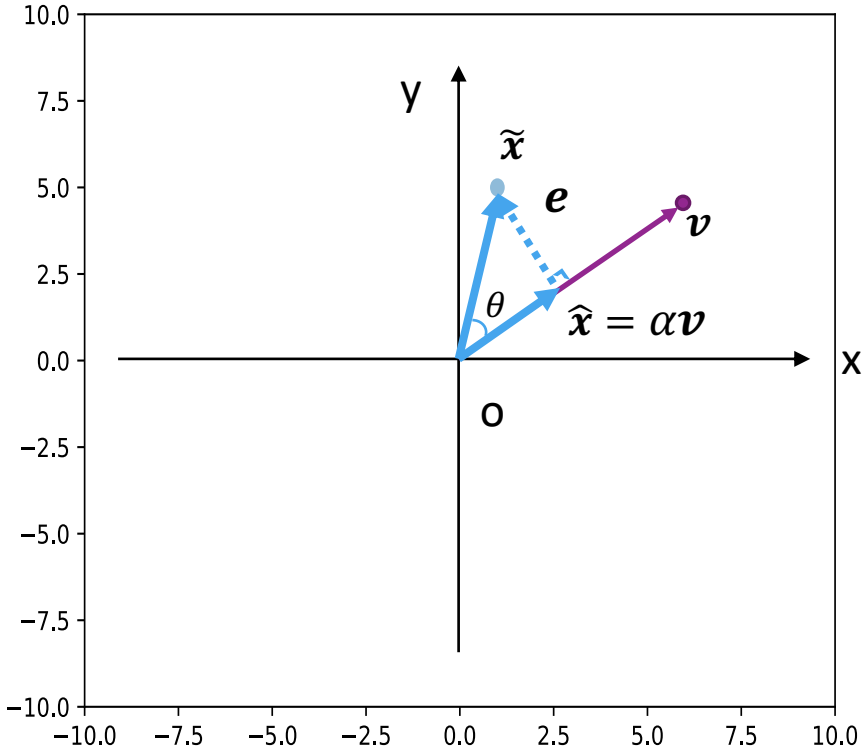


- Demean, so that the origin is 0.
- What is the key property of the new axis?

$$\max_{v} \mathbb{E}(\|\hat{x}\|^2)$$
$$= \max_{v} \mathbb{E}\left((\|\tilde{x}\|\cos(\theta))^2\right)$$
$$= \max_{v} \mathbb{E}\left(\left(\frac{\tilde{x}\cdot v}{\|v\|}\right)^2\right)$$
$$= \max_{v} \mathbb{E}\left((\tilde{x}\cdot v)^2\right)$$
$$s.t.\ \|v\| = 1$$

- $\hat{x} = \mathrm{Proj}_v(\tilde{x}) = \|\tilde{x}\|\cos(\theta) \times \frac{v}{\|v\|}$
- $\tilde{x}\cdot v = \|v\|\|\tilde{x}\|\cos(\theta) \rightarrow \|\tilde{x}\|\cos(\theta) = \frac{\tilde{x}\cdot v}{\|v\|}$
- If $\|v\| = 1$, $\|\tilde{x}\|\cos(\theta) = \tilde{x}\cdot v$

Geometric intuition



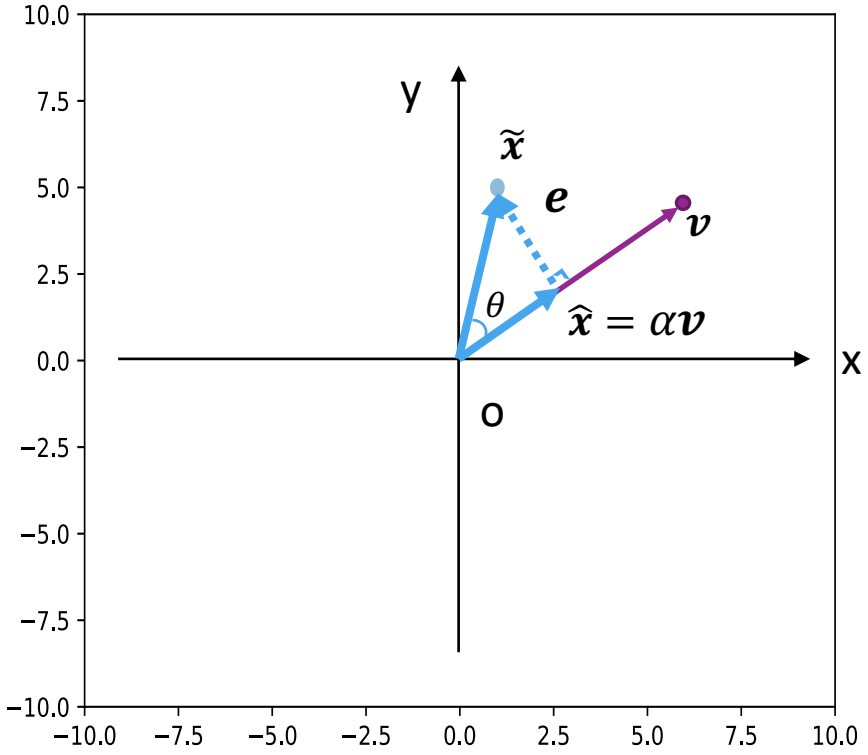
- Demean, so that the origin is 0.
- What is the key property of the new axis?

$$\max_{v} \mathbb{E}\left((\tilde{x} \cdot v)^2\right) \; s.t. \; \|v\| = 1$$

$$\begin{aligned} & var(\tilde{x} \cdot v) \\ &= \mathbb{E}\left(\left(\tilde{x} \cdot v - \mathbb{E}(\tilde{x} \cdot v)\right)^2\right) \\ &= \mathbb{E}\left((\tilde{x} \cdot v)^2\right) \end{aligned}$$

Minimizing approximation error is equivalent to maximizing directional variance (coordinate variance long a direction).

Geometric intuition



- Demean, so that the origin is 0.
- What is the key property of the new axis?

$$\max_v \mathbb{E}\left((\tilde{x} \cdot v)^2\right) s.t. \|v\| = 1$$

$$= \max_v \mathbb{E}\left((\tilde{x}v^T)^2\right)$$

$$= \max_v \mathbb{E}\left((\tilde{x}v^T)^T \tilde{x}v^T\right)$$

$$= \max_v \mathbb{E}(v\tilde{x}^T\tilde{x}v^T)$$

$$= \max_v v\mathbb{E}(\tilde{x}^T\tilde{x})\, v^T$$

$$\approx \max_v vQv^T \; s.t. \|v\| = 1$$

Q is the sample covariance

$$Q = \frac{1}{N}\tilde{X}^T\tilde{X}$$

Directional variance can be calculated from the sample covariance.

Optimization

$$max_{\boldsymbol{v}} \boldsymbol{v} Q \boldsymbol{v}^T \quad s.t. \|\boldsymbol{v}\| = 1$$

- Idea: shrink the solution set of $\boldsymbol{v}$ to a few candidates and then exhaustive search

- Theorem: any local maxima of the directional variance is an eigenvector of Q
  - Let $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p$ (column vectors) be the eigenvectors of Q : $Q\boldsymbol{v}_j = \lambda_j \boldsymbol{v}_j$
  - Sort eigenvalues in descending order: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$
    - Can show that eigenvalues are real and non-negative

Optimization

- Lagrange multiplier
- For an constrained problem: $\max_{\boldsymbol{v}} f(\boldsymbol{v})$ s.t. $g(\boldsymbol{v}) = 0$
- Build the Lagrange function: $L(\boldsymbol{v}, \lambda) = f(\boldsymbol{v}) + \lambda g(\boldsymbol{v})$
- For any optimal $\boldsymbol{v}^*$ for the original problem, there exists a λ^* such that $(\boldsymbol{v}^*, \lambda^*)$ is a stationary point for the Lagrange function
- Thus, the method of Lagrange multipliers yields a necessary condition for optimal point for the original constrained problem.

Optimization

- For our constrained problem: $\max_{\boldsymbol{v}} \boldsymbol{v}Q\boldsymbol{v}^T$ s.t. $\|\boldsymbol{v}\| = 1$
- Lagrange function: $L(\boldsymbol{v}, \lambda) = \boldsymbol{v}Q\boldsymbol{v}^T + \lambda(\boldsymbol{v}\boldsymbol{v}^T - 1)$
- Find the stationary point of the Lagrange function:

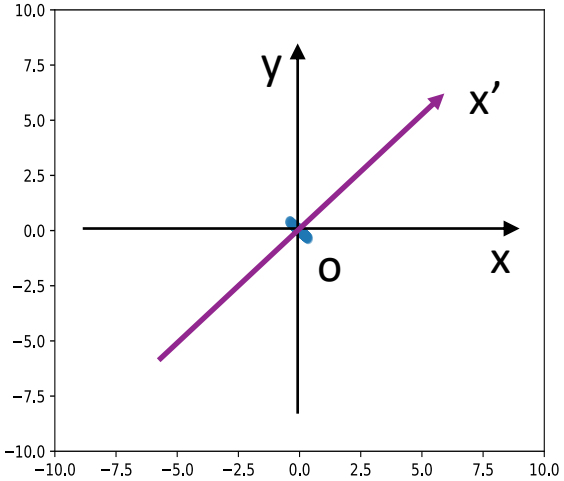
$$\frac{\partial L}{\partial \boldsymbol{v}} = 2\boldsymbol{v}Q + 2\lambda\boldsymbol{v} = 0$$
$$\rightarrow \boldsymbol{v}Q = -\lambda\boldsymbol{v}$$
$$\rightarrow Q^T\boldsymbol{v}^T = -\lambda\boldsymbol{v}^T$$
$$\rightarrow Q\boldsymbol{v}^T = -\lambda\boldsymbol{v}^T$$

- Any local maxima of the variance directional is an eigenvector

Optimization

- The candidate set of $\boldsymbol{v}$ is the eigenvectors of Q: $\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_p\}$
- The corresponding cost value $\boldsymbol{v}^T Q \boldsymbol{v}$: $\{\lambda_1, \dots, \lambda_p\}$
- Note from here onwards $\boldsymbol{v}$ represent column vectors
- Since $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$
- $\boldsymbol{v}_1$ is the solution for the problem $max_v \boldsymbol{v}^T Q \boldsymbol{v} \quad s.t. \|\boldsymbol{v}\| = 1$
- After removing all variance explained by $\boldsymbol{v}_1$, find the next max variance direction.

$$\boldsymbol{v}^T Q \boldsymbol{v} = \boldsymbol{v}^T \lambda \boldsymbol{v} = \lambda \boldsymbol{v}^T \boldsymbol{v} = \lambda$$



Principal Components

- Principal components: The eigenvectors of Q, $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p$
- Sorted by eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$
- Each vector is of dimension p
- Represents directions of maximal variance

Principal Components

- Key properties of Q
  - Symmetric: $Q_{ij} = Q_{ji}$
  - Eigenvalues are real and non-negative
- Key properties of PCs (here $\boldsymbol{v}_j$ are row vectors)
  - $\boldsymbol{v}_j \boldsymbol{v}_k^T = 1$ if $j = k$
  - $\boldsymbol{v}_j \boldsymbol{v}_k^T = 0$ if $j \neq k$
  - $VV^T = I$

Outline

- ❑ Dimensionality reduction
- ❑ Principal components and directions of variance (a geometric perspective)
- ❑ Computing PCs via the SVD
- ❑ Approximation with PCs
- ❑ Face example in python
- ❑ Extensions

Computing PCs via the SVD

- SVD provides a numerically more stable way to calculate PCs

$$Q = \frac{1}{N}\tilde{X}^T\tilde{X}$$
$$= \frac{1}{N}VS^TU^TUSV^T$$
$$= V\frac{S^TS}{N}V^T$$
$$\rightarrow QV = V\frac{S^TS}{N}$$

- Singular-Value Decomposition (SVD)

$$\forall\, M \in \mathbb{C}^{m\times n}, \exists\, r, U, S, V \; so\; that$$
$$M = USV^T$$

  - $U \in \mathbb{C}^{m\times r}, U^TU = I$
  - $S \in \mathbb{C}^{rxr}$ is a diagonal matrix
  - $V \in \mathbb{C}^{n\times r}, V^TV = I$
  - $r \leq \min(m, n)$ (Rank of the matrix)

- $\tilde{X} = USV^T$

Computing PCs via the SVD

- SVD provides a numerically more stable way to calculate PCs

$$QV = V\frac{S^TS}{N} \qquad or \qquad Q\boldsymbol{v_i} = \frac{S_{ii}^TS_{ii}}{N}\boldsymbol{v_i}$$

- Rows of V^T are eigenvectors of Q, e.g. PCs
- $\frac{S^TS}{N}$ are the eigenvalues (real and nonnegative)

- Singular-Value Decomposition (SVD)

$$\forall\, M \in \mathbb{C}^{m\times n}, \exists\, r, U, S, V \; so\; that$$
$$M = USV^T$$

  - $U \in \mathbb{C}^{m\times r}, U^TU = I$
  - $S \in \mathbb{C}^{rxr}$ is a diagonal matrix
  - $V \in \mathbb{C}^{n\times r}, V^TV = I$
  - $r \leq \min(m, n)$ (Rank of the matrix)

- $\tilde{X} = USV^T$

Outline

- ❑ Dimensionality reduction
- ❑ Principal components and directions of variance (a geometric perspective)
- ❑ Computing PCs via the SVD
- ❑ Approximation with PCs
- ❑ Face example in python
- ❑ Extensions

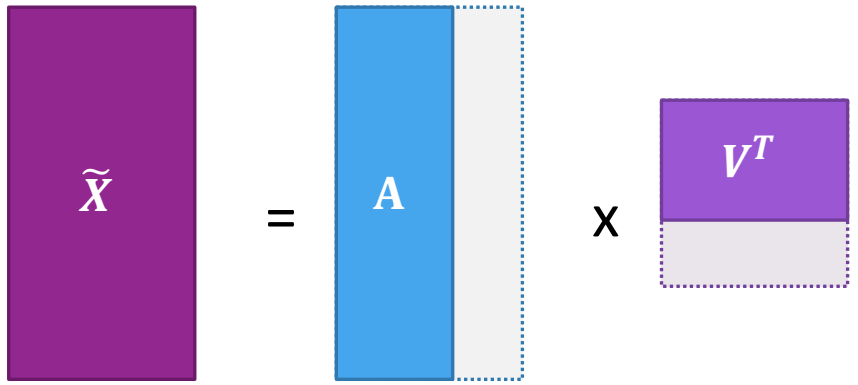
Average Approximation Error

- Given data $\tilde{\boldsymbol{x}}_i, i = 1, \ldots, N$
- Let $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p$ be the PCs
- Find coefficient expansion of each data sample:

$$\tilde{\boldsymbol{x}}_i = \sum_{j=1}^{p} \alpha_{ij} \boldsymbol{v}_j, \qquad \alpha_{ij} = \boldsymbol{v}_j^T \tilde{\boldsymbol{x}}_i$$

- Now consider approximation with d coefficients:

$$\hat{\boldsymbol{x}}_i = \sum_{j=1}^{d} \alpha_{ij} \boldsymbol{v}_j$$



Average Approximation Error

- Error in sample i:

$$\tilde{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_i = \sum_{j=d+1}^{p} \alpha_{ij} \boldsymbol{v}_j$$

- Theorem: Average error with a d PC approximation is:

$$\frac{1}{\mathrm{N}} \sum_{i=1}^{N} \|\tilde{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_i\|^2 = \sum_{j=d+1}^{p} \lambda_j$$

  - Sum of the smallest $p - d$ eigenvalues

Proportion of Variance (PoV)

- Total variance of data set: $\frac{1}{\mathrm{N}}\sum_{i=1}^{N}\|\tilde{\boldsymbol{x}}_i\|^2 = \sum_{j=1}^{p}\lambda_j$
- Average approximation error: $\frac{1}{\mathrm{N}}\sum_{i=1}^{N}\|\tilde{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_i\|^2 = \sum_{j=d+1}^{p}\lambda_j$
- The proportion of variance explained by d PCs is:

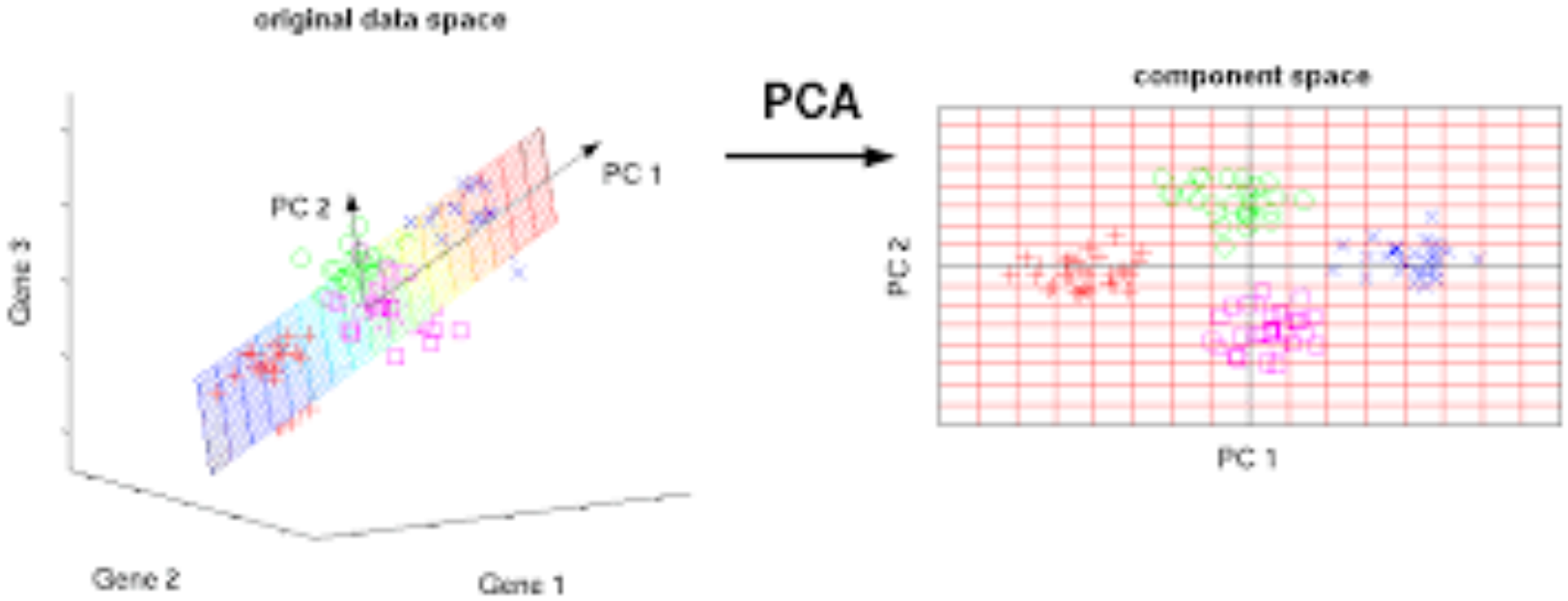
$$PoV(d) = \frac{\sum_{j=1}^{d}\lambda_j}{\sum_{j=1}^{p}\lambda_j}$$

  - Measure of approximation error in using d PCs

- Example: Suppose eigenvalues of sample covariance matrix are $10, 4, 0.2, 0.1, 0, 0, \ldots$
  - What is the PoV for $d = 1,2,3, \ldots$

Visualizing the Representation

- Finds a low-dimensional representation

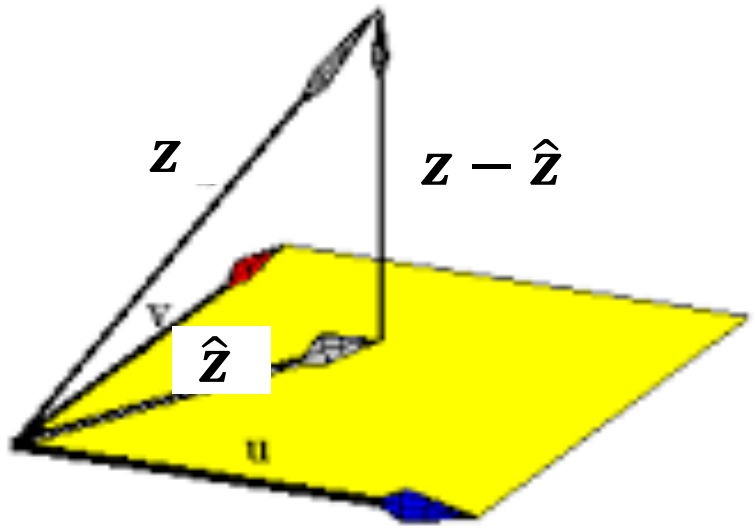


Geometry of Approximations

- Approximation can be interpreted geometrically
- Let V be set of all linear combinations

$$\sum_{j=1}^{d} \alpha_j \boldsymbol{v}_j$$

  - V is a vector space
  - Called the span of $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d$
- Given $\boldsymbol{z}$, $\hat{\boldsymbol{z}}$ is the closest vector in V to $\boldsymbol{z}$
- Called the projection of $\boldsymbol{z}$ onto V



Space spanned by $v_1, \ldots, v_d$

Outline

- ❑ Dimensionality reduction
- ❑ Principal components and directions of variance (a geometric perspective)
- ❑ Computing PCs via the SVD
- ❑ Approximation with PCs
- ❑ Face example in python
- ❑ Extensions

Example: Faces

Labeled Faces in the Wild Home

- Face images can be high-dimensional
  - We will use 50 x 37 = 1850 pixels
- But, there may be few degrees of freedom
- Can we reduce the dimensionality of this?
- Data Labelled Faces in the Wild project
  - http://vis-www.cs.umass.edu/lfw
  - Large collection of faces (13000 images)
  - Taken from web articles about 10 years ago

Loading the Data

- Built-in routines to load data is sciket-learn
- Can take several minutes the first time (Be patier

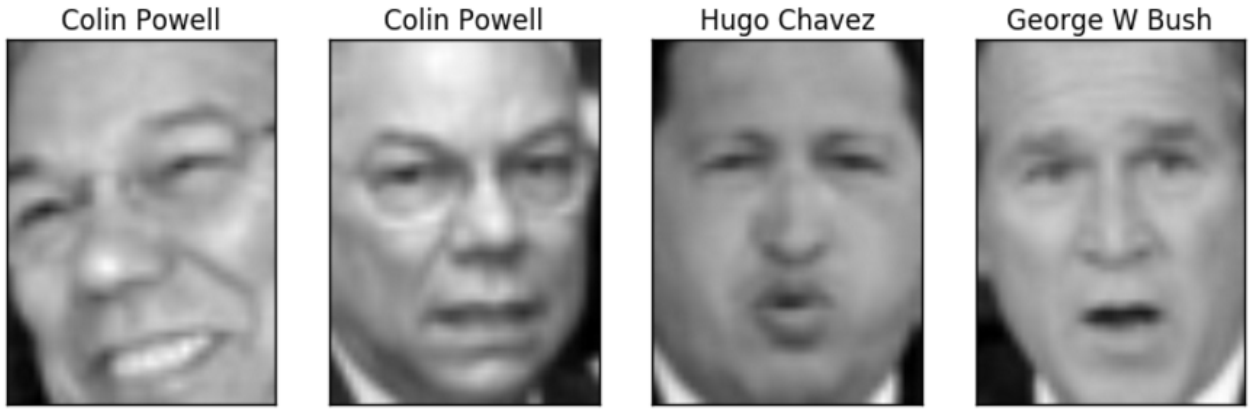
```
Image size      = 50 x 37 = 1850 pixels
Number faces    = 1288
Number classes = 7
```

```
from sklearn.datasets import fetch_lfw_people
lfw_people = fetch_lfw_people(min_faces_per_person=70, resize=0.4)
```

```
2016-11-14 14:15:30,862 Downloading LFW metadata: http://vis-www.cs.umass.edu/lfw/pairsDevTrain.txt
2016-11-14 14:15:30,958 Downloading LFW metadata: http://vis-www.cs.umass.edu/lfw/pairsDevTest.txt
2016-11-14 14:15:31,028 Downloading LFW metadata: http://vis-www.cs.umass.edu/lfw/pairs.txt
2016-11-14 14:15:31,294 Downloading LFW data (~200MB): http://vis-www.cs.umass.edu/lfw/lfw-funneled.tgz
2016-11-14 14:20:10,056 Decompressing the data archive to C:\Users\Sundeep\scikit_learn_data\lfw_home\lfw_funneled
2016-11-14 14:22:08,605 Loading LFW people faces from C:\Users\Sundeep\scikit_learn_data\lfw_home
2016-11-14 14:22:09,735 Loading face #00001 / 01288
2016-11-14 14:22:13,640 Loading face #01001 / 01288
```

Plotting the Data

- Some example faces
- You may be too young to remember them all




```
def plt_face(x):
    h = 50
    w = 37
    plt.imshow(x.reshape((h, w)), cmap=plt.cm.gray)
    plt.xticks([])
    plt.yticks([])

I = np.random.permutation(n_samples)
plt.figure(figsize=(10,20))
nplt = 4;
for i in range(nplt):
    ind = I[i]
    plt.subplot(1,nplt,i+1)
    plt_face(X[ind])
    plt.title(target_names[y[ind]])
```

Computing the SVD

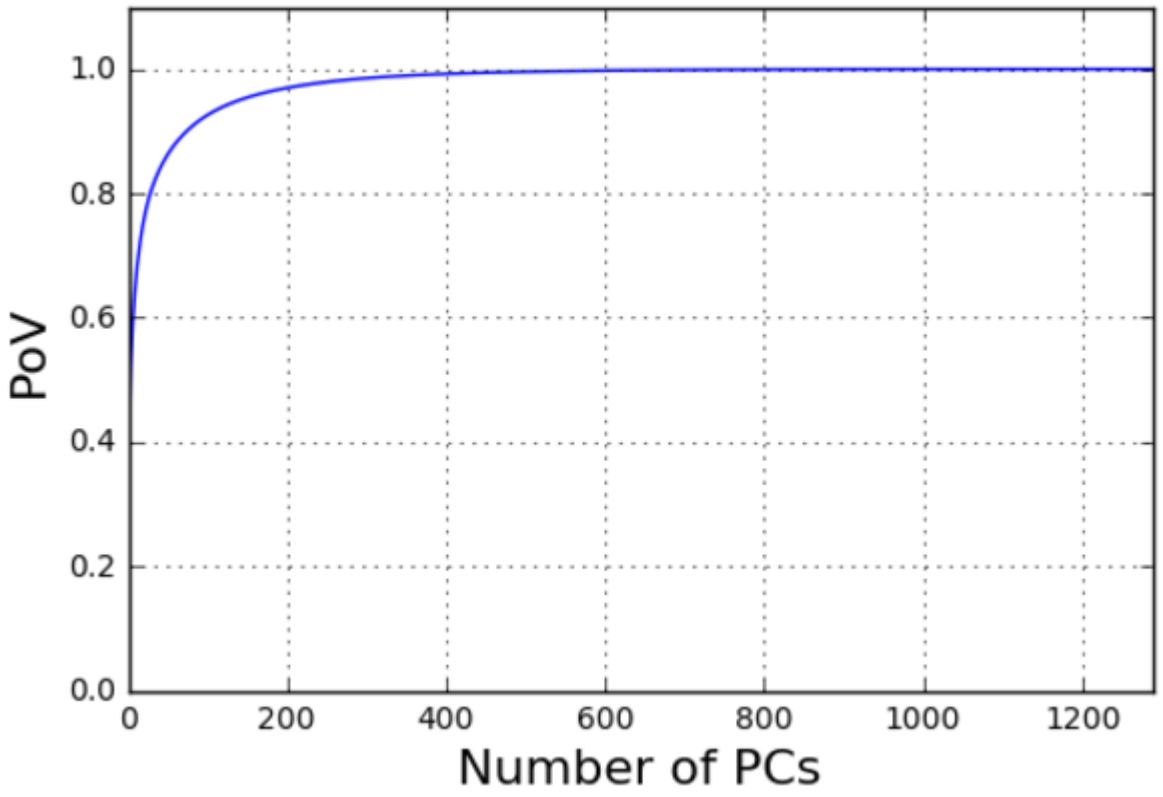
```
npix = h*w
Xmean = np.mean(X,0)
Xs = X - Xmean[None,:]
```

Then, we compute an SVD. Note that in python the SVD re

```
U,S,V = np.linalg.svd(Xs, full_matrices=False)
```

- Note efficient use of python broadcasting
- SVD:
  - Use full_matrices (avoids computing zero SVs)
  - Matrix V is what we call V^T (Different from MATLAB)

Finding the PoV



- Most variance explained in about 400 components
- Some reduction

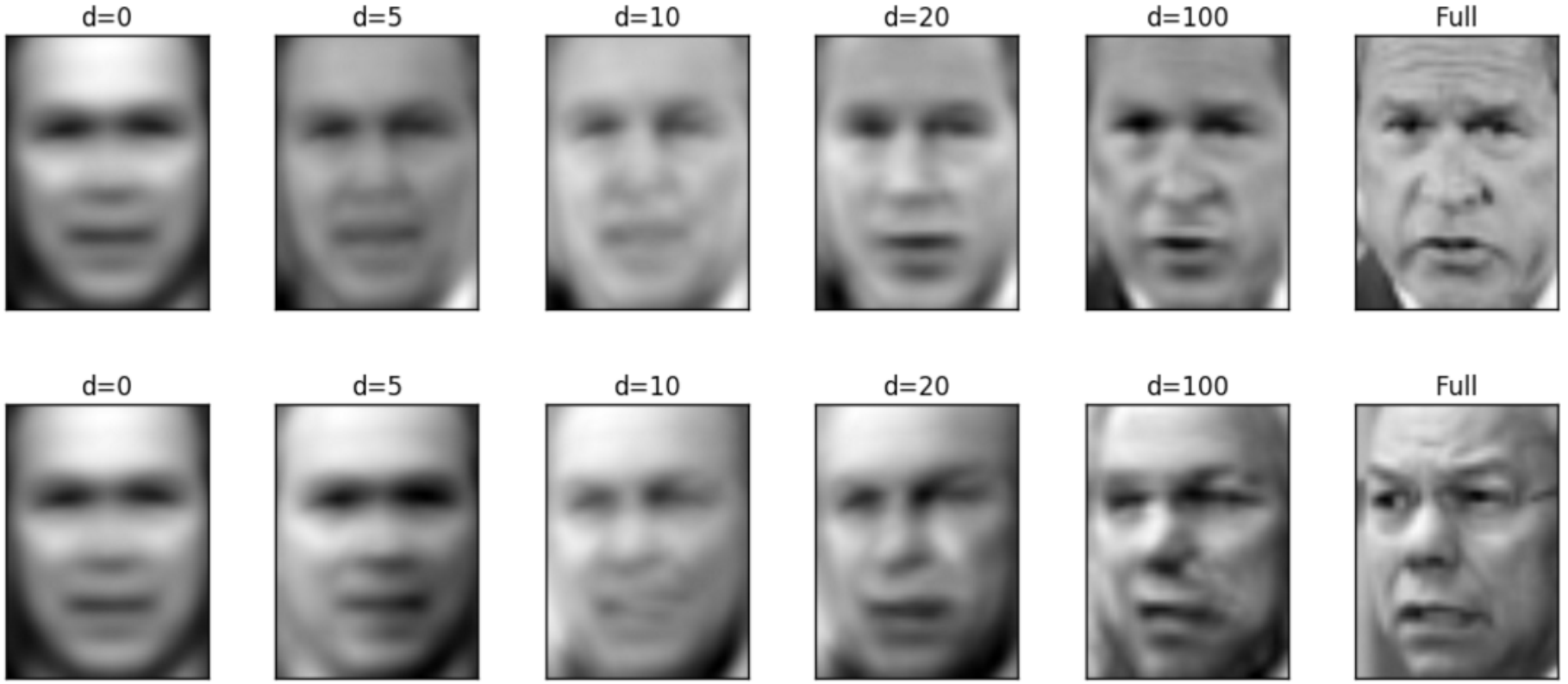
```
lam = S**2
PoV = np.cumsum(lam)/np.sum(lam)

plt.plot(PoV)
plt.grid()
plt.axis([1,n_samples,0, 1.1])
plt.xlabel('Number of PCs', fontsize=16)
plt.ylabel('PoV', fontsize=16)
```

Plotting Approximations

```
nplt = 2                # number of faces to plot
ds = [0,5,10,20,100]    # number of SVD approximations

# Select random faces
inds = np.random.permutation(n_samples)
inds = inds[:nplt]
nd = len(ds)

# Set figure size
plt.figure(figsize=(1.8 * (nd+1), 2.4 * nplt))
plt.subplots_adjust(bottom=0, left=.01, right=.99, top=.90, hspace=.35)

# Loop over figures
iplt = 0
for ind in inds:
    for d in ds:
        plt.subplot(nplt,nd+1,iplt+1)
        Xhati = (U[ind,:d]*S[None,:d]).dot(V[:d,:]) + Xmean
        plt_face(Xhati)
        plt.title('d={0:d}'.format(d))
        iplt += 1

    # Plot the true face
    plt.subplot(nplt,nd+1,iplt+1)
    plt_face(X[ind,:])
    plt.title('Full')
    iplt += 1
```

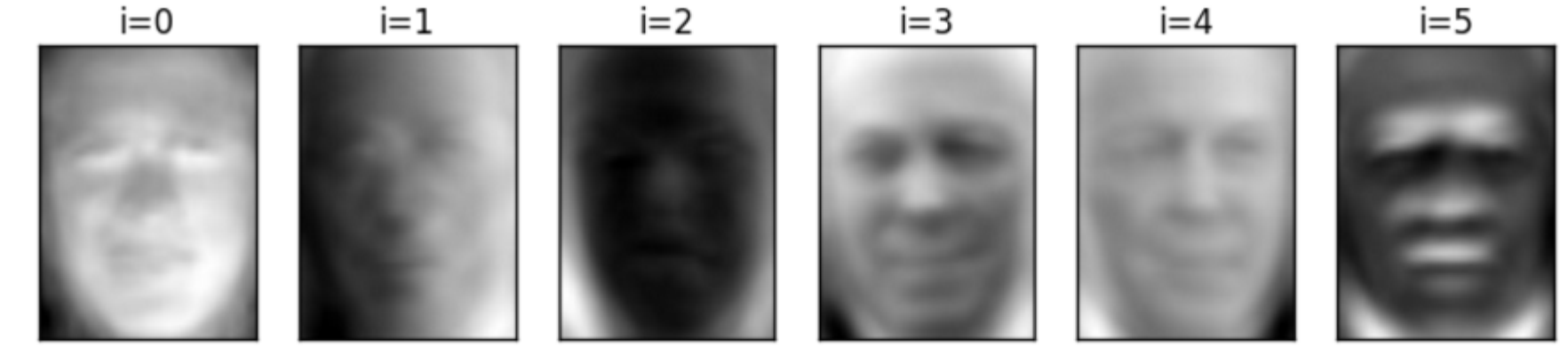
- Selection of figure sizes for subplot
- Note: Efficient computing of approx

Plotting the Approximations



Plotting the PCs

- The PCs can be plotted as well
- These are known as Eigenfaces!



How do we use Principle Components?

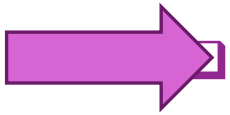
- PCs tell us “latent” factors of the data
  - Each sample is a weighted average of these PCs (eg. eigen-faces)
- Can use the coefficients as low-dimension features for classification
  - Early success with face recognition (using projection onto eigenfaces)
- Can use the coefficients for compression
  - JPEG image coder: uses DCT basis at block level
  - JPEG wavelet coder: uses Wavelet basis over entire image
- Denoising
  - Project noisy data onto the basis vectors
  - Reconstruct from low freq. basis vectors
- Data normalization: scaled coefficients will be independent and have unit variance.

Other Considerations

- Normalization: For most data, it is essential to standardize before computing PC
  - Otherwise, components with large values dominate small ones
- Sklearn has built in PCA routine (will explore in lab)
- Some texts do not subtract mean
  - Will be picked up as one of the PCs

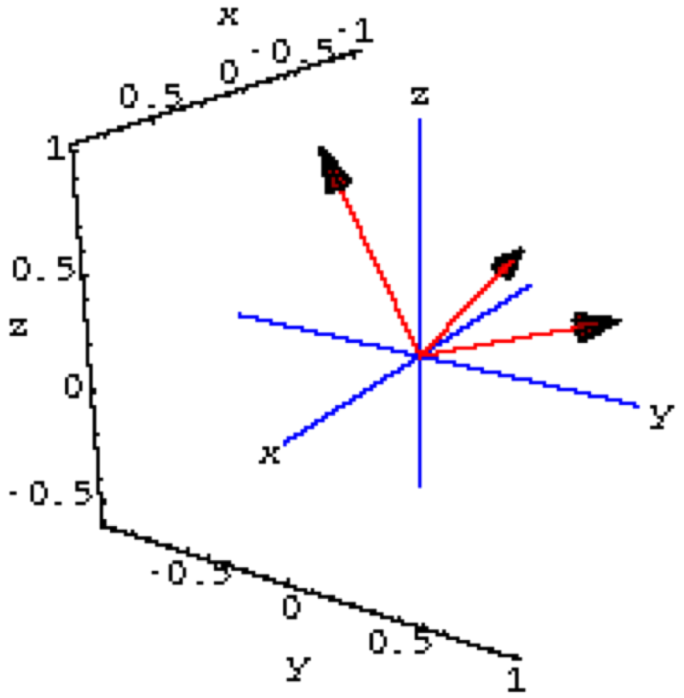
Outline

- ❑ Dimensionality reduction
- ❑ Principal components and directions of variance (a geometric perspective)
- ❑ Computing PCs via the SVD
- ❑ Approximation with PCs
- ❑ Face example in python
- ❑ Extensions



Orthonormal Sets and Bases

- Definition: A set of vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d$ are an orthonormal set if:
  - $\|\boldsymbol{v}_j\| = 1$ for all j (unit length)
  - $\boldsymbol{v}_j^T \boldsymbol{v}_k = 0$ if $j \neq k$ (perpendicular to one another)
- If $d = p$ then $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p$ is called an orthonormal basis
- Matrix form: If $\boldsymbol{V} = [\boldsymbol{v}_1 \ \ldots \boldsymbol{v}_d]$, then $\boldsymbol{V}^T \boldsymbol{V} = I_d$
- If $d = p$, then $\boldsymbol{V}$ is an orthogonal matrix



Coefficients in an Orthonormal Basis

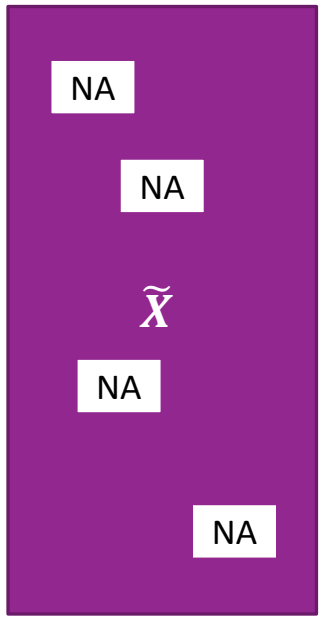
- For an orthonormal set V

$$\boldsymbol{\alpha} = V^T \boldsymbol{x}, \qquad \boldsymbol{x} = V\boldsymbol{\alpha}$$

- Examples of Orthonormal Basis:
  - PCs, a data driven orthonormal basis
  - Discrete Fourier Transform (normalized)
  - Discrete Cosine Transform
  - Wavelet transform

Wiberg's Method

- PCA with missing data



$$\underset{V}{\operatorname{argmin}} \min_{A} \left\| \tilde{X} - AV \right\|_F$$

$$s.t. VV^T = I$$

$$\tilde{X} \in \mathbb{R}^{N \times p}, A \in \mathbb{R}^{N \times d}, V \in \mathbb{R}^{d \times p}, d \leq p$$

$$\underset{V}{\operatorname{argmin}} \min_{A} \left\| W \circ \left(\tilde{X} - AV \right) \right\|_F$$

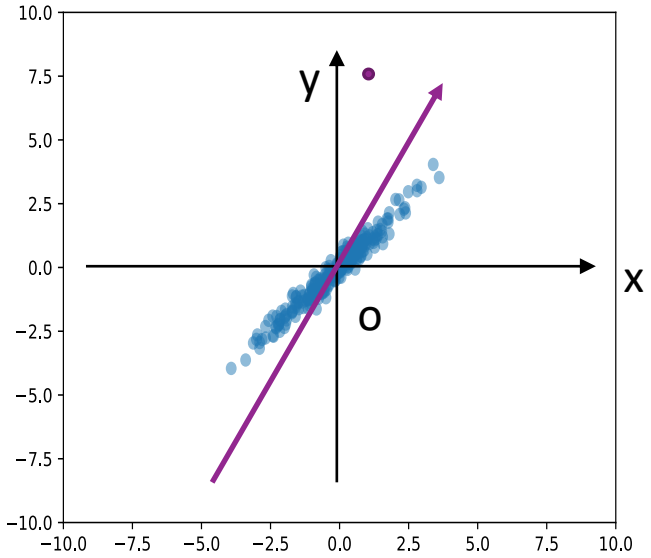
$$s.t. VV^T = I$$

$$\tilde{X} \in \mathbb{R}^{N \times p}, A \in \mathbb{R}^{N \times d}, V \in \mathbb{R}^{d \times p}, d \leq p$$

$$W_{ij} = 1 \; if \; valid \; data, otherwise \; 0$$

Robust PCA

- PCA is known sensitive to outlier



$$\underset{V}{\operatorname{argmin}} \min_{A} \left\| \tilde{X} - AV \right\|_F$$

$$s.t. VV^T = I$$

$$\tilde{X} \in \mathbb{R}^{N \times p}, A \in \mathbb{R}^{N \times d}, V \in \mathbb{R}^{d \times p}, d \leq p$$

⇩

$$\underset{V}{\operatorname{argmin}} \min_{A} \|L - AV\|_F + \lambda \|S\|_1$$

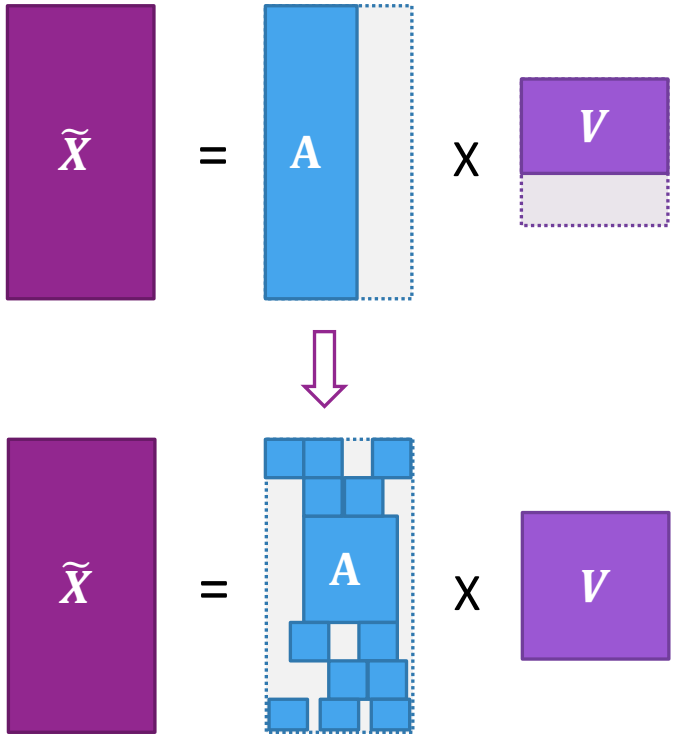
$$s.t. \tilde{X} = L + S$$

$$VV^T = I$$

$$\tilde{X} \in \mathbb{R}^{N \times p}, A \in \mathbb{R}^{N \times d}, V \in \mathbb{R}^{d \times p}, d \leq p$$

Sparse orthonormal transforms

- From low rank to sparsity



$$\underset{V}{\operatorname{argmin}} \min_{A} \left\| \tilde{X} - AV \right\|_F$$

$$s.t. VV^T = I$$

$$\tilde{X} \in \mathbb{R}^{N \times p}, A \in \mathbb{R}^{N \times d}, V \in \mathbb{R}^{d \times p}, d \leq p$$

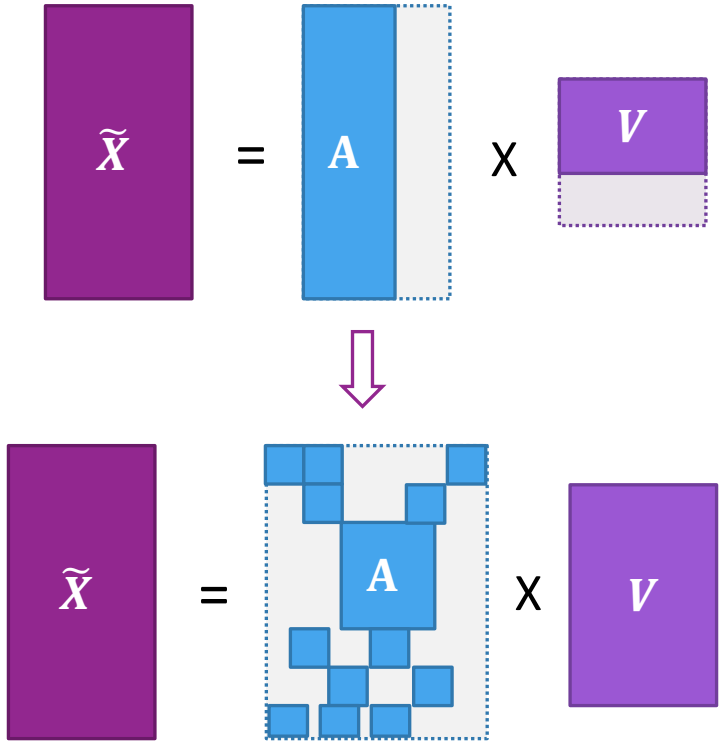
$$\underset{V}{\operatorname{argmin}} \min_{A} \left\| \tilde{X} - AV \right\|_F + \lambda \|A\|_1$$

$$s.t. VV^T = I$$

$$\tilde{X} \in \mathbb{R}^{N \times p}, A \in \mathbb{R}^{N \times p}, V \in \mathbb{R}^{p \times p}$$

KSVD

- More parsimonious representation with redundant dictionary



$$\underset{V}{\operatorname{argmin}} \min_{A} \left\| \tilde{X} - AV \right\|_F$$

$$s.t. VV^T = I$$

$$\tilde{X} \in \mathbb{R}^{N \times p}, A \in \mathbb{R}^{N \times d}, V \in \mathbb{R}^{d \times p}, d \leq p$$

$$\underset{V}{\operatorname{argmin}} \min_{A} \left\| \tilde{X} - AV \right\|_F$$

$$s.t.\ \tilde{X} \in \mathbb{R}^{N \times p}, A \in \mathbb{R}^{N \times d}, V \in \mathbb{R}^{d \times p}, (d \geq p)$$

Linear discriminative analysis

- Projects to the most discriminative direction
- Supervised learning with labels
- Simultaneously maximizing between classes variance and minimizing within class variance

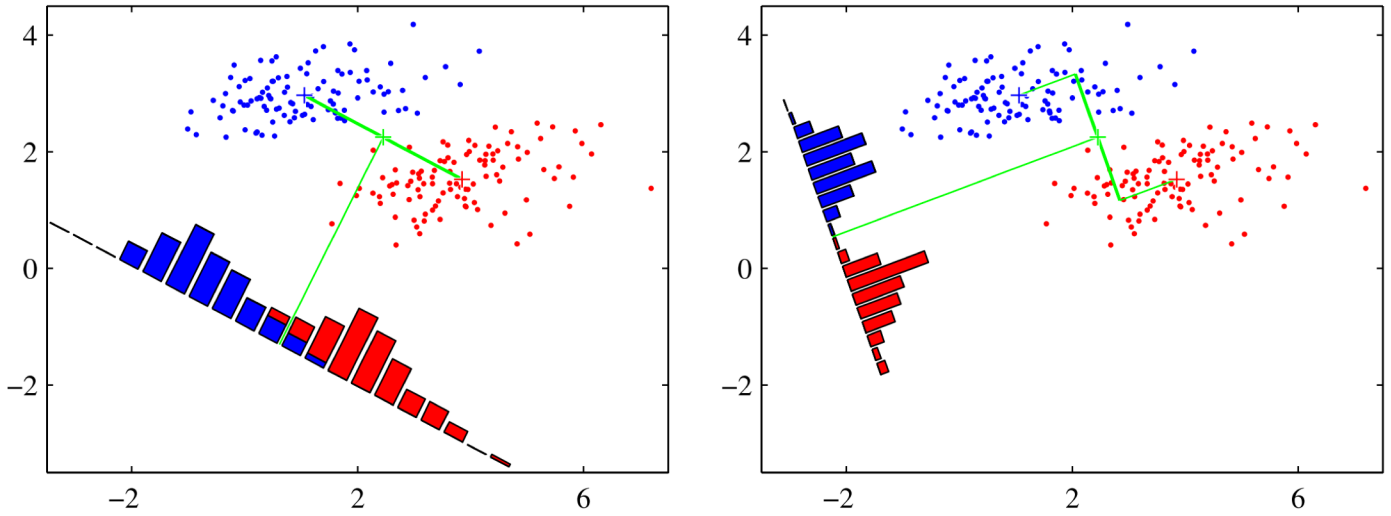


*figure from Bishop, Pattern Recognition and Machine Learning

Linear discriminative analysis

- Projects to the most discriminative direction
- Supervised learning with labels
- Simultaneously maximizing between classes variance and minimizing within class variance

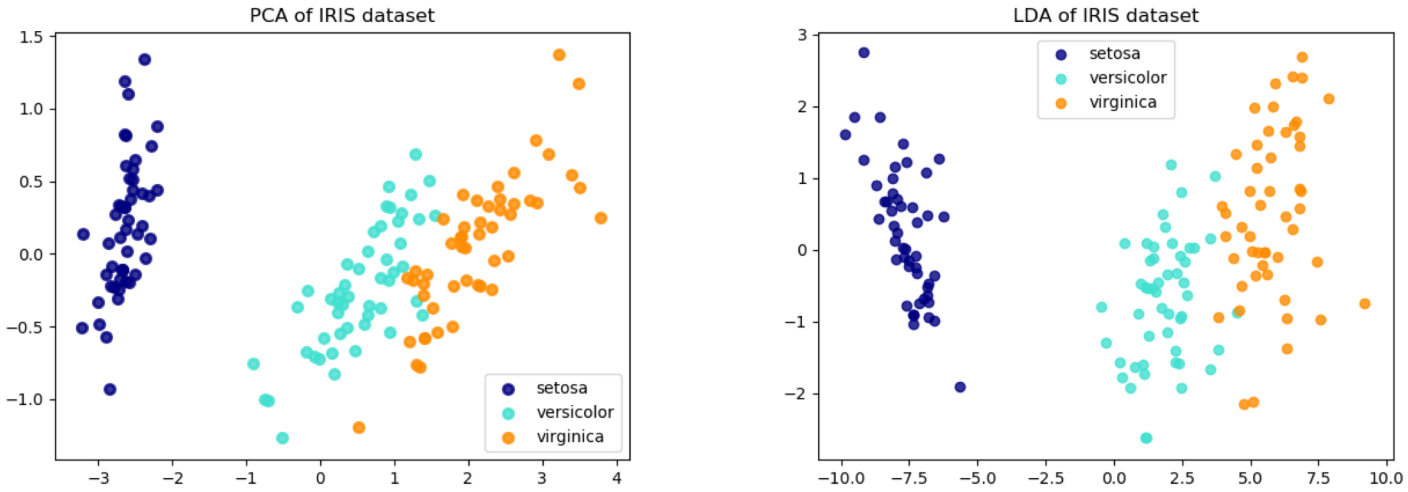
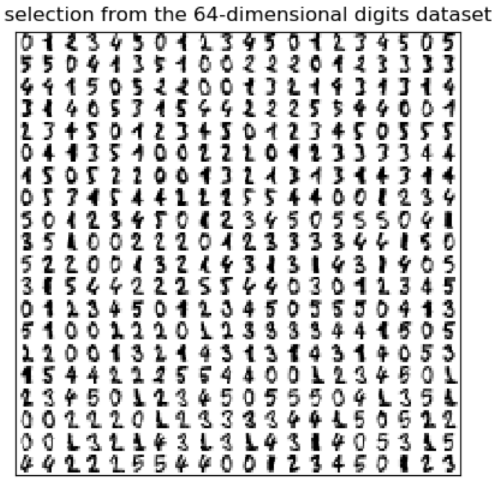
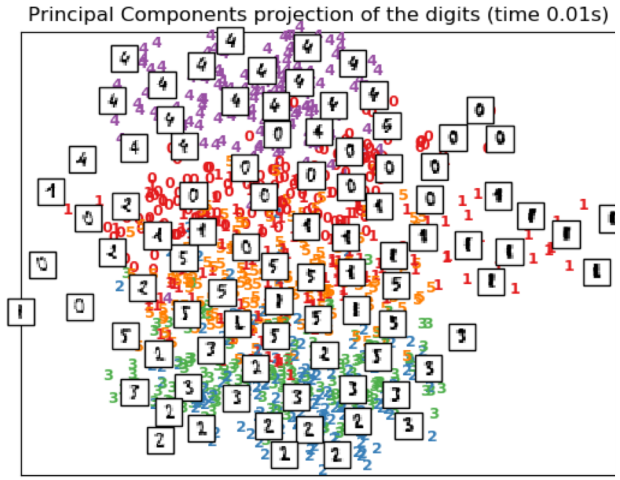


*figure from http://scikit-learn.org

Linear discriminative analysis

- Projects to the most discriminative direction
- Supervised learning with labels
- Simultaneously maximizing between classes variance and minimizing within class variance





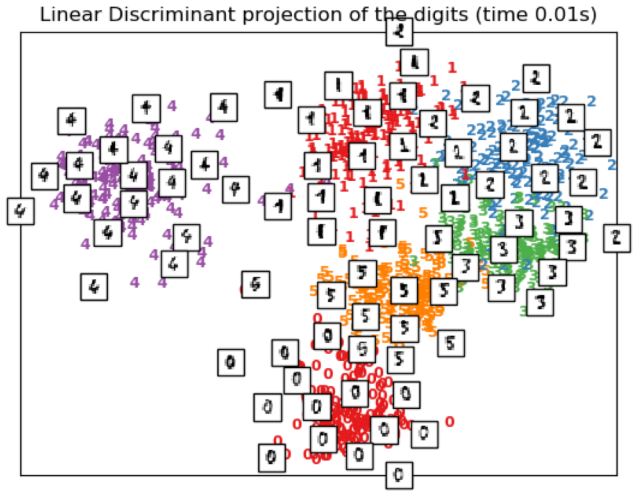
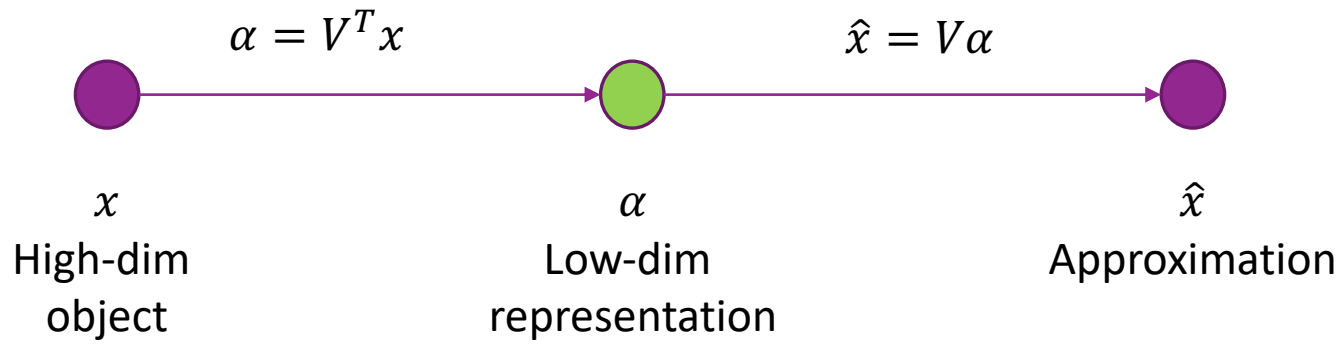


*figure from http://scikit-learn.org

State-of-the-Art: Auto-Encoders

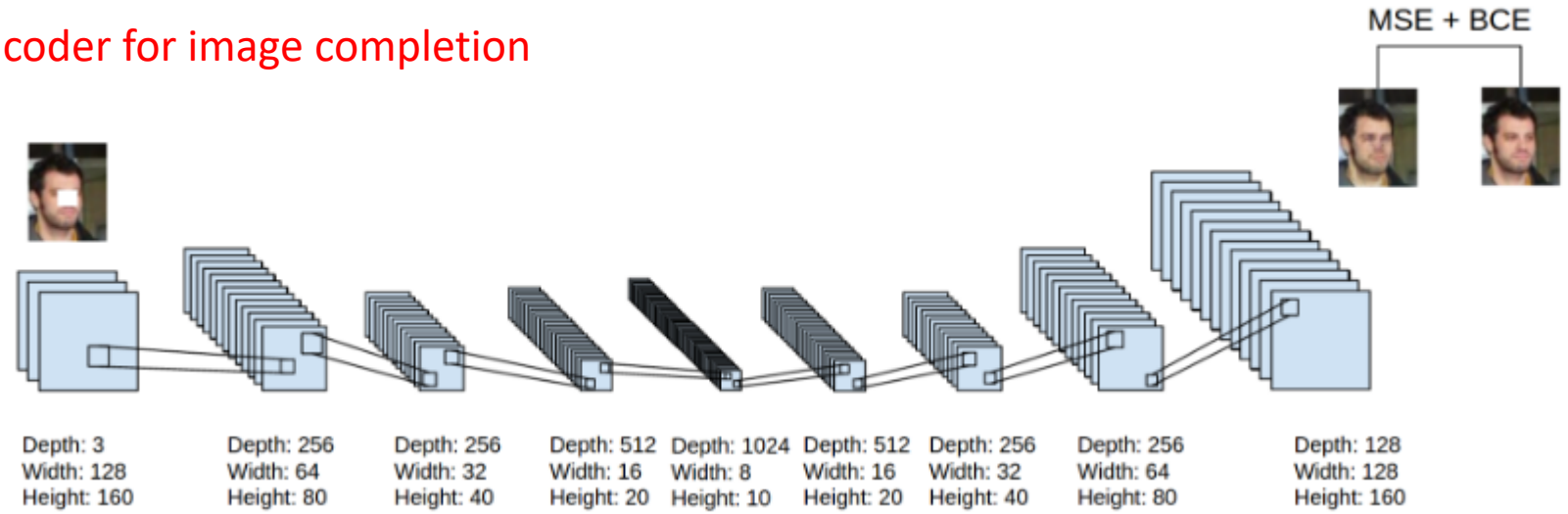
- PCA is a simple example of an autoencoder
- Tries to find low-dim representation
- Restricted to linear transforms
- Not very good for images and complex data



Deep Auto-Encoders

- Can use deep networks for learning complex latent representations and their inverses
  - http://www.cc.gatech.edu/~hays/7476/projects/Avery_Wenchen/
  - https://swarbrickjones.wordpress.com/2016/01/13/enhancing-images-using-deep-convolutional-generative-adversarial-networks-dcgans/ (Code in Theano not tensorflow)

Autoencoder for image completion



Representation with deep learning

- MNIST in 3D

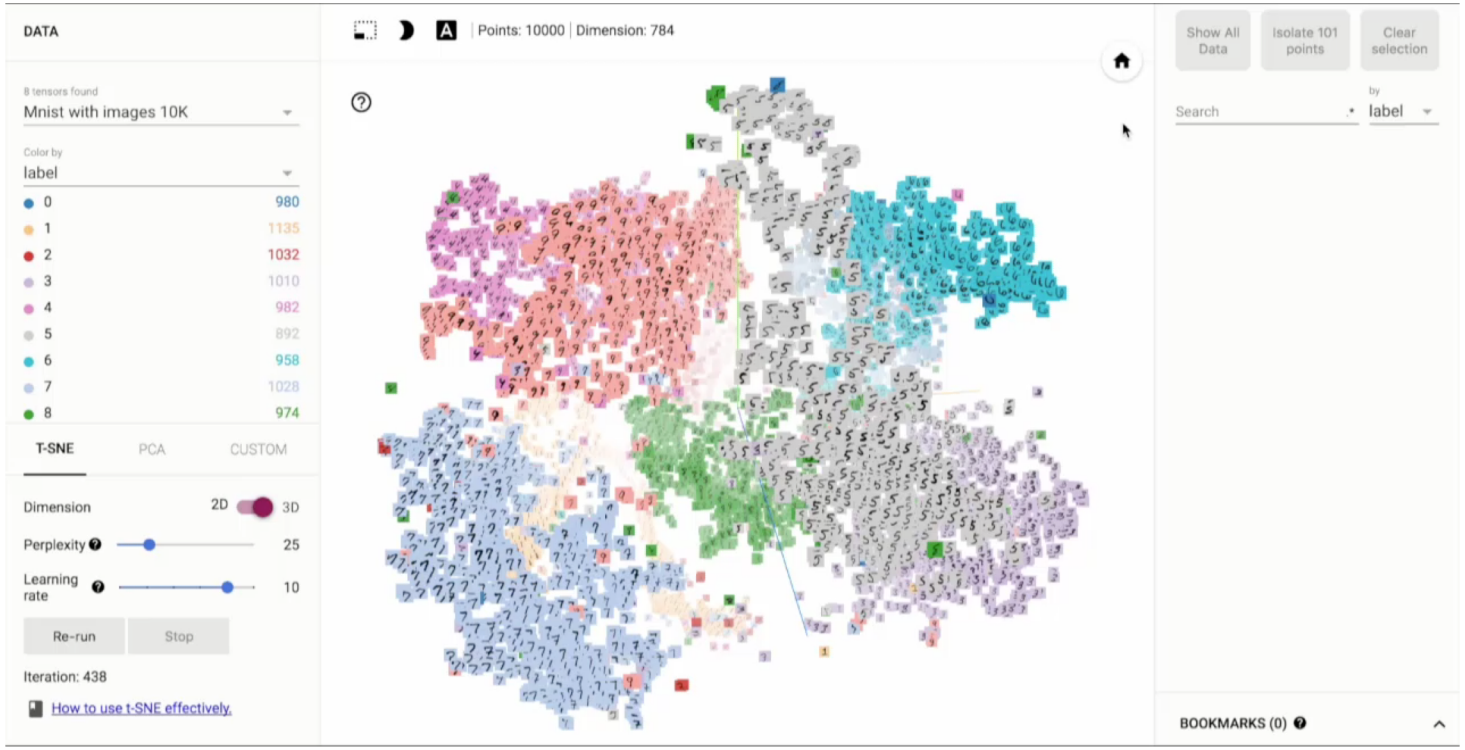


*figure from https:// tensorflow.org