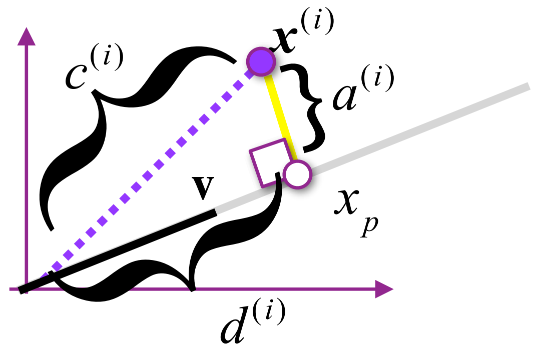
Do not distribute course material

mathematical background needed

Background material

Projecting $\mathbf{x}^{(i)} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ onto $\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$

when $\|\mathbf{v}\|_2 = \sqrt{v_1^2 + v_2^2} = 1$

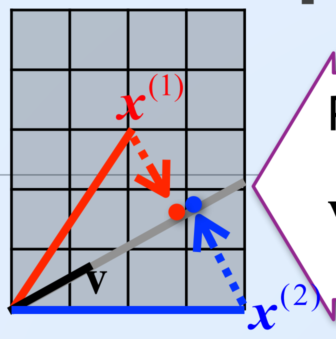


$$d^{(i)} = \mathbf{x}^{(i)T}\mathbf{v} = [x_1 \quad x_2] \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

$$\mathbf{x}_p^{(i)} = \left([x_1 \quad x_2] \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \right) \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = d^{(i)} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = d^{(i)}\mathbf{v}$$

By Pythagorus $a^2 + d^2 = c^2$

Example



Project onto $\mathbf{v} = \begin{bmatrix} 0.9 \\ 0.5 \end{bmatrix}$

I rounded $\mathbf{v} = \begin{bmatrix} 0.9 \\ 0.5 \end{bmatrix}$ quite a bit,

a few more decimals $\mathbf{v} = \begin{bmatrix} 0.894 \\ 0.447 \end{bmatrix}$

$$\mathbf{x}^{(1)} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

$$d^{(1)} = \mathbf{x}^{(1)T}\mathbf{v} = [2 \quad 3] \begin{bmatrix} 0.9 \\ 0.5 \end{bmatrix} = 1.8 + 1.3$$

$$\mathbf{x}_p^{(1)} = d^{(1)}\mathbf{v} = 3.1 \begin{bmatrix} 0.9 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 2.8 \\ 1.6 \end{bmatrix}$$

$$\mathbf{x}^{(2)} = \begin{bmatrix} 4 \\ 0 \end{bmatrix}$$

$$d^{(2)} = \mathbf{x}^{(2)T}\mathbf{v} = [4 \quad 0] \begin{bmatrix} 0.9 \\ 0.5 \end{bmatrix} = 3.6 + 0$$

$$\mathbf{x}_p^{(2)} = d^{(2)}\mathbf{v} = 3.6 \begin{bmatrix} 0.9 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 3.2 \\ 1.8 \end{bmatrix}$$

Decomposing Matrices

Singular Value Decomposition Theorem **(SVD)** we know that (stated without proof)
Any *matrix* X can be decomposed into $X=U\,S\,V^T$ where S is a **diagonal** matrix containing positive real values. V is **orthogonal** matrix (i.e. square with orthonormal columns), and U has orthonormal columns.

$V^{-1}=V^T$

Thus every ***symmetric*** matrix, $A = X^T X$, can be written as

$A = X^T X = (USV^T)^T(USV^T) = (VSU^T)(USV^T) = VDV^T$ where V is an **orthogonal** matrix, and D is a **diagonal** matrix

$$\begin{bmatrix} -0.47 & -0.37 & -0.47 & 0.46 & 0.46 \end{bmatrix} \begin{bmatrix} -0.47 \\ -0.37 \\ -0.47 \\ 0.46 \\ 0.46 \end{bmatrix} = 1$$

$$A = X^T_{\text{centered}} X_{\text{centered}} = \begin{bmatrix} 26.86 & 21.29 & 26.86 & -20.43 & -20.43 \\ 21.29 & 19.43 & 21.29 & -15.14 & -15.14 \\ 26.86 & 21.29 & 26.86 & -20.43 & -20.43 \\ -20.43 & -15.14 & -20.43 & 27.71 & 27.71 \\ -20.43 & -15.14 & -20.43 & 27.71 & 27.71 \end{bmatrix}$$

$$\begin{bmatrix} -0.36 & -0.41 & -0.36 & -0.54 & -0.54 \end{bmatrix} \begin{bmatrix} -0.47 \\ -0.37 \\ -0.47 \\ 0.46 \\ 0.46 \end{bmatrix} = 0$$

$$= \underbrace{\begin{bmatrix} -0.47 & -0.36 & -0.39 & -0.71 & 0. \\ -0.37 & -0.41 & 0.83 & -0. & -0. \\ -0.47 & -0.36 & -0.39 & 0.71 & -0. \\ 0.46 & -0.54 & -0.06 & 0. & -0.71 \\ 0.46 & -0.54 & -0.06 & -0. & 0.71 \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} 110.09 & 0 & 0 & 0 & 0 \\ 0 & 16.73 & 0 & 0 & 0 \\ 0 & 0 & 1.75 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{D} \underbrace{\begin{bmatrix} -0.47 & -0.37 & -0.47 & 0.46 & 0.46 \\ -0.36 & -0.41 & -0.36 & -0.54 & -0.54 \\ -0.39 & 0.83 & -0.39 & -0.06 & -0.06 \\ -0.71 & -0. & 0.71 & 0. & 0. \\ 0. & -0. & 0. & -0.71 & 0.71 \end{bmatrix}}_{V^T}$$

$$\underbrace{\begin{bmatrix} -0.47 & -0.36 & -0.39 & -0.71 & 0. \\ -0.37 & -0.41 & 0.83 & -0. & -0. \\ -0.47 & -0.36 & -0.39 & 0.71 & -0. \\ 0.46 & -0.54 & -0.06 & 0. & -0.71 \\ 0.46 & -0.54 & -0.06 & -0. & 0.71 \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} 0.47 & -0.37 & -0.47 & 0.46 & 0.46 \\ -0.36 & -0.41 & -0.36 & -0.54 & -0.54 \\ -0.39 & 0.83 & -0.39 & -0.06 & -0.06 \\ -0.71 & -0. & 0.71 & 0. & 0. \\ 0. & -0. & 0. & -0.71 & 0.71 \end{bmatrix}}_{V^T} = \begin{bmatrix} 1. & 0. & 0. & 0. & 0. \\ 0. & 1. & 0. & 0. & 0. \\ 0. & 0. & 1. & 0. & 0. \\ 0. & 0. & 0. & 1. & 0. \\ 0. & 0. & 0. & 0. & 1. \end{bmatrix}$$

Since V is an orthogonal matrix, what is VV^T?

A) It depends on the values in V
B) Identity matrix
C) A
D) 0 matrix

$$\begin{bmatrix}-0.47 & -0.37 & -0.47 & 0.46 & 0.46\end{bmatrix}\begin{bmatrix}-0.47\\-0.37\\-0.47\\0.46\\0.46\end{bmatrix}=1$$

$$\begin{bmatrix}-0.36 & -0.41 & -0.36 & -0.54 & -0.54\end{bmatrix}\begin{bmatrix}-0.47\\-0.37\\-0.47\\0.46\\0.46\end{bmatrix}=0$$

$$A = \begin{bmatrix} 26.86 & -20.43 & -20.43\\ 21.29 & -15.14 & -15.14\\ 26.86 & -20.43 & -20.43\\ 20.43 & 27.71 & 27.71\\ 20.43 & 27.71 & 27.71\end{bmatrix}$$

$$= \underset{V}{\begin{bmatrix}-0.47 & -0.36 & -0.39 & -0.71 & 0.\\ -0.37 & -0.41 & 0.83 & -0. & -0.\\ -0.47 & -0.36 & -0.39 & 0.71 & -0.\\ 0.46 & -0.54 & -0.06 & 0. & -0.71\\ 0.46 & -0.54 & -0.06 & -0. & 0.71\end{bmatrix}} \underset{D}{\begin{bmatrix}110.09 & 0 & 0 & 0 & 0\\ 0 & 16.73 & 0 & 0 & 0\\ 0 & 0 & 1.75 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\end{bmatrix}} \underset{V^T}{\begin{bmatrix}-0.47 & -0.37 & -0.47 & 0.46 & 0.46\\ -0.36 & -0.41 & -0.36 & -0.54 & -0.54\\ -0.39 & 0.83 & -0.39 & -0.06 & -0.06\\ -0.71 & -0. & 0.71 & 0. & 0.\\ 0. & -0. & 0. & -0.71 & 0.71\end{bmatrix}}$$

$$\underset{V}{\begin{bmatrix}-0.47 & -0.36 & -0.39 & -0.71 & 0.\\ -0.37 & -0.41 & 0.83 & -0. & -0.\\ -0.47 & -0.36 & -0.39 & 0.71 & -0.\\ 0.46 & -0.54 & -0.06 & 0. & -0.71\\ 0.46 & -0.54 & -0.06 & -0. & 0.71\end{bmatrix}} \underset{V^T}{\begin{bmatrix}-0.47 & -0.37 & -0.47 & 0.46 & 0.46\\ -0.36 & -0.41 & -0.36 & -0.54 & -0.54\\ -0.39 & 0.83 & -0.39 & -0.06 & -0.06\\ -0.71 & -0. & 0.71 & 0. & 0.\\ 0. & -0. & 0. & -0.71 & 0.71\end{bmatrix}} = \begin{bmatrix}1. & 0. & 0. & 0. & 0.\\ 0. & 1. & 0. & 0. & 0.\\ 0. & 0. & 1. & 0. & 0.\\ 0. & 0. & 0. & 1. & 0.\\ 0. & 0. & 0. & 0. & 1.\end{bmatrix}$$

1) https://www.stat.cmu.edu/~cshalizi/uADA/12/lectures/ch18.pdf

2) http://theory.stanford.edu/~tim/s17/l/l8.pdf

Or you can read the more complete set of notes:
a)http://web.stanford.edu/class/cs168/l/l7.pdf

b) http://theory.stanford.edu/~tim/s17/l/l8.pdf

c) http://web.stanford.edu/class/cs168/l/l9.pdf

Lecture Principal Component Analysis

PROF. LINDA SELLIE

Outline

In this lecture we are not augmenting the feature vector with an extra 1

- ❑ Reduce number of features and find latent features
  - Motivation
  - Intuition on keeping the variance of the data
  - Toy Example of projecting data onto a lower dimensional space
  - Which line should we project onto?
- ❑ Algorithm
- ❑ Examples
- ❑ Finding principle components
- ❑ When PCA doesn't work well

Unsupervised Learning

$$\left\{(\mathbf{x}^{(1)}, \cancel{y^{(1)}}), (\mathbf{x}^{(2)}, \cancel{y^{(2)}}), \ldots, (\mathbf{x}^{(N)}, \cancel{y^{(N)}})\right\}$$

$$\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}\}$$

Used for:

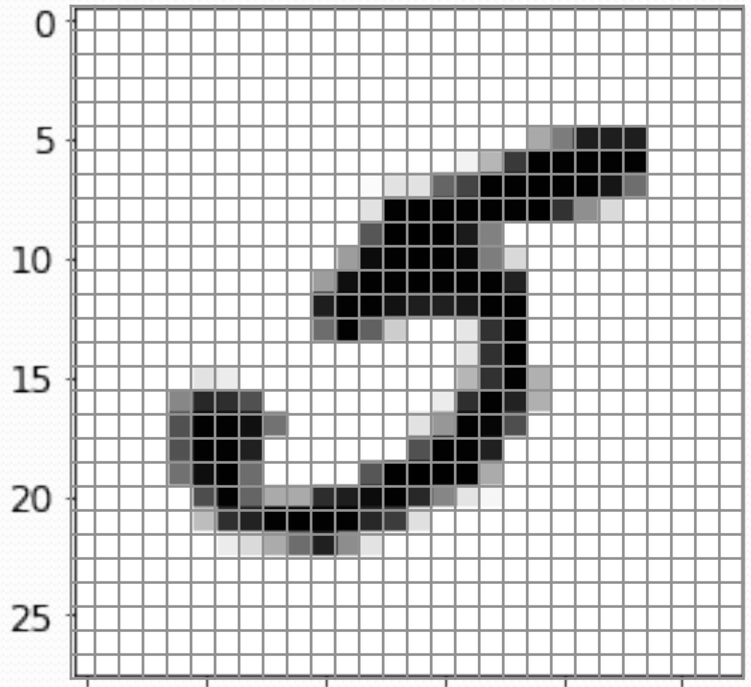
- **unsupervised learning problems**
- **as a preprocessing step before a supervised learning problem**

How can we remove useless dimensions

Raw data: potentially complex, high-dimensional

Goal is to find a transformation of the features of **x**: $z = \Phi(x)$ Where the number of features of **z** is less than the number of features of **x**.

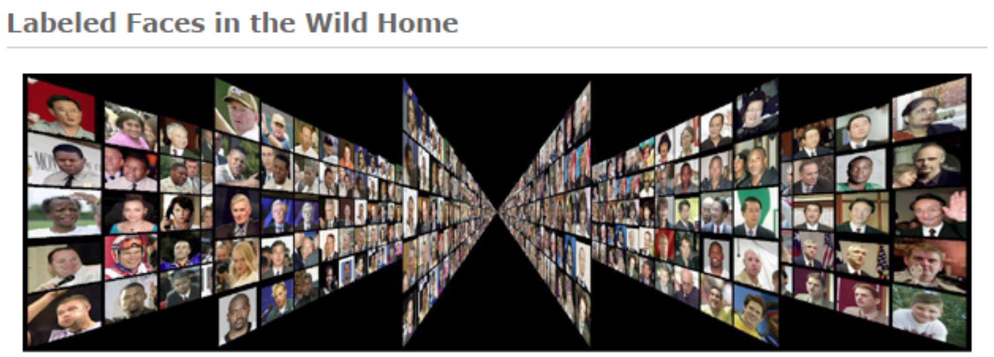
***We want to remove**: redundancy, less informative dimensions, and noise*



Data in form original gathered.
Can we find a "better" representation?

How do we define better?

Example: Faces



- ❑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

The curse of dimensionality

Having many features makes visualizing the data difficult

Many learning tasks get more difficult as the dimensions (# of features) increase:

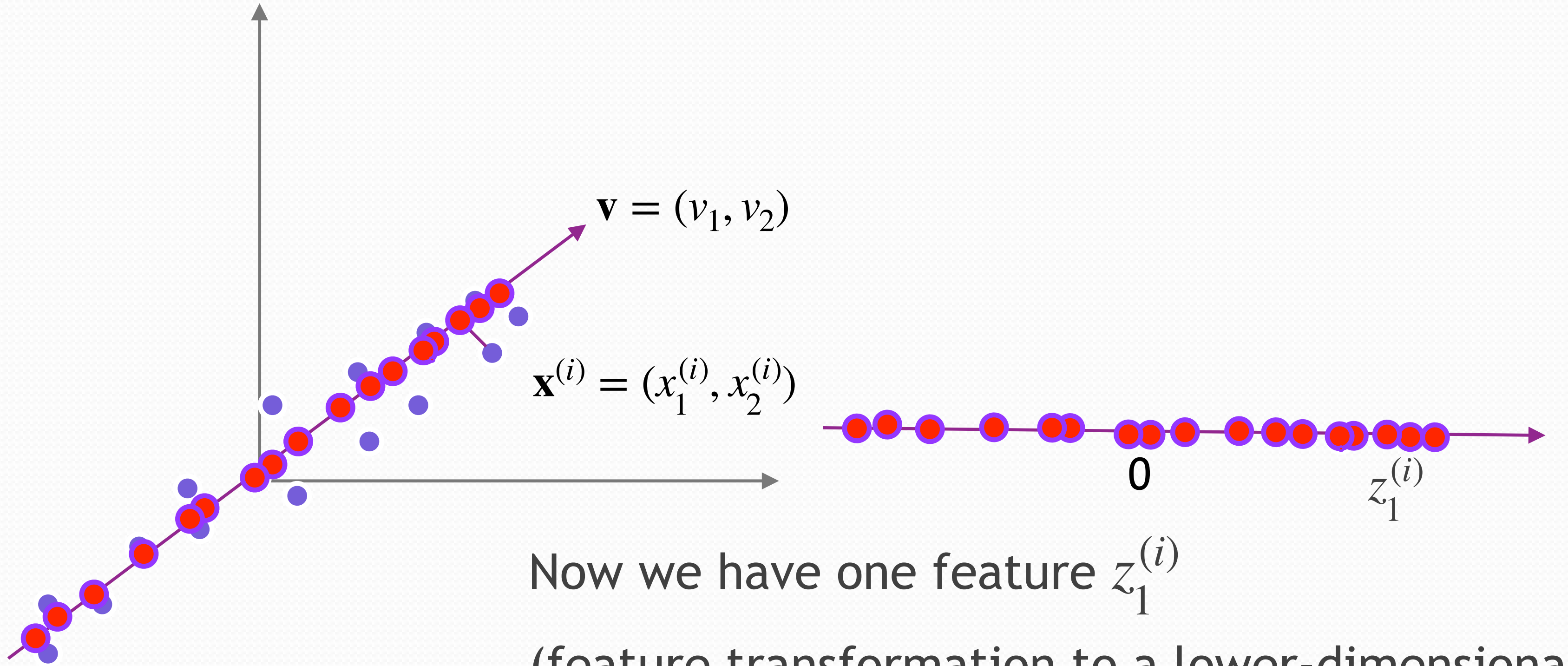
- computation (most learning problem's computational complexity scales polynomially with dimensionality)
- Noise starts affecting the computation - random fluctuations of dimensions not correlated with the outcome.
- Generalization is more difficult for many learning tasks (more features means more data is needed)

Outline

In this lecture we are not augmenting the feature vector with an extra 1

- ❑Reduce number of features and find latent features
  - Motivation
  - Intuition on keeping the variance of the data
  - Toy Example of projecting data onto a lower dimensional space
  - Which line should we project onto?
- ❑ Algorithm
- ❑ Examples
- ❑ Finding principle components
- ❑ When PCA doesn’t work well

Dimensionality reduction (Big idea)

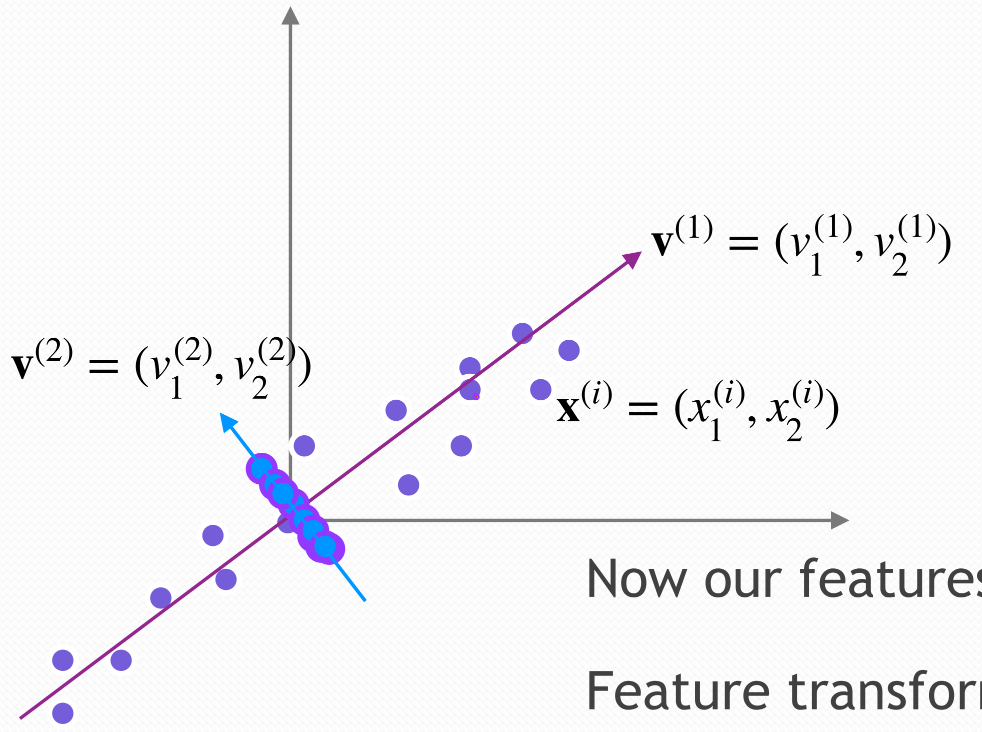

$\mathbf{v} = (v_1, v_2)$
$\mathbf{x}^{(i)} = (x_1^{(i)}, x_2^{(i)})$
0
$z_1^{(i)}$

Now we have one feature $z_1^{(i)}$

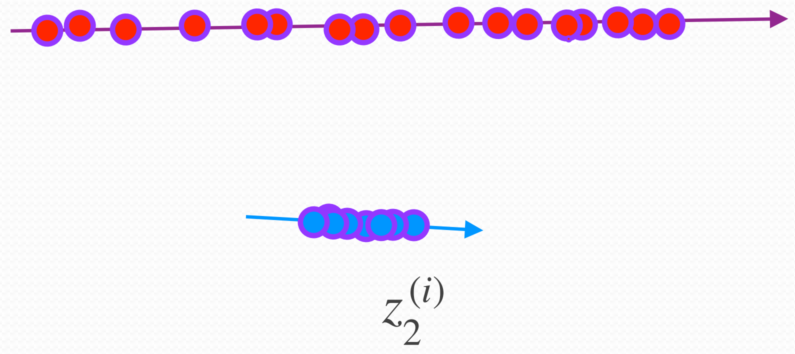
(feature transformation to a lower-dimensional space $\Phi(\mathbf{x}^{(i)}) = (z_1^{(i)})$)

If $\mathbf{v}$ is a unit vector, we can get an approximation to $(x_1^{(i)}, x_2^{(i)})$ by $z^{(i)}\mathbf{v}$

🤞 $\Phi^{-1}\left(\Phi(\mathbf{x}^{(i)})\right) = z_1^{(i)}\mathbf{v} \approx \mathbf{x}^{(i)}$ for all $i \in 1,\ldots,N$



$\mathbf{v}^{(1)} = (v_1^{(1)}, v_2^{(1)})$
$\mathbf{v}^{(2)} = (v_1^{(2)}, v_2^{(2)})$
$\mathbf{x}^{(i)} = (x_1^{(i)}, x_2^{(i)})$



Now our features are $(z_1^{(i)}, z_2^{(i)})$

Feature transformation $\Phi(\mathbf{x}^{(i)}) = (z_1^{(i)}, z_2^{(i)})$

If $\mathbf{v}^{(1)}, \mathbf{v}^{(2)}$ are unit vectors and perpendicular to each other, we can easily reconstruct $(x_1^{(i)}, x_2^{(i)}) = z_1^{(i)}\mathbf{v}^{(1)} + z_2^{(i)}\mathbf{v}^{(2)}$

PCA: Principle Component Analysis

Big idea:

1. For each feature, compute the mean. Let $\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}^{(i)}$

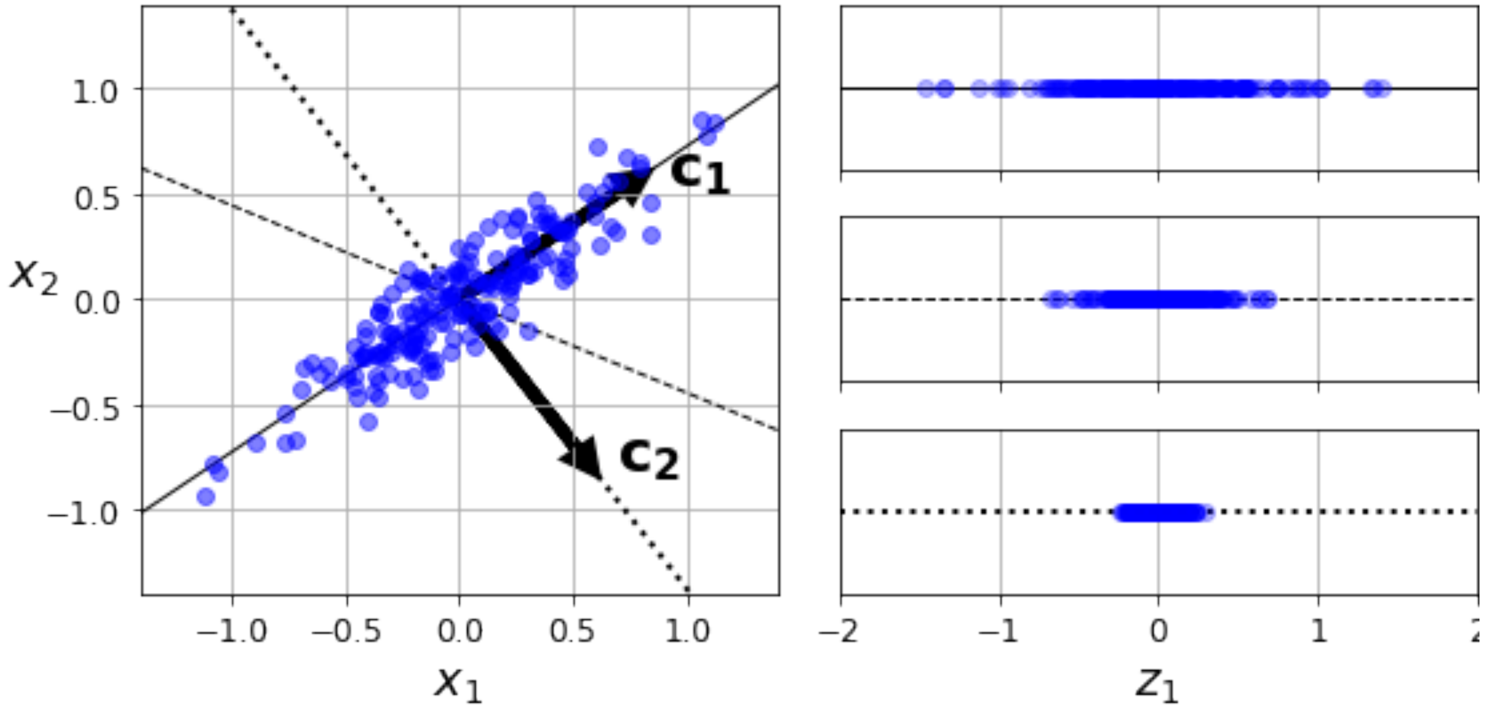
   (Or zero center the training examples $\mathbf{x}^{(i)}$)

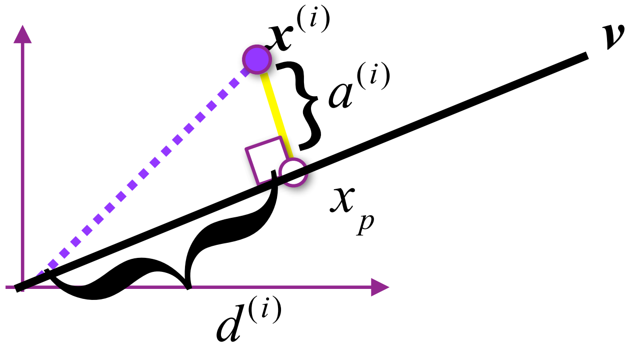
2. Find $k < d$ vectors in $\mathbb{R}^d$: $\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \cdots, \mathbf{v}^{(k)}$ which are unit vectors and perpendicular to each other (orthonormal)

3. For each training, example compute $\Phi(\mathbf{x}) = (z_1, z_2, \cdots, z_k)$ where $z_i = (\mathbf{x} - \bar{\mathbf{x}})^T \mathbf{v}^{(i)}$

The rest of the lecture is about determining how to choose $\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \cdots, \mathbf{v}^{(k)}$ such that $\Phi(\mathbf{x}^{(i)}) = (z_1^{(i)}, z_2^{(i)}, \cdots, z_k^{(i)})$ maximizes variance (and minimizes least-square reconstruction error)

Compression





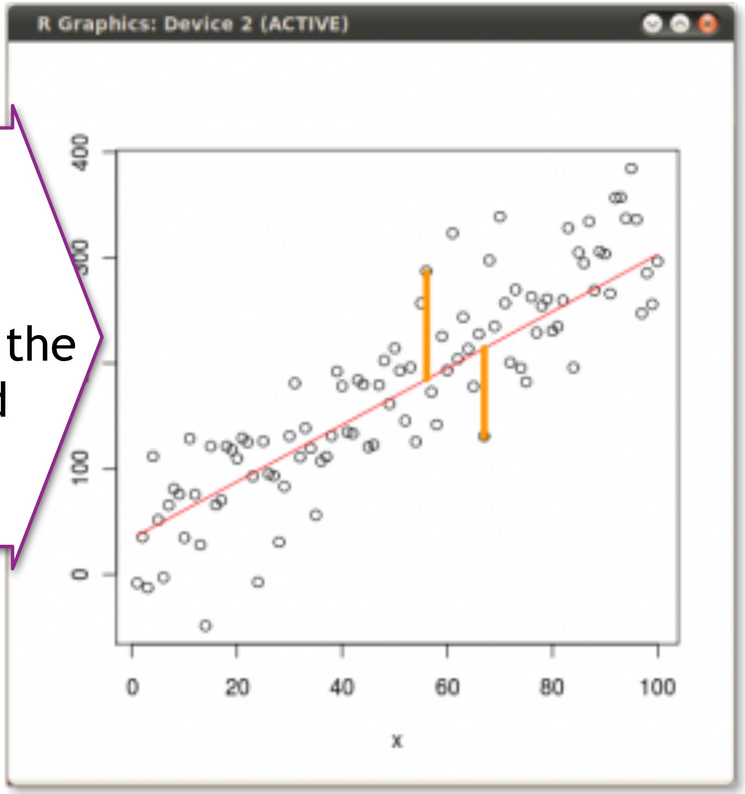
Note that minimizing $\sum_{i=1}^{N} a^{(i)2}$

is the same as maximizes $\sum_{i=1}^{N} d^{(i)2}$

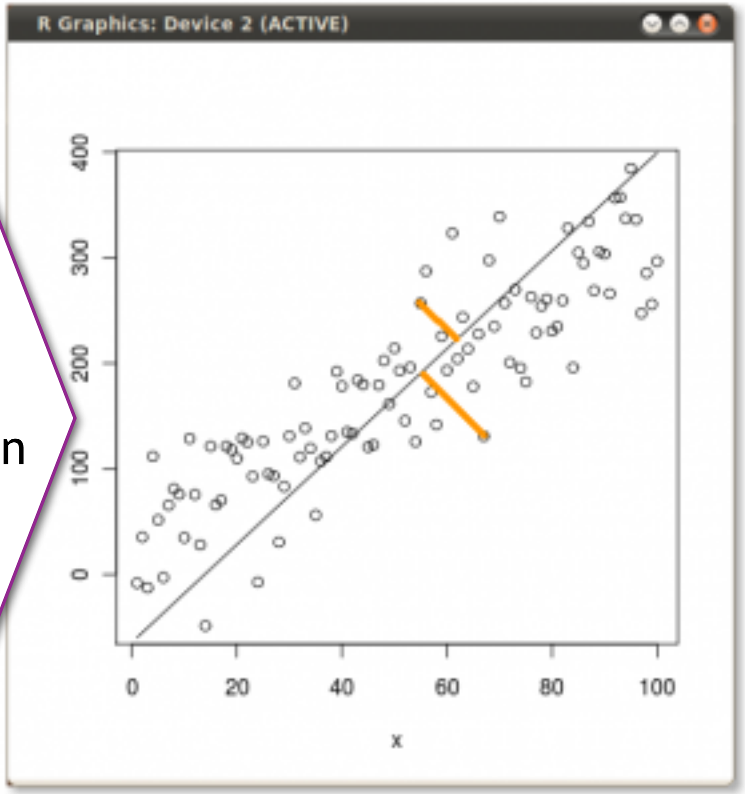
Picture on left from https://github.com/ageron/handson-ml/blob/master/08_dimensionality_reduction.ipynb

OLS

PCA



Error: vertical distance between the point and the line



Reconstruction error 2D to 1D: euclidean distance between the point and the line

Image from https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues

Outline

In this lecture we are not augmenting the feature vector with an extra 1

- ❑Reduce number of features and find latent features
  - Motivation
  - Intuition on keeping the variance of the data
  - Toy Example of projecting data onto a lower dimensional space
  - Which line should we project onto?
- ❑ Algorithm
- ❑ Examples
- ❑ Finding principle components
- ❑ When PCA doesn’t work well

Does the choice of basis matter?

http://web.stanford.edu/class/cs168/l/l7.pdf

Toy Example from Tim Roughgarden

- 4 foods rated on a scale from 1 to 10
- N = 4, and d = 4

	Kale	Taco Bell	Sashimi	Pop Tarts
Alice	10	1	2	7
Bob	7	2	1	10
Carolyn	2	9	7	3
Dave	3	6	10	2

- Each data point can be approximated by $\bar{\mathbf{x}} + z_1 v^{(1)} + z_2 v^{(2)}$

$$\bar{\mathbf{x}} = (5.5, 4.5, 5, 5.5) \quad \mathbf{v}^{(1)} = (3, -3, -3, 3) \quad \mathbf{v}^{(2)} = (1, -1, 1, -1)$$

Feature 1

Feature 2

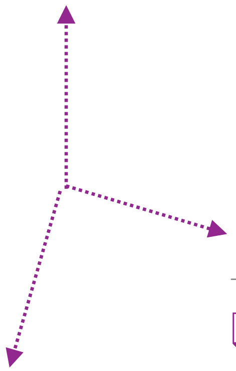
Then

- Alice: $(z_1,z_2) = (1,1)$
- Bob: $(z_1,z_2) = (1,-1)$
- Carolyn: $(z_1,z_2) = (-1,-1)$
- Dave: $(z_1,z_2) = (-1,1)$

Everyone has a $v^{(1)}$ coordinate z_1 and a $v^{(2)}$ coordinate z_2

N=4
d=2

	$v^{(1)}$	$v^{(2)}$
Alice	1	1
Bob	1	-1
Carolyn	-1	-1
Dave	-1	1



Toy Example from Th

The current basis is based on the questions we asked. If we had asked instead if they were 1) vegetarian and 2) health conscious we would have gotten a better set of features

.pdf

- 4 foods rated on a scale from 1 to 10
- N = 4, and d = 4

	Kale	Taco Bell	Sashimi	Pop Tarts
Alice	10	1	2	7
Bob	7	2	1	10
Carolyn	2	9	7	3
Dave	3	6	10	2

- Each data point can be approximated by $\bar{\mathbf{x}} + z_1 v^{(1)} + z_2 v^{(2)}$

$$\bar{\mathbf{x}} = (5.5, 4.5, 5, 5.5) \qquad \mathbf{v}^{(1)} = (3, -3, -3, 3) \qquad \mathbf{v}^{(2)} = (1, -1, 1, -1)$$

Feature 1

Feature 2

Then

- Alice: $(z_1,z_2) = (1,1)$
- Bob: $(z_1,z_2) = (1,-1)$
- Carolyn: $(z_1,z_2) = (-1,-1)$
- Dave: $(z_1,z_2) = (-1,1)$

Everyone has a $v^{(1)}$ coordinate z_1 and a $v^{(2)}$ coordinate z_2

N=4
d=2

	$v^{(1)}$	$v^{(2)}$
Alice	1	1
Bob	1	-1
Carolyn	-1	-1
Dave	-1	1

What is the “best” way to represent the data?

CAN WE FIND BETTER (LATENT) FEATURES

OUR NEW FEATURES WILL BE A LINEAR COMBINATION OF THE OLD FEATURES

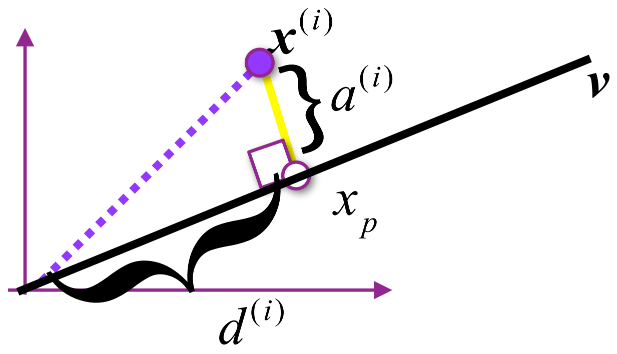
Outline

In this lecture we are not augmenting the feature vector with an extra 1

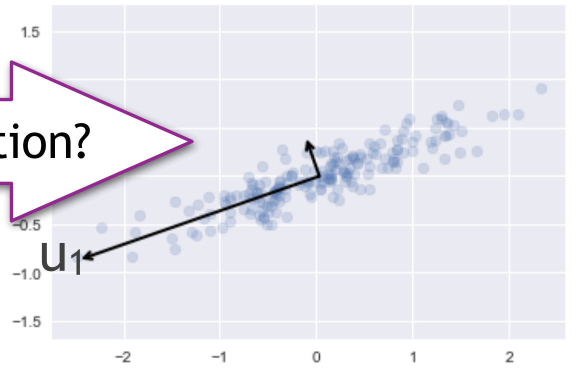
- Standard **unsupervised** preprocessing: zero centering, data normalization
- Reduce number of features and find latent features
  - Motivation
  - Intuition on keeping the variance of the data
  - Toy Example of projecting data onto a lower dimensional space
  - Which line should we project onto?
- Algorithm
- Examples
- Finding principle components
- When PCA doesn't work well

Data

- Principle Component Analysis (PCA) is an orthogonal projection of the data onto a lower-dimensional linear space that:
  - maximizes the variance of the projected data
  - minimizes the mean *squared* distances between the data poir
    and its projection onto the line



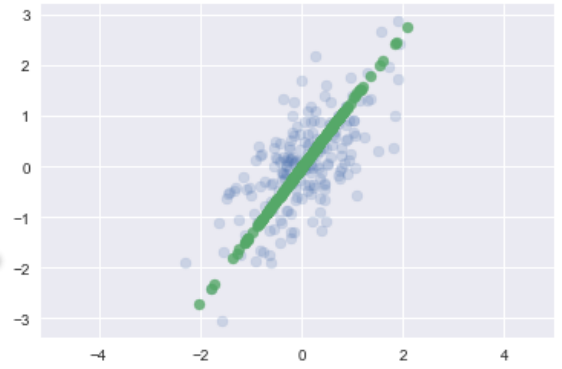
Noise or information?



- How can we find **v**, the direction of the largest variance?

Note that minimizing $\sum_{i=1}^{N} a^{(i)\,2}$ is the same as maximizes $\sum_{i=1}^{N} d^{(i)\,2}$

Goal is to find a new coordinate system in the direction of greatest variability



A proof can be found in: https://www.stat.cmu.edu/~cshalizi/uADA/12/lectures/ch18.pdf

Outline

In this lecture we are not augmenting the feature vector with an extra 1

- ❑ Standard **unsupervised** preprocessing: zero centering, data normalization
- ❑ Reduce number of features and find latent features
  - Motivation
  - Intuition on keeping the variance of the data
  - Toy Example of projecting data onto a lower dimensional space
  - Which line should we project onto?
- ❑ Algorithm
- ❑ Examples
- ❑ Finding principle components
- ❑ When PCA doesn't work well

PCA Algorithm

(This algorithm is modified from the book *Learning from Data* by Yasar Abu-Mostafa et al)

US coordinates of concepts
V concept vectors
S importance of concept

- PCA Algorithm:
- Inputs: The zero-centered data matrix X and k >=1
  1) Compute the SVD of X: [U,S,V]=svd (X)
  2) Let $V_k = [\mathbf{v}^{(1)},\dots,\mathbf{v}^{(k)}]$ be the first k columns of V
  3) The PCA-feature matrix is $Z = XV_k$

These vectors are the optimal coordinate basis

The values we ignore incur the least reconstruction error

k-features for each example

$$Z = \begin{pmatrix} - & \boldsymbol{x}^{(1)T} & - \\ - & \boldsymbol{x}^{(2)T} & - \\ & \vdots & \\ - & \boldsymbol{x}^{(N)T} & - \end{pmatrix} \begin{pmatrix} | & | & & | \\ \boldsymbol{v}^{(1)} & \boldsymbol{v}^{(2)} & \cdots & \boldsymbol{v}^{(k)} \\ | & | & & | \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}^{(1)T}\boldsymbol{v}^{(1)} & \cdots & \boldsymbol{x}^{(1)T}\boldsymbol{v}^{(k)} \\ \boldsymbol{x}^{(2)T}\boldsymbol{v}^{(1)} & & \boldsymbol{x}^{(2)T}\boldsymbol{v}^{(k)} \\ & \vdots & \\ \boldsymbol{x}^{(N)T}\boldsymbol{v}^{(1)} & \cdots & \boldsymbol{x}^{(N)T}\boldsymbol{v}^{(k)} \end{pmatrix} = \begin{pmatrix} - & \boldsymbol{z}^{(1)T} & - \\ - & \boldsymbol{z}^{(2)T} & \\ & \vdots & \\ - & \boldsymbol{z}^{(N)T} & - \end{pmatrix}$$

More information can be found here: https://stats.stackexchange.com/questions/134282/relationship-between-svd-and-pca-how-to-use-svd-to-perform-pca?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa

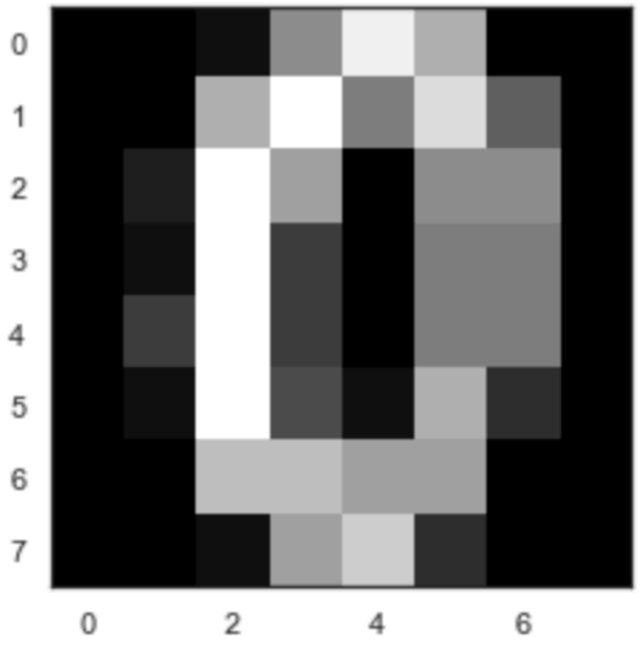
A nice tutorial https://davetang.org/file/Singular_Value_Decomposition_Tutorial.pdf
We are assuming the diagonal elements in S go from largest to smallest

Outline

In this lecture we are not augmenting the feature vector with an extra 1

- ❑ Reduce number of features and find latent features
  - Motivation
  - Intuition on keeping the variance of the data
  - Toy Example of projecting data onto a lower dimensional space
  - Which line should we project onto?
- ❑ Algorithm
- ❑ Examples
- ❑ Finding principle components
- ❑ When PCA doesn't work well

Preprocessing: Flatten



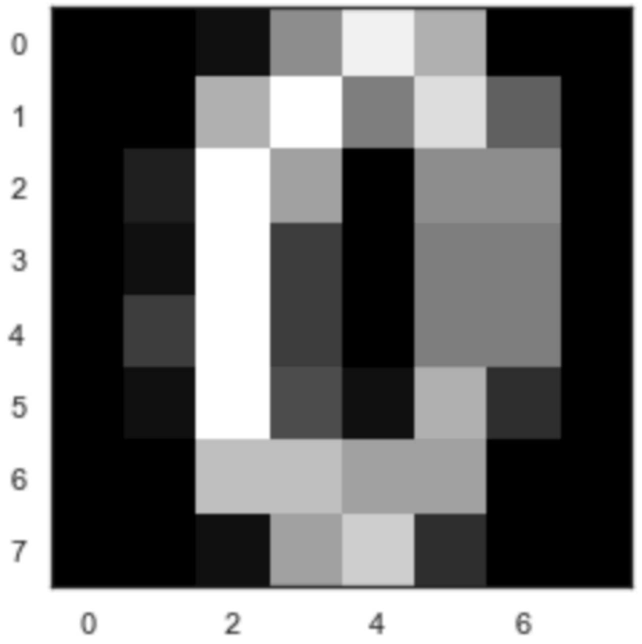
```
[ 0.  0.  1.  9. 15. 11.  0.  0.]
[ 0.  0. 11. 16.  8. 14.  6.  0.]
[ 0.  2. 16. 10.  0.  9.  9.  0.]
[ 0.  1. 16.  4.  0.  8.  8.  0.]
[ 0.  4. 16.  4.  0.  8.  8.  0.]
[ 0.  1. 16.  5.  1. 11.  3.  0.]
[ 0.  0. 12. 12. 10. 10.  0.  0.]
[ 0.  0.  1. 10. 13.  3.  0.  0.]
```

$\rightarrow$

Feature one

```
 0.
 0.
 1.
 9.
15.
11.
 0.
 0.
 0.
 0.
11.
16.
 8.
14.
 6.
 0.
 0.
 2.
16.
10.
 0.
 9.
 9.
 0.
 0.
 1.
16.
 4.
 0.
 8.
 8.
 0.
 0.
 4.
16.
 4.
 0.
 8.
 8.
 0.
 0.
 1.
16.
 5.
 1.
11.
 3.
 0.
 0.
 0.
12.
12.
10.
10.
 0.
 0.
 0.
 0.
 1.
10.
13.
 3.
 0.
 0.
```

Preprocessing: Flatten



```
[ 0.  0.  1.  9. 15. 11.  0.  0.]
[ 0.  0. 11. 16.  8. 14.  6.  0.]
[ 0.  2. 16. 10.  0.  9.  9.  0.]
[ 0.  1. 16.  4.  0.  8.  8.  0.]
```

$\rightarrow$

```
 0.
 0.
 1.
 9.
15.
11.
 0.
 0.
 0.
 0.
11.
16.
 8.
14.
 6.
 0.
 0.
 2.
16.
10.
 0.
 9.
 9.
 0.
 0.
 1.
16.
 4.
 0.
 8.
 8.
```

Feature two

In the next few slides, I will show the digits in the original 2D shape to illustrate how the lower dimensional space can still capture the essential details

Example 1

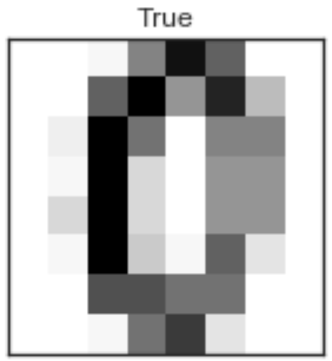
Adapted from the book *Python Data Science Handbook*

- Looking at 8x8 digits data set, we had 64 features, one for each pixel $\boldsymbol{x}^T = (x_1, x_2, \ldots, x_{64})$
- We can think of our data set as being in a basis - a pixel basis!

$$\text{image}(\boldsymbol{x}) = x_1 \cdot (\text{pixel 1}) + x_2 \cdot (\text{pixel 2}) + \cdots + x_{64} \cdot (\text{pixel 64})$$

- We could reduce the number of features by just selecting some of the original features.... (i.e. 8 pixels out of 64 original pixels.) We can do this by zeroing out all but 8 of the basis vectors!

$$\text{image}(\boldsymbol{x}) = x_1 \cdot (\text{pixel 1}) + x_2 \cdot (\text{pixel 2}) + \cdots + x_8 \cdot (\text{pixel 8})$$



- We notice that this will not generalize well
- What if we chose a different basis function?

Example 1

Adapted from the book *Python Data Science Handbook*

- Looking at 8x8 digits data set, we had 64 features, one for each pixel $\boldsymbol{x}^T = (x_1, x_2, \ldots, x_{64})$
- We can think of our data set as being in a basis - a pixel basis!

How many features do we need to use so we can recognize a digit? (Assuming I know the original mean and the features)

a) 0-5
b) 6-10
c) 11-15
d) 16-20
e) 21-30
f) More than 30

$\cdot(\text{pixel } 2) + \cdots + x_{64} \cdot (\text{pixel } 64)$

features by just selecting some of the original features…. (i.e. 8

Ve can do this by zeroing out all but 8 of the basis vectors!

$_2 \cdot (\text{pixel } 2) + \cdots + x_8 \cdot (\text{pixel } 8)$

- We notice that this will not generalize well
- What if we chose a different basis function?

Example 1

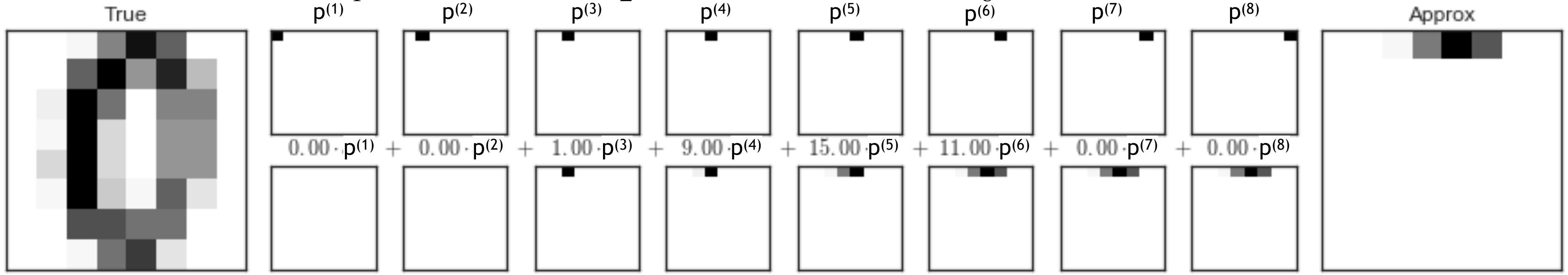
Adapted from the book *Python Data Science Handbook*

- Looking at 8x8 digits data set, we had 64 features, one for each pixel $\boldsymbol{x}^T = (x_1, x_2, \ldots, x_{64})$
- We can think of our data set as being in a basis - a pixel basis!

$$\text{image}(\boldsymbol{x}) = x_1 \cdot (\text{pixel }1) + x_2 \cdot (\text{pixel }2) + \cdots + x_{64} \cdot (\text{pixel }64)$$

- We could reduce the number of features by just selecting some of the original features…. (i.e. 8 pixels out of 64 original pixels.) We can do this by zeroing out all but 8 of the basis vectors!

$$\text{image}(\boldsymbol{x}) = x_1 \cdot (\text{pixel }1) + x_2 \cdot (\text{pixel }2) + \cdots + x_8 \cdot (\text{pixel }8)$$


True
$p^{(1)}$
$p^{(2)}$
$p^{(3)}$
$p^{(4)}$
$p^{(5)}$
$p^{(6)}$
$p^{(7)}$
$p^{(8)}$
Approx
$0.00 \cdot p^{(1)} + 0.00 \cdot p^{(2)} + 1.00 \cdot p^{(3)} + 9.00 \cdot p^{(4)} + 15.00 \cdot p^{(5)} + 11.00 \cdot p^{(6)} + 0.00 \cdot p^{(7)} + 0.00 \cdot p^{(8)}$

- We notice that this will not generalize well
- What if we chose a different basis function?

New Features?

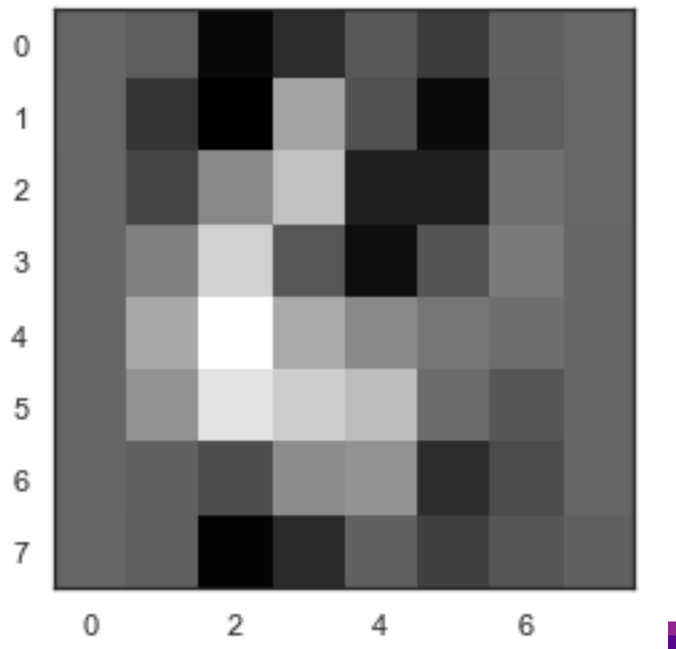
How many new features do we need till we can recognize a number ?

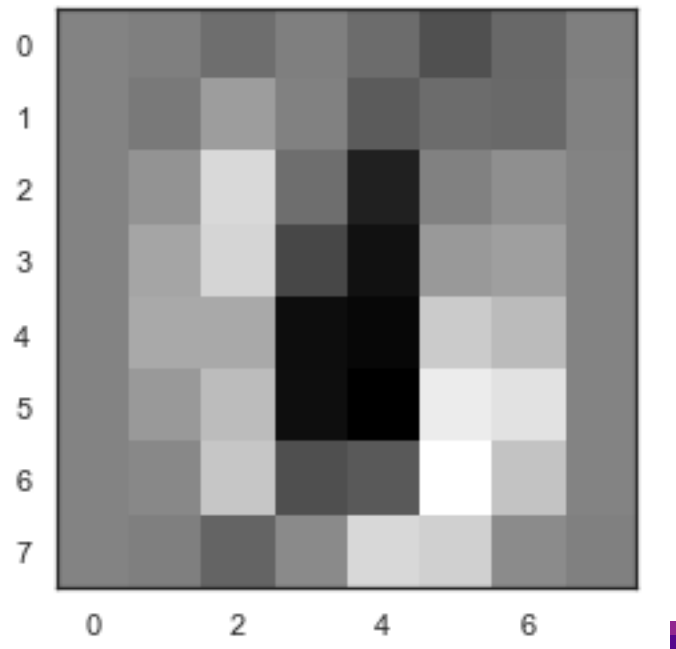
$\mathbf{v}^{(1)}$

```
[ 0.   -0.02 -0.22 -0.14 -0.03 -0.1  -0.01  0.  ]
[-0.   -0.12 -0.24  0.15 -0.05 -0.22 -0.01  0.  ]
[-0.   -0.08  0.08  0.22 -0.17 -0.16  0.03  0.  ]
[ 0.    0.06  0.25 -0.04 -0.21 -0.04  0.05  0.  ]
[ 0.    0.16  0.37  0.16  0.09  0.04  0.02  0.  ]
[ 0.    0.11  0.3   0.25  0.21  0.01 -0.04  0.  ]
[ 0.   -0.01 -0.06  0.09  0.11 -0.14 -0.06  0.  ]
[ 0.   -0.01 -0.24 -0.14 -0.01 -0.09 -0.04 -0.01]
```

$\mathbf{v}^{(2)}$

```
[ 0.   -0.01 -0.05 -0.01 -0.05 -0.12 -0.06 -0.01]
[-0.   -0.02  0.06 -0.01 -0.09 -0.05 -0.06 -0.  ]
[-0.    0.04  0.2  -0.05 -0.23 -0.    0.03 -0.  ]
[-0.    0.08  0.19 -0.14 -0.26  0.05  0.07  0.  ]
[ 0.    0.09  0.09 -0.27 -0.29  0.17  0.13  0.  ]
[ 0.    0.05  0.13 -0.27 -0.3   0.24  0.22  0.  ]
[ 0.    0.01  0.15 -0.12 -0.1   0.29  0.15  0.  ]
[-0.   -0.01 -0.07  0.02  0.19  0.18  0.02 -0.01]
```





Example 1 (cont)

$$x^T = (x_1, x_2, \ldots, x_{64})$$

- We could use new features (a different basis function, the PCA basis ($\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(64)}$))

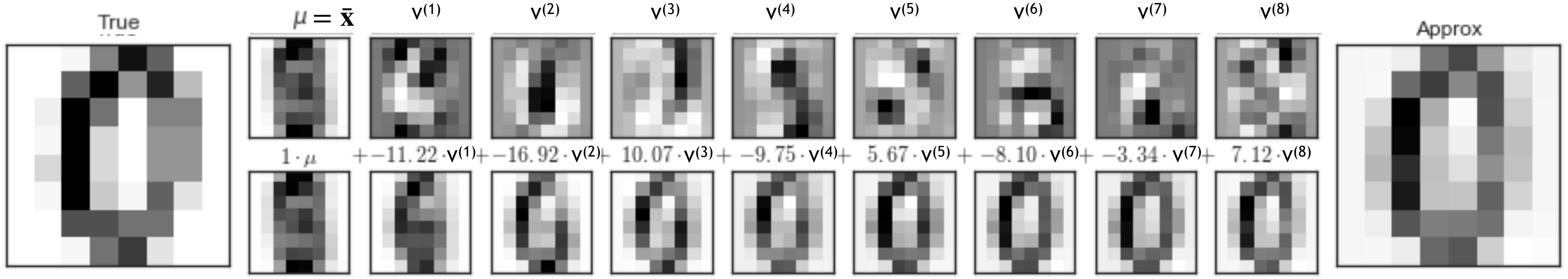
$$z = (x^T\mathbf{v}^{(1)}, x^T\mathbf{v}^{(2)}, \cdots, x^T\mathbf{v}^{(64)})^T = (z_1, z_2, \cdots, z_{64})^T \quad \text{thus} \quad \text{image}(x) = z_1\mathbf{v}^{(1)} + z_2\mathbf{v}^{(2)} + \cdots + z_{64}\mathbf{v}^{(64)}$$

- We could reduce the number of features by just selecting some of the original features (i.e. 8 basis vectors out of 64 original basis vectors), we can do this by zeroing out all but 8 of the basis vectors

$$\text{image}(x) = z_1\mathbf{v}^{(1)} + z_2\mathbf{v}^{(2)} + \cdots + z_8\mathbf{v}^{(8)}$$

We will zero center our data first and then choose the 8 new features in z-space

- The 8 PCA features (basis vectors are the 8 principle components) will be a low dimensional representation of our data

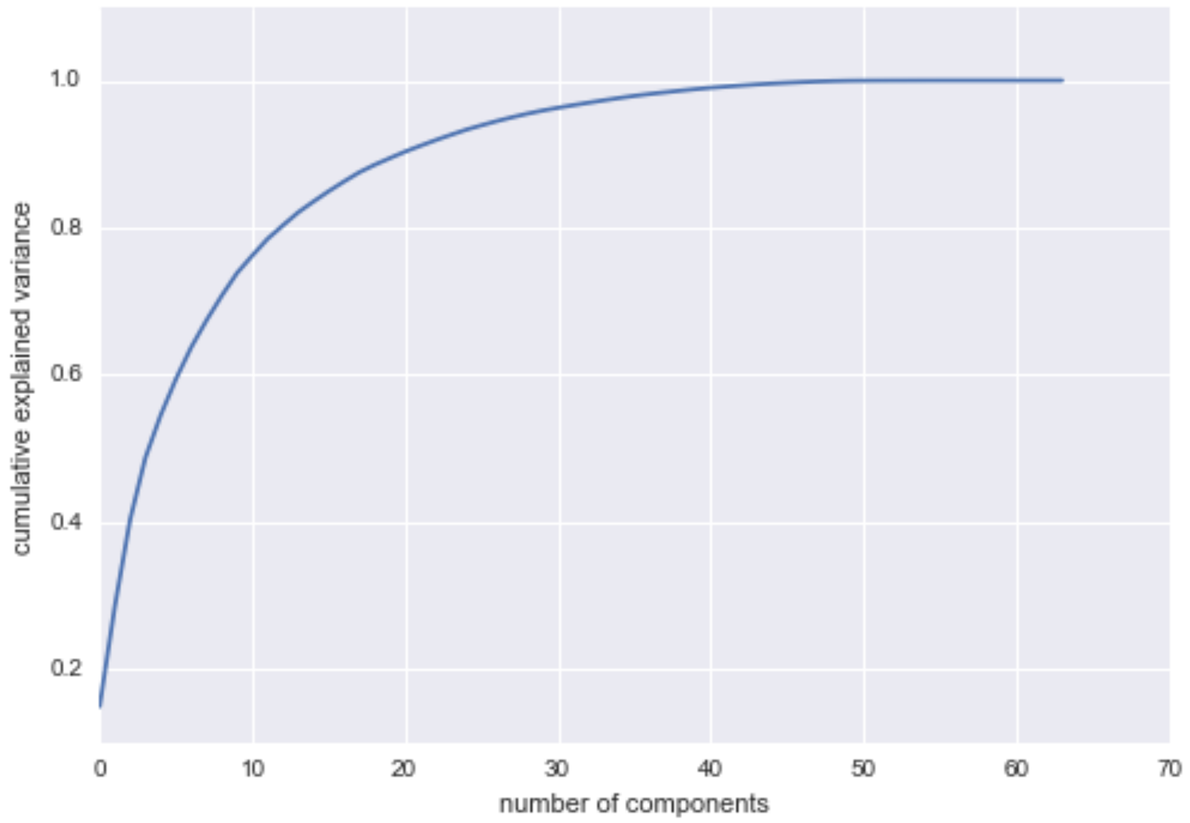


True
$\mu = \bar{\mathbf{x}}$
$v^{(1)}$
$v^{(2)}$
$v^{(3)}$
$v^{(4)}$
$v^{(5)}$
$v^{(6)}$
$v^{(7)}$
$v^{(8)}$
Approx
$1 \cdot \mu + -11.22 \cdot v^{(1)} + -16.92 \cdot v^{(2)} + 10.07 \cdot v^{(3)} + -9.75 \cdot v^{(4)} + 5.67 \cdot v^{(5)} + -8.10 \cdot v^{(6)} + -3.34 \cdot v^{(7)} + 7.12 \cdot v^{(8)}$

$$\mathbf{z} = \begin{bmatrix} -11.22 \\ -16.98 \\ 10.07 \\ -9.75 \\ 5.67 \\ -8.10 \\ -3.34 \\ 7.12 \end{bmatrix}$$

Example 1 (cont)

- ❑How many principle components should we chose?
- ❑ We can look at how much of the variance is explained by k principle components
- ❑ About 75% of the variance is explained by the first 10 principle components
- ❑ About 90% of the variance is explained by the first 20 principle components
- ❑ About 100% of the variance is explained with the first 50 principle components



Example 2 - PCA on faces

- Sklearn has a dataset called Labeled Faces in the Wild. The faces are of famous people collided on the internet. The faces are in jpeg.

62 x 47 pixels = 2914

Learn more at: https://pythonmachinelearning.pro/face-recognition-with-eigenfaces/

Computing the PCA using SVD

PCA Algorithm:

Inputs: The zero centered data matrix X and k >=1

1) Compute the SVD of X: [U,S,V]=svd (X)
2) Let $V_k = [v^{(1)},\ldots,v^{(k)}]$ be the first k columns of V
3) The PCA-feature matrix is $Z = XV_k$

*4) Mapping the reduced image *back to the original dimensions* by $\dot{X}=(XV_k)V_k^T$

In the code we add back the mean to "uncenter"

- To compute the PCA we first need to center the data

```
Xmean = np.mean(X,0)
Xs = X - Xmean
```

This is finding the mean. The 0 states that you should sum over the rows when computing the mean

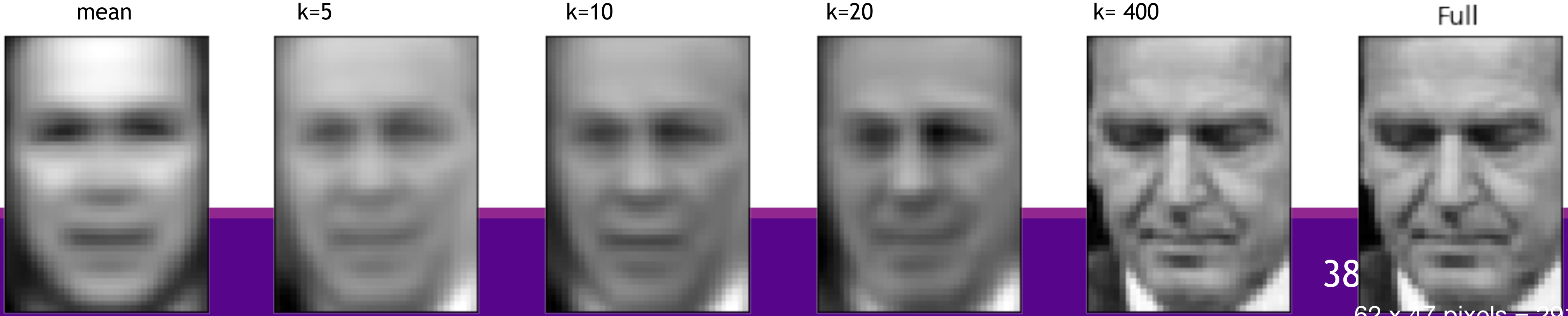
- Next we can compute the SVD using Sklearn's class svd

- Computing the projected images

  Project on to Z space: Xs V_k

  We can map the image back into the original dimensions and then add back the mean: (Xs V_k)V_k^T +Xmean

Looking at the faces after being mapped back

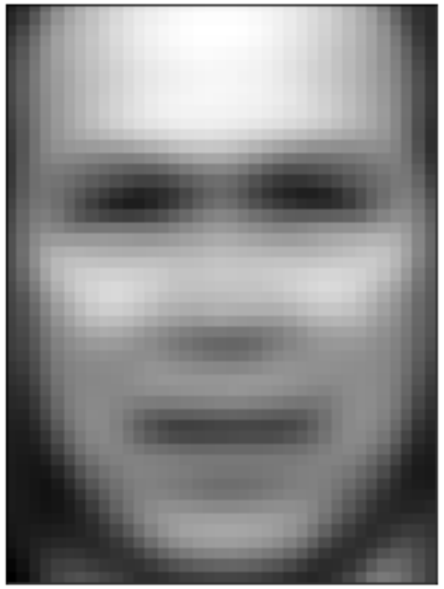


62 x 47 pixels = 2914

The average face!

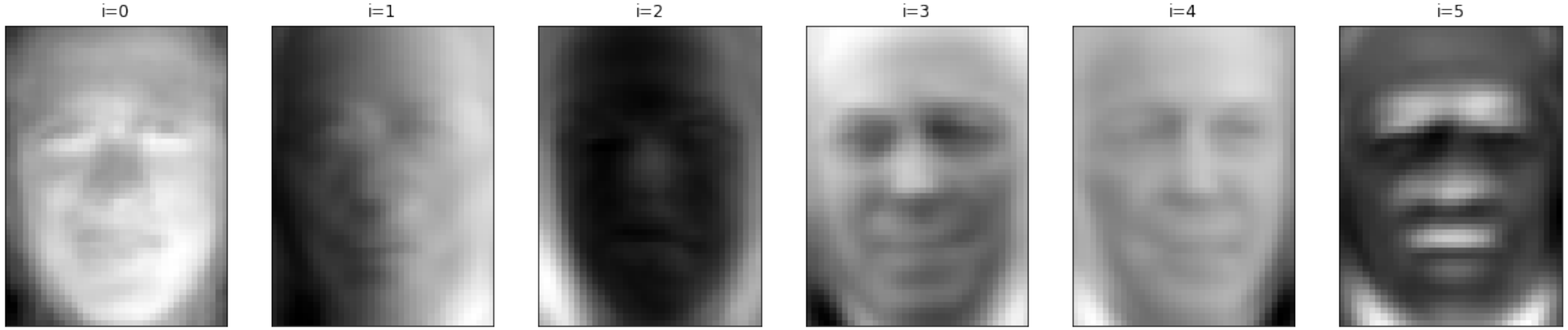
```
Xmean = np.mean(X,0)
```

```
plt_face(Xmean)
```

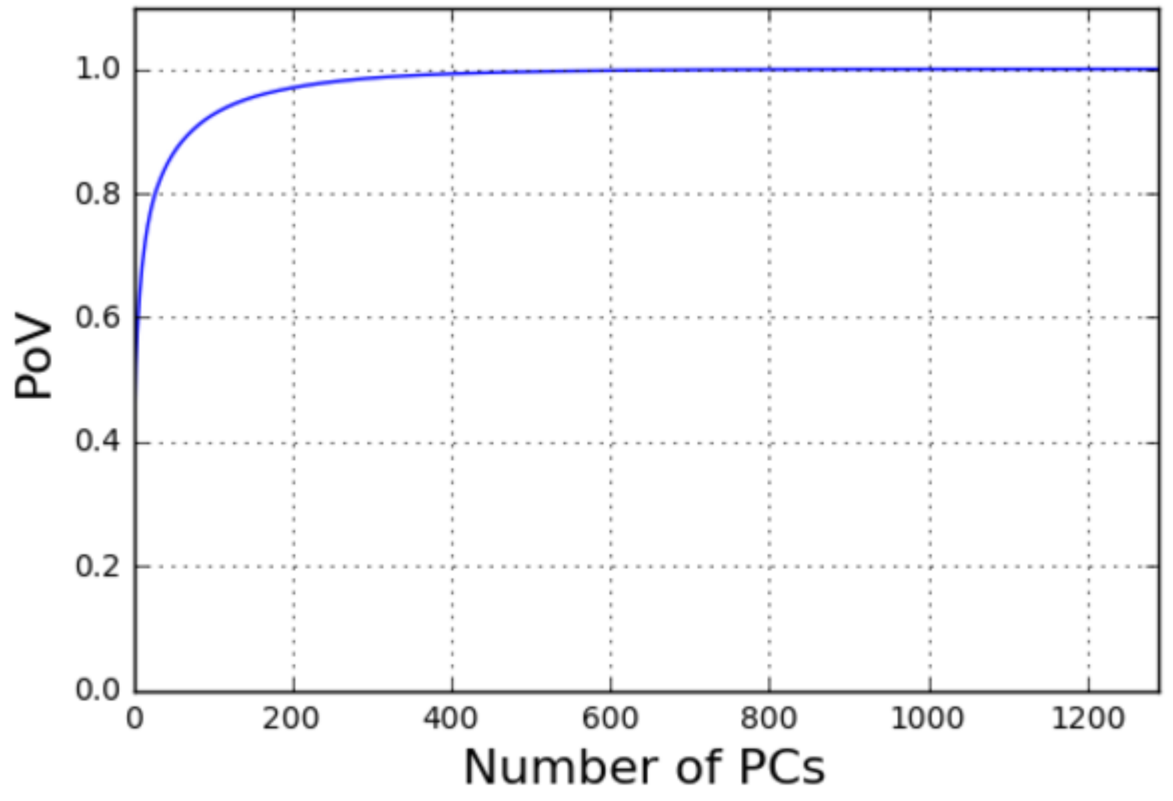


Looking at the the features i.e. eigenvectors)!

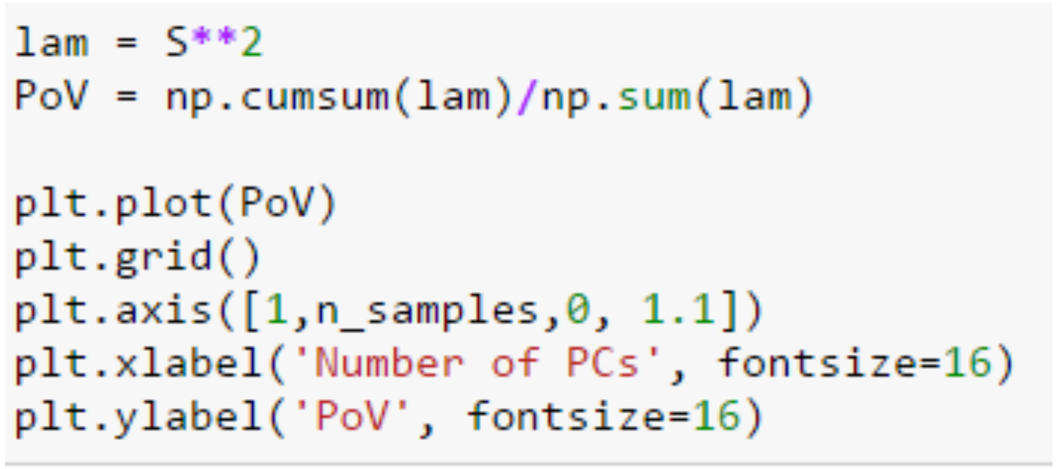
```
nplt = 6
plt.figure(figsize=(10, 20))
for i in range(nplt):
    plt.subplot(1,nplt,i+1)
    Vi = V[i,:]
    plt_face(V[i,:])
    plt.title('i={0:d}'.format(i))
```



Finding the PoV



- Most variance explained in about 400 components
- Some reduction

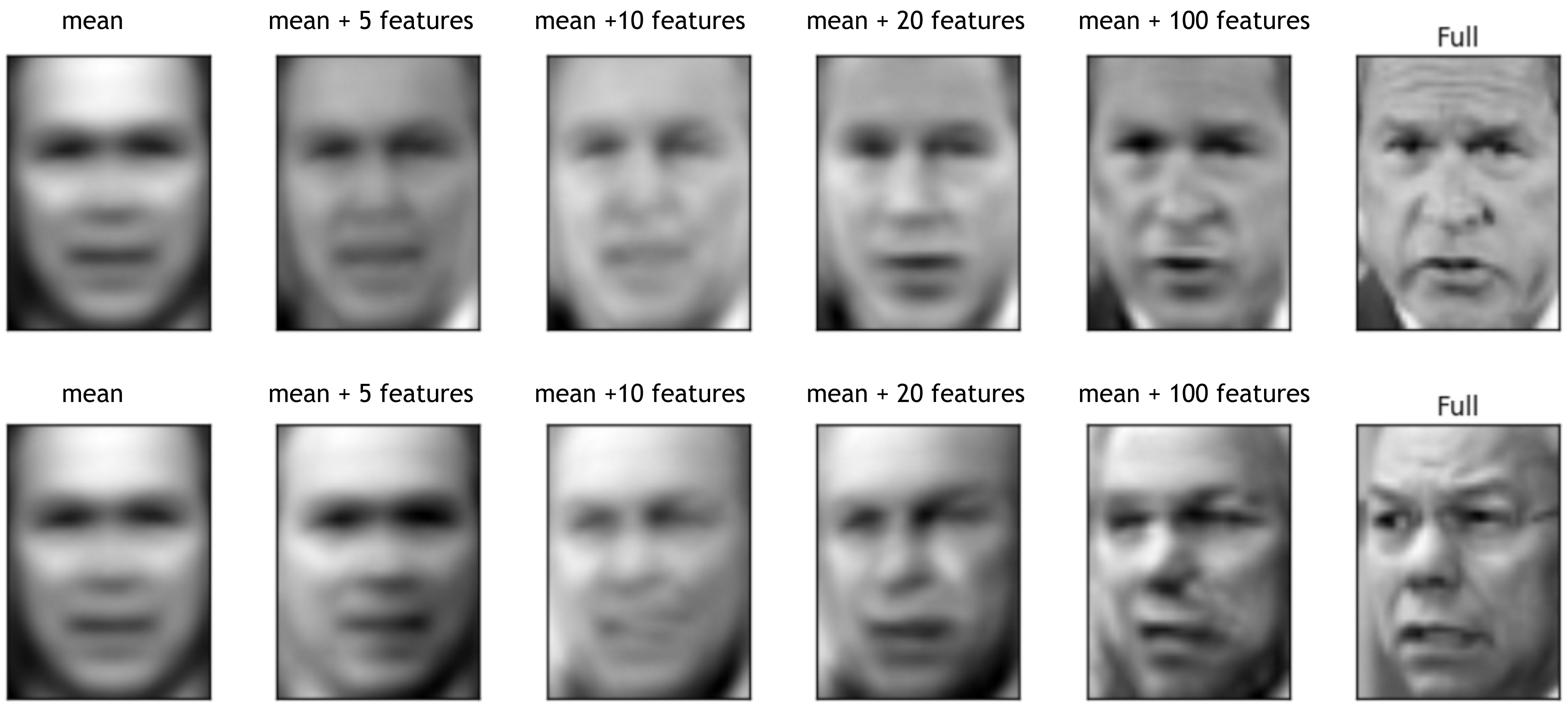


```
lam = S**2
PoV = np.cumsum(lam)/np.sum(lam)

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 the Approximations

Projecting back to the original feature space: $(Xs\ V_k)V_k^T + Xmean$



PCA Algorithm

(This algorithm is modified from the book *Learning from Data* by Yasar Abu-Mostafa et al)

- PCA Algorithm:
- Inputs: The centered data matrix X and k >=1
  1) Compute the SVD of X: [U,S,V]=svd (X)
  2) Let V_k = [$\mathbf{v}^{(1)}$,…,$\mathbf{v}^{(k)}$] be the first k columns of V
  3) The PCA-feature matrix is Z = XV_k

k-features for each example

The values we ignore incur the least reconstruction error

$$\hat{X} = (XV_k)V_k^T = ZV_k^T$$

If we map back to the origin feature space

$$Z = \begin{pmatrix} - & \boldsymbol{x}^{(1)T} & - \\ - & \boldsymbol{x}^{(2)T} & - \\ & \vdots & \\ - & \boldsymbol{x}^{(N)T} & - \end{pmatrix} \begin{pmatrix} | & | & & | \\ \boldsymbol{v}^{(1)} & \boldsymbol{v}^{(2)} & \cdots & \boldsymbol{v}^{(k)} \\ | & | & & | \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}^{(1)T}\boldsymbol{v}^{(1)} & \cdots & \boldsymbol{x}^{(1)T}\boldsymbol{v}^{(k)} \\ \boldsymbol{x}^{(2)T}\boldsymbol{v}^{(1)} & & \boldsymbol{x}^{(2)T}\boldsymbol{v}^{(k)} \\ & \vdots & \\ \boldsymbol{x}^{(N)T}\boldsymbol{v}^{(1)} & \cdots & \boldsymbol{x}^{(N)T}\boldsymbol{v}^{(k)} \end{pmatrix} = \begin{pmatrix} - & \boldsymbol{z}^{(1)T} & - \\ - & \boldsymbol{z}^{(2)T} & \\ & \vdots & \\ - & \boldsymbol{z}^{(N)T} & - \end{pmatrix}$$

More information can be found here: https://stats.stackexchange.com/questions/134282/relationship-between-svd-and-pca-how-to-use-svd-to-perform-pca?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa

A nice tutorial https://davetang.org/file/Singular_Value_Decomposition_Tutorial.pdf

We are assuming the diagonal elements in S go from largest to smallest

PCA Algorithm

(This algorithm is modified from the book *Learning from Data* by Yasar Abu-Mostafa et al)

- PCA Algorithm:
- Inputs: The centered data matrix X and k >=1
  1) Compute the SVD of X: [U,S,V]=svd (X)
  2) Let V_k = [$\mathbf{v}^{(1)}$,…,$\mathbf{v}^{(k)}$] be the first k columns of V
  3) The PCA-feature matrix is $Z = XV_k$

These vectors are the optimal coordinate basis

The values we ignore incur the least reconstruction error

k-features for each example

$$\hat{X} = (XV_k)V_k^T = ZV_k^T$$

If we map back to the origin feature space

$$Z = \begin{pmatrix} - & \mathbf{x}^{(1)T} & - \\ - & \mathbf{x}^{(2)T} & - \\ & \vdots & \\ - & \mathbf{x}^{(N)T} & - \end{pmatrix} \begin{pmatrix} | & | & & | \\ \mathbf{v}^{(1)} & \mathbf{v}^{(2)} & \cdots & \mathbf{v}^{(k)} \\ | & | & & | \end{pmatrix} = \begin{pmatrix} \mathbf{x}^{(1)T}\mathbf{v}^{(1)} & \cdots & \mathbf{x}^{(1)T}\mathbf{v}^{(k)} \\ \mathbf{x}^{(2)T}\mathbf{v}^{(1)} & & \mathbf{x}^{(2)T}\mathbf{v}^{(k)} \\ & \vdots & \\ \mathbf{x}^{(N)T}\mathbf{v}^{(1)} & \cdots & \mathbf{x}^{(N)T}\mathbf{v}^{(k)} \end{pmatrix} = \begin{pmatrix} - & \mathbf{z}^{(1)T} & - \\ - & \mathbf{z}^{(2)T} & \\ & \vdots & \\ - & \mathbf{z}^{(N)T} & - \end{pmatrix}$$

More information can be found here: https://stats.stackexchange.com/questions/134282/relationship-between-svd-and-pca-how-to-use-svd-to-perform-pca?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa

A nice tutorial https://davetang.org/file/Singular_Value_Decomposition_Tutorial.pdf
We are assuming the diagonal elements in S go from largest to smallest