# Do not distribute course material

You may not and may not allow others to reproduce or distribute lecture notes and course materials publicly whether or not a fee is charged.

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

- Some of the material is from Prof. Sundeep Rangan

# 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

# 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)

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

# Dimensionality Reduction

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

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

PCA Goal: *linearly* transforms unlabeled data set into a "better" set of features (aka new coordinate system)

$$\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}\} \xrightarrow{\Phi(\mathbf{x}) = \mathbf{z}} \{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(N)}\}$$

$$d > \tilde{d}$$

# Applications

Examples:

- 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

- Text data
- Financial data
- Transactional data

# Outline

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

- ❑ Reduce the number of features and find latent features
  - Motivation
  - Intuition on keeping the variance of the data
- ❑ Algorithm
- ❑ Examples
- ❑ Finding principle components
- ❑ When PCA doesn't work well

# Latent Features

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 $\bar{\mathbf{x}} + z_1 v^{(1)} + z_2 v^{(2)}$

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

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 |

# 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

- 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

# Feature Elimination

$$\Phi(\mathbf{x}) = [x_1]$$

# Dimensionality Reduction

Inspired by CMU 10:301 slides

# Pair share:
# Which do you prefer? Why?

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

Inspired by CMU 10:301 slides

# Dimensionality reduction (Big idea)

The $\mathbf{v}$ that maximizes this formula is the *first principle component*

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 $i \in 1, \ldots, N$

The **v** that maximizes this formula is the *second principle component*

Noise or information?

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 $(x_1^{(i)}, x_2^{(i)}) = z_1^{(i)}\mathbf{v}^{(1)} + z_2^{(i)}\mathbf{v}^{(2)}$

The **v** that maximizes this formula is the *second principle component*

Noise or information?

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 $(x_1^{(i)}, x_2^{(i)}) = z_1^{(i)}\mathbf{v}^{(1)} + z_2^{(i)}\mathbf{v}^{(2)}$

# Maximizing variance

Principle Component Analysis (PCA) finds a unit vector **v**:

- maximizes the variance of the projected data
- minimizes the mean *squared* distances between the data point and its projection onto the line

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

same as

$$\text{maximizes} \sum_{i=1}^{N} d^{(i)\,2}$$

How can we find **v**?

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

# PCA: Principle Component Analysis

**Big idea:**

1. Zero center the training data $\mathbf{x}^{(i)}$

Suggested to standardize data

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 $z_i = \mathbf{x}^T \mathbf{v}^{(i)}$

**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)**

# Linear Regression

Error: vertical distance between the point and the line

# PCA

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

# 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
- ❑ 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)

- 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 $V_k = [\mathbf{v}^{(1)},\ldots,\mathbf{v}^{(k)}]$ 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

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

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

# 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
- ❑ 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.]
```

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.]
```

→

```
 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 three

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

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

# 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?

# 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)

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

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

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

- 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

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

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

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

# 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

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

- 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

# Computing the PCA using SVD

- 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

**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"

- 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

# 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:
- Inputs: The centered data matrix X and k >=1
  1) Compute the SVD of X: [U,S,V]=svd (X)
  2) Let $V_k = [\mathbf{v}^{(1)},\ldots,\mathbf{v}^{(k)}]$ 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

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

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

# =

- PCA Algorithm:
- Inputs: The centered data matrix X and k >=1
  1) Compute the SVD of X: [U,S,V]=svd (X)
  2) Let $V_k = [\mathbf{v}^{(1)},\ldots,\mathbf{v}^{(k)}]$ 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

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

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

# PCA: Principle Component Analysis

**Linearly** projects examples into a **lower** dimensional **subspace**: $N \times d$ into $N \times k$ where $k < d$

PCA maintains as much of the original **variance** (and minimizes least square reconstruction error)

# When PCA Doesn't Work Well

- PCA doesn't find non-linear features
- One option is to transform the features before applying PCA
- To learn about kernels with PCA you can look at: https://en.m.wikipedia.org/wiki/Kernel_principal_component_analysis

# Summary

- PCA is a linear projection of our original feature space into a k dimensional vector space
- The new feature space is in a new coordinate system
- We choose the new coordinates such that:
  - maximizes variance
  - minimizes projection error (square loss)
- PCA is used for:
  - dimensionality reduction
  - visualization (if we choose k to be 2 or 3)
  - Compression (with loss)
  - De-noising (removes small variance in data)

# Additional notes

- The *data must be centered* before running PCA
- Optional: normalize none, some, or all the features
  - Normalizing will prevent arbitrary scales of different features affect the decisions made by the PCA algorithm
  - Not normalizing when features have similar scales with different variances will keep differences of features that are usually meaningful
- Choosing $k$ (i.e. size of subspace $\mathbb{R}^k$ for new features)
  - If performing PCA as a preprocessing step before running a learning algorithm, choose $k$ based on validation data performance
  - Optionally: choose $k$ based on how much of the original variance of X is maintained after projecting to a smaller dimension is $\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$. Note: X is noisy so this formula includes the noise.
  - Learn more at [https://ro-che.info/articles/2017-12-11-pca-explained-variance](https://ro-che.info/articles/2017-12-11-pca-explained-variance)

# Additional notes

- The *data must be centered* before running PCA
- Optional: normalize none, some, or all the features
  - Normalizing will prevent arbitrary scales of different features affect the decisions made by the PCA algorithm
  - Not normalizing when features have similar scales with different variances will keep differences of features that are usually meaningful
- Choosing $k$ (i.e. size of subspace $\mathbb{R}^k$ for new features)
  - If performing PCA as a preprocessing step before running a learning algorithm, choose $k$ based on validation data performance
  - Optionally: choose $k$ based on how much of the original variance of X is maintained after proj ion is $\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$. Note: X is noisy so this formula includes the noise.

The $\lambda_1, \lambda_2, \cdots, \lambda_d$ are the diagonal values in D = $S^2$

  - Learn more at https://ro-che.info/articles/2017-12-11-pca-explained-variance

The following slides were not presented in class

# Outline

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

- Standard **unsupervised** preprocessing: zero centering, data normalization
- Reduce the number of features and find latent features
  - Motivation
  - Intuition on keeping the variance of the data
- Algorithm
- Examples
- Finding principle components
- When PCA doesn't work well

We will focus on the case where $k=1$.

The more general case extends these ideas.

# Outline for finding principle components

Objective function

- How do we find arg $\max_v \mathbf{v}^T A \mathbf{v}$, where $A = X^T X$ and $\mathbf{v}$ is a unit vector
  - Thought experiment: if D is a diagonal matrix then $\mathbf{v} = \arg\max_v \mathbf{v}^T D \mathbf{v}$ is solved by setting $\mathbf{v} = \mathbf{e}_1$
  - Fact from linear algebra: any $A = X^T X$ can be written as $A = V D V^T$
  - Proving $V e_1 = \arg\max_v \mathbf{v}^T A \mathbf{v}$, where $A = X^T X = V D V^T$ and v is a unit vector
- Using a standard library to find V
  - Fact from linear algebra: any X can be written as $X = U D V^T$ (singular value decomposition, SVD)
  - There are many libraries to compute the SVD
  - Observe: $X^T X = (U S V^T)^T (U S V^T) = V D V^T$

# Outline for finding principle components

- What is the "*best*" lower dimensional space?
  - Maximizing variance of the the points projected onto a line **v** (or minimizing mean squared distance between points and their projection onto the line)
  - Goal: find arg $\max_v$ $\mathbf{v}^T A \mathbf{v}$, where $A = X^T X$ and **v** is a unit vector
- How do we find arg $\max_v$ $\mathbf{v}^T A \mathbf{v}$, where $A = X^T X$ and **v** is a unit vector
  - Thought experiment: if D is a diagonal matrix then $\mathbf{v}$ = arg $\max_v$ $\mathbf{v}^T D \mathbf{v}$ is solved by setting $\mathbf{v} = \mathbf{e}_1$
  - Fact from linear algebra: any $A = X^T X$ can be written as $A = V D V^T$
  - Proving $V e_1$ = arg $\max_v$ $\mathbf{v}^T A \mathbf{v}$, where $A = X^T X = V D V^T$ and v is a unit vector
- Using a standard library to find V
  - Fact from linear algebra: any X can be written as $X = U D V^T$ (singular value decomposition, SVD)
  - There are many libraries to compute the SVD
  - Observe: $X^T X = (U S V^T)^T (U S V^T) = V D V^T$

# Objective function

# Assumptions & Projection formula

The data is zero-centered(i.e., *the mean is 0*), $(\mathbf{x}^{(1)}+\mathbf{x}^{(2)}+ \ldots + \mathbf{x}^{(N)})/N = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$

- The goal is to maximize the variance of the projection of **the training data** onto the vector **v**
  - Let **v** be a unit vector $\|v\|_2 = 1$ Uniqueness
  - The distance of the projected point to the origin is $|\mathbf{x}^T\mathbf{v}|$

# Objective: maximizing the Variance of the data after projection

Given the zero-centered data (i.e., *the mean is 0*), $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}$ :

- The formula for the variance of a set of N points projected onto the line **v** (the **mean is 0** even after being projected onto **v**)

$$\text{var} = \frac{1}{N}\sum_{i=1}^{N}\Big(\overbrace{\boldsymbol{x}^{(i)T}\boldsymbol{v}}^{\mathbf{z}^{(i)}} - 0\Big)^2 = \frac{1}{N}\sum_{i=1}^{N}\Big(\boldsymbol{x}^{(i)T}\boldsymbol{v}\Big)^2$$

$$\mathbf{x}^{(i)\,T}\mathbf{v} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)})\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_d \end{bmatrix} = \mathbf{v}^T\mathbf{x}^{(i)}$$

maximizing the variance of the projection of **x**

$$\frac{1}{N}\sum_{i=1}^{N}\Big(\boldsymbol{x}^{(i)T}\boldsymbol{v}\Big)\Big(\boldsymbol{x}^{(i)T}\boldsymbol{v}\Big) = \frac{1}{N}\sum_{i=1}^{N}\Big(\boldsymbol{v}^T\boldsymbol{x}^{(i)}\Big)\Big(\boldsymbol{x}^{(i)T}\boldsymbol{v}\Big) = \boldsymbol{v}^T\left(\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}^{(i)}\boldsymbol{x}^{(i)T}\right)\boldsymbol{v}$$

- $\text{argmax}_{\mathbf{v}:||\mathbf{v}||=1}\frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}^{(i)T}\mathbf{v})^2$ is the same as $\text{argmax}_{\mathbf{v}:||\mathbf{v}||=1}\mathbf{v}^T\underbrace{X^TX}\mathbf{v}$

$\frac{1}{N}X^TX$ is the sample covariance matrix

# Objective: maximizing the Variance of the data after projection

Given the zero-centered data (i.e., *the mean is 0*), $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}$ :

- The formula for the variance of a set of N points projected onto the line **v** (the **mean is 0** even after being projected onto **v**)

The **v** that maximizes this formula is the *first principle component*

$$\frac{1}{N}\sum_{i=1}^{N}\Big(\overbrace{\boldsymbol{x}^{(i)T}\boldsymbol{v}}^{\mathbf{z}^{(i)}}-0\Big)^2 = \frac{1}{N}\sum_{i=1}^{N}\Big(\boldsymbol{x}^{(i)T}\boldsymbol{v}\Big)^2$$

$$\mathbf{x}^{(i)\,T}\mathbf{v} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)})\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_d \end{bmatrix} = \mathbf{v}^T\mathbf{x}^{(i)}$$

maximizing the variance of the projection of **x**

$$\frac{1}{N}\sum_{i=1}^{N}\Big(\boldsymbol{x}^{(i)T}\boldsymbol{v}\Big)\Big(\boldsymbol{x}^{(i)T}\boldsymbol{v}\Big) = \frac{1}{N}\sum_{i=1}^{N}\Big(\boldsymbol{v}^T\boldsymbol{x}^{(i)}\Big)\Big(\boldsymbol{x}^{(i)T}\boldsymbol{v}\Big) = \boldsymbol{v}^T\left(\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}^{(i)}\boldsymbol{x}^{(i)T}\right)\boldsymbol{v}$$

- $\text{argmax}_{\mathbf{v}:||\mathbf{v}||=1}\frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}^{(i)T}\mathbf{v})^2$ is the same as $\text{argmax}_{\mathbf{v}:||\mathbf{v}||=1}\mathbf{v}^T\underbrace{X^TX}\mathbf{v}$

$\frac{1}{N}X^TX$ is the sample covariance matrix

# Objective: maximizing the Variance of the data after projection

Given the zero-centered data (i.e., *the mean is 0*), $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}$ :

- The formula for the variance of a set of N points projected onto the line **v** (the **mean is 0** even after being projected onto **v**)

The **v** that maximizes this formula is the *first*

$$= \frac{1}{N}\sum^{N}\Big(\overbrace{\mathbf{x}^{(i)T}\mathbf{v}}^{\mathbf{z}^{(i)}} - 0\Big)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{x}^{(i)T}\boldsymbol{v}\right)^2$$

$$\mathbf{x}^{(i)\,T}\mathbf{v} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)})\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_d \end{bmatrix} = \mathbf{v}^T\mathbf{x}^{(i)}$$

Same a s $X^TX$, e.g. if d=2

$$\begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(N)} \\ x_2^{(1)} & x_2^{(2)} & \cdots & x_2^{(N)} \end{bmatrix}\begin{bmatrix} x_1^{(1)} & x_2^{(1)} \\ x_1^{(2)} & x_2^{(2)} \\ \cdots & \\ x_1^{(N)} & x_2^{(N)} \end{bmatrix}$$

**Outer** product. If d=2

$$\begin{bmatrix} \sum_{i=1}^N x_1^{(i)}x_1^{(i)} & \sum_{i=1}^N x_1^{(i)}x_2^{(i)} \\ \sum_{i=1}^N x_2^{(i)}x_1^{(i)} & \sum_{i=1}^N x_2^{(i)}x_2^{(i)} \end{bmatrix}$$

$$\boldsymbol{v}) = \left(\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}^{(i)}\boldsymbol{x}^{(i)T}\right)\boldsymbol{v}$$

- $\text{argmax}_{\mathbf{v}:||\mathbf{v}||=1}\frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}^{(i)T}\mathbf{v})^2$ is the same as $\text{argmax}_{\mathbf{v}:||\mathbf{v}||=1}\mathbf{v}^T\underbrace{X^TX}\mathbf{v}$

$\frac{1}{N}X^TX$ is the sample covariance matrix

# Toy Example

from https://web.stanford.edu/class/cs246/slides/06-dim_red.pdf

| | The Martian | Interstellar | Inception | Titanic | The Notebook | |
|---|---|---|---|---|---|---|
| X= | [1, | 1, | 1, | 0, | 0], | Billy |
| | [3, | 3, | 3, | 0, | 0], | Ellie |
| | [4, | 4, | 4, | 0, | 0], | Sam |
| | [5, | 5, | 5, | 0, | 0], | Pat |
| | [0, | 2, | 0, | 4, | 4], | Tully |
| | [0, | 0, | 0, | 5, | 5], | Liz |
| | [0, | 1, | 0, | 2, | 2] | Mo |

$$X_{centered} = \begin{bmatrix} -0.86 & -1.29 & -0.86 & -1.57 & -1.57 \\ 1.14 & 0.71 & 1.14 & -1.57 & -1.57 \\ 2.14 & 1.71 & 2.14 & -1.57 & -1.57 \\ 3.14 & 2.71 & 3.14 & -1.57 & -1.57 \\ -1.86 & -0.29 & -1.86 & 2.43 & 2.43 \\ -1.86 & -2.29 & -1.86 & 3.43 & 3.43 \\ -1.86 & -1.29 & -1.86 & 0.43 & 0.43 \end{bmatrix}$$

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

$$\text{argmax}_{\mathbf{v}:||\mathbf{v}||=1} \mathbf{v}^T \begin{bmatrix} 26.86 & 21.29 & 26.86 & -20.43 & -20.43 \\ 21.29 & 19.43 & 21.29 & -15.14 & -15.14 \\ 26.86 & 21.29 & 26.86 & -20.43 & -20.43 \\ -20.43 & -15.14 & -20.43 & 27.71 & 27.71 \\ -20.43 & -15.14 & -20.43 & 27.71 & 27.71 \end{bmatrix} \mathbf{v}$$

# Outline for finding principle components

- What is the "*best*" lower dimensional space?
  - Maximizing variance of the the points projected onto a line v (or minimizing mean squared distance between points and their projection onto the line)
  - Goal: find arg $\max_v v^T A v$, where $A = X^T X$ and v is a unit vector

Optimizer

- Using a standard library to find V
  - Fact from linear algebra: any X can be written as $X = U D V^T$ (singular value decomposition, SVD)
  - There are many libraries to compute the SVD
  - Observe: $X^T X = (U S V^T)^T (U S V^T) = V D V^T$

# Outline for finding principle components

- What is the "*best*" lower dimensional space?
  - Maximizing variance of the the points projected onto a line v (or minimizing mean squared distance between points and their projection onto the line)
  - Goal: find arg $\max_v v^T A v$, where $A = X^T X$ and v is a unit vector
- How do we find arg $\max_v v^T A v$, where $A = X^T X$ and v is a unit vector
  - Thought experiment: if D is a diagonal matrix then $v = \arg\max_v v^T D v$ is solved by setting $v = e_1$
  - Fact from linear algebra: any $A = X^T X$ can be written as $A = V D V^T$
  - Proving $V e_1 = \arg\max_v v^T A v$, where $A = X^T X = V D V^T$ and v is a unit vector
- Using a standard library to find V
  - Fact from linear algebra: any X can be written as $X = U D V^T$ (singular value decomposition, SVD)
  - There are many libraries to compute the SVD
  - Observe: $X^T X = (U S V^T)^T (U S V^T) = V D V^T$

# Simplification

SUPPOSE A IS A DIAGONAL MATRIX

$$A = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_d \end{pmatrix}$$

# Stretching a vector

Largest

$$A = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_d \end{pmatrix}$$

- If A is a diagonal matrix with nonnegative entries $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_d \geq 0$

  then A maps $\mathbf{v}^T = [v_1, v_2, \ldots, v_d]$ to $A\mathbf{v}$ where the i$^{th}$ coordinate, $v_i$, is stretched by a factor of $\lambda_i$

- For example if A = $\begin{pmatrix} 3 & 0 \\ 0 & 0.5 \end{pmatrix}$ then A**v** will map $v_1$ to $3v_1$ and $v_2$ to $(0.5)v_2$

➜

- Which unit vector v will be "stretched" the most for the diagonal matrix A?

**v** = **e**$^{(1)}$ = $(1,0)^T$

General case: **v** = **e**$^{(1)}$ = $(1,0, \ldots, 0)^T$

Using the approach from http://theory.stanford.edu/~tim/s17/l/l8.pdf

# First Principle Component

Largest

$$A = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_d \end{pmatrix}$$

- Which unit vector $\mathbf{v}$ maximizes $\mathbf{v}^T(A\mathbf{v})$

$$\mathbf{v}^T(A\mathbf{v}) = (v_1, v_2, \ldots, v_d)\begin{pmatrix} \lambda_1 v_1 \\ \lambda_2 v_2 \\ \vdots \\ \lambda_d v_d \end{pmatrix} = \sum_{i=1}^{d} v_i^2 \lambda_i = \lambda_1 \cdot v_1^2 + \lambda_2 \cdot v_2^2 + \cdots + \lambda_d v_d^2$$

Weighted average of $\lambda_i$

Pair share: where should we put the weight in $\mathbf{v}$ to maximize $\lambda_1 \cdot v_1^2 + \lambda_2 \cdot v_2^2 + \cdots + \lambda_d v_d^2$

Using the approach from http://theory.stanford.edu/~tim/s17/l/l8.pdf

# First Principle Component

Largest

$$A = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_d \end{pmatrix}$$

- Which unit vector $\mathbf{v}$ maximizes $\mathbf{v}^T(A\mathbf{v})$

$$\mathbf{v}^T(A\mathbf{v}) = (v_1, v_2, \ldots, v_d)\begin{pmatrix} \lambda_1 v_1 \\ \lambda_2 v_2 \\ \vdots \\ \lambda_d v_d \end{pmatrix} = \sum_{i=1}^{d} v_i^2 \lambda_i = \lambda_1 \cdot v_1^2 + \lambda_2 \cdot v_2^2 + \cdots + \lambda_d v_d^2$$

Weighted average of $\lambda_i$

Maximized by putting all the weight on $\lambda_1$ by choosing $\mathbf{v} = \mathbf{e}^{(1)} = (1,0, \ldots, 0)^T$

Using the approach from http://theory.stanford.edu/~tim/s17/l/l8.pdf

# Toy Example

$X^T_{centered} X_{centered} = A = \begin{bmatrix} 110.09 & 0 & 0 & 0 & 0 \\ 0 & 16.73 & 0 & 0 & 0 \\ 0 & 0 & 1.75 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

$$\text{argmax}_{\mathbf{v}:||\mathbf{v}||=1} \mathbf{v}^T \begin{bmatrix} 110.09 & 0 & 0 & 0 & 0 \\ 0 & 16.73 & 0 & 0 & 0 \\ 0 & 0 & 1.75 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{v} = 110.09 \cdot v_1^2 + 16.73 \cdot v_2^2 + 1.75 \cdot v_3^2 + 0 \cdot v_4^2 + 0 \cdot v_5^2$$

$$\leq 110.09 \cdot (v_1^2 + v_2^2 + v_3^2 + v_4^2 + v_5^2)$$

$$v_1^2 + v_2^2 + v_3^2 + v_4^2 + v_5^2 = 1$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 110.09 & 0 & 0 & 0 & 0 \\ 0 & 16.73 & 0 & 0 & 0 \\ 0 & 0 & 1.75 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = 110.09$$

# General Case

$A = X^T X$

# Orthonormal Matrix Properties

All a matrix does is rotate/permute and stretch vectors.

If the columns of V are orthonormal then multiplying a vector **v** by V doesn't change the length of **v**, i.e.

$$\|V\mathbf{v}\|_2 = \|\mathbf{v}\|_2$$

PAIR SHARE: WHAT DOES THIS MEAN? WE COULD ALSO SAY V IS ORTHOGONAL.

Proof: $\|V\mathbf{v}\|_2^2 = (V\mathbf{v})^T V\mathbf{v} = \mathbf{v}^T V^T V\mathbf{v} = \mathbf{v}^T\mathbf{v} = \|\mathbf{v}\|_2^2$

$\|V\mathbf{v}\|_2, \|\mathbf{v}\|_2$ HAVE THE SAME LENGTH (NORM)

Note that: $V^T V = I$

Ideas taken from http://theory.stanford.edu/~tim/s17/l/l8.pdf

# First Principle Component

$\lambda_1 \geq \lambda_2 \geq \cdots \lambda_d \geq 0$

$$D = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_d \end{pmatrix}$$

Matrix A can be written: $A = VDV^T$ where D is a diagonal matrix and V is orthogonal.

What is the direction of "maximum stretch" for A?

Answer: find the vector **v** that is mapped to $\mathbf{e}^{(1)}$ under $V^T$.

This is the vector $\mathbf{v} = V\mathbf{e}^{(1)}$. FIRST COLUMN OF V

$\mathbf{e}^{(1)} = (1,0,0,\ldots,0)^T$

$\|\mathbf{v}\| = \|V\mathbf{e}^{(1)}\| = 1$

We can see this by: $\mathbf{v}^T A\mathbf{v} = \mathbf{e}^{(1)T}V^T AV\mathbf{e}^{(1)} = \mathbf{e}^{(1)T}V^T VDV^T V\mathbf{e}^{(1)} = \mathbf{e}^{(1)T}D\mathbf{e}^{(1)} = \lambda_1$

Ideas taken from http://theory.stanford.edu/~tim/s17/l/l8.pdf

# Finding k Principle Components

$$A = X^T X = VDV^T$$

The ***first principle component*** **v** is the first row of $V^T$ (i.e.the first column of V, thus we are looking for $\mathbf{v}^{(1)} = V\mathbf{e}^{(1)}$)

The ***second principle component*** **v** is the second row of $V^T$ (i.e.the first column of V, thus we are looking for $\mathbf{v}^{(2)} = V\mathbf{e}^{(2)}$) and so on.

Or

Repeat the original procedure where the data set has been updated $\mathbf{x}^{(i)} := \mathbf{x}^{(i)} - (\mathbf{x}^{(i)T}\mathbf{v})\mathbf{v}$

Ideas taken from http://theory.stanford.edu/~tim/s17/l/l8.pdf

# Observation

## Computing the $i^{th}$ principal component

After finding the first principle component we can repeat the procedure on a new data matrix X

$$X = \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \\ \vdots \\ \mathbf{x}^{(N)} \end{bmatrix} \rightarrow X = \begin{bmatrix} \mathbf{x}^{(1)} - (\mathbf{x}^{(1)}\mathbf{v}^{(1)})\mathbf{v}^{(1)} \\ \mathbf{x}^{(2)} - (\mathbf{x}^{(2)}\mathbf{v}^{(1)})\mathbf{v}^{(1)} \\ \vdots \\ \mathbf{x}^{(N)} - (\mathbf{x}^{(N)}\mathbf{v}^{(1)})\mathbf{v}^{(1)} \end{bmatrix}$$

For each example we remove the part of the data that is explained by $\mathbf{v}_1$

By repeating this procedure we can find the first k principle components (i.e. first k eigenvectors)

# Toy Example: set **v** to be the first column of V and then compute $\mathbf{v}^T A \mathbf{v}$

| | The Martian | Interstellar | Inception | Titanic | The Notebook | |
|---|---|---|---|---|---|---|
| | [1, | 1, | 1, | 0, | 0], | Billy |
| | [3, | 3, | 3, | 0, | 0], | Ellie |
| X= | [4, | 4, | 4, | 0, | 0], | Sam |
| | [5, | 5, | 5, | 0, | 0], | Pat |
| | [0, | 2, | 0, | 4, | 4], | Tully |
| | [0, | 0, | 0, | 5, | 5], | Liz |
| | [0, | 1, | 0, | 2, | 2] | Mo |

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

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

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

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 110.09 & 0 & 0 & 0 & 0 \\ 0 & 16.73 & 0 & 0 & 0 \\ 0 & 0 & 1.75 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = 110.09$$

Original dataset from https://web.stanford.edu/class/cs246/slides/06-dim_red.pdf

# Toy Example

from https://web.stanford.edu/class/cs246/slides/06-dim_red.pdf

| | The Martian | Interstellar | Inception | Titanic | The Notebook | |
|---|---|---|---|---|---|---|
| X= | [1, | 1, | 1, | 0, | 0], | Billy |
| | [3, | 3, | 3, | 0, | 0], | Ellie |
| | [4, | 4, | 4, | 0, | 0], | Sam |
| | [5, | 5, | 5, | 0, | 0], | Pat |
| | [0, | 2, | 0, | 4, | 4], | Tully |
| | [0, | 0, | 0, | 5, | 5], | Liz |
| | [0, | 1, | 0, | 2, | 2] | Mo |

$$X_{centered} = \begin{bmatrix} -0.86 & -1.29 & -0.86 & -1.57 & -1.57 \\ 1.14 & 0.71 & 1.14 & -1.57 & -1.57 \\ 2.14 & 1.71 & 2.14 & -1.57 & -1.57 \\ 3.14 & 2.71 & 3.14 & -1.57 & -1.57 \\ -1.86 & -0.29 & -1.86 & 2.43 & 2.43 \\ -1.86 & -2.29 & -1.86 & 3.43 & 3.43 \\ -1.86 & -1.29 & -1.86 & 0.43 & 0.43 \end{bmatrix}$$

$$X_{centered}\ V_2 = \begin{bmatrix} -0.86 & -1.29 & -0.86 & -1.57 & -1.57 \\ 1.14 & 0.71 & 1.14 & -1.57 & -1.57 \\ 2.14 & 1.71 & 2.14 & -1.57 & -1.57 \\ 3.14 & 2.71 & 3.14 & -1.57 & -1.57 \\ -1.86 & -0.29 & -1.86 & 2.43 & 2.43 \\ -1.86 & -2.29 & -1.86 & 3.43 & 3.43 \\ -1.86 & -1.29 & -1.86 & 0.43 & 0.43 \end{bmatrix} \begin{bmatrix} -0.47 & -0.37 \\ -0.36 & -0.41 \\ -0.39 & 0.83 \\ -0.71 & -0. \\ 0. & -0. \end{bmatrix} = \begin{bmatrix} 1.317 & 1.124 \\ 3.95 & 3.372 \\ 5.267 & 4.496 \\ 6.583 & 5.62 \\ -2.9 & 5.121 \\ -4.559 & 5.37 \\ -1.45 & 2.56 \end{bmatrix}$$

Data matrix in Z-space

# Toy Example

from https://web.stanford.edu/class/cs246/slides/06-dim_red.pdf

| | The Martian | Interstellar | Inception | Titanic | The Notebook | |
|---|---|---|---|---|---|---|
| X= | [1, | 1, | 1, | 0, | 0], | Billy |
| | [3, | 3, | 3, | 0, | 0], | Ellie |
| | [4, | 4, | 4, | 0, | 0], | Sam |
| | [5, | 5, | 5, | 0, | 0], | Pat |
| | [0, | 2, | 0, | 4, | 4], | Tully |
| | [0, | 0, | 0, | 5, | 5], | Liz |
| | [0, | 1, | 0, | 2, | 2] | Mo |

$$X_{centered}\ V_2 = \begin{bmatrix} -0.86 & -1.29 & -0.86 & -1.57 & -1.57 \\ 1.14 & 0.71 & 1.14 & -1.57 & -1.57 \\ 2.14 & 1.71 & 2.14 & -1.57 & -1.57 \\ 3.14 & 2.71 & 3.14 & -1.57 & -1.57 \\ -1.86 & -0.29 & -1.86 & 2.43 & 2.43 \\ -1.86 & -2.29 & -1.86 & 3.43 & 3.43 \\ -1.86 & -1.29 & -1.86 & 0.43 & 0.43 \end{bmatrix} \begin{bmatrix} -0.47 & -0.37 \\ -0.36 & -0.41 \\ -0.39 & 0.83 \\ -0.71 & -0. \\ 0. & -0. \end{bmatrix} = \begin{bmatrix} 1.317 & 1.124 \\ 3.95 & 3.372 \\ 5.267 & 4.496 \\ 6.583 & 5.62 \\ -2.9 & 5.121 \\ -4.559 & 5.37 \\ -1.45 & 2.56 \end{bmatrix}$$

Data matrix in Z-space

Since $D = \begin{bmatrix} 110.09 & 0 & 0 & 0 & 0 \\ 0 & 16.73 & 0 & 0 & 0 \\ 0 & 0 & 1.75 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

we have maintained $\dfrac{110.09 + 16.73}{110.09 + 16.73 + 1.75 + 0} = \dfrac{126.83}{128.57}$ of the original variance

# Toy Example

from https://web.stanford.edu/class/cs246/slides/06-dim_red.pdf

| | The Martian | Interstellar | Inception | Titanic | The Notebook | |
|---|---|---|---|---|---|---|
| X= | [1, | 1, | 1, | 0, | 0], | Billy |
| | [3, | 3, | 3, | 0, | 0], | Ellie |
| | [4, | 4, | 4, | 0, | 0], | Sam |
| | [5, | 5, | 5, | 0, | 0], | Pat |
| | [0, | 2, | 0, | 4, | 4], | Tully |
| | [0, | 0, | 0, | 5, | 5], | Liz |
| | [0, | 1, | 0, | 2, | 2] | Mo |

$$X_{centered}\ V_2 = \begin{bmatrix} -0.86 & -1.29 & -0.86 & -1.57 & -1.57 \\ 1.14 & 0.71 & 1.14 & -1.57 & -1.57 \\ 2.14 & 1.71 & 2.14 & -1.57 & -1.57 \\ 3.14 & 2.71 & 3.14 & -1.57 & -1.57 \\ -1.86 & -0.29 & -1.86 & 2.43 & 2.43 \\ -1.86 & -2.29 & -1.86 & 3.43 & 3.43 \\ -1.86 & -1.29 & -1.86 & 0.43 & 0.43 \end{bmatrix} \begin{bmatrix} -0.47 & -0.37 \\ -0.36 & -0.41 \\ -0.39 & 0.83 \\ -0.71 & -0. \\ 0. & -0. \end{bmatrix} = \begin{bmatrix} 1.317 & 1.124 \\ 3.95 & 3.372 \\ 5.267 & 4.496 \\ 6.583 & 5.62 \\ -2.9 & 5.121 \\ -4.559 & 5.37 \\ -1.45 & 2.56 \end{bmatrix}$$

Data matrix in Z-space

Variance of data explained by $\mathbf{v}^{(1)}$

$$\text{Since } D = \begin{bmatrix} 110.09 & 0 & & & \\ 0 & 16.73 & 0 & & \\ 0 & 0 & 1.75 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

we have maintained $\dfrac{110.09 + 16.73}{110.09 + 16.73 + 1.75 + 0} = \dfrac{126.83}{128.57}$ of the original variance

# Outline for finding principle components

- What is the "*best*" lower dimensional space?
  - Maximizing variance of the the points projected onto a line **v** (or minimizing mean squared distance between points and their projection onto the line)
  - Goal: find arg $\max_{\mathbf{v}}$ $\mathbf{v}^T A \mathbf{v}$, where $A = X^T X$ and v is a unit vector
- How do we find arg $\max_{\mathbf{v}}$ $\mathbf{v}^T A \mathbf{v}$, where $A = X^T X$ and **v** is a unit vector
  - Thought experiment: if D is a diagonal matrix then $\mathbf{v} = \arg\max_{\mathbf{v}} \mathbf{v}^T D \mathbf{v}$ is solved by setting $\mathbf{v} = \mathbf{e}_1$
  - Fact from linear algebra: any $A = X^T X$ can be written as $A = V D V^T$
  - Proving $V\mathbf{e}_1 = \arg\max_{\mathbf{v}} \mathbf{v}^T A \mathbf{v}$, where $A = X^T X = V D V^T$ and v is a unit vector

Better Optimizer

# Outline for finding principle components

- What is the "*best*" lower dimensional space?
  - Maximizing variance of the the points projected onto a line **v** (or minimizing mean squared distance between points and their projection onto the line)
  - Goal: find arg $\max_v$ $\mathbf{v}^T A \mathbf{v}$, where $A = X^T X$ and v is a unit vector
- How do we find arg $\max_v$ $\mathbf{v}^T A \mathbf{v}$, where $A = X^T X$ and **v** is a unit vector
  - Thought experiment: if D is a diagonal matrix then $\mathbf{v}$ = arg $\max_v$ $\mathbf{v}^T D \mathbf{v}$ is solved by setting $\mathbf{v} = \mathbf{e}_1$
  - Fact from linear algebra: any $A = X^T X$ can be written as $A = V D V^T$
  - Proving $V\mathbf{e}_1$ = arg $\max_v$ $\mathbf{v}^T A \mathbf{v}$, where $A = X^T X = V D V^T$ and v is a unit vector
- Using a standard library to find V
  - Fact from linear algebra: any X can be written as $X = U S V^T$ (singular value decomposition, SVD)
  - There are many libraries to compute the SVD
  - Observe: $X^T X = (U S V^T)^T (U S V^T) = V D V^T$

# Computing the PCA using SVD

X is zero centered

Goal: Factorize $A = X^T X = VDV^T$ to find $V_k$ (eigenvectors) and $\lambda_1, \lambda_2, \ldots, \lambda_k$ (eigenvalues)

Instead use SVD of X to find $V_k$ (eigenvectors) and $\lambda_1, \lambda_2, \ldots, \lambda_k$(eigenvalues).

Advantage: didn't compute $X^TX$, which can be extremely large if there are many features.

By **SVD,** factorize $X = U S V^T$.

$A = X^TX = (U S V^T)^T(U S V^T) = (V S^T U^T)(U S V^T)$.

- $U^T U = I$ (identity matrix).
- S is a diagonal matrix, then $S^T = S$. Let $D = S^T S = S^2$

Thus $A = X^TX = (V S U^T)(U S V^T) = VS^2V^T = VDV^T$.

# 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
- Algorithm
- Examples
- Finding principle components
- When PCA doesn't work well

# When PCA Doesn't Work Well

- PCA doesn't find non-linear features
- One option is to transform the features before applying PCA
- To learn about kernels with PCA you can look at: https://en.m.wikipedia.org/wiki/Kernel_principal_component_analysis

# Summary

- PCA is a linear projection of our original feature space into a k dimensional vector space
- The new feature space is in a new coordinate system
- We choose the new coordinates such that:
  - maximizes variance
  - minimizes projection error (square loss)
- PCA is used for:
  - dimensionality reduction
  - visualization (if we choose k to be 2 or 3)
  - Compression (with loss)
  - De-noising (removes small variance in data)

# Additional notes

- The *data must be centered* before running PCA
- Optional: normalize none, some, or all the features
  - Normalizing will prevent arbitrary scales of different features affect the decisions made by the PCA algorithm
  - Not normalizing when features have similar scales with different variances will keep differences of features that are usually meaningful
- Choosing $k$ (i.e. size of subspace $\mathbb{R}^k$ for new features)
  - If performing PCA as a preprocessing step before running a learning algorithm, choose $k$ based on validation data performance
  - Optionally: choose $k$ based on how much of the original variance of X is maintained after projecting to a smaller dimension is $\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$. Note: X is noisy so this formula includes the noise.
  - Learn more at [https://ro-che.info/articles/2017-12-11-pca-explained-variance](https://ro-che.info/articles/2017-12-11-pca-explained-variance)

# Additional notes

- The *data must be centered* before running PCA
- Optional: normalize none, some, or all the features
  - Normalizing will prevent arbitrary scales of different features affect the decisions made by the PCA algorithm
  - Not normalizing when features have similar scales with different variances will keep differences of features that are usually meaningful
- Choosing $k$ (i.e. size of subspace $\mathbb{R}^k$ for new features)
  - If performing PCA as a preprocessing step before running a learning algorithm, choose $k$ based on validation data performance
  - Optionally: choose $k$ based on how much of the original variance of X is maintained after proj ion is $\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$. Note: X is noisy so this formula includes the noise.

The $\lambda_1, \lambda_2, \cdots, \lambda_d$ are the diagonal values in D = $S^2$

  - Learn more at https://ro-che.info/articles/2017-12-11-pca-explained-variance

# Extra slides

YOU ARE NOT RESPONSIBLE FOR THE MATERIAL ON THE NEXT TWO SLIDES

# Finding the top k principle components

$V_{1:k}$

The first k columns of V are the top k principle components

How do we compute these?

**Algorithm 1**

Power Iteration

Given matrix $A = X^\top X$:

- Select random unit vector $\mathbf{u}^{(0)}$
- For $i = 1, 2, \ldots,$ set $\mathbf{u}^{(1)} = A^i \mathbf{u}^{(0)}$. If $\mathbf{u}^{(i)}/\|\mathbf{u}^{(i)}\| \approx \mathbf{u}^{(i-1)}/\|\mathbf{u}^{(i-1)}\|$, then return $\mathbf{u}^{(i)}/\|\mathbf{u}^{(i)}\|$.

This works well if $\lambda_1, \lambda_2, \ldots, \lambda_k$ are not close in value to each other. Theorem: with prob ≥1/2 over the choice of $\mathbf{u}^{(0)}$, for t≥1,

$$|(A^T\mathbf{u}^{(0)})\mathbf{v}^{(1)}| \geq 1 - 2\sqrt{d}\left(\frac{\lambda_2}{\lambda_1}\right)^t$$

Note that if $\lambda_1 = \lambda_2$, then the algorithm will return a vector in the space spanned by the two principle components

Mathematicians call $\lambda_1, \lambda_2, \ldots, \lambda_k$ the first k eigenvalues and $V_{1:k}$ the first k eigenvectors

*"PCA boils down to computing the k eigenvectors of the covariance matrix $X^TX$ that have the largest eigenvalues."*

Remember: $\mathbf{x} = \sum_{i=1}^{d} z_i \mathbf{v}^{(i)} = \sum_{i=1}^{d} (\mathbf{x}^T \mathbf{v}^{(i)}) \mathbf{v}^{(i)}$

# Observation

Since $A = VDV^T$ th $A^2 = AA = (VDV^T)(VDV^T) = VD^2V^T$ thus $A^i = VD^iV^T$

i IS THE EXPONENT

We note that $A^i$ has the same principle components as $A$, but the diagonal values are $\lambda_1^i, \lambda_2^i, \ldots, \lambda_d^i$

# Computing the i$^{\text{th}}$ principal component

After finding the first principle component we can repeat the procedure on a new data matrix X

$$X = \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \\ \vdots \\ \mathbf{x}^{(N)} \end{bmatrix} \rightarrow X = \begin{bmatrix} \mathbf{x}^{(1)} - (\mathbf{x}^{(1)}\mathbf{v}^{(1)})\mathbf{v}^{(1)} \\ \mathbf{x}^{(2)} - (\mathbf{x}^{(2)}\mathbf{v}^{(1)})\mathbf{v}^{(1)} \\ \vdots \\ \mathbf{x}^{(N)} - (\mathbf{x}^{(N)}\mathbf{v}^{(1)})\mathbf{v}^{(1)} \end{bmatrix}$$

For each example we remove the part of the data that is explained by $\mathbf{v}^{(1)}$

By repeating this procedure we can find the first k principle components (i.e. first k eigenvectors)