DSC 190

Machine Learning: Representations

Lecture 8 | Part 1

Dimensionality Reduction

High Dimensional Data

- Data is often high dimensional (many features)

- Example: Netflix user
  - Number of movies watched
  - Number of movies saved
  - Total time watched
  - Number of logins
  - Days since signup
  - Average rating for comedy
  - Average rating for drama
  - ⋮

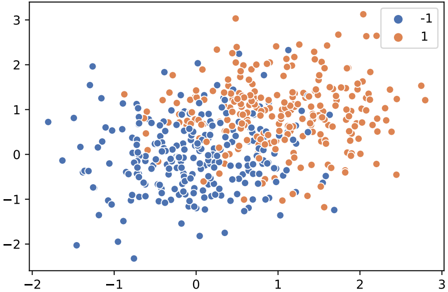
High Dimensional Data

- More features can give us more information
- But it can also cause problems
- **Today**: how do we reduce dimensionality without losing too much information?

More Features, More Problems

- Difficulties with high dimensional data:
  1. Requires more compute time / space
  2. Hard to visualize / explore
  3. The “curse of dimensionality”: it’s harder to learn

Experiment



- On this data, low 80% train/test accuracy
- Add 400 features of pure noise, re-train
- Now: 100% train accuracy, **58%** test accuracy
- **Overfitting!**

Task: Dimensionality Reduction

- We’d often like to **reduce** the dimensionality to improve performance, or to visualize.
- We will typically lose information
- Want to minimize the loss of useful information

Redundancy

- Two (or more) features may share the same information.
- Intuition: we may not need all of them.

Today

- Today we'll think about reducing dimensionality from $\mathbb{R}^d$ to $\mathbb{R}^1$

- Next time we'll go from $\mathbb{R}^d$ to $\mathbb{R}^{d'}$, with $d' \leq d$

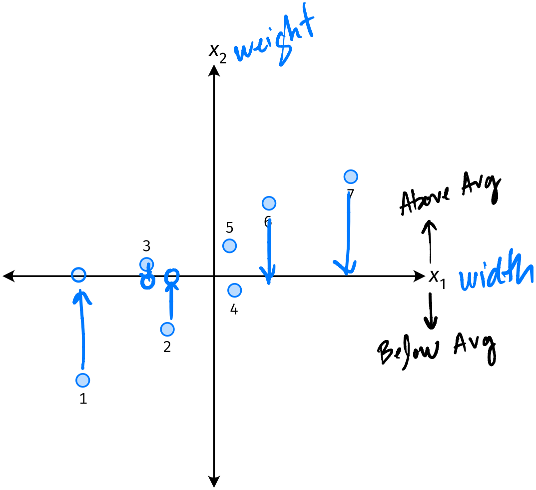
Today's Example

- Let's say we represent a phone with two features:
  - x_1: screen width
  - x_2: phone weight
- Both measure a phone's "size".
- Instead of representing a phone with both x_1 and x_2, can we just use a single number, z?
  - Reduce dimensionality from 2 to 1.

First Approach: Remove Features

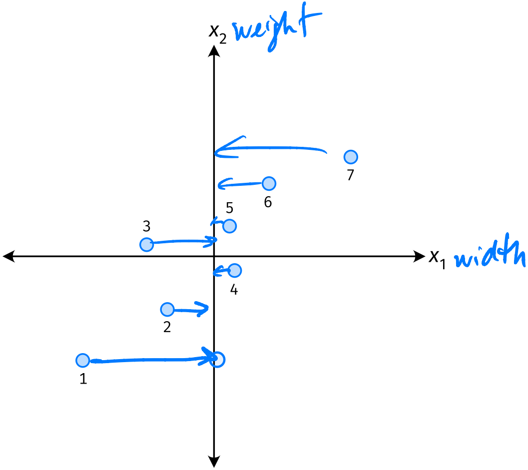
- Screen width and weight share information.
- **Idea:** keep one feature, remove the other.
- That is, set new feature $z = x_1$ (or $z = x_2$).

Removing Features



- Say we set $z^{(i)} = \vec{x}_1^{(i)}$ for each phone, i.
- Observe: $z^{(4)} > z^{(5)}$.
- Is phone 4 really "larger" than phone 5?

Removing Features



- Say we set $z^{(i)} = \vec{x}_2^{(i)}$ for each phone, i.
- Observe: $z^{(3)} > z^{(4)}$.
- Is phone 3 really "larger" than phone 4?

Better Approach: Mixtures of Features

- **Idea**: z should be a combination of x_1 and x_2.

- One approach: linear combination.

$$\vec{u} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

$$\begin{aligned} z &= u_1 x_1 + u_2 x_2 \\ &= \vec{u} \cdot \vec{x} \end{aligned}$$

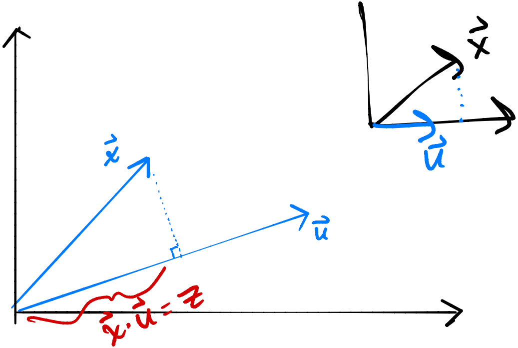
- $u_1, \ldots, u_2$ are the mixture coefficients; we can choose them.

Normalization

- Mixture coefficients generalize proportions.
- We could assume, e.g., $|u_1| + |u_2| = 1$.
- But it makes the math easier if we assume $\sqrt{u_1^2 + u_2^2} = \sqrt{1}$.
- Equivalently, if $\vec{u} = (u_1, u_2)^T$, assume $\|\vec{u}\| = 1$

$z = \vec{x} \cdot \vec{u}$

Geometric Interpretation



- z measures how much of $\vec{x}$ is in the direction of $\vec{u}$
- If $\vec{u} = (1, 0)^T$, then $z = x_1$
- If $\vec{u} = (0, 1)^T$, then $z = x_2$

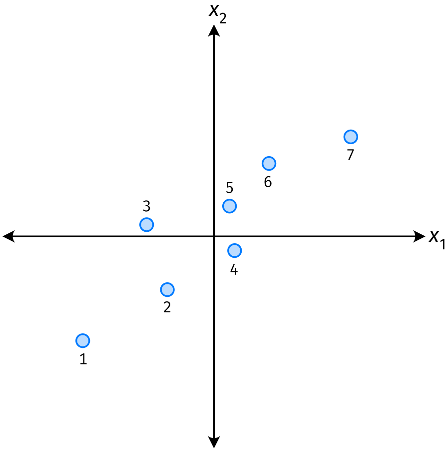
$u = (1, 0)$ $\quad z = \vec{u} \cdot \vec{x} = (1, 0)^T \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = x_1$

Choosing $\vec{u}$

- Suppose we have only two features:
  - x_1: screen size
  - x_2: phone thickness
- We'll create single new feature, z, from x_1 and x_2.
  - Assume $z = u_1 x_1 + u_2 x_2 = \vec{x} \cdot \vec{u}$
  - Interpretation: z is a measure of a phone's size
- How should we choose $\vec{u} = (u_1, u_2)^T$?

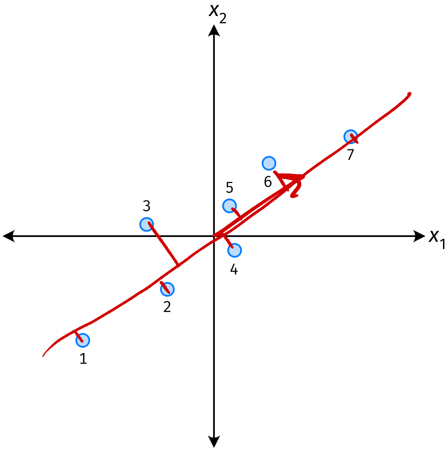
Example

$u = (1, 0)^T$



- $\vec{u}$ defines a direction
- $\vec{z}^{(i)} = \vec{x}^{(i)} \cdot \vec{u}$ measures position of $\vec{x}$ along this direction

Example



- Phone "size" varies most along a diagonal direction.
- Along direction of "max variance", phones are well-separated.
- **Idea:** $\vec{u}$ should point in direction of "max variance".

Our Algorithm (Informally)

- **Given**: data points $\vec{x}^{(1)}, \ldots, \vec{x}^{(n)} \in \mathbb{R}^d$
- Pick $\vec{u}$ to be the direction of “max variance”
- Create a new feature, z, for each point:

$$z^{(i)} = \vec{x}^{(i)} \cdot \vec{u}$$

PCA

- This algorithm is called **Principal Component Analysis**, or **PCA**.
- The direction of maximum variance is called the **principal component**.

Exercise

Suppose the direction of maximum variance in a data set is

$$\vec{u} = (1/\sqrt{2}, -1/\sqrt{2})^T$$

Let

- $\vec{x}^{(1)} = (3, -2)^T$
- $\vec{x}^{(2)} = (1, 4)^T$

What are $z^{(1)}$ and $z^{(2)}$?

$$z^{(1)} = \vec{x}^{(1)} \cdot \vec{u} = (3,-2)^T \cdot (1/\sqrt{2}, -1/\sqrt{2})^T$$

$$= \frac{3}{\sqrt{2}} + \frac{2}{\sqrt{2}} = \frac{5}{\sqrt{2}}$$

$$z^{(2)} = \vec{x}^{(2)} \cdot \vec{u} = (1,4)^T \cdot (1/\sqrt{2}, -1/\sqrt{2})^T$$

$$= \frac{1}{\sqrt{2}} - \frac{4}{\sqrt{2}} = \frac{-3}{\sqrt{2}}$$

Problem

- How do we compute the “direction of maximum variance”?

DSC 190

Machine Learning: Representations

Lecture 8 | Part 2

Covariance Matrices

Variance

- We know how to compute the variance of a set of numbers $X = \{x^{(1)}, \dots, x^{(n)}\}$:

$$\mathrm{Var}(X) = \frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - \mu)^2$$

- The variance measures the "spread" of the data

Generalizing Variance

- If we have two features, x_1 and x_2, we can compute the variance of each as usual:

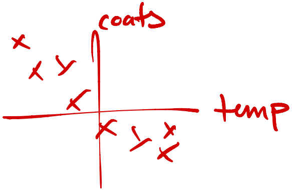
$$\text{Var}(x_1) = \frac{1}{n} \sum_{i=1}^{n} (\vec{x}_1^{(i)} - \mu_1)^2$$

$$\text{Var}(x_2) = \frac{1}{n} \sum_{i=1}^{n} (\vec{x}_2^{(i)} - \mu_2)^2$$

- Can also measure how x_1 and x_2 vary together.

Measuring Similar Information

- Features which share information if they *vary together*.
  - A.k.a., they “co-vary”
- Positive association: when one is above average, so is the other
- Negative association: when one is above average, the other is below average

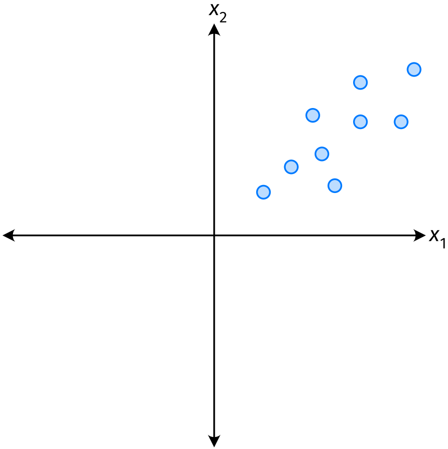


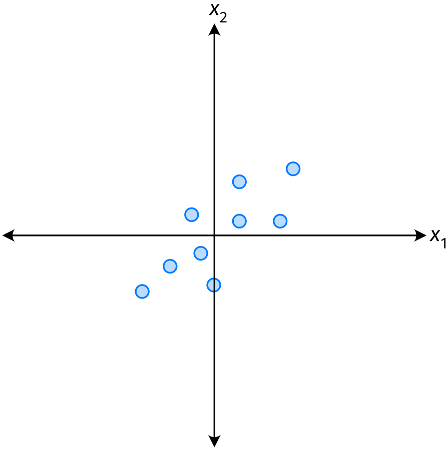
Examples

- Positive: temperature and ice cream cones sold.
- Positive: temperature and shark attacks.
- Negative: temperature and coats sold.

Centering

- First, it will be useful to **center** the data.





Centering

- Compute the mean of each feature:

$$\mu_j = \frac{1}{n} \sum_{1}^{n} \vec{x}_j^{(i)}$$

- Define new centered data:

$$\vec{z}^{(i)} = \begin{pmatrix} \vec{x}_1^{(i)} - \mu_1 \\ \vec{x}_2^{(i)} - \mu_2 \\ \vdots \\ \vec{x}_d^{(i)} - \mu_d \end{pmatrix}$$

Centering (Equivalently)

- Compute the mean of all data points:

$$\vec{\mu} = \frac{1}{n} \sum_{1}^{n} \vec{x}^{(i)}$$

- Define new centered data:

$$\vec{z}^{(i)} = \vec{x}^{(i)} - \vec{\mu}$$

Exercise

Center the data set:

$$\vec{x}^{(1)} = (1, 2, 3)^T \rightarrow (1,2,3)-(0,1,2)=(1,1,1)^T$$
$$\vec{x}^{(2)} = (-1, -1, 0)^T \rightarrow (-1,-2,-2)^T$$
$$\vec{x}^{(3)} = (0, 2, 3)^T$$

$$\vec{\mu} = \frac{\vec{x}^{(1)} + \vec{x}^{(2)} + \vec{x}^{(3)}}{3} = \frac{1}{3}(0, 3, 6)^T = (0, 1, 2)^T$$

Quantifying Co-Variance

- One approach is as follows[1].

$$\mathrm{Cov}(x_i, x_j) = \frac{1}{n}\sum_{k=1}^{n} \vec{x}_i^{(k)} \vec{x}_j^{(k)}$$

  - For each data point, multiply the value of feature i and feature j, then average these products.

  - This is the **covariance** of features i and j.

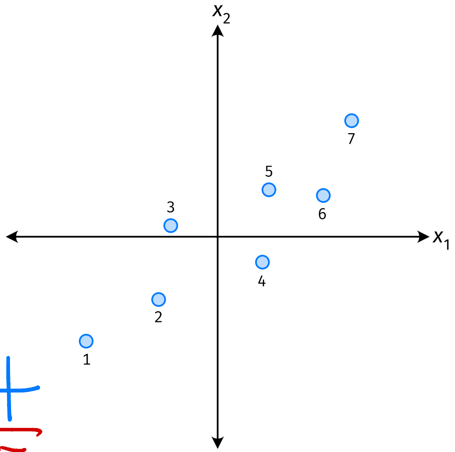
[1] Assuming centered data

Quantifying Covariance

- Assume the data are **centered**.

$$\text{Covariance} = \frac{1}{7}\sum_{i=1}^{7} \vec{x}_1^{(i)} \times \vec{x}_2^{(i)}$$

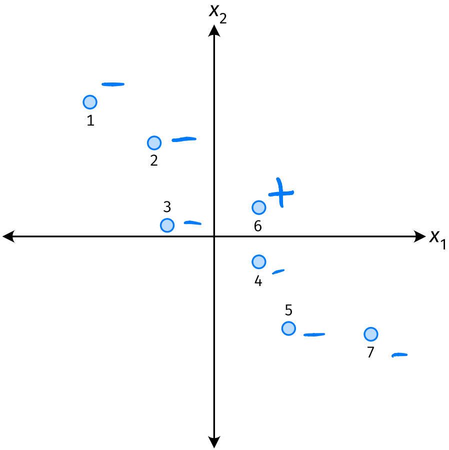
$$\frac{+}{1} \quad \frac{+}{2} \quad \frac{-}{3} \quad \frac{-}{4} \quad \frac{+}{5} \quad \frac{+}{6} \quad \frac{+}{7}$$



Quantifying Covariance

- Assume the data are **centered**.

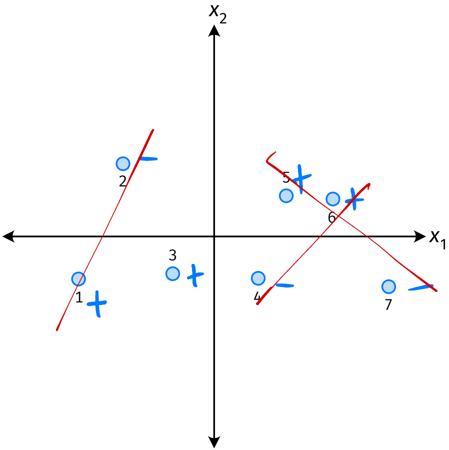
$$\text{Covariance} = \frac{1}{7}\sum_{i=1}^{7} \vec{x}_1^{(i)} \times \vec{x}_2^{(i)}$$



Quantifying Covariance

- Assume the data are **centered**.

$$\text{Covariance} = \frac{1}{7}\sum_{i=1}^{7} \vec{x}_1^{(i)} \times \vec{x}_2^{(i)}$$



Quantifying Covariance

- The **covariance** quantifies extent to which two variables vary together.

- Assume we have centered the data.

- The **sample covariance** of feature i and j is:

$$\sigma_{ij} = \frac{1}{n} \sum_{k=1}^{n} \vec{x}_i^{(k)} \vec{x}_j^{(k)}$$

Exercise

True or False: $\sigma_{ij} = \sigma_{ji}$? True

$$\sigma_{ij} = \frac{1}{n} \sum_{k=1}^{n} \vec{x}_i^{(k)} \vec{x}_j^{(k)}$$

Covariance Matrices

$$\begin{array}{cc} & \overset{d}{\longleftrightarrow} \\ d\updownarrow & \begin{pmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} & \cdots & \sigma_{1d} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} & \cdots & \\ & & \ddots & & \end{pmatrix} \end{array}$$

- Given data $\vec{x}^{(1)}, \ldots, \vec{x}^{(n)} \in \mathbb{R}^d$.
- The **sample covariance matrix** C is the $d \times d$ matrix whose ij entry is defined to be σ_{ij}.

$$\sigma_{ij} = \frac{1}{n} \sum_{k=1}^{n} \vec{x}_i^{(k)} \vec{x}_j^{(k)}$$

Observations

- Diagonal entries of C are the variances.
- The matrix is **symmetric**!

Note

- Sometimes you'll see the sample covariance defined as:

$$\sigma_{ij} = \frac{1}{n-1} \sum_{k=1}^{n} \vec{x}_i^{(k)} \vec{x}_j^{(k)}$$

  Note the $1/(n-1)$
- This is an **unbiased** estimator of the population covariance.
- Our definition is the **maximum likelihood** estimator.
- In practice, it doesn't matter: $1/(n-1) \approx 1/n$.
- For consistency, in this class use $1/n$.

Computing Covariance

- There is a “trick” for computing sample covariance matrices.
- Step 1: make $n \times d$ data matrix, X
- Step 2: make Z by centering columns of X
- Step 3: $C = \frac{1}{n} Z^T Z$

$$X = \left(\xrightarrow{d} \right) \Big\downarrow n$$

Computing Covariance (in code)[2]

```
>>> mu = X.mean(axis=0)
>>> Z = X - mu
>>> C = 1 / len(X) * Z.T @ Z
```

[2] Or use `np.cov`

DSC 190

Machine Learning: Representations

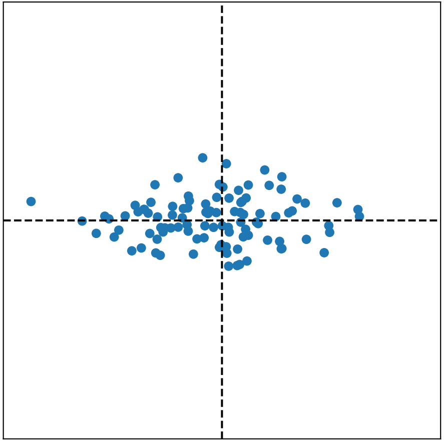
Lecture 8 | Part 3

Visualizing Covariance Matrices

Visualizing Covariance Matrices

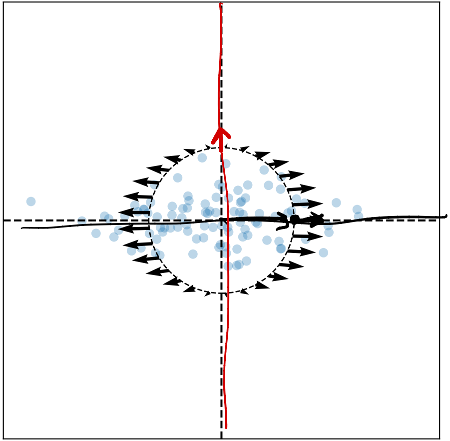
- Covariance matrices are symmetric.
- They have axes of symmetry (eigenvectors and eigenvalues).
- What are they?

Visualizing Covariance Matrices



$$C \approx \begin{pmatrix} 5 & 0 \\ 0 & 2 \end{pmatrix}$$

Visualizing Covariance Matrices

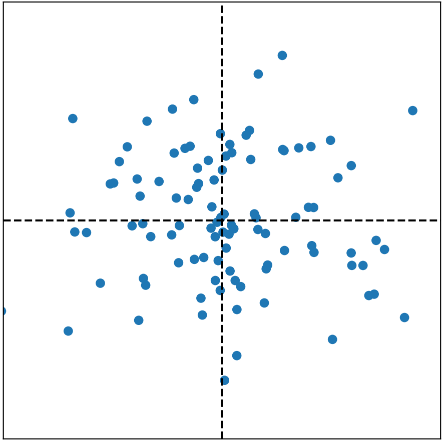


Eigenvectors:

$\vec{u}^{(1)} \approx (1,0)^T \quad \lambda_1 = 5$

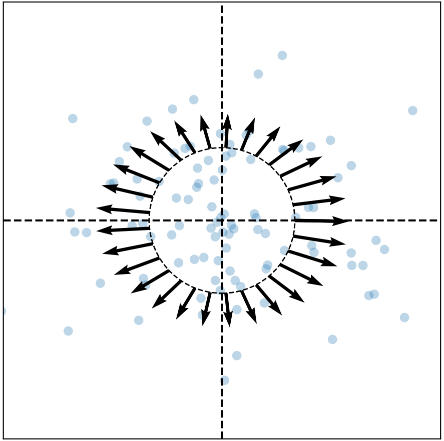
$\vec{u}^{(2)} \approx (0,1)^T \quad \lambda_2 = 2$

Visualizing Covariance Matrices



$$C \approx \begin{pmatrix} 5 & 0 \\ 0 & 5 \end{pmatrix}$$

Visualizing Covariance Matrices

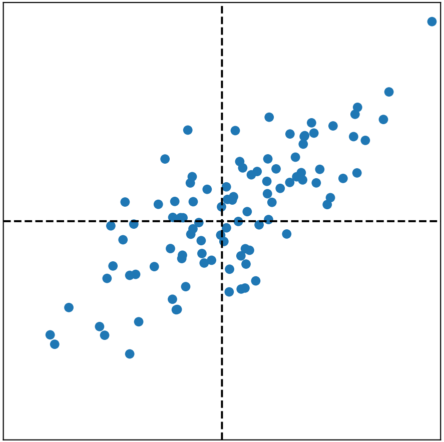


Eigenvectors:

$\vec{u}^{(1)} \approx$ Any direction

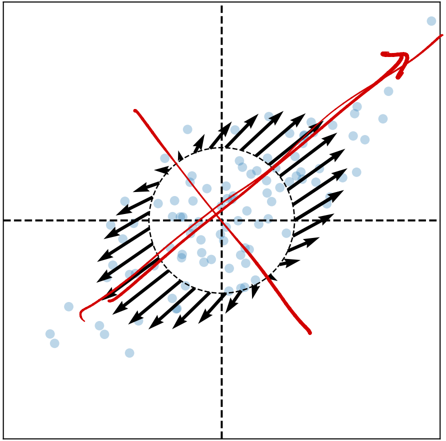
$\vec{u}^{(2)} \approx$

Visualizing Covariance Matrices



$$C \approx \begin{pmatrix} 5 & 3 \\ 3 & 3 \end{pmatrix}$$

Visualizing Covariance Matrices



Eigenvectors:

$\vec{u}^{(1)} \approx (1/\sqrt{2}, 1/\sqrt{2})^T$

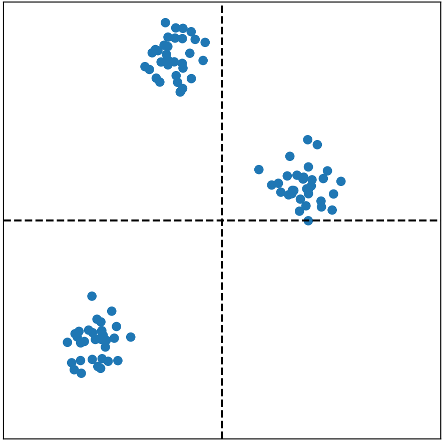
$\vec{u}^{(2)} \approx$

Intuitions

- The **eigenvectors** of the covariance matrix describe the data's "principal directions"
  - C tells us something about data's shape.
- The **top eigenvector** points in the direction of "maximum variance".
- The **top eigenvalue** is proportional to the variance in this direction.

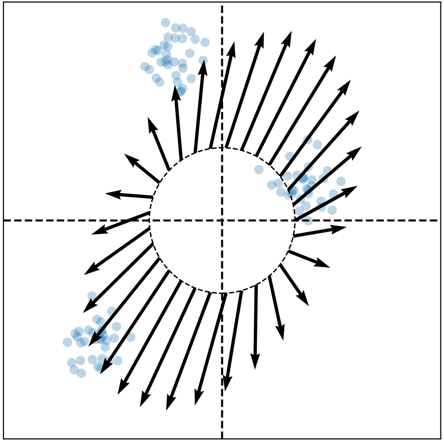
Caution

- The data doesn't always look like this.
- We can always compute covariance matrices.
- They just may not describe the data's shape very well.



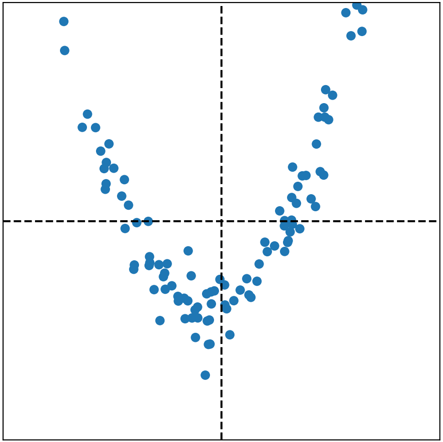
Caution

- The data doesn't always look like this.
- We can always compute covariance matrices.
- They just may not describe the data's shape very well.



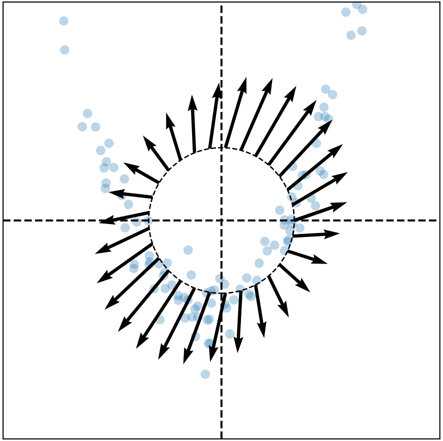
Caution

- The data doesn't always look like this.
- We can always compute covariance matrices.
- They just may not describe the data's shape very well.



Caution

- The data doesn't always look like this.
- We can always compute covariance matrices.
- They just may not describe the data's shape very well.



DSC 190

Machine Learning: Representations

Lecture 8 | Part 4

PCA, More Formally

The Story (So Far)

- We want to create a single new feature, z.

- Our idea: $z = \vec{x} \cdot \vec{u}$; choose $\vec{u}$ to point in the "direction of maximum variance".

- Intuition: the top eigenvector of the covariance matrix points in direction of maximum variance.

More Formally...

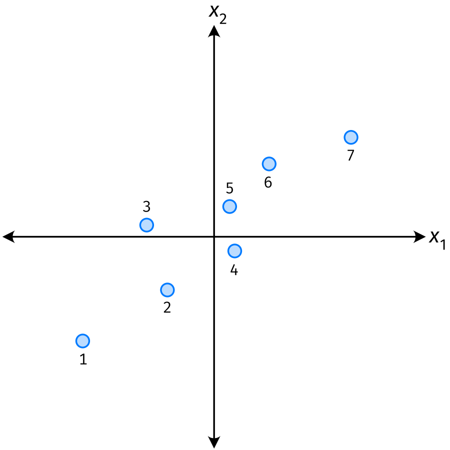
- We haven’t actually defined “direction of maximum variance”
- Let’s derive PCA more formally.

Variance in a Direction

- Let $\vec{u}$ be a unit vector.
- $z^{(i)} = \vec{x}^{(i)} \cdot \vec{u}$ is the new feature for $\vec{x}^{(i)}$.
- The variance of the new features is:

$$\begin{aligned}\text{Var}(z) &= \frac{1}{n}\sum_{i=1}^{n}(z^{(i)} - \mu_z)^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}\left(\vec{x}^{(i)} \cdot \vec{u} - \mu_z\right)^2\end{aligned}$$

Example



Note

- If the data are centered, then $\mu_z = 0$ and the variance of the new features is:

$$\begin{aligned}\text{Var}(z) &= \frac{1}{n}\sum_{i=1}^{n}(z^{(i)})^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}\left(\vec{x}^{(i)} \cdot \vec{u}\right)^2\end{aligned}$$

Goal

- The variance of a data set in the direction of $\vec{u}$ is:

$$g(\vec{u}) = \frac{1}{n}\sum_{i=1}^{n}\left(\vec{x}^{(i)} \cdot \vec{u}\right)^2$$

- Our goal: Find a unit vector $\vec{u}$ which maximizes g.

Claim

$$\frac{1}{n}\sum_{i=1}^{n}\left(\vec{x}^{(i)} \cdot \vec{u}\right)^2 = \vec{u}^T C \vec{u}$$

Our Goal (Again)

- Find a unit vector $\vec{u}$ which maximizes $\vec{u}^T C \vec{u}$.

Claim

- To maximize $\vec{u}^T C \vec{u}$ over unit vectors, choose $\vec{u}$ to be the top eigenvector of C.

- Proof:

PCA (for a single new feature)

- **Given**: data points $\vec{x}^{(1)}, \ldots, \vec{x}^{(n)} \in \mathbb{R}^d$

1. Compute the covariance matrix, C.
2. Compute the top eigenvector $\vec{u}$, of C.
3. For $i \in \{1, \ldots, n\}$, create new feature:

$$z^{(i)} = \vec{u} \cdot \vec{x}^{(i)}$$

A Parting Example

- MNIST: 60,000 images in 784 dimensions
- Principal component: $\vec{u} \in \mathbb{R}^{784}$
- We can project an image in $\mathbb{R}^{784}$ onto $\vec{u}$ to get a single number representing the image

Example