CS5841/EE5841 Machine Learning

Lecture 6: Clustering and dimensionality reduction

Evan Lucas

Overview

- Course updates
- Dimensionality reduction
- Clustering

Class updates

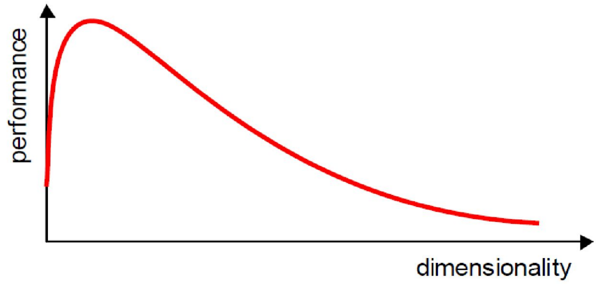
- HW1 due soon
- HW2 progress?
- HW3 released last Friday
- Clustering/dimensionality reduction quiz instead of homework?
- Decision tree/ensemble learning quiz coming up
- Probability quiz deadlines extended
- Review day for half of class on Wednesday

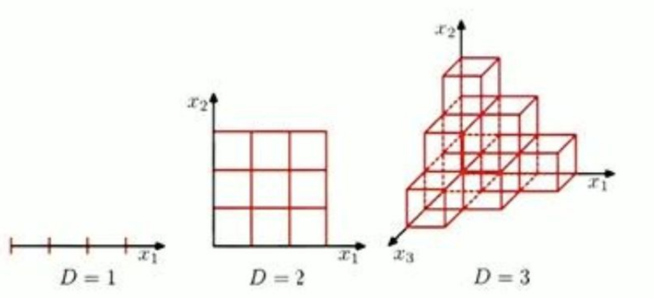
Dimensionality Reduction

Bishop 12

Curse of dimensionality

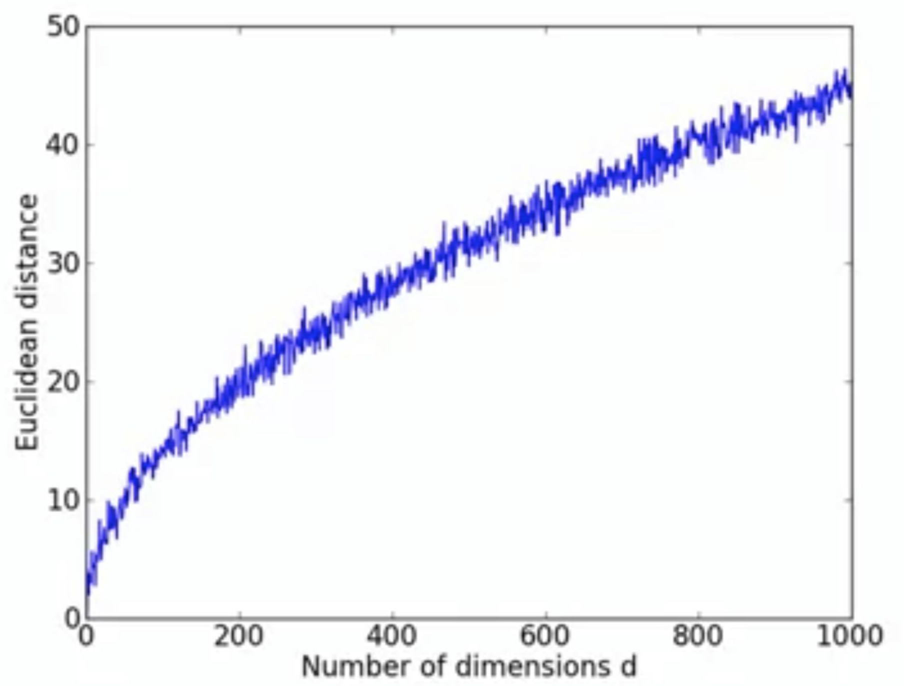
- Increasing the number of features will not always improve classification accuracy.
- In practice, the inclusion of more features might actually lead to worse performance.
- When dimensionality increases, the volume of the space increases exponentially so that the data becomes sparce.
- The number of training examples required increases exponentially with dimensionality **d**.





Curse of dimensionality (KNN example)

- In KNN, points tend to never be close together in high dimensional spa



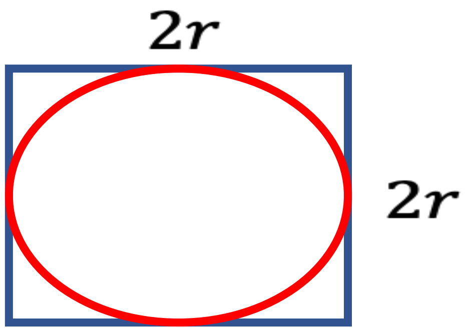
Credit: Bill Howe

Curse of dimensionality (KNN example)

- Consider a **hypersphere** with radius r and dimension d
- Distance between center and the corners is $r\sqrt{d}$
- Consider **Square** with edge of length $2r$
  - $\sqrt{r^2 + r^2}$
- Consider **hypercube** with edge of length $2r$

$$\frac{V_{hypersphere}}{V_{hypercube}} = \frac{\pi^{d/2}}{d2^{d-1}\Gamma(d/2)} \rightarrow 0 \text{ as } d \rightarrow \infty$$

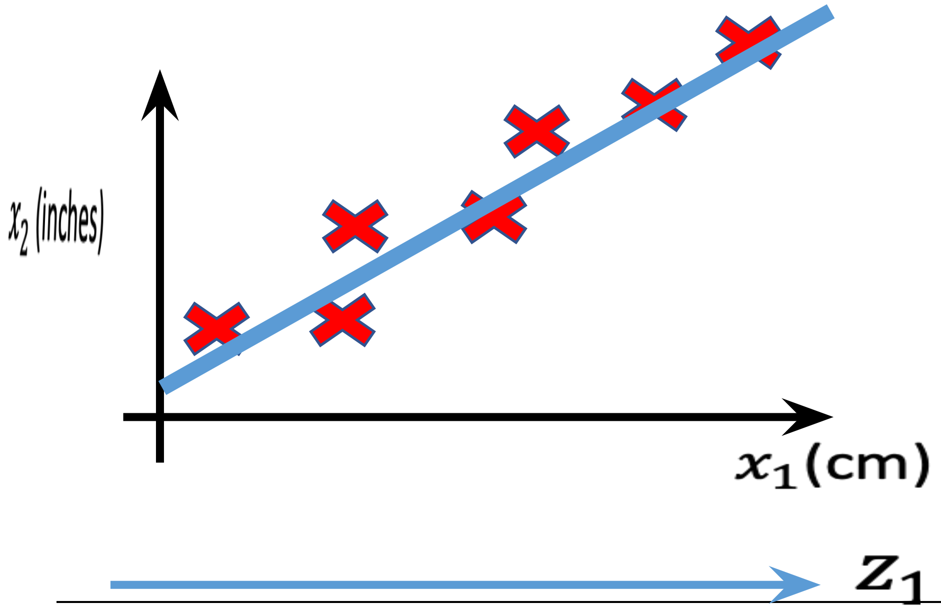
- Hypercube consist almost entirely of the “corners”



d= 2

Data Compression

- Reduces the required time and storage space
- Removing multi-collinearity improves the interpretation of the parameters of the machine learning model.



$$x^{(1)} \in R^2 \rightarrow z^{(1)} \in R$$

$$x^{(2)} \in R^2 \rightarrow z^{(1)} \in R$$

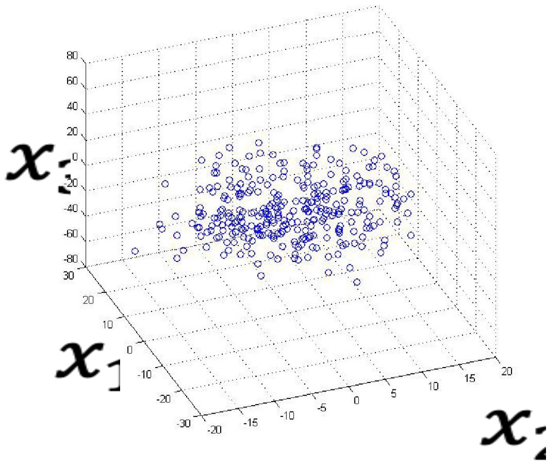
$$\vdots$$

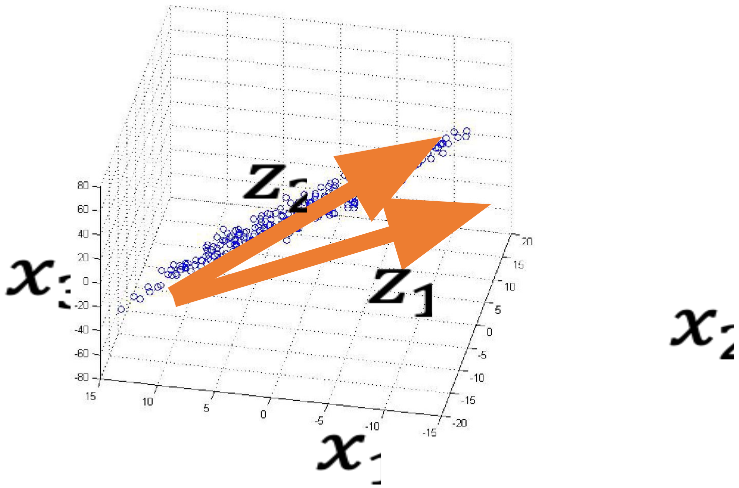
$$x^{(m)} \in R^2 \rightarrow z^{(m)} \in R$$

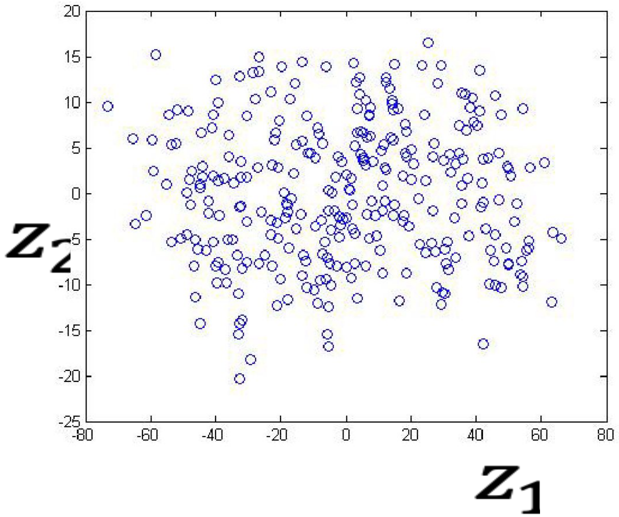
Data Compression

- Reduce data from 3D to 2D

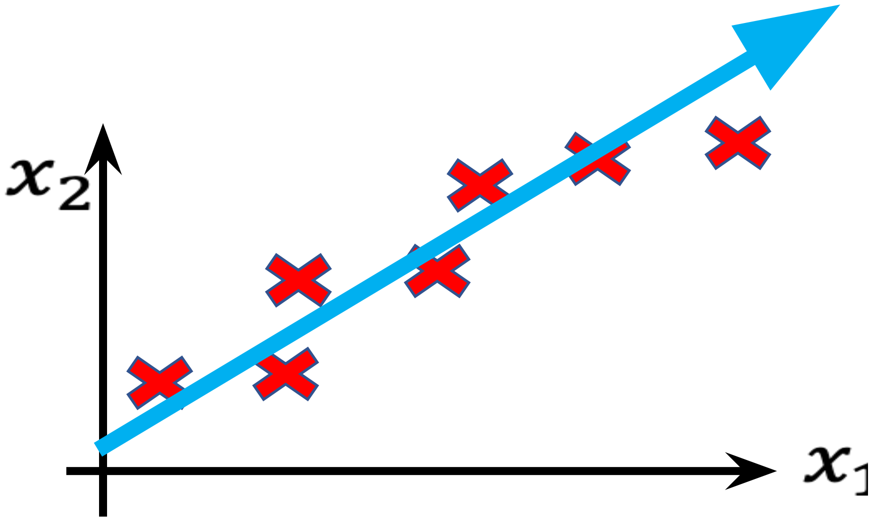
In reality: 1000D to 100D





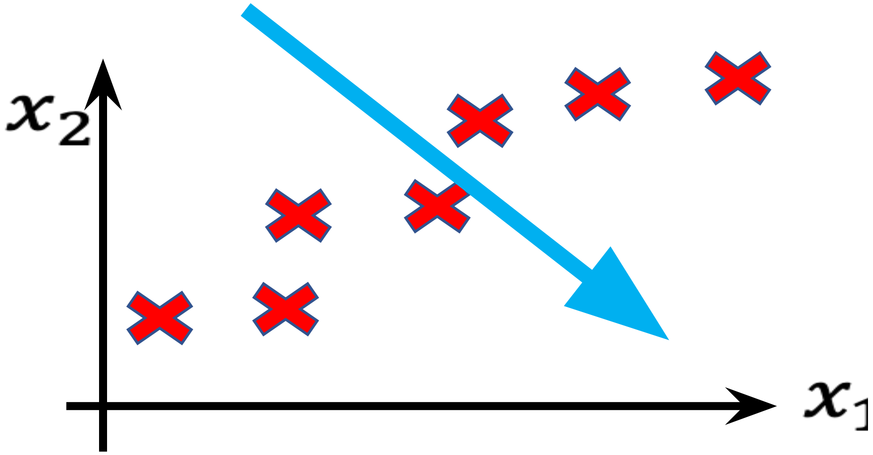


Principal Component Analysis (PCA)



- Find a single line onto which to project this data

Principal Component Analysis (PCA)

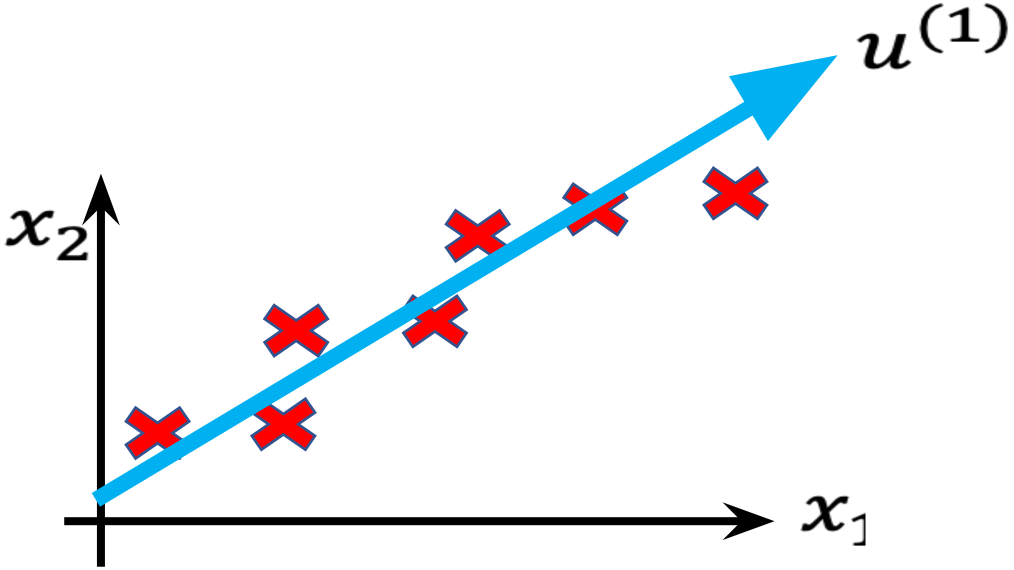


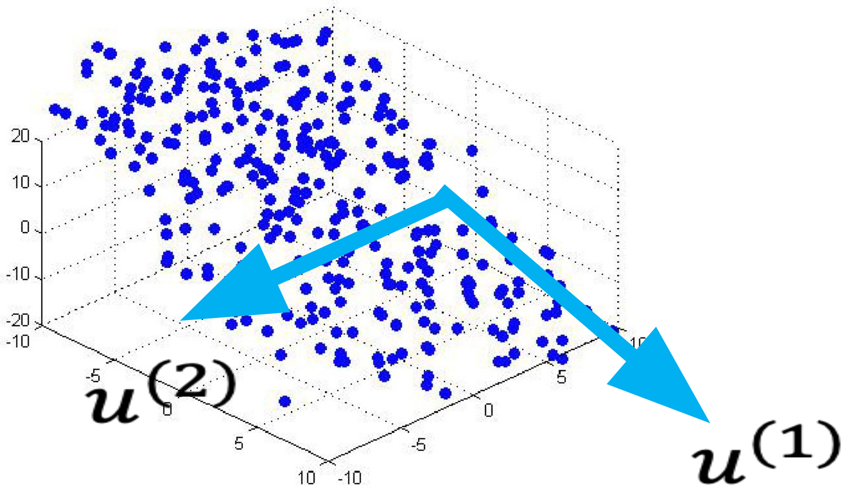
- Find a single line onto which to project this data

minimize the projection error

Principal Component Analysis (PCA)

- PCA tries to find the surface (a straight line in this case) which has the minimum projection error

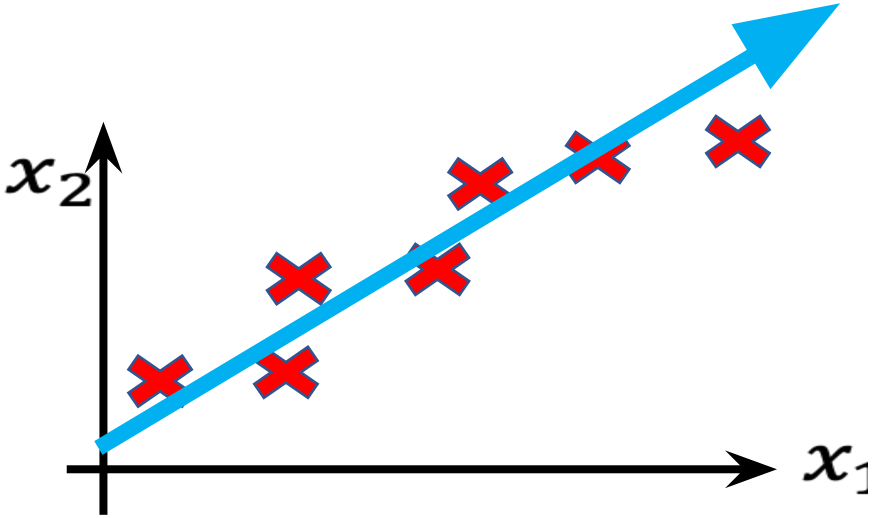




- Reduce n-D to k-D: find $u^{(1)}, u^{(2)}, \cdots, u^{(k)} \in R^n$ onto which to project the data, so as to minimize the projection error

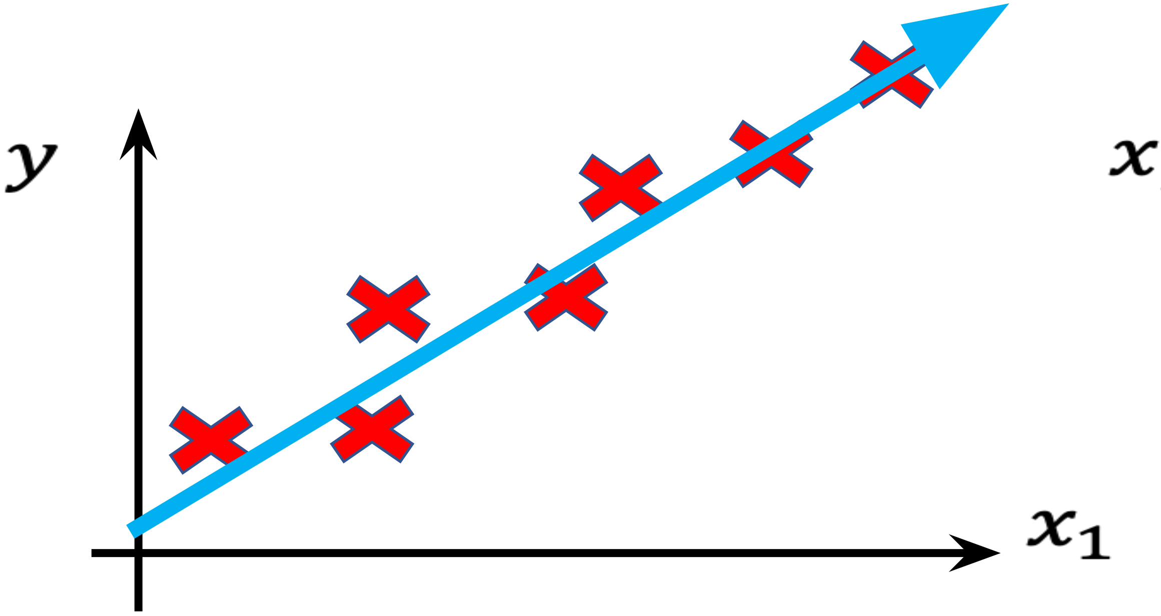
Feature scaling is important before PCA

- Normalization/Standardization is Important before PCA

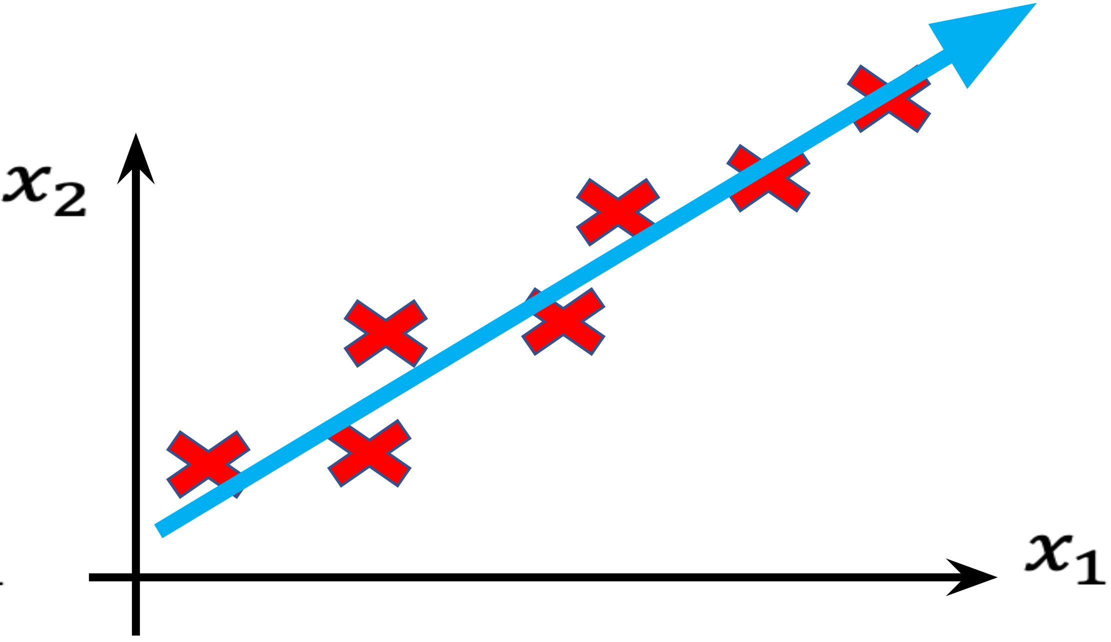


Linear regression vs. PCA

- Bonus point: What is the difference?



Linear Regression
vertical distance
Predict y



PCA
orthogonal distance
There is no "y"

Data pre-processing

- Training set: $x^{(1)}, x^{(2)}, \cdots, x^{(m)}$
- Preprocessing (feature scaling/mean normalization)

$$\mu_j = \frac{1}{m} \sum_i x_j^{(i)}$$

Replace each $x_j^{(i)}$ with $x_j - \mu_j$

If different features on different scales, scale features to have comparable range of values

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{s_j}$$

s_j: Biggest – smallest, Standard deviation (more commonly)

Principal Component Analysis Algorithm

- Goal: Reduce data from n-dimensions to k-dimensions
- Step 1: Compute "covariance matrix"

$$\Sigma = \frac{1}{m}\sum_{i=1}^{n}(x^{(i)})(x^{(i)})^{\mathsf{T}}$$ → an [n x n] matrix

- Step 2: Compute "eigenvectors" of the covariance matrix (e.g., using singular value decomposition)

```
[U, S, V] = svd(Σ)
```

$$\Sigma \mathrm{u_i} = \lambda_i \mathrm{u_i}$$

we assume eigenvalue $\lambda_1 > \lambda_2 > \cdots > \lambda_n$
and $u_1, u_2, \ldots u_n$ are the corresponding eigenvectors

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \ldots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

Principal Component Analysis Algorithm

- Goal: Reduce data from n-dimensions to k-dimensions
- Principal components: $u^{(1)}, u^{(2)}, \cdots, u^{(k)} \in R^n$
  - Just take the first *k-vectors* from U (first k columns)

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \ldots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

- Ureduce = U(:, 1:k) → an [n x k] matrix
- z = Ureduce$^{\mathrm{T}}$ * x → a [k x 1] vector

$$z = U_{reduce}^{T} x$$

PCA algorithm summary

- Preprocessing
- Calculate sigma (covariance matrix)
- Calculate eigenvectors with **svd**
- Take k vectors from U (`Ureduce = U(:, 1:k)`)
- Calculate z ($z = U_{reduce}^{T} x$)

Clustering

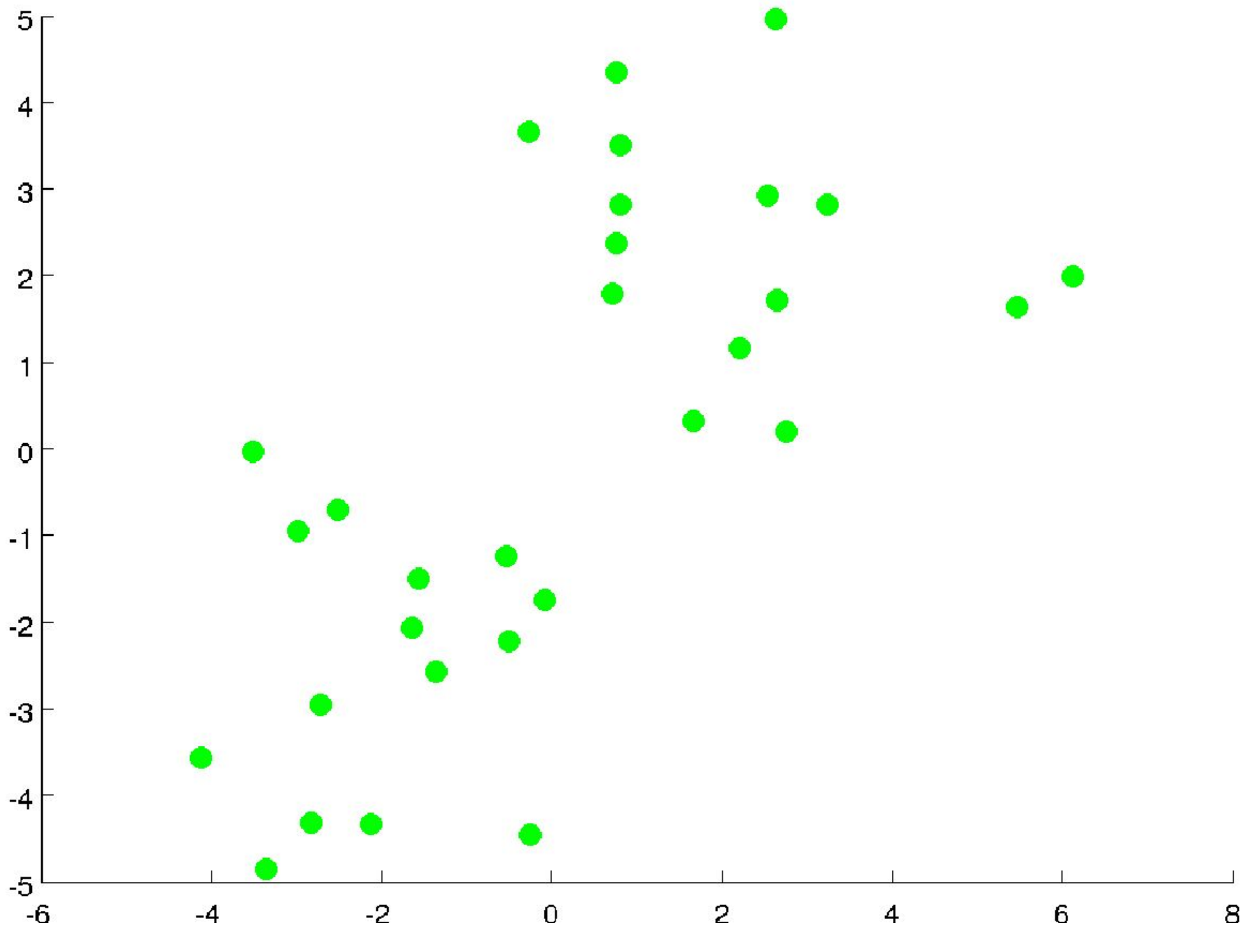
ESL 14.3

Clustering Examples

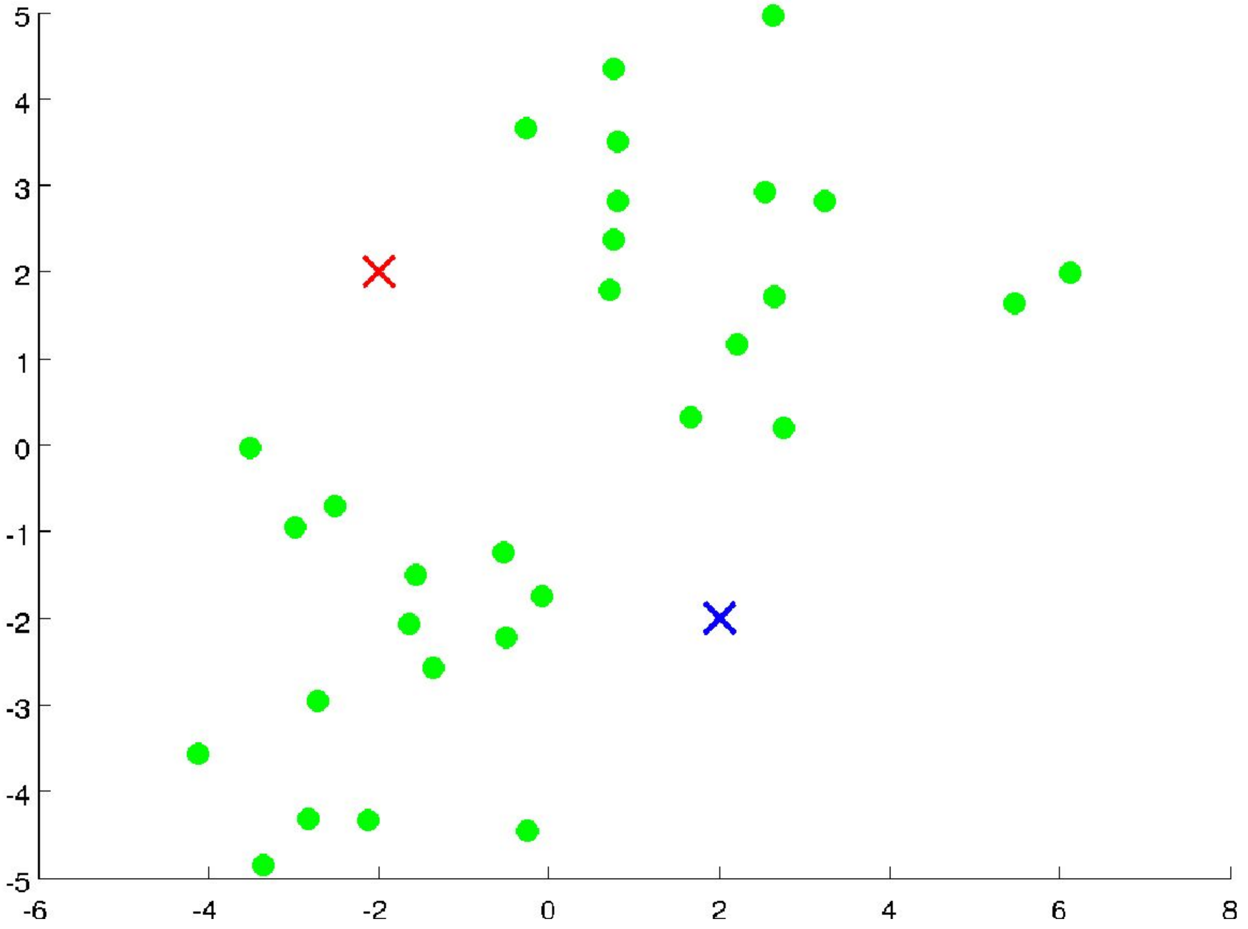
- **Market segmentation**
  - group customers into different market segments
- **Organizing computer clusters** and data centers
  - for network layout and location
- **Astronomical data analysis**
  - understanding galaxy formation

K-means

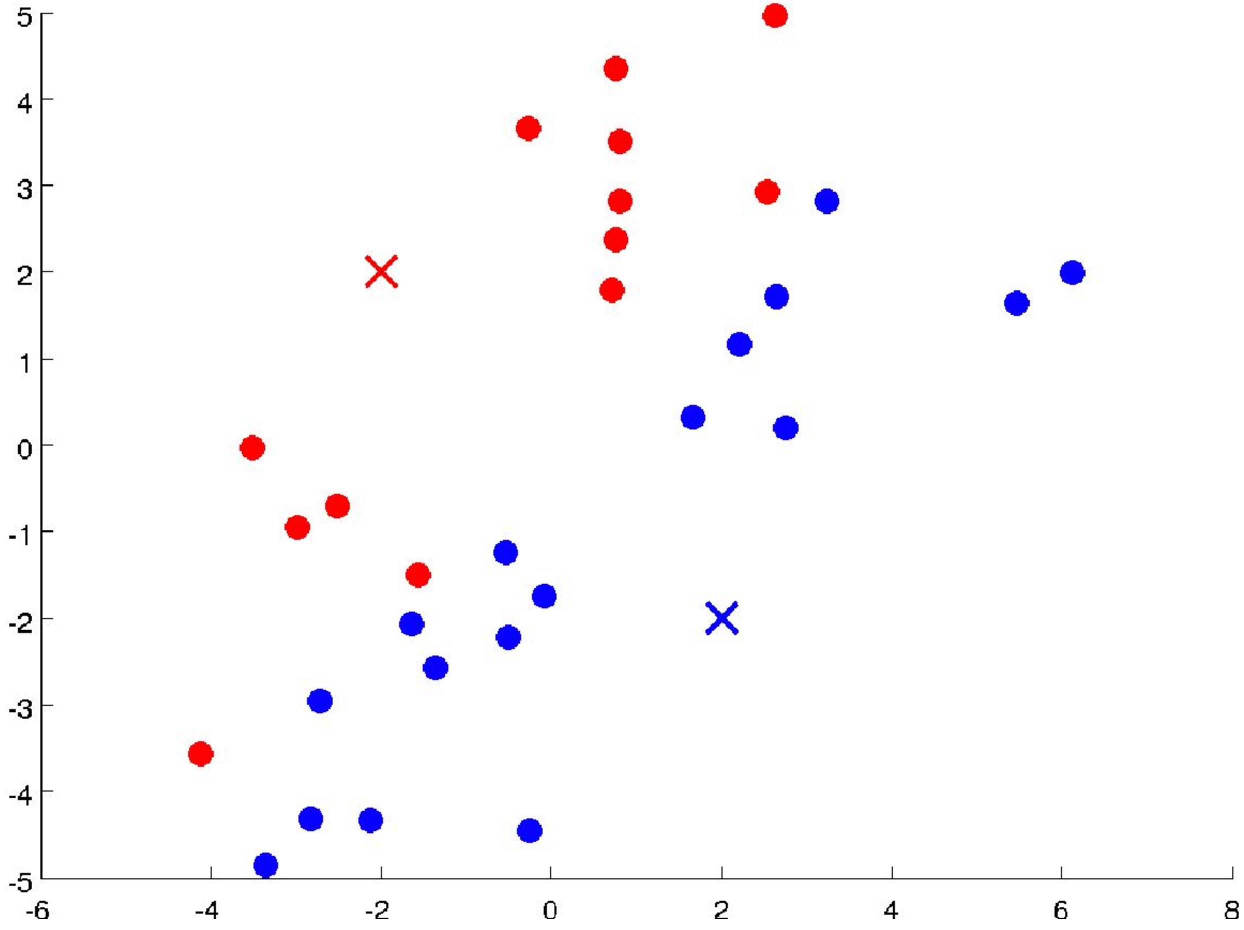
Take unlabeled data and group into two clusters



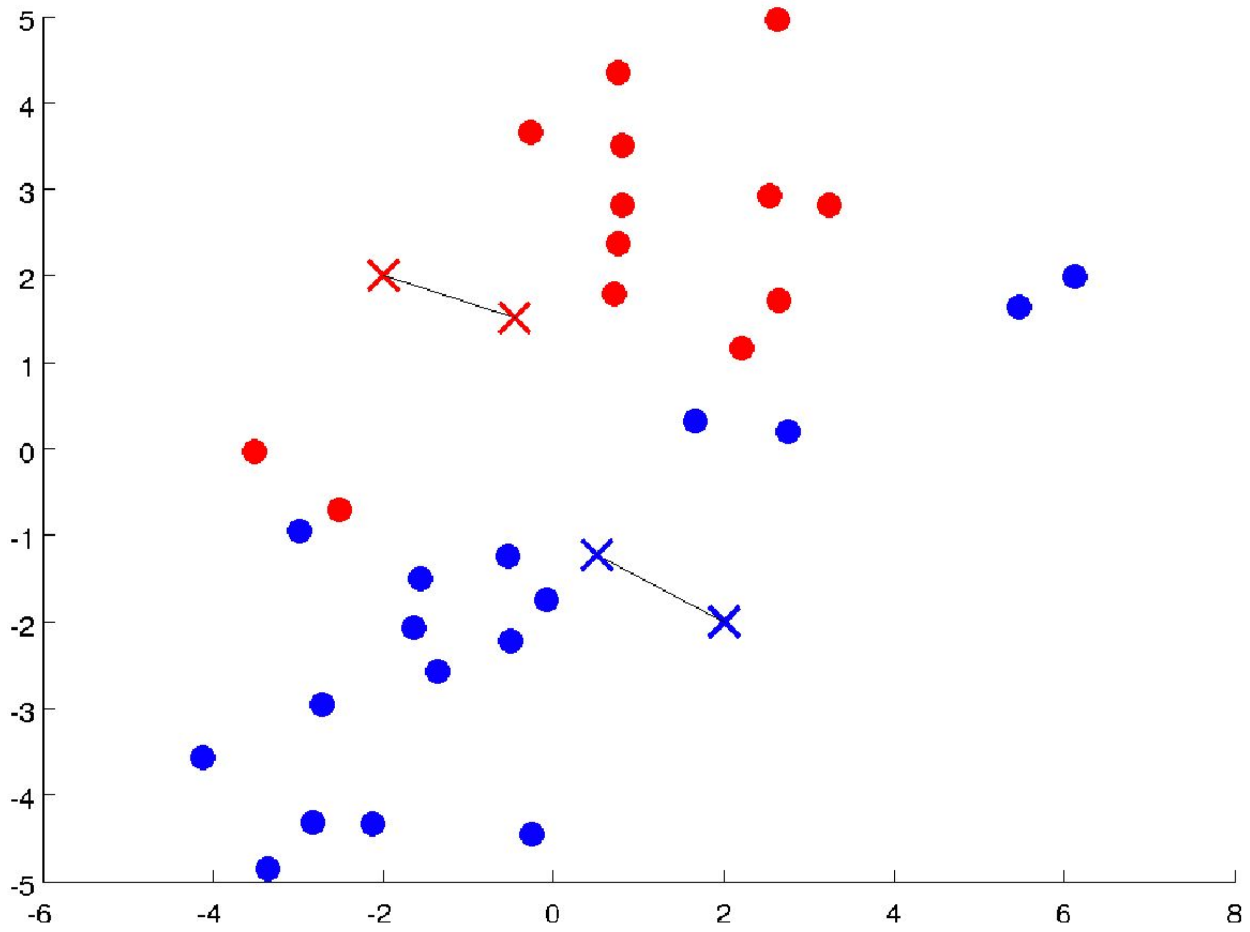
Randomly allocate *K* points as cluster centroids



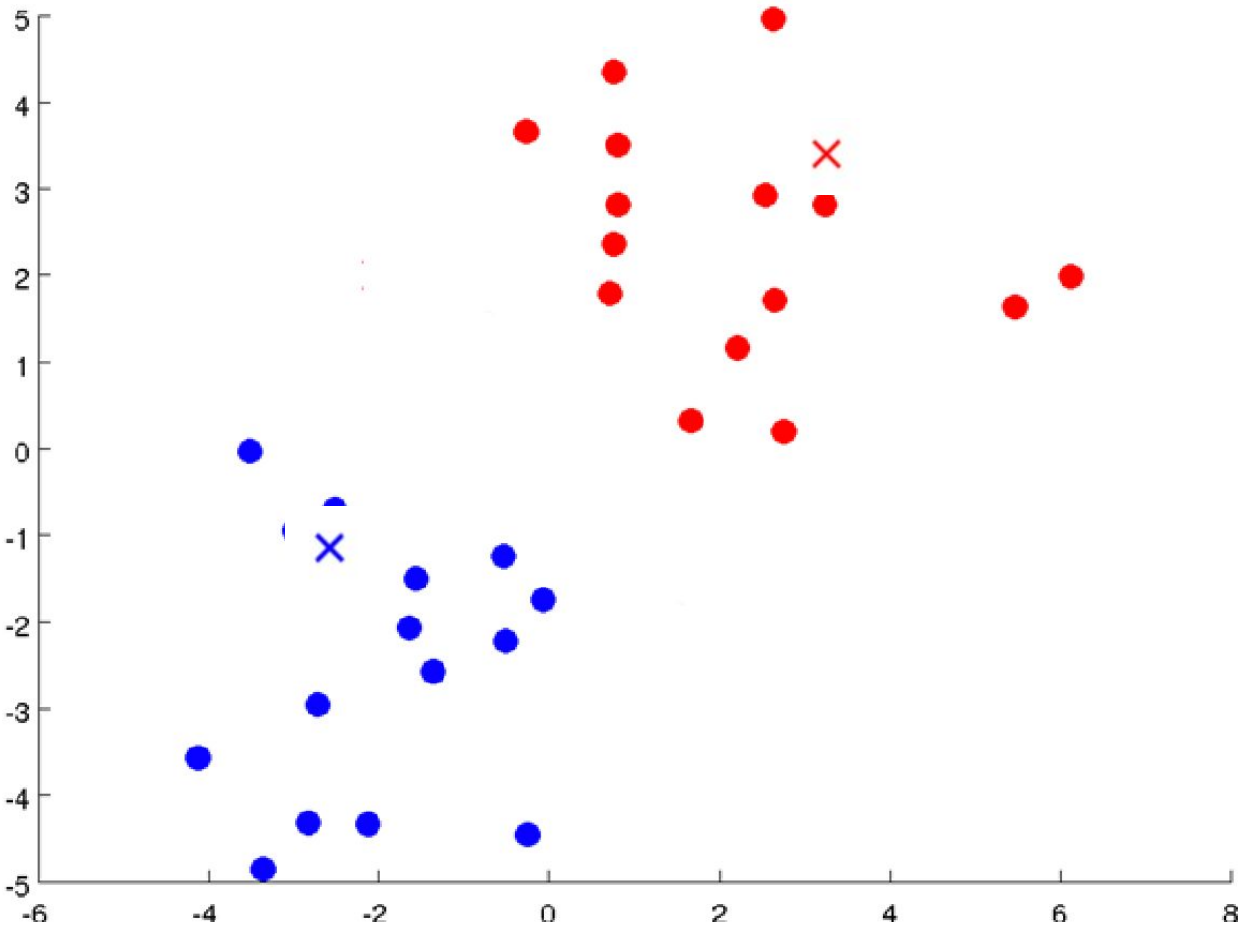
Assign each point depending on which centroid it’s closest to



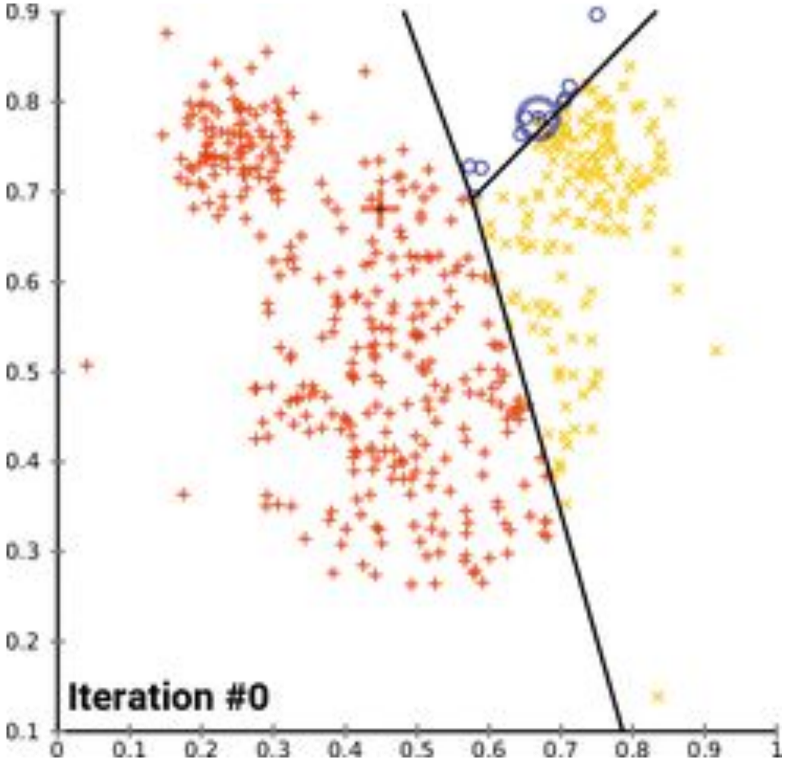
Move centroid to the average centroid of all points

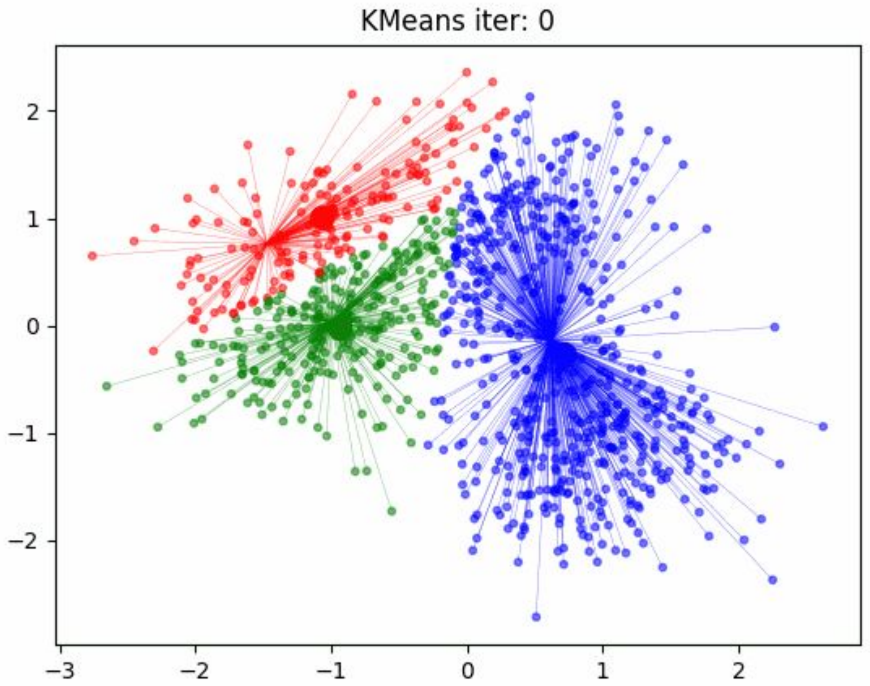


Iterate



Animations


Iteration #0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


KMeans iter: 0
2
1
0
−1
−2
−3
−2
−1
0
1
2

K-means algorithm

- Input:
  - K (number of clusters)
  - Training set $\{x^{(1)}, x^{(2)}, x^{(3)}, \cdots, x^{(m)}\}$

K-means algorithm

- Randomly initialize K cluster centroids $\mu_1, \mu_2, \cdots, \mu_K \in \mathbb{R}^n$

```
Repeat{
        for $i$ = 1 to $m$
                $c^{(i)}$ := index (from 1 to $K$) of cluster centroid
                closest to $x^{(i)}$
                                        Cluster assignment step

        for $k$ = 1 to $K$
                $\mu_k$ := average (mean) of points assigned to
        cluster $k$

                                        Centroid update step
}
```

K-means optimization objective

- $c^{(i)}$ = Index of cluster (1, 2, ... K) to which example $x^{(i)}$ is currently assigned
- μ_k = cluster centroid k ($\mu_k \in \mathbb{R}^n$)
- $\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned
- Optimization objective:

$$J(c^{(1)}, \cdots, c^{(m)}, \mu_1, \cdots, \mu_K) = \frac{1}{m}\sum_{i=1}^{m} \left\| x^{(i)} - \mu_{c^{(i)}} \right\|^2$$

$$\min_{\substack{c^{(1)},\cdots,c^{(m)} \\ \mu_1,\cdots,\mu_K}} J(c^{(1)}, \cdots, c^{(m)}, \mu_1, \cdots, \mu_K)$$

Example:

$x^{(i)} = 5$

$c^{(i)} = 5$

$\mu_{c^{(i)}} = \mu_5$

K-means algorithm

```
Randomly initialize $K$ cluster centroids $\mu_1, \mu_2, \cdots, \mu_K \in \mathbb{R}^n$

Repeat{
        for $i$ = 1 to $m$
                $c^{(i)}$ := index (from 1 to $K$) of cluster centroid closest to $x^{(i)}$

        for $k$ = 1 to $K$
                $\mu_k$ := average (mean) of points assigned to cluster $k$
}
```

Cluster assignment step

$$J(c^{(1)}, \cdots, c^{(m)}, \mu_1, \cdots, \mu_K) = \frac{1}{m} \sum_{i=1}^{m} \left\| x^{(i)} - \mu_{c^{(i)}} \right\|^2$$

Centroid update step

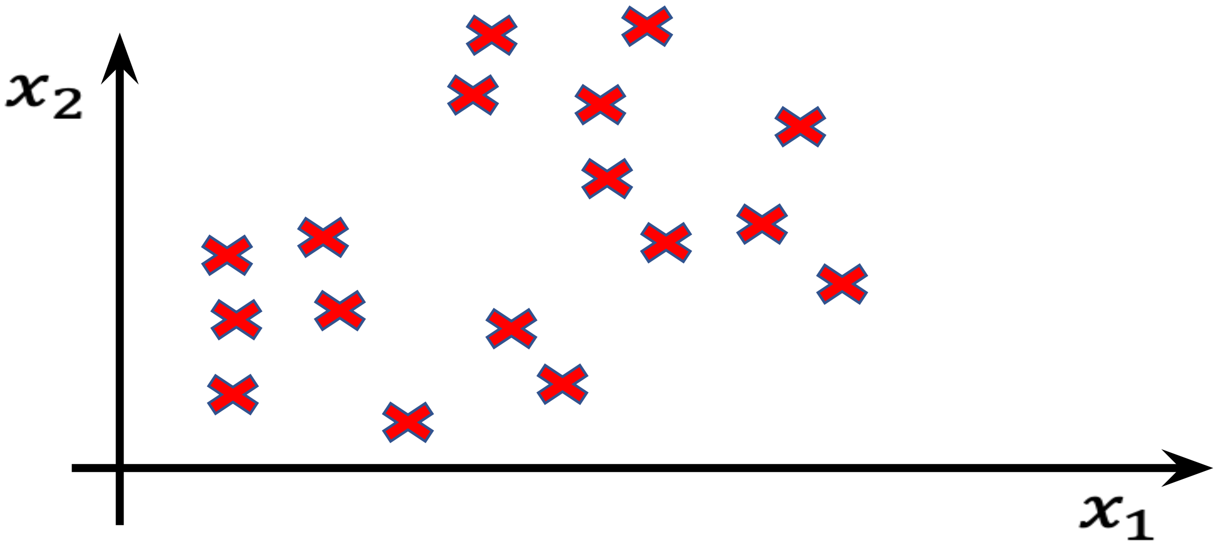
$$J(c^{(1)}, \cdots, c^{(m)}, \mu_1, \cdots, \mu_K) = \frac{1}{m} \sum_{i=1}^{m} \left\| x^{(i)} - \mu_{c^{(i)}} \right\|^2$$

K-means algorithm

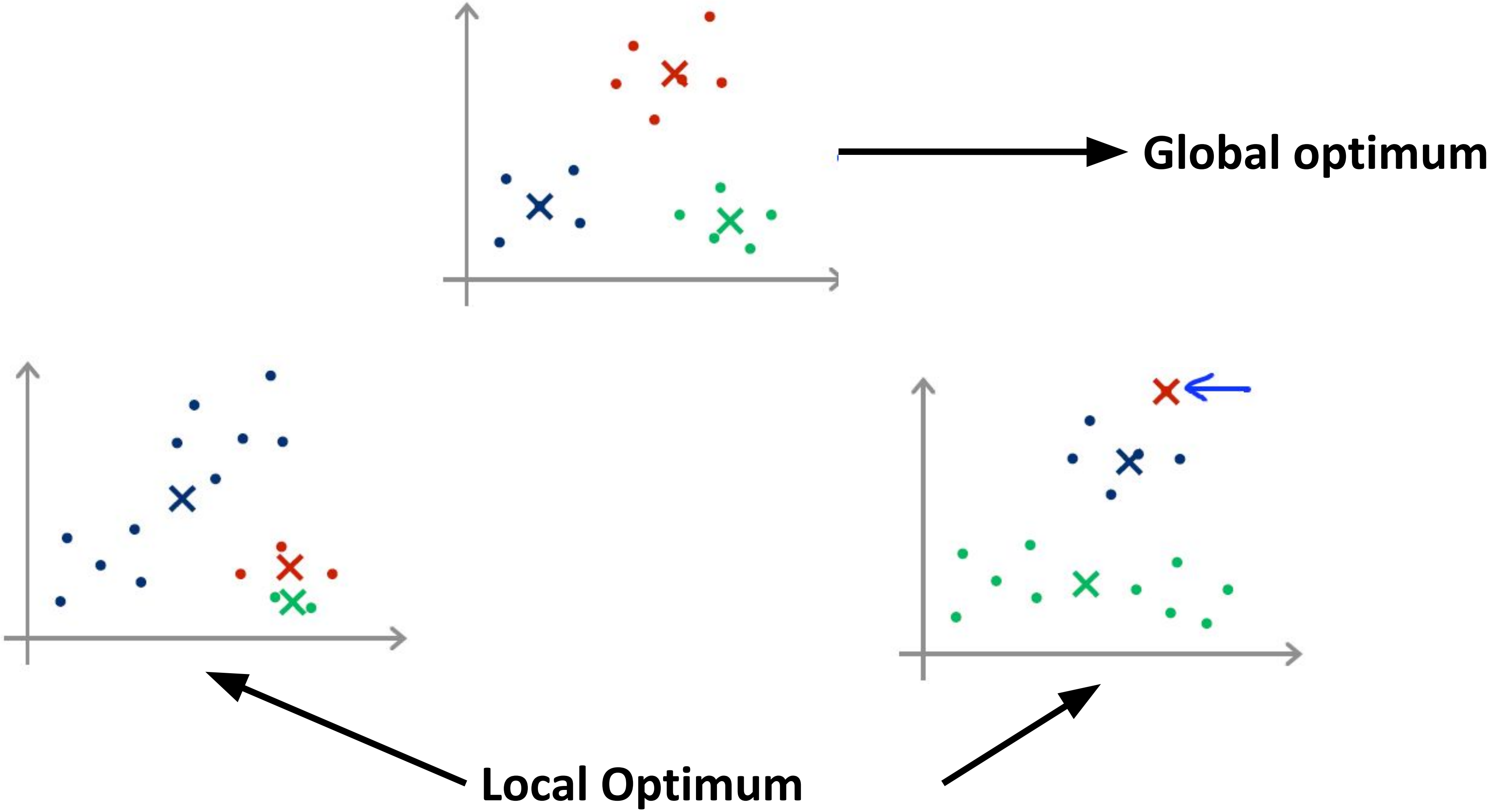
- Stopping criterion
  - no (or minimum) re-assignments of data points to different clusters
  - no (or minimum) change of centroids
  - minimum decrease in the optimization objective
  - Maximum iterations

Random initialization

- Randomly pick K training examples
- Set $\mu_1, \mu_2, \cdots, \mu_K$ equal to those K examples



Local Optimum



Global optimum

Local Optimum

Solutions to Local Optimum

- Multiple random initialization
- Furthest Point Heuristic

1) Multiple random initialization

```
For i = 1 to 100 {
        Randomly initialize K-means.
        Run K-means. Get $c^{(1)}, \cdots, c^{(m)}, \mu_1, \cdots, \mu_K$
        Compute the cost function (distortion)
                $J(c^{(1)}, \cdots, c^{(m)}, \mu_1, \cdots, \mu_K)$
}
```

Pick clustering that gave the lowest cost

$$J(c^{(1)}, \cdots, c^{(m)}, \mu_1, \cdots, \mu_K)$$

2) Furthest Point Heuristic

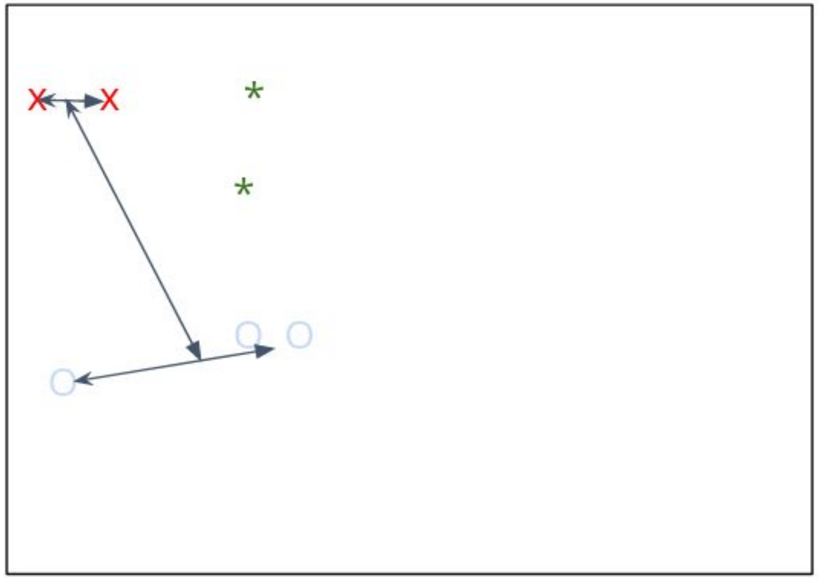
```
Choose $\mu_1$ arbitrarily (or at random)

For j = 2 to K
        Pick $\mu_j$ among data points $x^{(1)}, x^{(2)}, \cdots, x^{(m)}$ that is
        farthest from previously chosen $\mu_1, \mu_2, \cdots, \mu_{j-1}$
```

Slide credit: Maria-Florina Balcan

How to choose K?

- Cluster validity indices!!
  - An assessment of how "good" a given clustering is
  - Generally based on inter-cluster separation and intra-cluster spread
  - Ex: Davies-Bouldin Index, Silhouette Score, etc.

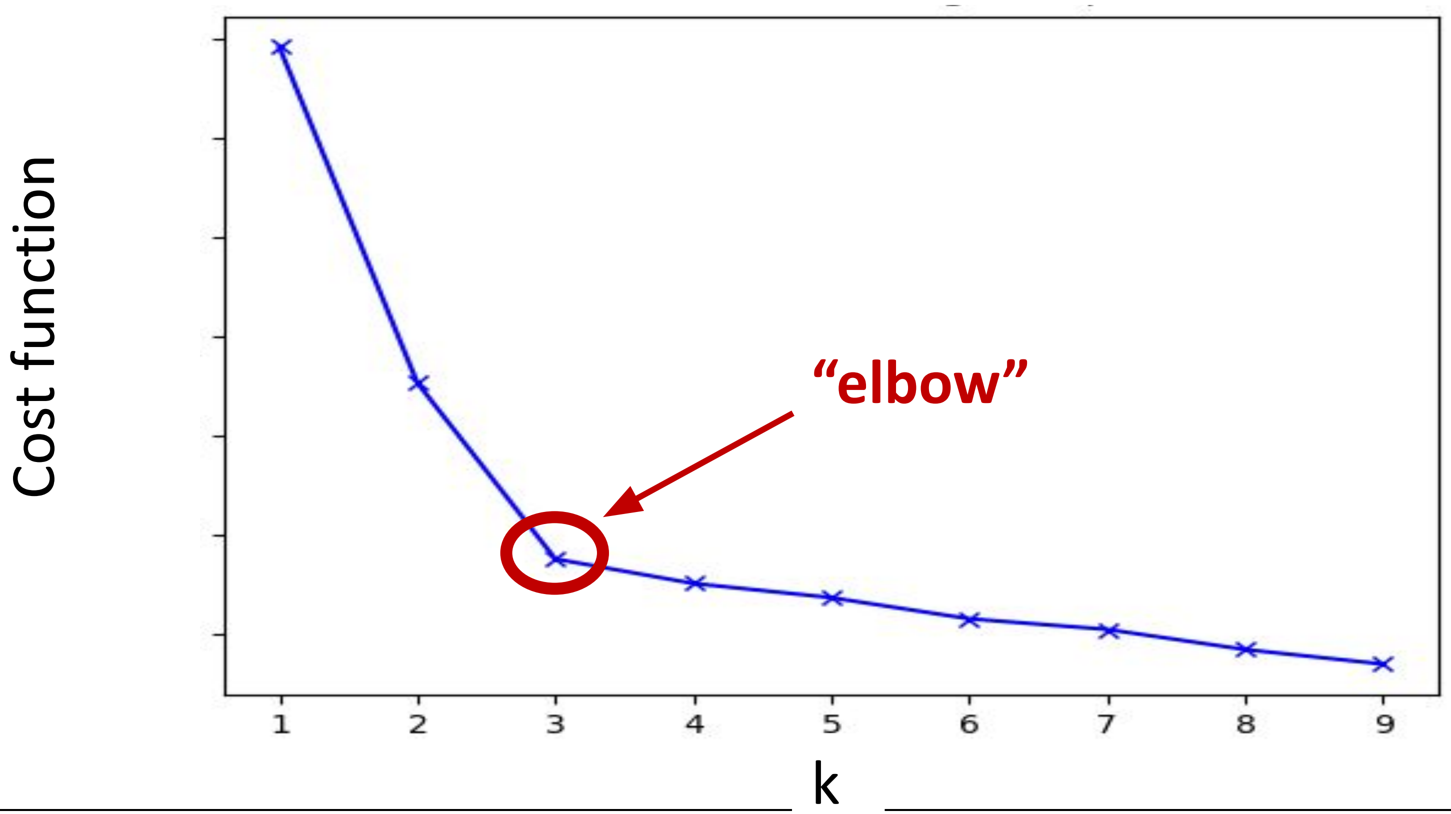
- Can also use visualizations to do it manually



Credit: Andrew Ng

How to choose K?

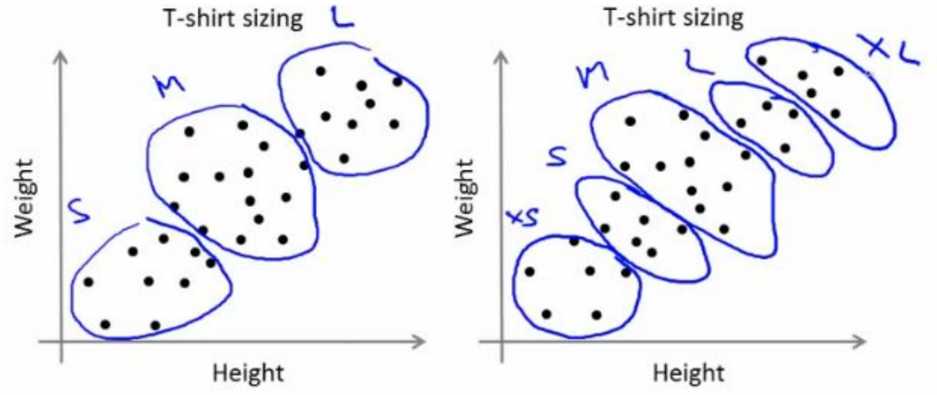
- 1) Elbow method



Credit: Andrew Ng

How to choose K?

- 2) Try multiple K and use performance from downstream tasks for selection (ie. "engineering decision")



K=3 or K=5?

- cost of making extra sizes vs. how well distributed the products are
- How important are those sizes though?

Credit: Andrew Ng

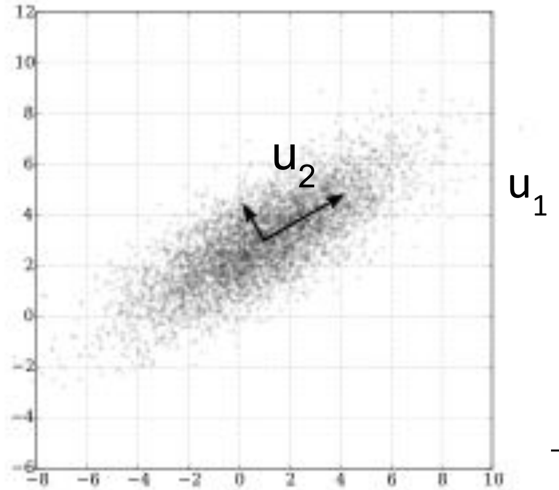
Summary

- k-means is the most popular clustering algorithm
- Pros
  - Simple: easy to understand and to implement
  - Efficient
    - Time complexity: O(tkn), where n is the number of data points, k is the number of clusters, and t is the number of iterations.
    - Since both k and t are small. k-means is considered a linear algorithm.
- Cons
  - The user needs to specify k
  - Sensitive to outliers
  - Not suitable for special data structure

Geometric interpretation of PCA

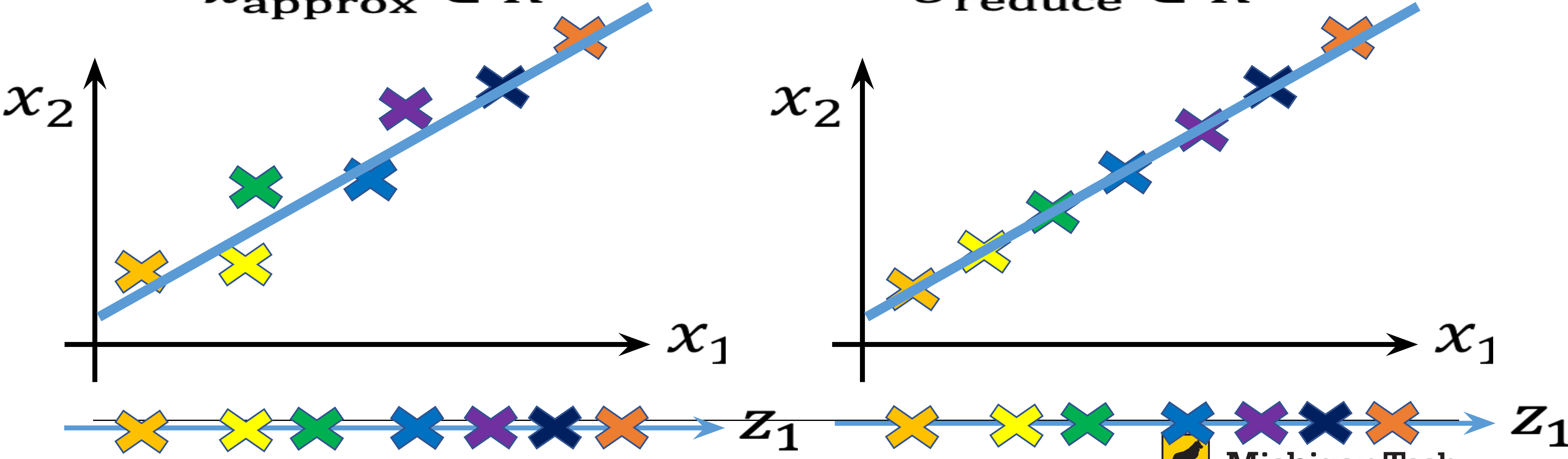
- PCA chooses the eigenvectors of the covariance matrix corresponding to the largest eigenvalues.
- The eigenvalues correspond to the variance of the data along the eigenvector directions.
- Therefore, PCA projects the data along the directions where the data varies most.

u_1: direction of max variance
u_2: orthogonal to u_1



Reconstruction from compressed representation

- Compression: $z^{(i)} = U_{\text{reduce}}^{\top} x^{(i)}$
- Reconstruction: $x_{\text{approx}}^{(i)} = U_{\text{reduce}} z^{(i)}$
- $x_{\text{approx}}^{(i)} \in R^{n}$ $U_{\text{reduce}} \in R^{n \times k}$



Application of PCA

- Compression
  - Reduce memory/disk needed to store data
  - Speed up learning algorithm
- Visualization (k=2, k=3)

- Bad use of PCA
  - Use it to prevent over-fitting
    - Fewer features □ less likely overfitting?
    - Might work OK, but not a good way to address overfitting
    - PCA does not look at y, may throw away some critical information
    - Use regularization instead.

Questions + Comments?