# UNSUPERVISED LEARNING

## Clustering

Unsupervised Learning: Introduction

* Supervised learning training set: {(*x*(1), *y*(1)), (*x*(2), *y*(2)), (*x*(3), *y*(3)), …, (*x*(*m*), *y*(*m*))}
* Unsupervised learning training set: {*x*(1), *x*(2), *x*(3), …, *x*(*m*)}
* Clustering algorithm: structure it finds is dataset grouped in clusters
* Applications of clustering

K-Means Algorithm

* Cluster assignment, move centroid
* Randomly initialize K cluster centroids (μ …)
* Repeat:
* Cluster assignment: c(i) := index of cluster centroid closest to x(i)? (value of k that minimizes the distance) min(k)||x(i) – μk||2
* Move centroid: μk := average(mean) of points assigned to cluster k
* Eliminate centroid with no points assigned, could randomly reinitialize
* Works for non-(well-)separated clusters 🡺 market segmentation

Optimization Objective

* Objective
* c(i): keep track of index/cluster of *x*(*i*)
* μk: cluster centroid location
* : cluster centroid of cluster to which *x*(*i*) has been assigned
* J: Distortion (cost function)
* Minimize J wrt c holding μ, minimize J wrt μ

Random Initialization

* K < m
* Randomly pick K training examples
* Set μ = examples
* Bad initialization would result in local optima: try multiple random initializations (low K)

Choosing Number of Clusters

* Most common to choose manually, looking at visualizations and/or algorithm output
* Elbow method, downstream purpose

# DIMENSIONALITY REDUCTION

## Motivation

Motivation I: Data Compression

Motivation II: Visualization

## Principal Component Analysis

PCA Problem Formulation

* Different from linear regression

PCA Algorithm

* Data preprocessing: mean normalization/feature scaling if needed
* Covariance matrix Σ: Sigma 🡪
* Eigenvectors of Σ: single value decomposition – svd more stable than eig
* Ureduce: z = UreduceT *x*

## Applying PCA

Reconstruction From Compressed Representation

* Xapprox = Ureduce . z

Choosing Number of Principal Components

* Minimize average square projection errors
* Total variation in data: average of (square) lengths of training examples/how far training examples are from origin (average)
* Ratio <= .01 (99% of variance retained)
* Get ration form matrix S ([U, S, V]):
* Run svd once

Advice for Applying PCA

* Supervised learning speedup
* define mapping by running only on training set, then apply to other sets
* Application
* compression
* reduce memory/disk storage
* speed up learning algorithm
* choose k - % of variance retained
* visualization
* k = 2 - 3
* Preventing overfitting: bad use of PCA (throwing away data/labels) – better to use regularization
* Run ML system without PCA first