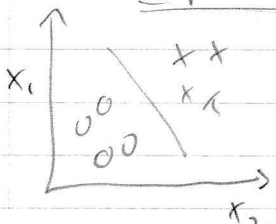
Machine Learning - Week 8 - Unsupervised Learning

T1 - Clustering

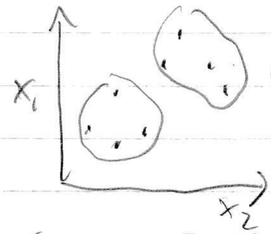
L1 - Unsupervised Learning Introduction

Recall: supervised learning



Training Set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots (x^{(m)}, y^{(m)})\}$

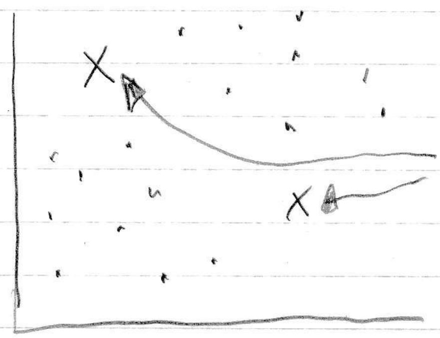
Unsupervised Learning



Training Set: $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(m)}\}$

Applications of Clustering
- Market segmentation
- Social Network Analysis
- Organize computing clusters
- Astronomical Data Analysis

L2 - K-Means Algorithm



Input: - K (number of clusters)
- Training set $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$, $x^{(i)} \in \mathbb{R}^n$ (drop $x_0 = 1$ convention)

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

Repeat {
  for $i = 1$ to m
    $c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$ — $\min_k \|x^{(i)} - \mu_k\|^2$ } cluster assignment step
  for $k = 1$ to K
    μ_K := average (mean) of points assigned to cluster k } Move centroid step

L3 - Optimization objective

Recall:

$c^{(i)}$ = index of cluster $(1, 2, \ldots, K)$ 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:

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

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

Distortion Algorithm

L4 - Random Initialization

Should have $K < m$

Randomly pick K training examples

Set $\mu_1, \ldots, \mu_K$ equal to these K examples

For i=1 to 100 {

Randomly initialize K means.

Run ~~K~~-Means. Get $c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K$

Compute cost function (distortion)

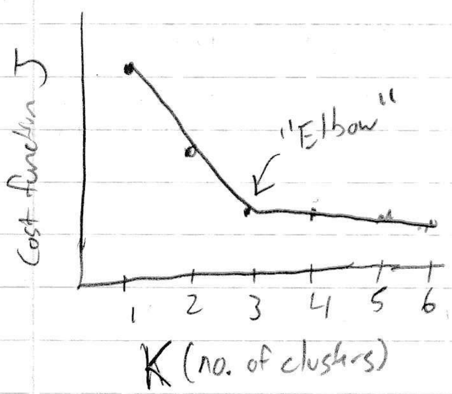
$\rightarrow J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$

}

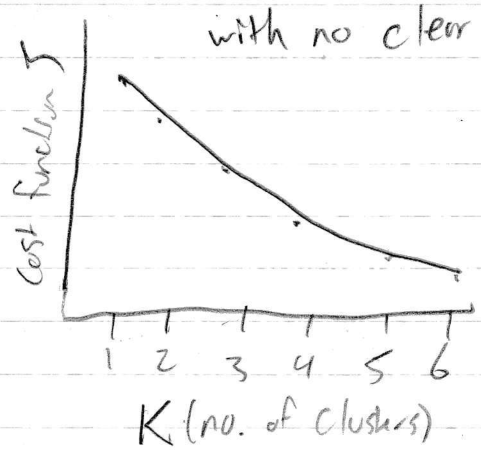
Pick clustering that gave the lowest cost $J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$

L5 - Choosing the number of clusters

Elbow method:



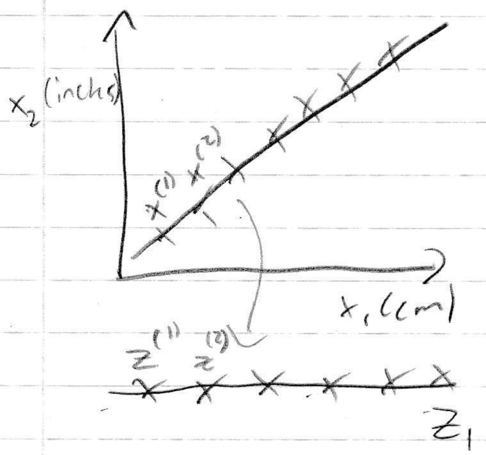
But could end up with no clear "elbow"



Sometimes, you're running K-means to get clusters to use for some later/downstream purpose. Evaluate K-means based on a metric for how well it performs for that later purpose.

T2 - Motivation for Dimensionality Reduction

L1 - Data Compression



Reduce data from 2D to 1D

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

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

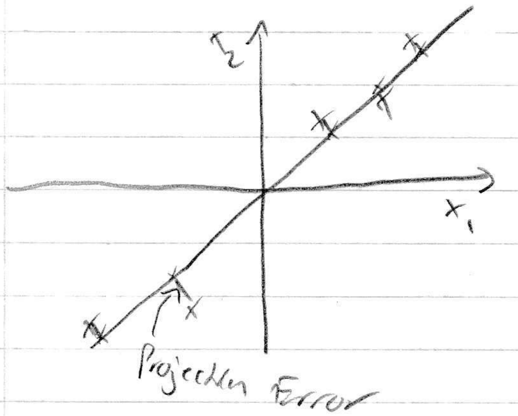
$\vdots$

L2 - Data Visualization

Watch video

T3 - Principal Component Analysis

L1 - Problem Formulation



$x \in \mathbb{R}^2$

Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error

General case: Reduce from n-dimension to K-dimensions: Find K vectors $u^{(1)}$, $u^{(2)}, \ldots, u^{(K)}$ onto which to project the data, so as to minimize the projection error.

L2 - Principal Component Analysis Algorithm

Data preprocessing

Training set: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$

Preprocessing (feature scaling/mean normalization):

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

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

If different features on different scales (e.g. x_1 = size of house, x_2 = # of bedrooms) scale features to have comparable range of values

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

Principle Component Analysis (PCA) algorithm

Reduce data from n-dimensions to K-dimensions.

Compute "covariance matrix":

$$\text{sigma} \rightarrow \Sigma = \frac{1}{m}\sum_{i=1}^{n} \underbrace{(x^{(i)})}_{n\times 1}\underbrace{(x^{(i)})^T}_{1\times n}$$

$\leftarrow$ Vectorized $= \left(\frac{1}{m}\right) \cdot X' \cdot X$

svd $\rightarrow$ Singular Value decomposition.

Compute "eigenvectors" of matrix Σ:

[U, S, V] = svd(Sigma); $\leftarrow$ Octave code

$\uparrow$ $n \times n$ matrix

$$U = \Bigg[\underbrace{u^{(1)} \; u^{(2)} \; u^{(3)}}_{K} \; \dots \; u^{(n)}\Bigg] \qquad U \in \mathbb{R}^{n\times n}$$

$$x \in \mathbb{R}^n \rightarrow z \in \mathbb{R}^K$$

$$z = \underbrace{\begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(K)} \\ | & | & & | \end{bmatrix}}_{n\times K}^{T} X = \underbrace{\underbrace{\begin{bmatrix} — (u^{(1)})^T — \\ \vdots \\ — (u^{(K)})^T — \end{bmatrix}}_{K\times n} \underbrace{X}_{n\times 1}}_{K\times 1}$$

$\uparrow$ Ureduce

T4 - Applying PCA

L1 - Reconstruction from Compressed Representation

Recall

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

$$z \in \mathbb{R} \to x \in \mathbb{R}^2$$

$$x_{approx} = U_{reduce}' z$$

L2 - Choosing the Number of Principal Components

Choosing k (number of principal components)

Average squared projection error: $\frac{1}{m}\sum_{i=1}^{m} \| x^{(i)} - x_{approx}^{(i)} \|^2$

Total variation in the data: $\frac{1}{m}\sum_{i=1}^{m} \| x^{i} \|^2$

Typically, choose k to be smallest value so that

$$\frac{\frac{1}{m}\sum_{i=1}^{m} \| x^{(i)} - x_{approx}^{(i)} \|^2}{\frac{1}{m}\sum_{i=1}^{m} \| x^{(i)} \|^2} \leq 0.01 \quad (1\%)$$

"99% of variance is retained"

Algorithm:
Try PCA with $k=1$
Compute $U_{reduce}, z^{(1)}, z^{(2)}, \ldots, z^{(m)}, x_{approx}^{(1)}, \ldots, x_{approx}^{(m)}$
Check if this is ≤ 0.01? If not, increase k and repeat

} very inefficient

But with $[U, S, V] = svd(Sigma)$

$$S = \begin{bmatrix} s_{11} & & 0 \\ & s_{22} & \\ 0 & & \ddots \; s_{nn} \end{bmatrix}$$

For given k, this $= 1 - \frac{\sum_{i=1}^{k} s_{ii}}{\sum_{i=1}^{n} s_{ii}}$, so check if ≤ 0.01

Summary:

[U, S, V] = svd(Sigma)

Pick smallest value of K for which

$$\frac{\sum_{i=1}^{K} S_{ii}}{\sum_{i=1}^{m} S_{ii}} \geq 0.99 \quad \text{(99\% of variance retained)}$$

L3 - Advice for applying PCA

Supervised learning speedup.

$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$

Extract inputs:

Unlabeled dataset: $x^{(1)}, x^{(2)}, \ldots, x^{(m)} \in \mathbb{R}^{10000}$

$\downarrow$ PCA

$z^{(1)}, z^{(2)}, \ldots, z^{(m)} \in \mathbb{R}^{1000}$

New training set:

$(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \ldots, (z^{(m)}, y^{(m)})$

$$h_\theta(z) = \frac{1}{1+e^{-\theta^T z}}$$

Note: Mapping $x^{(i)} \rightarrow z^{(i)}$ should be defined by running PCA only on the training set. This mapping can be applied as well to the examples $x_{cv}^{(i)}$ and $x_{test}^{(i)}$ in the cross validation and test sets.

Bad use of PCA: To prevent overfitting

Use $z^{(i)}$ instead of $x^{(i)}$ to reduce the number of features to $K < n$.

Thus, fewer features, less likely to overfit.

This might work ok, but isn't a good way to address overfitting. Use regularization instead.