Contents

I Week 8 **1**
0.1 Supervised Learning vs. Unsupervised Learning . . . 2
0.2 Clustering algorithm: K-means . . . 2
0.2.1 Principle . . . 2
0.3 Optimization Objective . . . 3
0.4 Random Initialization of cluster centroid . . . 4
0.5 Choose the Number of Clusters K . . . 5
0.6 Dimensionality Reduction . . . 5
0.6.1 Data Visualization . . . 5
0.6.2 Principal Component Analysis (PCA) algorithm . . . 6
0.6.3 Choose the number of principal components . . . 8
0.7 Advice for applying PCA . . . 8
0.7.1 applying PCA for supervised learning speedup . . . 8
0.7.2 Application of PCA . . . 9

Appendices **10**
.1 Cost function versus iteration . . . 11

Clustering algorithm: Unsupervised Learning

March 26, 2018

Part I

Week 8

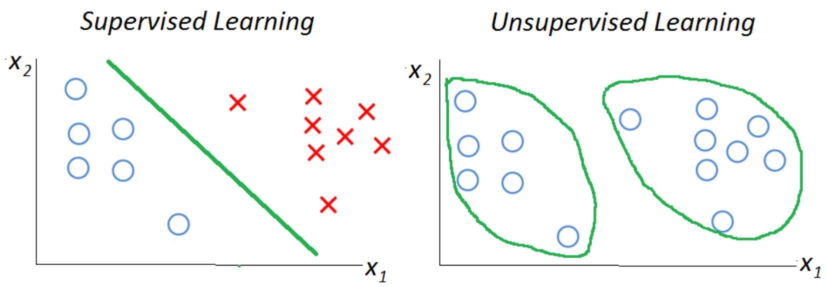
0.1 Supervised Learning vs. Unsupervised Learning

- **Supervised Learning** $\rightarrow$ Training set is of the form:

$$\left[(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), ...(x^{(m)}, y^{(m)})\right]$$

- **Unsupervised Learning** $\rightarrow$ Training set is of the form (no labels y):

$$\left[x^{(1)}, x^{(2)}, ..., x^{(m)}\right]$$



0.2 Clustering algorithm: K-means

0.2.1 Principle

$K-$means is a iterative algorithm:

1. inputs:
   - K: number of clusters that we want to find in the data set
   - Training set: $\left[x^{(1)}, x^{(2)}, ...x^{(m)}\right]$ where $x^{(i)} \in \mathbb{R}^n$ (by convention, $x_0 = 1$ is dropped).
2. Randomly initialize (set the location) of K cluster centroids: $\mu_1, \mu_2, \mu_3, ..., \mu_K \in \mathbb{R}^n$.
3. Algorithm:

```
Repeat {
          1) Cluster assignment step:
          Go thru each example and assign them to the closest
          cluster centroid
          for i = 1 to m:
                    c[i] := index (1 to K) of cluster centroid closest to x[i]
          end

          2) Move centroid step:
          Take the mean of all points associated with a cluster
          and move the centroid to that new position.
          for i = 1 to K:
                    mu[k] := mean of points assigned to cluster k
          end
          Reiterate until convergence==>
```

4. Mathematically:
   - In the cluster assignment step, c[i] ($c^{(i)}$) is the value of k that minimizes $||x^{(i)} - \mu_k||^2$ (J is minimized with respect to all $c^{(i)}$, while holding all μ_k fixed)
   - In the 'move centroid step', J is minimized with respect to all μ_k, and holding all $c^{(i)}$ fixed

- Example: Let's consider $x^{(1)}$, $x^{(5)}$, $x^{(6)}$, $x^{(10)}$, and assign them to cluster 2 (μ_2). Therefore: $c^{(1)} = 2$, $c^{(5)} = 2$, $c^{(6)} = 2$, $c^{(10)} = 2$. The new position of μ_2 is: $\mu_2 = \frac{1}{4}\left[x^{(1)} + x^{(5)} + x^{(6)} + x^{(10)}\right]$ ($\in \mathbb{R}^n$)

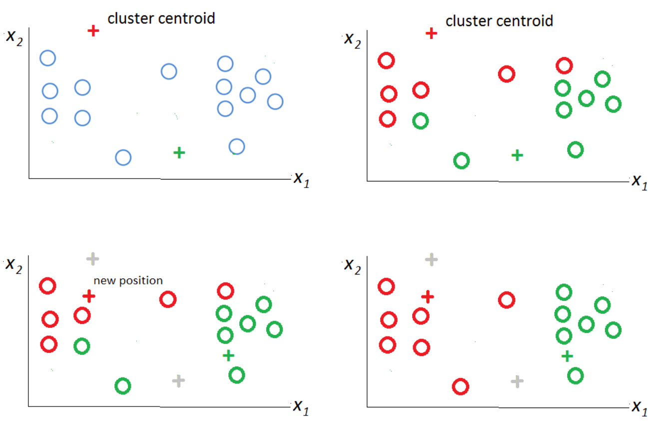


Figure 1: Illustration of the K-means algorithm

- Note: if a cluster cannot find any points to assign, by convention the cluster is eliminated.

0.3 Optimization Objective

- $c^{(i)}$ is the index of cluster (1, 2,...,K) to which example $x^{(i)}$ is currently assigned.

- μ_k = cluster centroid k ($\mu_k \in \mathbb{R}^n$ with $k \in \{1, 2, ..., K\}$)
- $\mu_{c^{(i)}}$ = cluster centroid of the cluster to which example $x^{(i)}$ has been assigned
  $x^{(1)} \rightarrow 5 \Rightarrow C^{(i)} = 5 \rightarrow \mu_{c^{(i)}} = \mu_5$

- **Optimization objective** :

$$J(c^{(1)}, ...c^{(m)}, \mu_1, \mu_2, ..., \mu_k) = \frac{1}{m}\sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}||^2 \tag{1}$$

  This cost function is sometimes called 'Distorsion'.

- **Cost function minimization**:

$$min_{\substack{c^{(1)},...c^{(m)} \\ \mu_1,\mu_2,...,\mu_k,}} J\left(c^{(1)}, ...c^{(m)}, \mu_1, \mu_2, ..., \mu_k\right) \tag{2}$$

0.4 Random Initialization of cluster centroid

- Should have $K < m$
- Randomly pick K **training examples** and set $\mu_1, \mu_2, ...\mu_K$ equals to those K examples: $\mu_1 = x^{(i)}$, $\mu_2 = x^{(j)}$ where i, j random.

- Local optimum: Depending on the initialization, $K-$means can converge to different solution (the distorsion function could be trapped in local optimum). One method is to try multiple random initialization

```
for i = 1 to 100
        randomly initialize K—means
        Run $K—$means and get c[1], ...c[m], mu[1], ...,mu[k] \\
        Compute cost  function distorsion
end
Pick clustering that gives the lowest cost.
```

Additional notes:

- for small Nbrs of clusters, doing multiple Random Initialization helps in giving good clustering.
- If $K > 10$, multiple clustering is less likely to give better solution.

0.5 Choose the Number of Clusters K

One known method to determine an appropriate cluster size is the 'Elbow Method', where the optimum cluster size is given by the Elbow of J versus k. However, in reality, the curve $J(k)$ rarely show a clear a elbow feature.

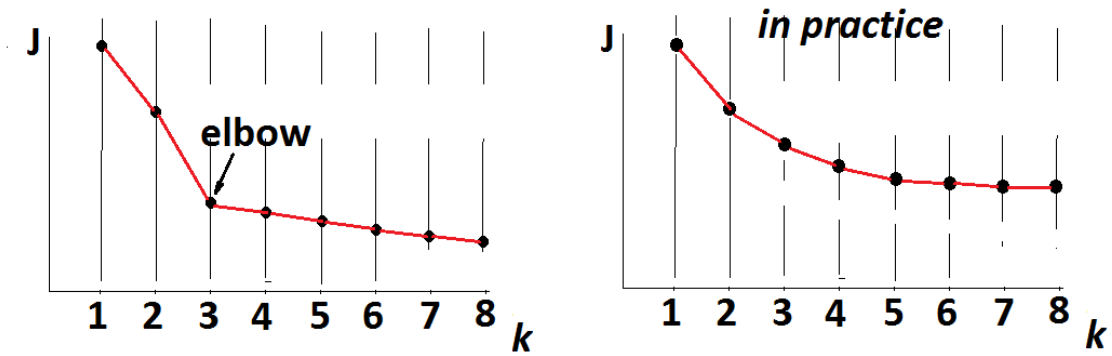


Figure 2: Illustration of the K-means algorithm

0.6 Dimensionality Reduction

Dimensionality reduction enables to run algorithm more quickly. For highly correlated dimensions, it is useful to perform Dimensionality Reduction.

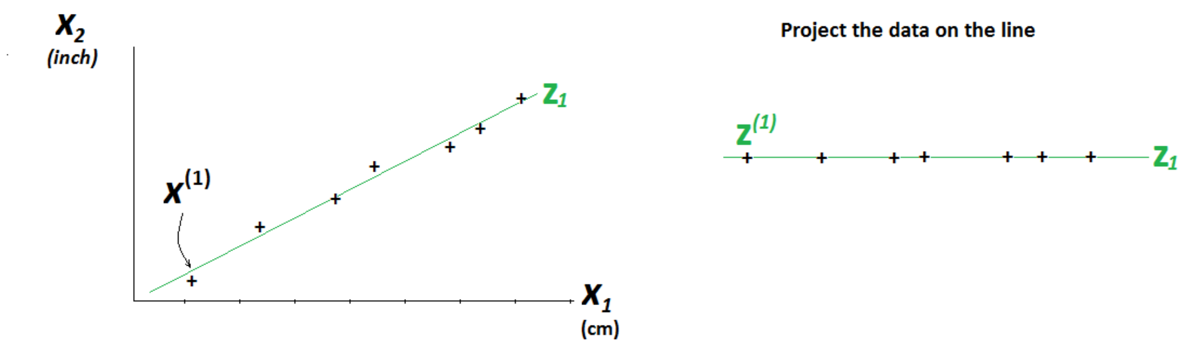


Figure 3: Illustration of dimensionality reduction 2D to 1D.

- Dimensiontionality reduction 2D to 1D:

$$x^{(1)}, ..., x^{(m)} \in \mathbb{R}^2 \Rightarrow z^{(1)}, ..., z^{(m)} \in \mathbb{R} \tag{3}$$

- Dimensiontionality reduction 3D to 2D:

$$x^{(1)}, ..., x^{(m)} \in \mathbb{R}^3 \Rightarrow z^{(1)}, ..., z^{(m)} \in \mathbb{R}^2 \tag{4}$$

0.6.1 Data Visualization

Dimensionality reduction from n-dimension to 2 dimensions is performed to visualize the data.

0.6.2 Principal Component Analysis (PCA) algorithm

PCA is used for dimensionality reduction.

1. Training set: $x^{(1)}$, ..., $x^{(m)}$
2. Always normalization:

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)} \tag{5}$$

   and replace each $x_j^{(i)} \leftarrow (x_j^{(i)} - \mu_j)$

3. Feature scaling
   and replace each $x_j^{(i)} \leftarrow \frac{(x_j^{(i)} - \mu_j)}{s_j}$, where s_j is the standard deviation.
4. PCA tries to find a lower dimensional surface/line onto which the projection error is minimized.

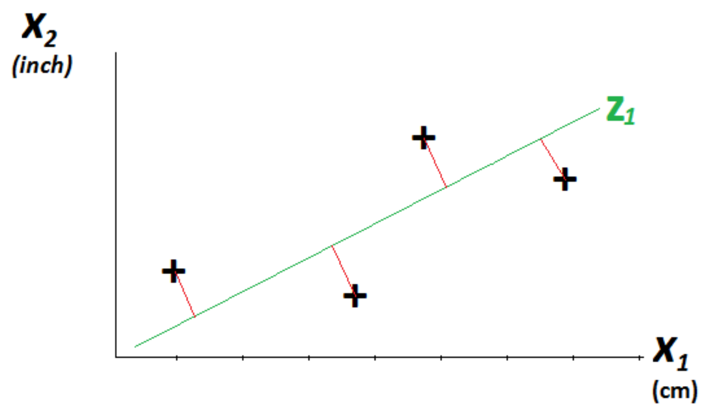


Figure 4

Formulation

- Dimensiontionality reduction n-D to k-D: Find k direction vectors $(u^{(1)}, ..., u^{(k)})$ onto which to project the data, so as to minimize the projection error.

- PCA is not linear regression

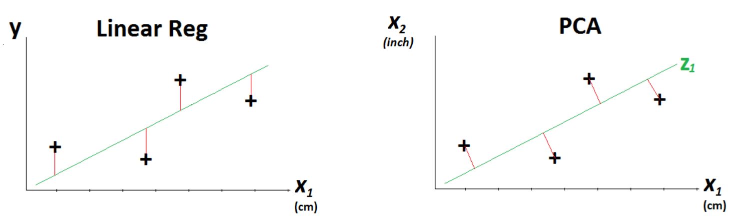


Figure 5

- PCA algorithm

1. Compute 'covariance matrix' $\Sigma = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)})(x^{(i)})^T$, where $x^{(i)}$ is $(n \times 1)$, and Σ is $(n \times n)$

2. Compute 'eigenvectors' of matrix Σ

```
[U, S, V] =svd(sigma)
```

'svd' (singular value decomposition) is a MatLab function outputing 3 matrices.

– U is a $n \times n$ matrix:

$$U = \begin{bmatrix} \vdots & \vdots & \vdots & & \vdots \\ u^{(1)} & u^{(2)} & u^{(3)} & & u^{(n)} \\ \vdots & \vdots & \vdots & & \vdots \end{bmatrix} \tag{6}$$

If we want to reduce the data to k dimensions, we take the 1st k column vectors $(u^{(1)}, u^{(2)}, .., u^{(k)}) = U_{\text{reduce}}(n \times k)$.

$$z^{(i)} = U_{\text{reduce}}^T x^{(i)} \tag{7}$$

where $z^{(i)} \in \mathbb{R}^k$

– Vectorized implementation of Σ with $X = \begin{bmatrix} ---(x^{(1)})^T--- \\ ----- \\ ------ \\ ---(x^{(m)})^T--- \end{bmatrix}$

```
Sigma = 1/m * (X' * X)
[U, S, V] = svd(Sigma)
Ureduce = U(:,1:k)
z = Ureduce' * X
```

Remember that for PCA $x^{(i)} \in \mathbb{R}^n$ (without $x_0 = 1$ by convention)

Reconstruction from compressed representation

$$x_{\text{approx}} = U_{\text{reduce}} * z \tag{8}$$

U_{reduce} is $(n \times k0$ and z $(k \times 1) \Rightarrow x_{\text{approx}}$ is $\mathbb{R}^n$

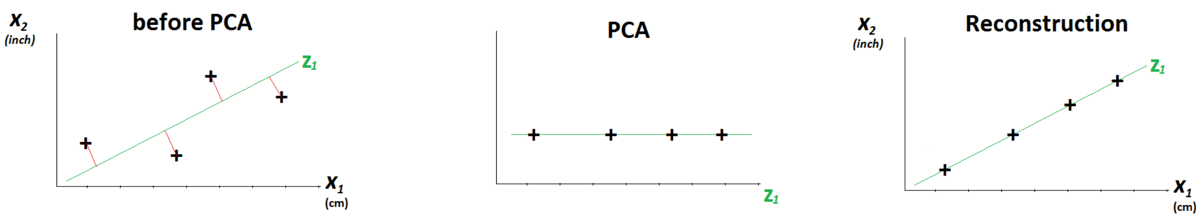


Figure 6

0.6.3 Choose the number of principal components

- PCA minimizes Average Squared Projection Error:

$$\frac{1}{m}\sum_{i=1}^{m}||x^{(i)} - x^{(i)}_{\text{app}}||^2 \tag{9}$$

- Total variation in the data:

$$\frac{1}{m}\sum_{i=1}^{m}||x^{(i)}||^2 \tag{10}$$

Typically, choose k to be the smallest value so that:

$$\frac{\frac{1}{m}\sum_{i=1}^{m}||x^{(i)} - x^{(i)}_{\text{app}}||^2}{\frac{1}{m}\sum_{i=1}^{m}||x^{(i)}||^2} \leq 0.01(or 0.05, 0.1) \tag{11}$$

which is interpreted as '99% (95, 90) of the variance is retained'.

- Implementation

```
Iteration
        Try PCA with k =1,
                Compute Ureduce, z[1], z[2], ..., z[m]
                                 xapp[1], xapp[2], ..., xapp[m]
                Check if Ratio of AverSquareProjection/TotalVariance <= 0.01
                k = k+1
```

This can be achieved with S-vector output by svd MatLab function. S is a diagonal matrix $n \times n$:

$$S = \begin{bmatrix} S_{11} & 0 & 0 & .. & 0 \\ 0 & S_{22} & 0 & .. & 0 \\ . & . & . & .. & . \\ . & . & . & .. & . \\ . & . & . & .. & . \\ 0 & . & . & .. & S_{nn} \end{bmatrix} \tag{12}$$

For a given value of k:

$$1 - \frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \leq 0.01 \tag{13}$$

for 99% of variance retained (which is a measure of the projection error).

0.7 Advice for applying PCA

0.7.1 applying PCA for supervised learning speedup

- inputs Assume a dataset $\{(x^{(1)}, y^{(1)}), ..., (x^{(m)}, y^{(m)})\}$.
  Let's say $x^{(i)} \in \mathbb{R}^{10000}$ (which is often found in computer vision where images coudl be (100x100)px.

- Extract inputs:

  - Unlabeled dataset $\{x^{(1)}, ..., x^{(m)}\} \in \mathbb{R}^{10000}$.
  - Apply PCA
  - $\{z^{(1)}, ..., z^{(m)}\}$

- New training set: $\{(z^{(1)}, y^{(1)}), ..., (z^{(m)}, y^{(m)})\}$

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

0.7.2 Application of PCA

- Compression application:

  - Reduce memory/disk need to store data
  - speed up learning algorithm

  Choose k using % of variance retained.

- Visualization application: k is typically 2 or 3

Appendices

.1 Cost function versus iteration

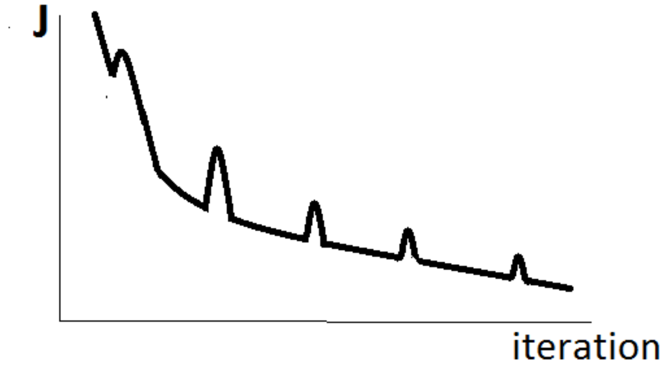


Figure 7: J vs. iteration : It's not possible for the cost function to sometime increase. There must be a bug in the code.