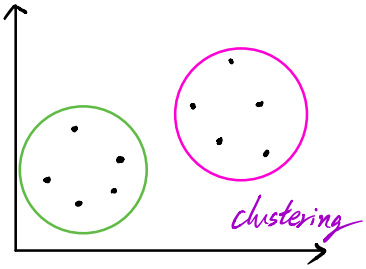
Unsupervised Learning



unlabeled data:

training set: $\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$

K-means: a clustering algorithm

input :- k (no. of clusters)

- training set $\{x^{(1)}, x^{(2)}, x^{(3)}, \cdots, x^{(m)}\}$ $\quad x^{(i)} \in \mathbb{R}^n$

optimization objective

$c^{(i)}$ = index of cluster $(1, 2, \cdots, K)$ to which example $x^{(i)}$ is currently assigned

μ_k = cluster centroid k $(\mu_k \in \mathbb{R}^n)$

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

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

step: randomly initialize k cluster centroids $\mu_1, \mu_2, \cdots, \mu_k \in \mathbb{R}^n$

Repeat {

cluster assignment step — minimize $J(\cdots)$ $c^{(1)}, \cdots, c^{(m)}$ holding $\mu_1, \cdots, \mu_k$ fixed

for $i = 1$ to m

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

$\min_k \| x^{(i)} - \mu_k \|^2$ → $c_i := k$

move centroid — minimize $J(\cdots)$ $\mu_1, \cdots, \mu_k$

for $k = 1$ to K

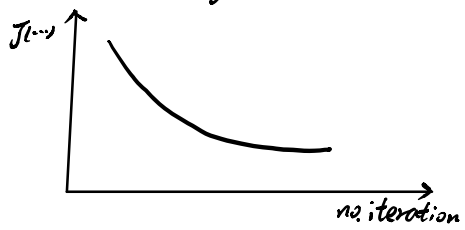
μ_k := average (mean) of points assigned to cluster k

$\mu_k := \frac{1}{t} \sum_{\substack{i=1 \\ c_i = k}}^{m} x^{(i)}$ $\quad t$ = no. of $c_i = k$, if $t = 0$, eliminate a cluster

$\mu_k \in \mathbb{R}^n$

}

the process of iteration always decrease $J(\cdots)$



random initialization:

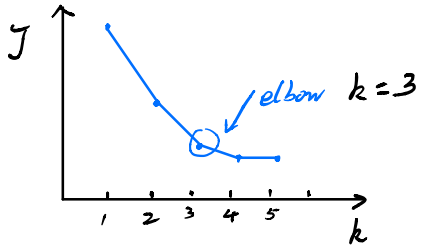
should have $k<m$, randomly pick k training example, set $\mu_1, \cdots, \mu_k$ equal to these k examples, random initialization may reach local optima result, to solve this: using many times of random initialization:

```
for i=1 to 100 {
    random initidize μ1, μ2, ···, μk
    run k-means
    compute J(···)
}
```

pick clustering that gave the lowest cost $J(\cdots)$

Choose the numbers of clusters $-k$:

elbow method:



plot J-K graph and find elbow point whese J decrease speed apporently slow down.

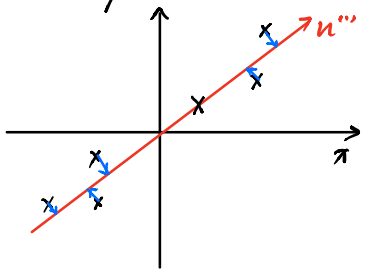
Principal Component Analysis (PCA)

reduce data from high dimension to low dimension.

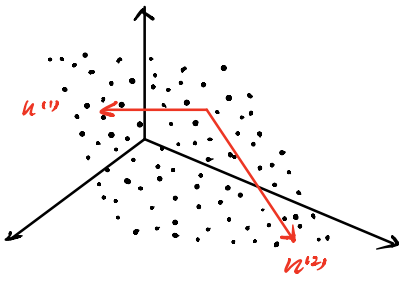
$$X = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(m)} \end{bmatrix} \xrightarrow{PCA} Z = \begin{bmatrix} z^{(1)} \\ z^{(2)} \\ \vdots \\ z^{(m)} \end{bmatrix}$$

$x^{(i)} \in \mathbb{R}^n$ $\quad$ $z^{(i)} \in \mathbb{R}^k$

PCA problem formulation:



$m=2$, $k=1$ $\quad$ 2D → 1D

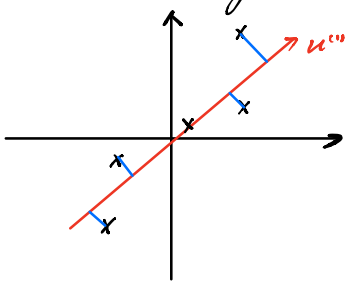


$m=3$, $k=2$ $\quad$ 3D → 2D

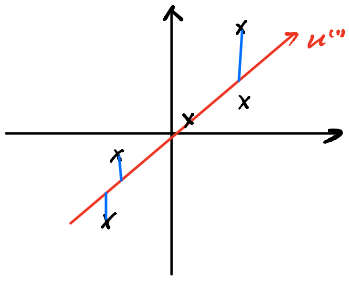
reduce from 2D to 1D: find a direction $u^{(1)} \in \mathbb{R}^n$ onto which to project the data so as to minimize the projection error.

reduce from nD to kD. find k vectors $u^{(1)}, u^{(2)}, \dots, u^{(k)}$ onto which to project the data so as to minimize the projection error.

PCA is not linear regression:



PCA



linear regression

blue line: the error

PCA Algorithm

1. training set: $x^{(1)}, x^{(2)}, \cdots, x^{(m)}$, $\quad n$

   reduce data from n-dimension to k-dimension

   $(n > k)$

2. preprocessing (feature scaling/mean normalization):

   $\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$ $\quad$ ensure every feature has zero mean

   $x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{s_j}$ $\quad$ s_j can be $\max x_j - \min x_j$

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

3. compute covariance matrix

   $$\Sigma = \frac{1}{m} \sum_{i=1}^{n} \underset{n \times 1}{(x^{(i)})} \underset{1 \times n}{(x^{(i)})^T} \qquad \Sigma \in \mathbb{R}^{n \times n} \qquad \Sigma = \text{sigma}$$

4. compute eigenvectors of matrix

   octave: `[u,s,v] = svd(sigma);`

   $$U = \underbrace{\begin{bmatrix} | & | & | & & | \\ u^{(1)} & u^{(2)} & u^{(3)} & \cdots & u^{(n)} \\ | & | & | & & | \end{bmatrix}}_{\text{select } u^{(1)} \sim u^{(k)}} \in \mathbb{R}^{n \times n} \qquad u^{(i)} \in \mathbb{R}^n$$

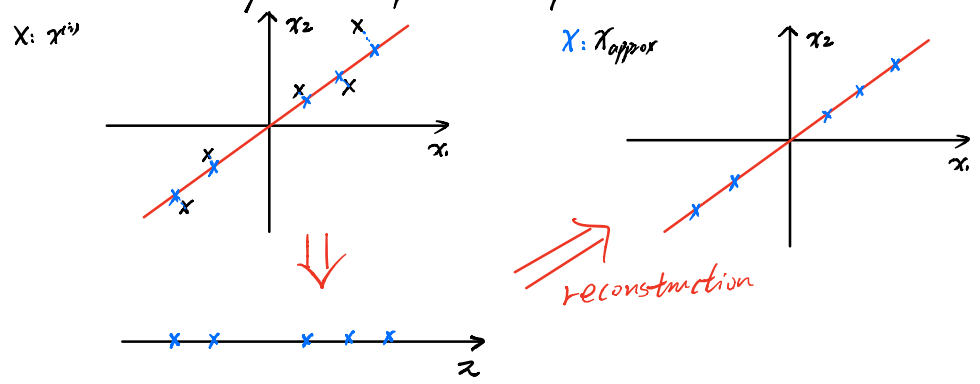
   octave: `Ureduce = U(:,1:k);`

   $$U_{reduce} = \begin{bmatrix} | & | & | & & | \\ u^{(1)} & u^{(2)} & u^{(3)} & \cdots & u^{(k)} \\ | & | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times k}$$

   octave: `z = Ureduce' * x;`

   $$z^{(i)} = \underset{k \times n}{(U_{reduce})^T} \underset{n \times 1}{x^{(i)}} \in \mathbb{R}^k$$

Reconstruction from compressed representation:



$z = (U_{reduce})^T x$ $\quad$ reconstruction $\quad$ $x_{approx}^{(i)} = \underset{n \times k}{U_{reduce}} \cdot \underset{k \times 1}{z^{(i)}}$

$z \in \mathbb{R}^k$ $\quad$ $x_{approx} \in \mathbb{R}^n$

choose the number of principle components — k

try PCA with $k = 1, 2, 3, \cdots$ }

compute $U_{reduce}, z^{(1)}, z^{(2)}, \cdots, z^{(m)}, x_{approx}^{(1)}, \cdots, x_{approx}^{(m)}$

check if $\dfrac{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)}\|^2} \le 0.01$

} $\Leftrightarrow$

Octave:

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

$$S = \begin{bmatrix} S_{11} & & & & \\ & S_{22} & & & \\ & & S_{33} & & \\ & & & \ddots & \\ & & & & S_{nn} \end{bmatrix}$$

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

choose the largest k. (99% of variance retained)

advice for applying PCA

training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$

↓ extract

unlabeled dataset: $x^{(1)}, x^{(2)}, \dots, x^{(m)} \in \mathbb{R}^n$

↓ PCA

$z^{(1)}, z^{(2)}, \dots, z^{(m)} \in \mathbb{R}^k$

new training set: $\{(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \dots, (z^{(m)}, y^{(m)})\}$

Note: map $x^{(i)} \to z^{(i)}$ 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 set

application of PCA

- compression
  - reduce memory/disk need to store data
  - speed up learning algorithm
- visualization ($k=2, 3$ for plotting graph)

Note: DO NOT use PCA to reduce features for preventing overfitting, which is a bad idea. This won't work, choose to regularization instead.

Design Machine Learning System:

- get training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$
- run PCA to reduce features, get $\{(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \dots, (z^{(m)}, y^{(m)})\}$ (Note: think if it's necessary)
- training hypothesis model on $\{(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \dots, (z^{(m)}, y^{(m)})\}$
- test on test set: map $x_{test}^{(i)}$ to $z_{test}^{(i)}$, run $h_\theta(z)$ on $\{(z_{test}^{(1)}, y_{test}^{(1)}), (z_{test}^{(2)}, y_{test}^{(2)}), \dots, (z_{test}^{(m)}, y_{test}^{(m)})\}$