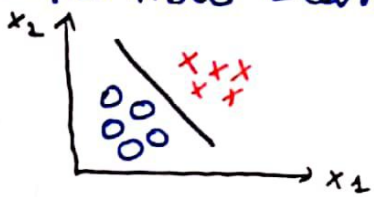
W8 – Unsupervised Learning Intro

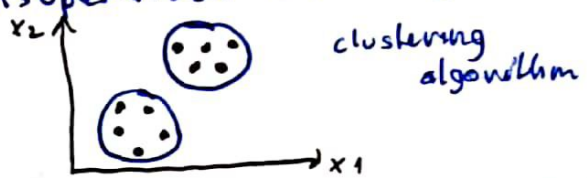
- İlk unsupervised learning algorithmimiz olan Clustering'den bahsedeceğiz.
- Bildiğimiz gibi, we learn from unlabeled data for unsupervised learning.

Supervised Learning



- Training Set: $\{(x^{(1)}, y^{(1)}), \ldots (x^{(m)}, y^{(m)})\}$
- Goal is to find a decision boundary that seperates positive and negative labeled examples.

Unsupervised Learning



- Training Set: $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots x^{(m)}\}$
- No labels (y's).
- Our goal is to make algorithm find some structure in the data.
- Algorithm that finds classes like above is called a clustering algorithm

- Our first unsupervised learning algorithm is Clustering. Although there will be other types of unsupervised learning algorithms that finds other types of structure or other types of patterns in the data other than clusters.

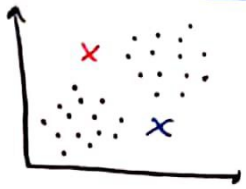
What is clustering good for?

- Market segmentation (alıcıları gruplarım ve farklı davranırım)
- Social Network Analysis (kullanıcıları gruplar)
- Organize Data Centers (determine which computers are tend to work together)
- Astronomical Data Analysis (understand galaxy formation)

- In the next video we'll start to talk about specific clustering algorithm.

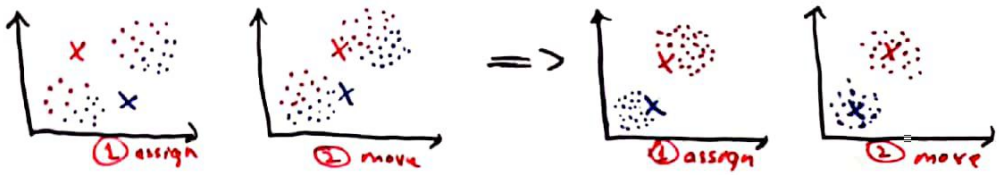
- In the clustering problem we are given an unlabeled data-set and we would like to have an algorithm automatically group the data into coherent subsets or into coherent clusters for us.
- The K-means algorithm is by far the most popular and widely used clustering algorithm.

K-MEANS ALGORITHMS



- Diyelim ki soldaki gibi bir data set olsun. Bunu iki cluster'a ayırmak istiyorum.
- K-means algorithm, önce iki adet noktayı randomly initialize eder. Bunlara cluster centroids denir.

- K-means is an iterative algorithm and it does 2 things.
  - Step 1 • cluster assignment step (her training ex'i yakınlığına göre bir gruba dahil edersin)
  - Step 2 • move centroid step (centroid'i yeni elemanların mean'ine taşırsın)



✓ K-means is converged.

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

- $x^{(i)} \in \mathbb{R}^n$ (drop $x_0 = 1$ convention)

What K-means algorithm does:

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

→ Repeat {

cluster assignment:
- for i = 1 to m → $\|x^{(i)} - \mu_k\|^2$'nin hangi k için min olduğu bulunur ve $c^{(i)}$'ye bu değer atanır!

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

move centroid:
- for k = 1 to K

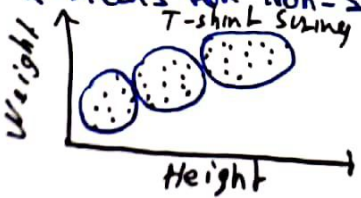
μ_k := average (mean) of points assigned to cluster k

} → Diyelim ki μ_2 centeroidine $x^{(1)}, x^{(5)}, x^{(6)}, x^{(10)}$ atandı yani $c^{(1)} = 2, c^{(5)} = 2, c^{(6)} = 2, c^{(10)} = 2$

→ Then $\mu_2 = \frac{1}{4}[x^{(1)} + x^{(5)} + x^{(6)} + x^{(10)}] \in \mathbb{R}^n$

(?) What if there is a cluster with "0" points assigned to it? In that case we can eliminate that centroid (common thing to do.). Second option is you can just randomly reinitialize that cluster centroid.

K-means for Non-Separated Clusters



- T-shirt üretiminin farklı insanların boyutuna bakıp 3 size çıkaracaksın S-M-L 3 clusters.
- Data iyi ayrılamayacak gibi görünse de K-means ayırır. Aralık ilk kümeye göre small, diğerine göre large T-shirt tasarlayabiliriz

W8 – Clustering Optimization Example

- Most of the supervised learning algorithms we've seen (lin. reg., log. reg. and so on) have an optimization objective or some cost function that the algorithm was trying to minimize. K-means also has a cost function that it's trying to minimize.
- It helps us in 2 means: 1st: It will help us debug the learning algorithm 2nd: To help k-means find better costs for ders and avoid the local optima. We'll se 2nd one later.

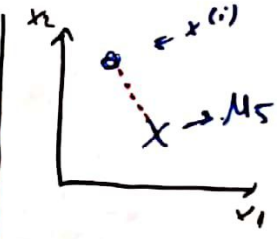
While K-means is running we gonna keep track of 2 sets of variables:

- $c^{(i)}$: index of cluster $(1, 2, \dots K)$ to which example $x^{(i)}$ is currently assigned.
- μ_k : cluster centroid k $(\mu_k \in \mathbb{R}^n)$ → K : # of cluster (total) → k : $\{1, 2, \dots K\}$

- $\mu_{c^{(i)}}$: to denote cluster centroid of cluster to which example $x^{(i)}$ has been assigned.
  - If $x^{(i)}$ is assigned to cluster 5. Then $c^{(i)} = 5$. Thus $\mu_{c^{(i)}} = \mu_5$

Optimization Objective

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



Nasıl önceden best Θ bulduysak burada da amaç best $c^{(1)}, \dots, c^{(m)}, \mu_1, \dots \mu_K$ bulmak

This cost function is also called Distortion

K-means Algorithm was:
Randomly initialize K cluster centroids $\mu_1, \mu_2, \dots \mu_K \in \mathbb{R}^n$
Repeat { ↱ cluster assignment: minimize $J(\dots)$ w.r.t. $c^{(1)}, c^{(2)}, \dots, c^{(m)}$ holding $\mu_1, \mu_2 \dots \mu_K$ fixed

```
for i = 1 to m
    c(i) := index (from 1 to K) of cluster centroid closest to x(i)
```

```
for k = 1 to K
    μk := mean of points assigned to cluster k
```

↳ Move centroid: minimize $J(\dots)$ w.r.t. $\mu_1, \dots \mu_K$

}

(*) Burada ∝ falan yok dolayısıyla Cost Function asla artamaz. Her adımda azalır.

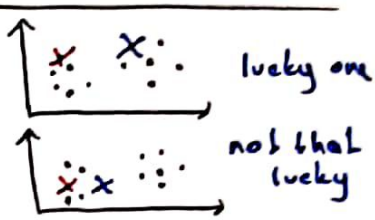
- Yani cost function aslında her training ex.'in kendi centroidine olan uzaklığının karesinin toplamı. Bunu minimize etmek istiyoruz.

- We'll talk about how to initialize K-means and more importantly this will lead us into a discussion how to make K-means avoid local optima as well.

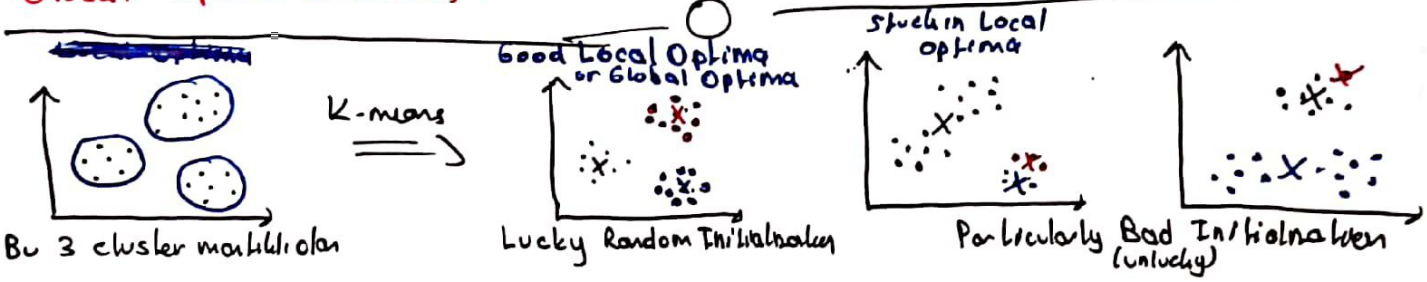
K-means algorithm'in ilk adımı şuydu:
- Randomly initialize K cluster centeroids $\mu_1, \mu_2, \dots \mu_K \in \mathbb{R}^n$
- Bu işlem için bundan farklı method var, fakat one method is much more recommended than the others. Let's talk about that option:

Random Initialization

Let's say $K = 2$



- Should have $K < m$.
- Randomly pick K training examples
- Set $\mu_1, \dots, \mu_K$ equal to these K examples.
- $\mu_1 = x^{(i)}$, $\mu_2 = x^{(j)}$ randomly chosen examples.
- Thus K-means can end up different solutions depending on the initialization.
- So depending on the initialization K-means can end up at Local Optima or Global Optima (desired):



- Local Optima dediğimiz $J(c^{(1)}, \dots c^{(m)}, \mu_1, \dots \mu_K)$'nin local optimasıdır.

- Thus, to avoid stucking in bad local optimas we can try multiple random initializations.

Way of applying multiple random Initialization:

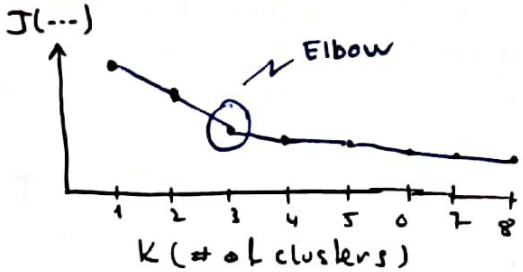
- for i = 1 to 100 {
  - → Randomly initialize K-means
  - → Run K-means. Get $c^{(1)}, \dots c^{(m)}, \mu_1, \dots, \mu_K$
  - → Compute cost function (distortion): $J(\dots)$ (100 farklı $J(\dots)$ lardan)
  }
- Pick clustering that gave lowest cost $J(\dots)$

- If K = 2 - 10 doing multiple random initialization can sometimes make you find a better local optima. But if K is large having multiple random initialization is less likely to make a huge difference, muhtemelen yine daha iyi sonuç verir ama büyük fark olması zor.

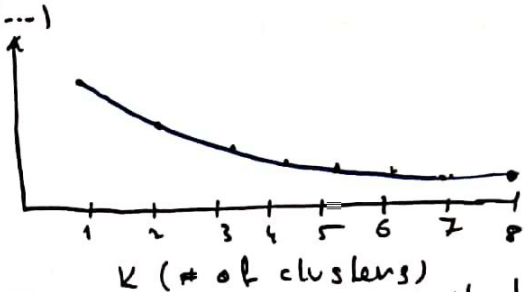
VIII.8 – Choosing the Number of Clusters, K

- To be honest, there isn't a great way of automatically choosing K. Most common way is still choosing it manually by looking at visualizations or by looking at the output of the clustering algorithm on something else.
- Bir daha'ya baksak bile kac cluster olmasi gerektiginin mutlak bir cevabi yoktur.

Elbow Method to Choose K: People sometimes use it to ~~determine~~ choose K.



- Elbow'a kadar distortion goes down rapidly then goes down slowly. Maybe K=3 is the right choice.



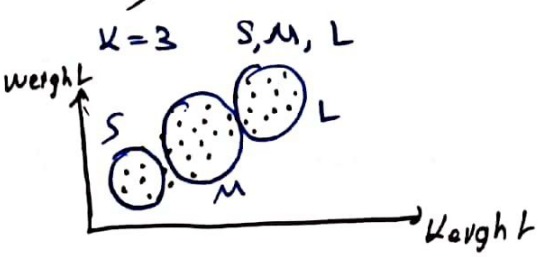
- Genelde boyle bir graph ile karşılaşsın burada kesin bir elbow yok. Soldaki plotla karşılaşınsak iyi bir şey kolayca seçebilirsin.

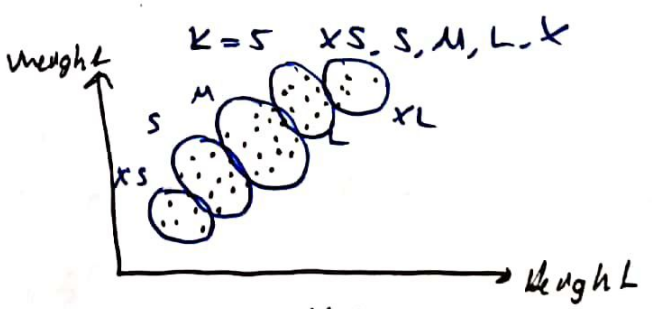
⭐ **Summary of Elbow Method:** Graph'i deneyip soldaki gibi çıkarsa ne alâ falan it is a long shot.

⭐ Grafiğin sürekli azalması MUST değil belki K=5 için K=3'ten daha büyük bir cost verebilir. Bu demektir ki kötü bir local minimum'a takıldı, random initialization multiple deneyip K-means'i tekrar çalıştırabilirsin.

Another Method to Choose K: Sometimes, you're running K-means ~~for~~ 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.

- Let's say you wanna use K-means for market segmentation like in the t-shirt sizing example below.





⭐ Burada ontik kara vermek var hangisinin daha iyi olduğuna bakılma 5 cluster olsun müşterilere daha iyi mi uysun yoksa bazı müşterilen aradığın, bulamasın ama daha ucuza mı maledelim?

W8 - Dimensionality Reduction -

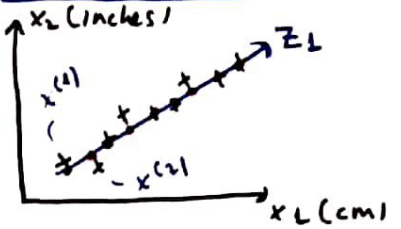
- We'll start talking about a second type of unsupervised learning problem called dimensionality reduction.
- There are couple of different reasons why one might want to do dimensionality reduction:

Motivation I: Data Compression

- Data compression allow us to use less computer memory and also speed up our learning algorithms.

Let's talk about what is dimensionality reduction?

2D to 1D



- Bir bilmiyorum ama bir çok feature içinden x_1 ve x_2 aynı sadece birimi farklı. This gives us a highly redundant representation.
- Aynı feature'ı ölçmek için hem x_1 hem x_2 kullanmak yerine tek feature ile işi halledebilirim.
- cm ve inch verileri rounded olduğu için tam bir çizgi takip etmiyorlar.

Dimensionality reduction için datanın takip ettiği line belirlenir ve datalar tek tek bu line'a project edilir. Sonuçta x_1-x_2 yerine z_1 adında bir feature elde edilir.

x_1 ile x_2 arasında bu kadar basit bir straight line (inches to cm) olmasa dahi basit features highly correlated olabilir, straight line'a yakın (matematiksel olarak olması şart değil!) yine dim. red. iş yapar.

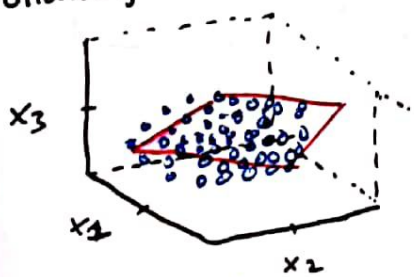
- $x^{(1)} \in \mathbb{R}^2 \rightarrow z^{(1)} \in \mathbb{R}$
- $x^{(2)} \in \mathbb{R}^2 \rightarrow z^{(2)} \in \mathbb{R}$
- $x^{(m)} \in \mathbb{R}^2 \rightarrow z^{(m)} \in \mathbb{R}$

Summary: If we allow ourselves to approximate the original data set by projecting all of my original examples onto the blue line then I need only one number to specify the position of a point on the line and therefore I can represent each training examples by 1 feature instead of 2. This is a approximation.

3D to 2D Reduction

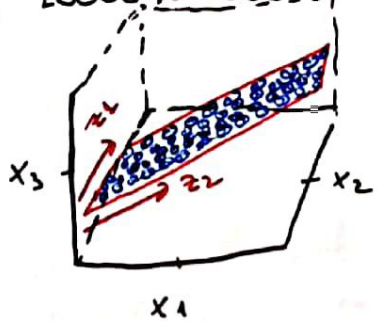
10000D to 100D gibi olur burada anlaşılması için 2D ve 3D

Genelde bu reduction kullanılıyor.

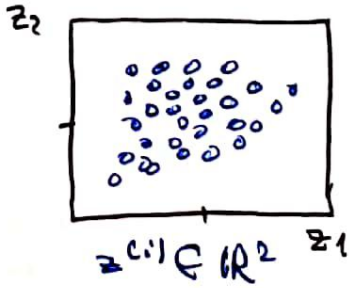


$x^{(i)} \in \mathbb{R}^3$

- Maybe all of these data roughly lies on the red plane.



- What we can do with dimensionality reduction is to project the data to a 2D plane



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

- Now I can represent any training ex. by z_1 and z_2 instead of x_1, x_2, x_3.

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \qquad z^{(i)} = \begin{bmatrix} z_1^{(i)} \\ z_2^{(i)} \end{bmatrix}$$

- In the last video we talked about dimensionality reduction for the purpose of compressing the data.
- In this video we are gonna talk about the second application of dimensional reduction and that is to visualize the data. Data visualization bir çok problemi çözmekte yararlı olur.

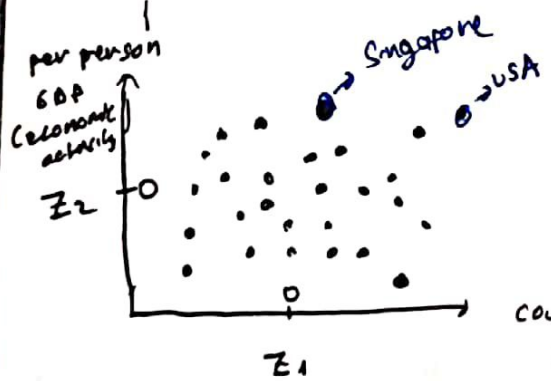
Country	x_1 GDP	x_2 Per capita GDP	x_3 Human Development Index	x_4 Life expectancy	x_5 Poverty Index	x_6 Mean household income	...
Canada	...	...	...	...	...	...	...
China	⋮	⋮	⋮	⋮	⋮	⋮	⋮
India	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Russia	⋮	⋮	⋮	⋮	⋮	⋮	⋮
USA	⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮							

$x \in \mathbb{R}^{50}$

- How do we visualize this data if we have 50 features for each country. It is very difficult to plot 50 dimensional data.
- Using dimensionality reduction what we can do is instead of representing each country by $x^{(i)} \in \mathbb{R}^{50}$ we can come up with a different feature representation $z^{(i)} \in \mathbb{R}^2$. If z_1 and z_2 somehow summarizes my numbers we can plot these countries in $\mathbb{R}^2$ and understand the countries little bit better.

Country	z_1	z_2
Canada	...	...
China	⋮	⋮
India	⋮	⋮
⋮		

- Reduce data from 50D to 2D
- When you do that, the meaning of z_1 and z_2 is not very clear. It is often up to us to figure out roughly what these features z_1, z_2 means



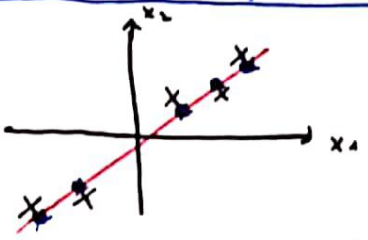
- You might find that given these 50 features these are the 2 main dimensions of the deviation.

- That's how we can use dimensionality reduction for visualization purpose.

- In the next part we will start to develop a specific algorithm called PCA (Principal Component Analysis) which will allow us to do this and also the earlier application we talked about of compressing the data

- For the problem of dimensionality reduction by far the most popular algorithm is principal component analysis (PCA)
- Let's formulate precisely, exactly what we would like PCA to do.

Principal Component Analysis Problem Formulation

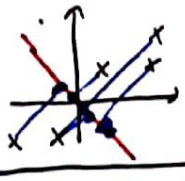


- Let's say we have a data set as here. $x \in \mathbb{R}^2$
- I want to reduce the dimension from 2D to 1D in other words I would like to find a line onto which to project the data.

- Yukarıda çizilen kırmızı çizgi iyi bir line'a benziyor ama neden? Çünkü siyah orjinal noktaların project edilmiş mavi noktalara olan uzaklıkları toplamı çok az. Bu demek oluyor ki kırmızı çizgi dimensionality reduction için iyi bir çizgi

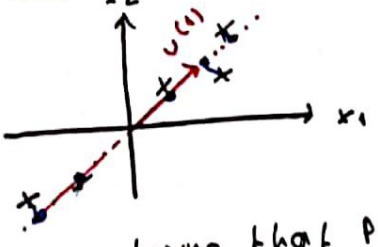
- So what PCA does is, it tries to find lower dimensional surface (a line in this case) onto which to project the data so that the sum of squares of distances b/w original data points and projected ones is minimized. dear.
- Gerçek data X ile projected data • arasındaki mesafeye denen "projection error."

- Before applying PCA, it's standard practice to first perform mean normalization and feature scaling so that features should have zero mean and should have comparable ranges of values.



- Yukarıdaki yerine yandaki gibi bir çizgi garddüğü gibi projection errorları artıyor. Bu yüzden bu seçim yanlış olurdu.

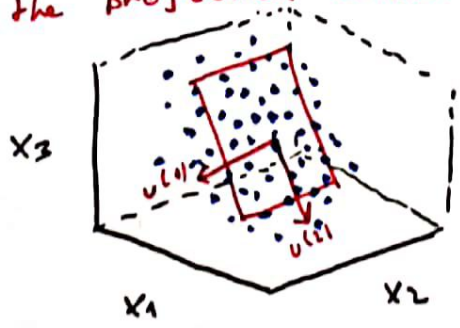
Let's write out the PCA problem a little more formally. $\mathbb{R}^2$ in this case



- 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.

- I'm hoping that PCA will find the vector $u^{(1)}$ above. So that when I project the data onto the line that I define by extending out the $u^{(1)}$ vector, and end up with pretty small projection errors. • $u^{(1)}$ veya $-u^{(1)}$ farketmez aynı line!

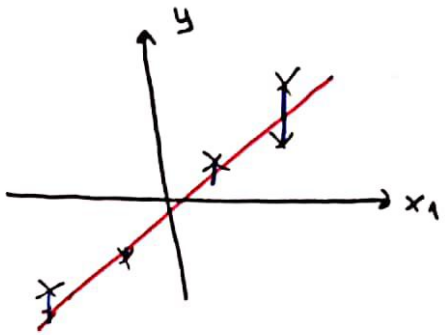
More General Case: Reduce n-dimension to k-dimension: Find k vectors $u^{(1)}, u^{(2)}, \dots u^{(k)}$ onto which to project the data so as to minimize the projection error.

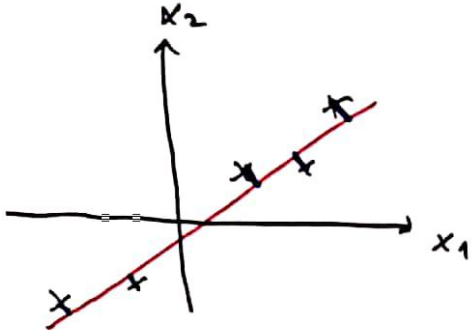


- Burada 3D to 2D dim. red. yapıyoruz. $u^{(1)}$ ve $u^{(2)}$ vektörleri birlikte 2D bir surface oluşturuyorlar. Data bu surface'e project edilecek. PCA'in görevi datanın min hatayla project edileceği surface'i bulmak.
- 2D yerine kD de aynı! 2D surface yerine kD surface tanımlanmış oluyor.

PCA is not linear regression:

- Despite some cosmetic similarity, these are actually totaly different algorithms.





- Lin. Reg. ican we are trying to predict the value of some variable y, given some input features x. We fit a line and minimize the squared errors b/w points and the line.
- Hatayı bakarsan mavi çizgilen vertical!
- In contrast in PCA, it tries to minimize the magnitude of different kind of errors (shortest distance b/w point x and red line)
- Ayrıca linear regression'ın'da there is this distinguished variable y we are trying to predict whereas in PCA there is no special variable y that we are trying to predict.

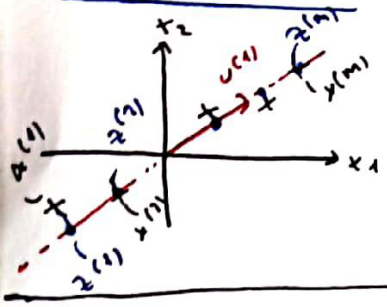
Summary: PCA is trying to find a lower dimensional surface onto which to project the data, so as to minimize the squared projection error. (squared distance b/w each point and its projection)

Sıradaki derste bu lower dimensional surface'ı nasıl bulurus ona bakacağız.

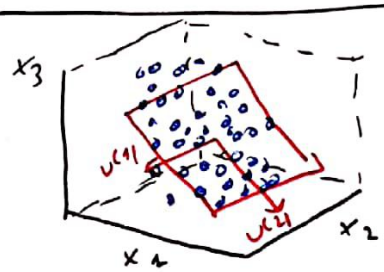
Data preprocessing before applying PCA:

- Given a training set of n unlabeled examples it is important to always perform mean normalization and depending on the data maybe perform feature scaling.
- Training set: $x^{(1)}, x^{(2)}, \dots x^{(m)}$
  - $\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$
  - Replace each $x_j^{(i)}$ with $x_j - \mu_j$
  - If different features on different scales perform feature scaling
- Feature Scaling / Mean Normalization: μ_j: mean for each feature

PCA Review:



- Reduce from 2D to 1D
- $x^{(i)} \in \mathbb{R}^2 \rightarrow z^{(i)} \in \mathbb{R}$



- Reduce 3D → 2D
- $x^{(i)} \in \mathbb{R}^2 \rightarrow z^{(i)} \in \mathbb{R}^2$
- $z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$

(*) What PCA has to do → Compute $u^{(1)}, u^{(2)}, \dots u^{(k)}$ vectors.
→ Compute $z^{(1)}, z^{(2)}, \dots z^{(m)}$ points.

PCA Algorithm

- Assume we wanna reduce data from n-D to k-D

→ Compute "covariance matrix": $\underset{n\times n}{\Sigma} = \frac{1}{m}\sum_{i=1}^{m} \underset{n\times 1}{(x^{(i)})} \cdot \underset{1\times n}{(x^{(i)})^T}$

→ Compute "eigenvectors" of matrix Σ: $[U, S, V] = \text{svd}(\Sigma)$ ↳ eig(Σ)

→ $U = \begin{bmatrix} | & | & | & & | \\ u^{(1)} & u^{(2)} & u^{(3)} & \dots & u^{(n)} \\ | & | & | & & | \end{bmatrix}_{n\times n}$ where $U \in \mathbb{R}^{n\times n}$ → $u^{(1)}, \dots u^{(k)}$ aradığımız surface'i oluşturur.

k → we are gonna use first k column of U

→ Sıradaki adım $x \in \mathbb{R}^n$ için $z \in \mathbb{R}^k$'yı bulmalıyız. $\begin{bmatrix} z_1 \\ \vdots \\ z_k \end{bmatrix} \leftarrow \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$

→ $U_{reduced} = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(k)} \\ | & | & & | \end{bmatrix}_{n\times k}$ → $z^{(i)} = \underset{k\times n}{U_{reduced}^T} \cdot \underset{n\times 1}{x^{(i)}}$ → $\begin{bmatrix} -(u^{(1)})^T- \\ -(u^{(2)})^T- \\ \vdots \\ -(u^{(k)})^T- \end{bmatrix}_{k\times n} \cdot x_{n\times 1}$

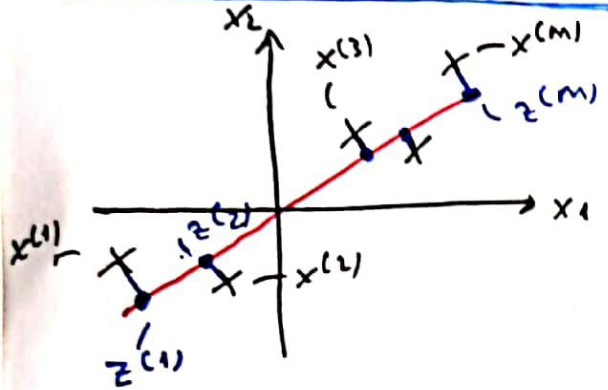
$k \times 1$

PCA Algorithm Summary

- After mean normalization (ensure every feature has zero mean) and optionally feature scaling:
- Sigma $= \frac{1}{m}\sum_{i=1}^{m} (x^{(i)}) \cdot (x^{(i)})^T$ → if X is given as $X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \dots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & & \dots & x_n^{(2)} \\ \vdots & & & & \vdots \\ x_1^{(m)} & x_2^{(m)} & & \dots & x_n^{(m)} \end{bmatrix}_{m\times n}$ → then $\text{Sigma} = \left(\frac{1}{m}\right) * X' * X$
- `[U, S, V] = svd(Sigma);`
- `Ureduced = U(:, 1:k);` // Grab first k columns
- `z = Ureduced' * x;` // Similar to K-means $x \in \mathbb{R}^n$ no $x_0 = 1$ convention

- I was talking about PCA as a compression algorithm where you may have say, a 1000-D data and compress it to 100-D feature vector.
- There should be a way to go back from this compressed representation back to an approximation of your original high-dimensional data.
- So given $z^{(i)}$ which may be 100-Dimensional, how do you go back to the original representation $x^{(i)}$ which maybe a 1000-Dimensional.

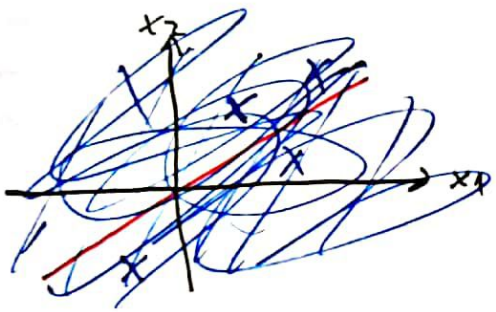
Reconstruction from Compressed Representation

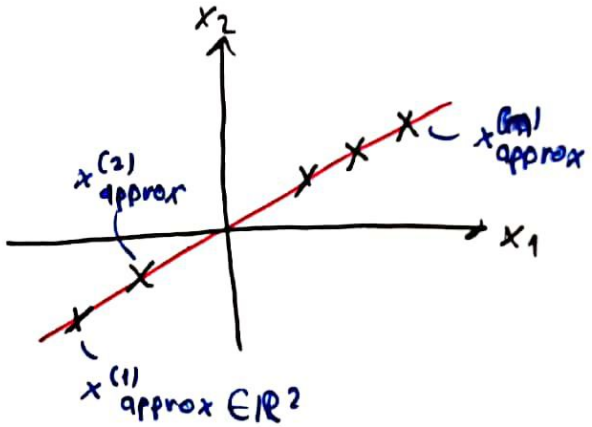


- We know: $z = U_{reduced}^T \cdot x$
- So if $z \in \mathbb{R}$ is given $\longrightarrow$ what is $x \in \mathbb{R}^2$
- $x_{approx} = U_{reduced} \cdot z$

$n \times 1 \quad = \quad n \times k \quad \cdot \quad k \times 1$

$n \times 1$





W8 - Choosing the # of Principal Components

Choosing k (number of principal components):

- Here are couple of useful concepts to choose k:
  - Avenage squared projection erron: $\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - x_{approx}^{(i)}\|^2$ (original data $\mathbb{R}^n$, projected data $\mathbb{R}^n$) — PCA bu değeri min. etmeye çalışır.
  - Total variation in the data: $\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)}\|^2$: On average how far are my training ex.s from being zero.

(1)

Typically choose k to be smallest value so that:

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

Baren %5, %10 da olabilir.

"99% of variance is retained"

~~In order to retain 99% of the variance~~ you can often reduce the dimension of the data significantly and retain most of the variance. Because for most real life data-sets many features are highly connelated.

Implementation:

Algorithm I

- Try PCA with $k=1$
- Compute $U_{reduced}, z^{(1)}, z^{(2)}, \dots z^{(m)}, x_{approx}^{(1)}, \dots, x_{approx}^{(m)}$
- Check if 99% of variance is retained:

$$\frac{\frac{1}{m}\sum \dots}{\frac{1}{m}\sum \dots} \leq 0.01 \; ?$$

- If 99% of var. is ret. use current k! Else go back and increase k. So we are trying to find the minimum k that provides 99% of variance is retained.

(2) Very ineffecient her adimda PCA uygulayerek! Çok fazla hesaplama var.

Algorithm II (Easier)

- $[U, S, V] = \text{svd}(\text{Sigma})$
- $S = \begin{bmatrix} S_{11} & & 0 \\ & S_{22} & \\ 0 & & \ddots \\ & & & S_{nn} \end{bmatrix}$
- For a given k, retained variance value can be calculated using the S only. S'i de yalnızca başta bir kez bulmamız yeterli:

$$\text{retained variance} = 1 - \frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}}$$

- Mesela $k=2$ için $1 - \frac{S_{11}+S_{22}}{S_{11}+S_{22}+\dots+S_{nn}}$
- Sınayla k'yi artır if 99% variance retained sağlanmışsa dur!

-W8- Advice for Applying PCA -

- In an earlier video, we've said that PCA sometimes can be used to speed up the running time of a learning algorithm.

Supervised Learning Speedup: Most common use of PCA.

- Let's say we have training set: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots (x^{(m)}, y^{(m)})$
- And let's say examples $x^{(i)}$ are very high dimensional $x^{(i)} \in \mathbb{R}^{10000}$ Mesela computer vision ile uğraşıyorsun diyelim 100x100 px resimlerle uğraşıyorsak bu 10000 features yapar.
- Feature vectors dimension 10 000 olunca running a learning algorithm might be slow. PCA ile dimensionality reduction yapıp algoritmayı hızlandırabiliriz.

(1) First Extract Inputs:
- Unlabeled dataset: $x^{(1)}, x^{(2)}, \dots x^{(m)} \in \mathbb{R}^{10\,000}$
- Apply PCA now I have $z^{(1)}, z^{(2)}, \dots z^{(m)} \in \mathbb{R}^{1000}$

(2) Now I have a new training set:
- $(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \dots, (z^{(m)}, y^{(m)})$
- Now I can take this reduced dimension training set and feed it to a learning algorithm maybe a NN or log.reg. $h_\theta(z) = \frac{1}{1+e^{-\theta^T z}}$
- If we have a new example maybe a new test example x, map it through the same mapping $x \to z \to h_\theta(z) \to$ prediction

(3) Final Note: Mapping $x^{(i)} \to z^{(i)}$ 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 sets.
- Mapping yaparken bir Ureduced bulurum. Bu Ureduced is like a parameter that is learned by PCA and we should be fitting our parameters only to our training sets and not to our cross validation or test sets.
- Then having found Ureduced on having found the parameters for feature scaling where the mean normalization and the scale that you divide the features by to get them on to comparable scales. Having found all those PARAMETERS on the training set you can then apply the same mapping to other examples maybe in your cross-validation or test sets.

(4) Reducing the data from 10000 D to 1000 D is not un-realistic. For most of the problems we can reduce the data by 10x and still retain the most of the variance yani we are barely affecting the classification accuracy.

Application of PCA Summary:

- Compression
  - Reduce memory/disk needed to store data
  - Speed up learning algorithm

  ] Bu uygulamalar için choose k by % of variance retain. >90% iyi bir değerdir.

- Visualization] choose $k=2$ or $k=3$.

Bad use of PCA: To prevent overfitting:

- People use $z^{(i)}$ instead of $x^{(i)}$ to reduce # of features to $k<n$
- Thus, fewer features, less likely to overfit.
- It might actually work OK. But it is not a good way to address overfitting

→ Regularization is a much better way for overfitting problem:

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \boxed{\frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2}$$

- PCA doesn't use the values y. Thus it throws some information without knowing the y values. It may throw away some valuable information. Regularization will work better because this minimization problem actually know what the values y are. Less likely to throw away some valuable information.

PCA is sometimes used where it shouldn't be. Don't use it if you don't need it:

- Mesela bir ML system design ediliyor:
  - Get training set $\{(x^{(1)}, y^{(1)}), \dots (x^{(m)}, y^{(m)})\}$
  - Run PCA $x^{(i)}$ in dimension to get $z^{(i)}$ ✗ Genelken olabilir!
  - Train logistic regression on $\{(z^{(1)}, y^{(1)}), \dots, (z^{(m)}, y^{(m)})\}$
  - Test on test set: Map $x_{test}^{(i)}$ to $z_{test}^{(i)}$. Run $h_\theta(z)$ on $\{(z_{test}^{(1)}, y_{test}^{(1)}), \dots (z_{test}^{(m)}, y_{test}^{(m)})\}$

⇒ How about doing the whole thing without using PCA?

Before implementing PCA, first try running whatever you want to do with the original/raw data $x^{(i)}$. Only if that doesn't do what you want, then implement PCA and consider using $z^{(i)}$.

PCA is one of the most powerful unsupervised learning algorithms.