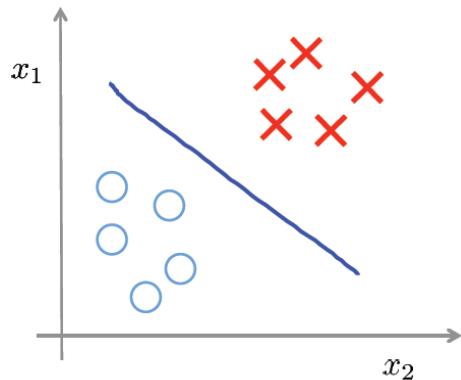


الأسبوع الثامن

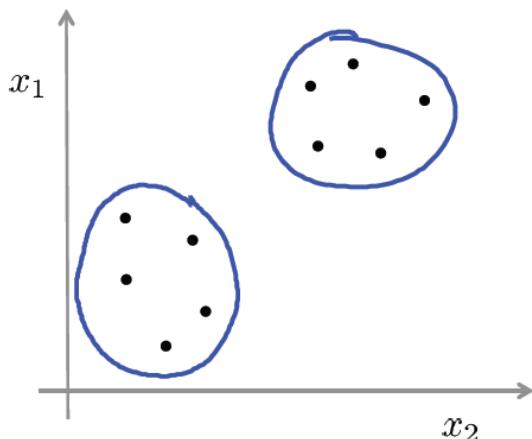
Unsupervised Learning

- كل ما سبق تعلمه كان ، الـ supervised learning يعني باعطي الآلة ، مجموعة من X & y وهي بتجيب الخوارزم ●
الآن هنبدأ في الـ unsupervised learning ومعناه اني معدنيش y ، بس عندي اكستات ، وديه اللي هديها للخوارزم ،
وهو هيقسمها لمجموعات حسب تقاربها او تشابهها

- تجنبنا للغمطة بين الـ supervised و الـ unsupervised ، هنشوف المثال ده :

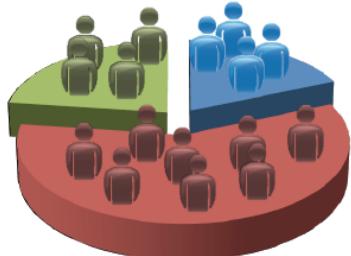


- هنا مثل للـ supervised ، إن عندي عدد من البيانات اللي عندي فيها قيمة y & X و بيقوم الخوارزم بالفصل بينهم ، عن طريق الـ best fit line

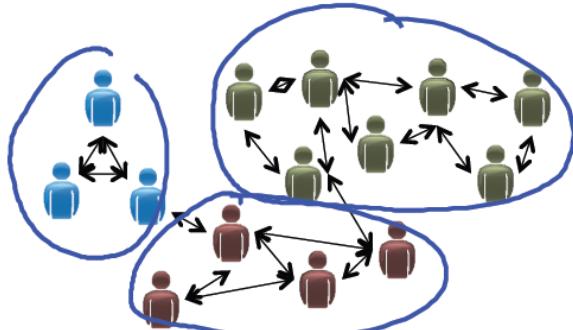


- بينما في الـ unsupervised انا عندي بيانات مدخلة بس (إكستات) ، من غير وايات (النقط شبه بعض مش متميزة زي اللي فوق) . والخوارزم بيقسم النقط بناء على الصفات المتشابهة و التقارب ●
ومن استخداماته :

Applications of clustering



→ Market segmentation



→ Social network analysis



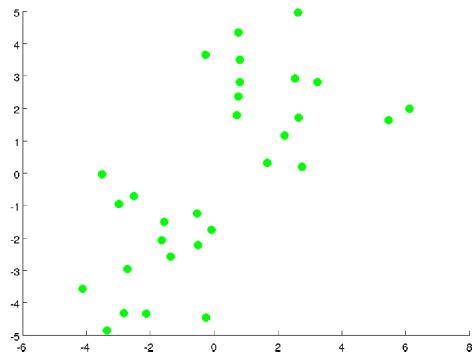
→ Organize computing clusters



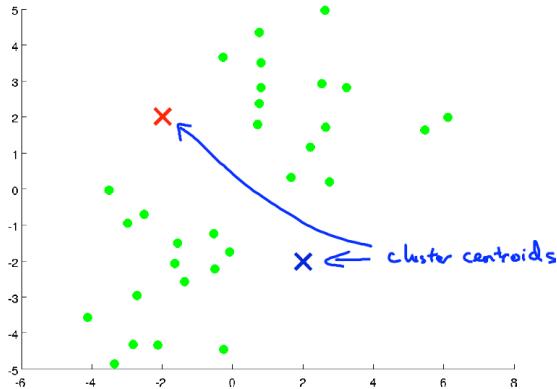
→ Astronomical data analysis

- تقسيم شرائح العملاء (target group) كل مجموعة لها صفات محددة
- تقسيم الناس في الشبكات الاجتماعية ، حسب صفاتهم
- تقسيم بيانات خاصة بالحسابات
- تقسيم النجوم و المجرات

- أشهر خوارزم لعمل clustering هو ما يسمى الـ K mean
- تعالى نفهم هو بيعمل ايه
- نفرض ان عندنا نقط غير معنونة زي كدة :



- أول خطوة الـ K mean بيعملها ، انه بيعمل نقطتين عشوائيتين ، اسمهم cluster centroid مرکز العنقود ، زي كدة :



الـ K mean بيعمل خطوتين بالتوالي , وبيكررهم :

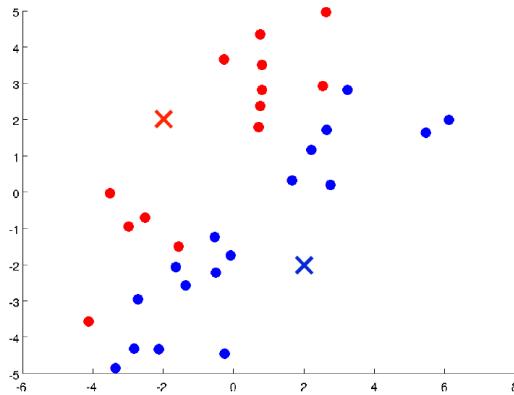
أولا : انه يشوف المسافات بين كل نقطة من العينة (العينة) و المركزين , ويخلطي كل نقطة من العينة

تابعة للمركز الأقرب ليها

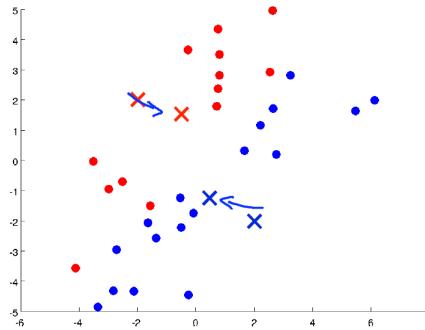
ثانيا : انه يغير مكان المركزين , ويعيد الخطوة تاني

● يعني ايه ؟

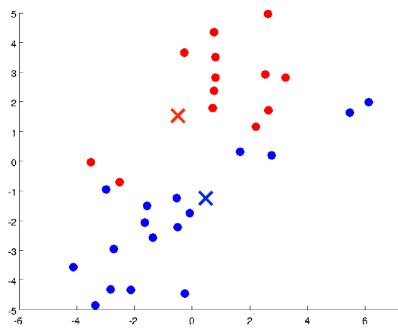
يعني هو عمل المركزين النقط الحمرا و الزرقا , وبعدها يخلطي كل النقط الخضرا , يا تابعة لدعي او ديه , حسب هي اقرب لمين , زي الصورة ديه :



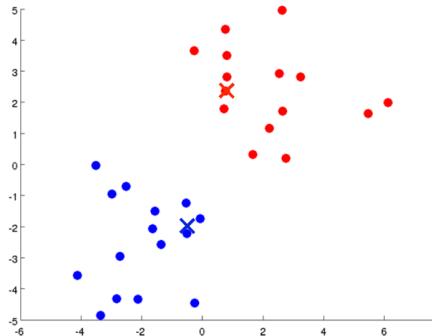
تيجي الخطوة الثانية , ان الخوارزم عايز يضبط النقط اكتر , فهيعمل تحريك للنقطة الزرقا, في اتجاه منتصف تكتل النقط الزرقا , وكذلك في الحمرا , فهتبقي زي كدة :



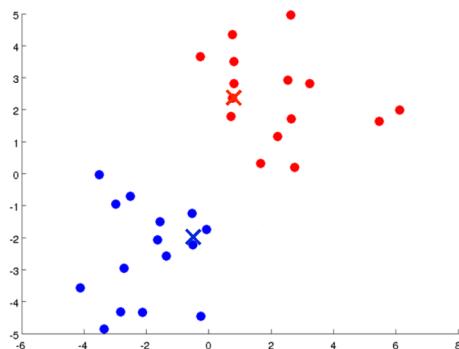
○ لما تتنقل النقطتين ، هنعيد الخطوة الاولى تاني ، اننا نشوف مين اقرب و نقسم تاني زي كدة :



○ نعمل تحريك تاني ، لمركز الالوان الجديدة ، واعادة تلوين



○ بعد تكرار مرة كمان ، نلاقي ان تم عمل التقسيمة السليمة ، لغاية لما نلاقي ان مفيش تحريك تاني متاح



○ وبالتالي ، أيا كانت قيمة النقط اللي تم تحديدها على الجراف ، فالـ K هيقدر يعدل وضعها بالطريقة ديه
○ كمان ده ينطبق علي اي عدد من العناقيد ، مش لازم 2 ، لو اتعمل 5 نقاط هتعمل برضه

• المدخلات :

- خوارزم الـ K mean يبيجيه مدخلين هما :
 - عدد العناقيد K (يعني هعمل كام مجموعة من النقط)
 - البيانات المدخلة X_i
- لاحظ ان عدد العناقيد , احيانا بيكون ليه طريق عشان احدهه , يعني ممكن تكون انا مخطط اني هقسم الجمفور عندي لاربع مجموعات , او اني اخلي الخوارزم هو يختار الافضل حسب ما هو شايف , وده هنشوفه قدام

• تعالى نتعرف علي طريقة الكود الخاص بالخطوتين دول :

```
for i = 1 to m
   $c^{(i)}$  := index (from 1 to K ) of cluster centroid
  closest to  $x^{(i)}$   $\min_{\text{all } k} \|x^{(i)} - \mu_k\|^2$ 
```

○ الخطوة الاولى هي الاختيار :

- و هي تحديد اي نقطة تتبع اي مركز فيهم , وده عن طريق تتابع الـ norm بين كل نقطة , وبين المركز هنسمي المركز μ على اعتبار ان الـ K الكابيتال هي عدد العناقيد , بينما الـ k السمول , هو رقم العنقود , اللي بيبدأ من 1 لغاية K قيمة الـ norm هتكون :

$$\text{Get } k \text{ to minimize } == C^i = \|X^i - \mu_k\|$$

- يعني بجيبي قيمة المسافة بين كل اكس وبين المركز الاول و الثاني , واشوف انهي k اللي هتعمل اقل C
- احيانا بيعملو تربيع لقيمة , وهي هي مفيش فرق

○ الخطوة الثانية الازاحة :

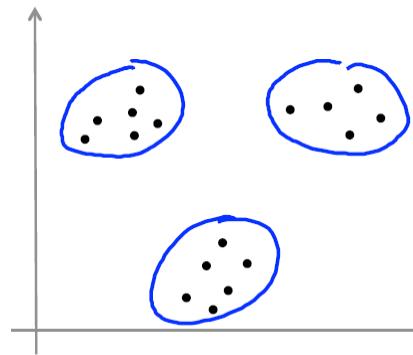
$$\mu_k := \text{average (mean) of points assigned to cluster } k \rightarrow \begin{matrix} x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)} \\ \rightarrow c^{(1)}=2, c^{(2)}=2, c^{(3)}=2, c^{(4)}=2 \end{matrix}$$

$$\mu_2 = \frac{1}{4} [x^{(1)} + x^{(2)} + x^{(3)} + x^{(4)}] \in \mathbb{R}^n$$

- بعد ما اعمل تقسيم مجموعات , بقول ان قيمة μ ه تكون متوسط النقط اللي تم اختيارها , يعني اجمعهم و اقسمهم علي عددهم

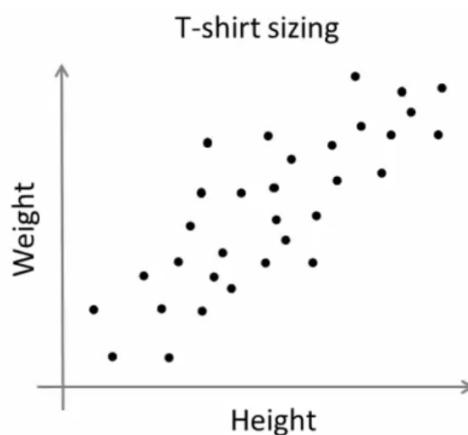
• المجموعات المنسجمة :

- لاحظ ان تقسيم الحاجات مش دايما بيكون سهل , يعني لو بصينا هنا

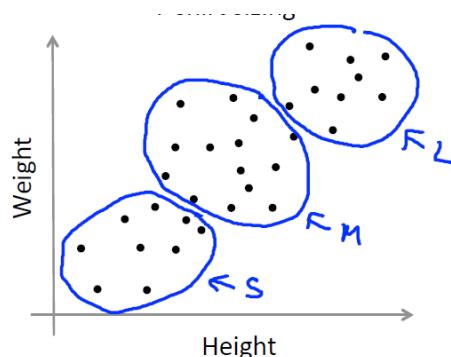


- هنا لقي النقط مفصولة عن بعض بشكل كويس ، وان التفريق بينهم متاح وسهل الي حد كبير

بينما هنا



- واضح ان البيانات متداخلة مع بعض بشكل كبير ، وان الارقام منسجمة و مفيش فروق واضحة بينهم ، فمثلا ، لو عايز اقيش اوزان و اطوال الناس ، عشان اقسمهم مجموعات عشان اعمل تلات مقاسات من تيشرت ، فعايز اقسمهم S , M , L



- ساعتها ممكن الـ K mean يقسم المجموعات بقدر الامكان حسب المتاح زي كدة

الـ Optimization Objective •

- يعني ايه ؟ يعني الدالة او القيمة اللي بنحاول نقللها بقدر الامكان عشان نوصل للقيم المثلالية

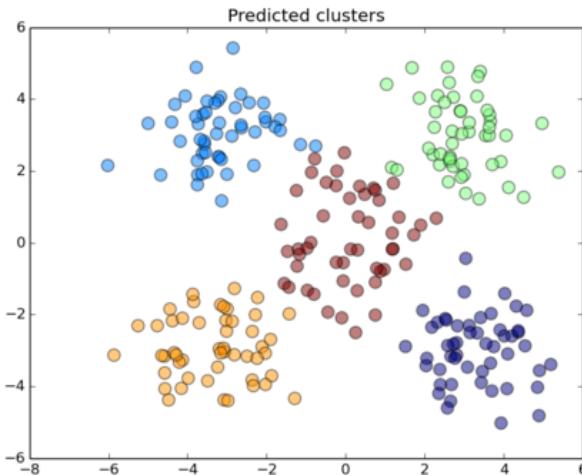
- زي ما شفنا في الـ supervised cost هي الـ J اللي بنحاول نقلها بقدر الامكان ، عشان نوصل لأفضل قيم للثيتا اللي تحقق لنا التقسيم (في الكلاسيفيكاشن) او التوقع (في الريجريشن)
- و برضه هنا في الـ K mean عندنا رقم للـ Optimization عشان نشوف افضل تقسيم للنقاط اي ان الوصول للـ Optimization لقيمة الـ K معناهاني اقدر احدد مكان المركزين بالضبط ، اللي بيجمعو حولهم النقط بالشكل المثالي ، بحيث يكون مجموع المسافات بين كل مركز ، والنقط التابعة ليه اقل ما يكون

- لعمل الـ Optimization محتاج حاجتين :
 - قيمة C وهو خاص بالتقسيمات ، اللي هو الرقم الخاص بالمجموعة الاولى او الثانية و هكذا
 - قيمة μ وهي خاصة بمركز كل مجموعة منهم

- من القيمتين دول ، ممكن ادمجهما مع بعض و اقول ان $\mu_{c^{(i)}}$ معناها انه بجيب الميو (مركز المجموعة) الخاص برقم المجموعة اللي اسمه C
- و ممكن ساعتها نصيغ قيمة الـ J التكلفة ، علي انها مجموع مربعات المسافات بين كل اكس ، وبين المركز الخاص بيها ، وقسمتها على m اللي هو عدد نقاط المجموعة ديها

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

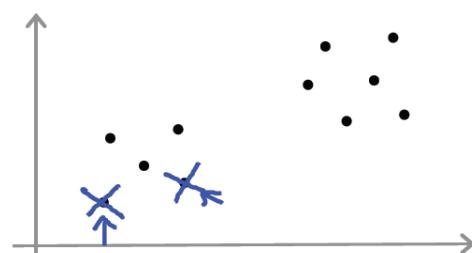
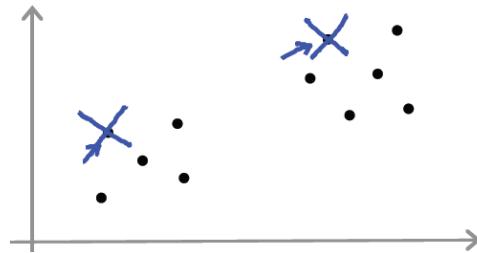
- الطرف الايسر معناه ان المدخلات بتاعت الـ J هي السيهات و الميوهات
- فلو عنديمجموعات كدة ، قيمة الـ J هي مجموع مربعات الفروق بين كل نقطة و المركز بتاعتها على m



- واحيانا بنسميها الـ distortion cost function

-
- خطوات الخوارزم:

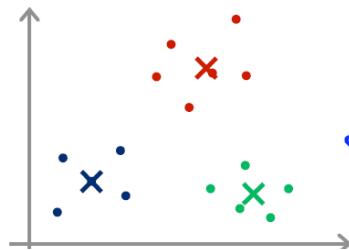
- سنتعلم هنا الخطوة الاولى لبدأ عمل الـ K means و كيفية تجنب الوقوع في فخ القيم الصغيرة المحلية local minima
- سنبدأ من كيفية اختيار موقع المركز لكل مجموعة
- واضح ان عدد الـ K لابد ان يكون اقل من m , فمستحيل عمل عدد مجموعات يساوي او اكتر من عدد العناصر
- فلو عندنا نقط زي كدة يفضل اني اختار نقط كل مركز زي دول ، عشان نوصل للتقسيم الامثل :



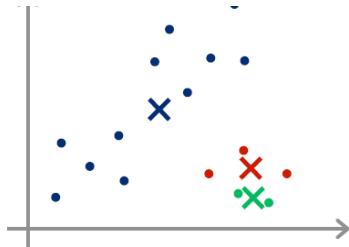
- وبينما ، لو عملت اختيار نقط زي ديه ، هيعمل تقسيم غير مناسب



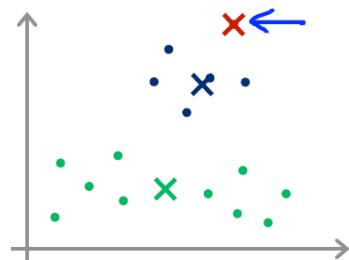
- لو عندي هنا مجموعة من النقط ، فممكنت تقسم ، بالشكل ده :



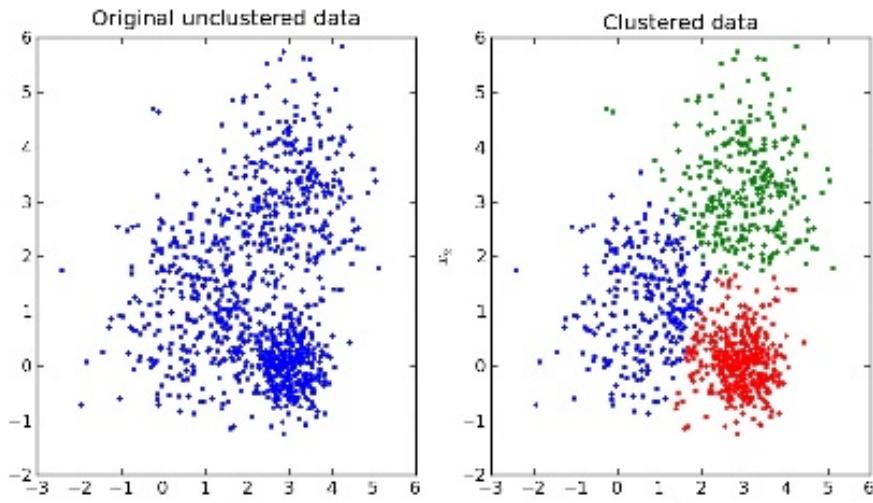
- واللي بيدو انه تقسيم مثالاً ، بينما ممكن لو تم اختيار نقط مختلفة ان النقط تقسيم كدة



او كدة



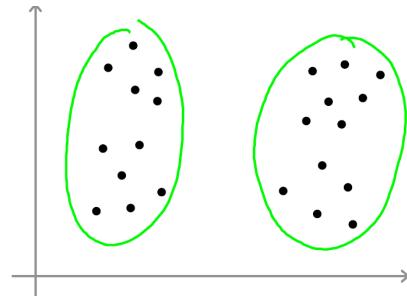
- فمن الواضح ان التقسيم الاول , هو تقسيم مثالي , يعني عمل تجميع لكل الي شبه بعض مع بعض , بينما الثاني و الثالث غير سليم , لانه مقدرش يعمي تقسيم سليم
- عشان نفهمها اكتر , تخيل معايا اني باعمل تقسيم لمجموعة من الناس حسب محور اكس (ذكاءه) ومحور واي (تعليميه) , فالتقسيم الاول يبدو تقسيم منطقي و متجانس , بينما الثاني و الثالث غير معبر عن اي حاجة
- وهذا عايزين نربط بين ده ومفهوم **global minima Vs local minima**
- في الـ **supervised** كنا بنتعامل مع الجلوبال , عليه انه النتيجة الافضل مطلقا , بينما اللوكال , علي انه احده النتائج لكن فيه افضل منه
- نفس الموضوع هنا , التقسيم الاول جلوبال , والثاني و الثالث لوكال
- واضح ان الحل الامثل , قيمة L الكلية بتكون اقل ما يكون , بينما في الحلين الثاني و الثالث (اللي فيه مشاكل) بتكون قيمة L اعلي شوية من قيمة L في الجلوبال
- وعشان اوصل للجلوبال , لازم اعمل الـ **K mean** اكتر من مرة , لغاية لما اتأكد اني وصلت للنتيجة الافضل
- واحيانا بيوصل العدد لـ 100 محاولة , واحسب اقل L فيه
- و خد بالك لو عدد المجموعات **clusters** قليلة (اقل من 10) فهم جدا انك تعمل محاولات كتيرة , لان احتمالية الوصول لـ **local** بيكون اكبر بكثير , من العدد الكبير لـ **clusters**
- طبعا انك تعتمد على العامل البصري عشان تقيم مين فيهم افضل , لان النتائج بتبقى ضخمة جدا , وشكلها قريب من بعضه , فمش هتعرف تميزها الا بالارقام , زي كدة



- عدد المجموعات :

- اختيار عدد المجموعات يخضع لعدد من العوامل و الخطوات الهامة
- احيانا يتم اختيار عدد المجموعات بالنظر ، و عبر رؤية البيانات التي تم توزيعها على محوري X_1 ، X_2 ، فيمكنك تمييز العدد التقريري للمجموعات

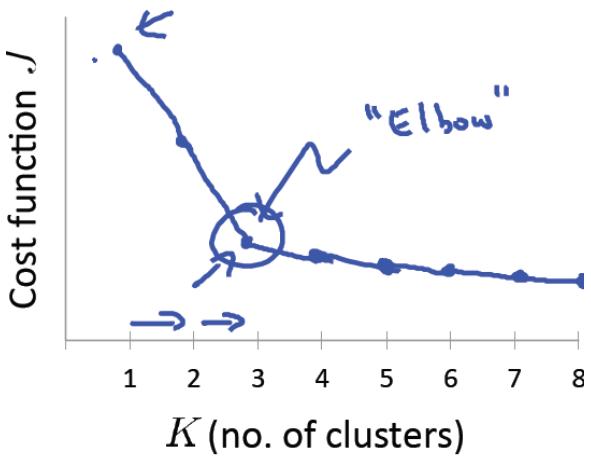
- بس مش دايما يكون سهل نجيب العدد ، يعني البيانات ديه :



- هنا متقسمين مجموعتين ، بينما ممكن يتقسمو بسهولة تلات او اربع مجموعات ، و هتظل البيانات سليمة

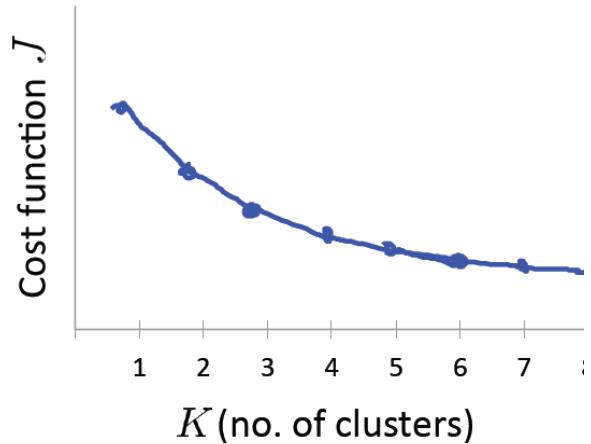
- واحد الطرق المستخدمة لحساب العدد المناسب ، هي طريقة الكوع Elbow method

- والمقصود بيها ، اني ممكن اعمل جراف بين عدد المحاولات الممكنة ، وقيمة L في كل مرة ، و اختيار العدد المناسب ، الي من ناحية يكون فيه قيمة K مناسبة ليها (عدد معقول من المجموعات) و في نفس الوقت يكون قيمة الـ L قليلة بما يكفي
- ولما ارسم ، هشوف القيمة اللي ه تكون عاملة زي الكوع ، بحيث فيها تقليل جامد ، بعدها نسبة التقليل بتكون اهدي



○ فمن الواضح هنا ان قيمة 3 هي العدد المناسب اللي ممكن اختاره

○ بي خد بالك انك لما تعمل طريق الكوع , ممكن ميكونش الكوع واضح بما في الكفاية زي كدة :

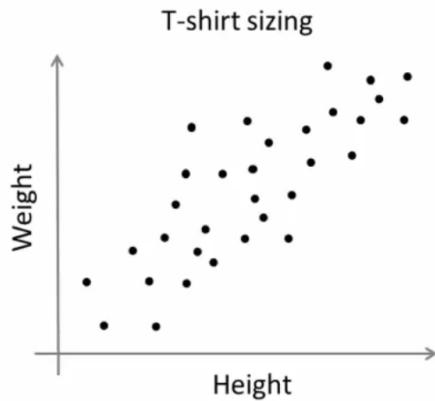


○ ساعتها اختار رقم تقريري , بما يتناسب مع احتياجاتي

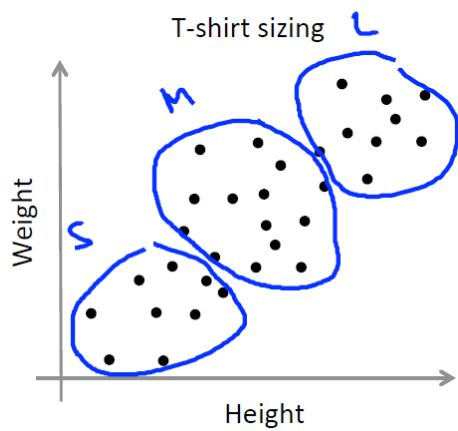
● خطوة مهمة لازم تعملها بعد ما تقسم المجموعات , وهي التقييم الفعلي للسوق

○ يعني انا دلوقتي اخترت اني اقسم البيانات عندي لـ 3 مجموعات , بعد ما اعمل كدة , ابدا اشوف العوامل الثانية المتعلقة بالسوق , واللي ملهاش علاقة بالـ ML زي تكلفة انشاء الحاجة ديه , و مدي رضاء العملاء عليك , وتأثيرها على السوق , ومدي منافسة المنافسين لديك و هكذا

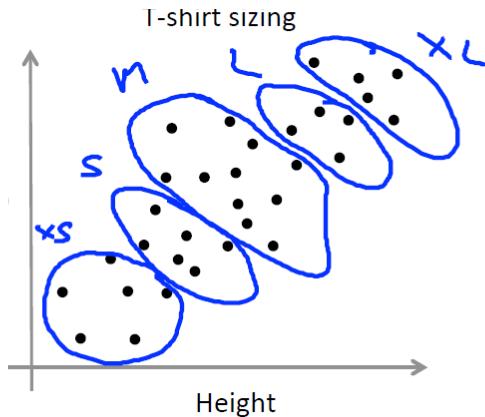
○ فمثلا هنا , عندا مجموعة من البيانات الخاصة بطول و وزن العملاء , اللي عايزين نصمم تيشيرتات على اساسها ,
فيما تري الافضل نقسم كام مجموعة ؟



- ممكن نقسم 3 مجموعات زي كدة S , M , L



- أو ممكن نقسمهم 5 مجموعات زي كدة XS , S , M , L , XL



- و اتخاذ القرار ده افضل ولا ده افضل يعتمد علي : سعر عمل نماذج للتيشيرتات ، وتكلفة عمل احجام مختلفة ، ومدى رضي العملاء ، وتنوع المنافسين ، و هكذا

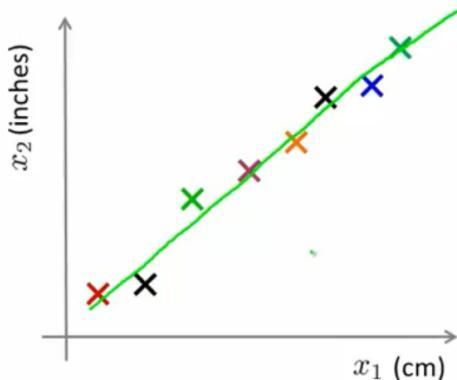
- كدة احنا عرفنا ايه هو الـ K mean تعالى نتعرف على نوع تاني اسمه dimensionaly reduction
-

• تقليل الأبعاد dimensionaly reduction

- هناك عدد من المميزات التي تجعل استخدام تكنيك تقليل الأبعاد بالغ الأهمية , مثل ضغط البيانات و ضغط البيانات لا يستخدم لتقليل المساحات المستخدمة على الهاارد ديسك , لكن لعمل تسريع اكثر للعملية الحسابية نفسها
- الفكرة :
- فكرة تقليل الأبعاد , معناها الغاء عدد من الـ features لأنها مرتبطة بـ features تانية موجودة بالفعل
- يعني لو عندي مثلاً من ضمن الـ features الخاصة بسيارة , هو طولها بالقدم , وطولها المتر , وزنها بالكيلو , وزنها بالرطل , و هل هي اوتوماتيك ولا مانوبل , وهي فيه دبرياج ولا لا
- فمن الواضح ان الطول بالقدم هو مرتبط بالمتر , والوزن بالكيلو مرتبط بالرطل , و ان لو هي اوتوماتيك , فاكيد مش هيكون فيها دبرياج
- فممكن الغي عدد من الـ features اللي فيه features تانية مرتبطة بيها , اختصاراً لوقت الـ processing
- خد بالك ان الاصل انك متعملاً بالغاء الا لعناصر مرتبطة ببعض 100% (زي تغيير الوحدات من متر لقدم) , الا ان ممكن يحصل تجاوز كنوع من تخفيض المعلومات اني احذف احد الـ features اللي مرتبطة بشكل كبير في البيانات عندي بعنصر تاني , حتى لو مكانتش 100%
- يعني لو عندي feature خاص بالحي اللي ساكن فيه الطالب , و feature تاني هو الحالة المادية لاسرتة , ولو لقينا ان دايماً كل اللي ساكنين في التجمع اغنياء , وكل ساكنى امية فقراً , وكل ساكنى الهرم من متوسطي الحال , ساعتها ممكن احذف feature العنوان تماماً , لأن بالفعل فيه feature تاني بيعبر عنه بشكل كبير
- صحيح ان التعبير مش 100% , لكن قد يكون احياناً تخفيض البيانات و تسريع العملية , اولي من الحصول على زيادة في الكفاءة , والفرق بسيط

• اختصار الارقام ممكن يكون من بعدين بعد واحد , او من 3 ابعاد لبعدين

- من بعدين بعد واحد معناها : لو عندنا اتنين features مش مرتبطين ببعض 100% , لكن بنسبة معينة , وعايزين نحولها لـ feature واحد , فنشوف الرسم هنا هيكون عامل ازاي

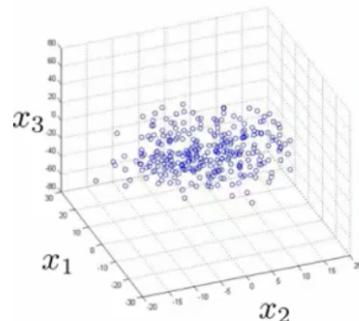


- فلو مسكتنا الخط الـ best fit ليهم , و فرشناه خط افقي , هنلاقي الخط كدة

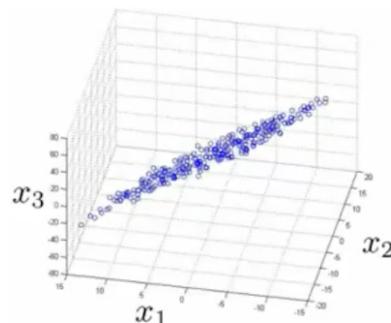


- وبالتالي ، بدل ما كان عندنا قيمتين X_1 ، X_2 ، فعندها قيمة واحدة و هي Z_1 اللي بتعبّر عن القيمتين
 - ولو واحد بالك رياضيا، قيمة Z_1 هتساوي $(X_1 + X_2)^2$ لنقطة 1 ونفس البيانات لباقي الزدات
-

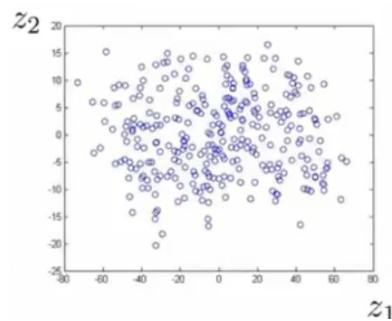
- عرفنا من شوية ازاي حول من 2D لـ 1D ، تعالى نعرف ازاي حول من 3D لـ 2D
- فلو مثلاً شفنا هنا ، نقط موجودة في 3 أبعاد ، اكس 1 و 2 و 3



- وقلنا اننا عايزين نقلها لبعدين بس ، فممكّن نلغي X_3 بنفس التكتيك اللي فات لالغاء بعد ، عشان يكون رسم مسطح فقط إكس 1 و 2 ، بحيث تكون عاملة كدة في الـ 3D



- او تكون عاملة كدة في الـ 2D



- و ساعتها الابعاد بدل ما كانت اكس 1 , 2 , 3 , هي دالة في اكس 1
3 , 2 ,

- و هنا كل زد اية هتكون عبارة عن فيكتور , زد 1 , و زد 2

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

- لاحظ ان اللي شفناه هو تحويل بسيط , من 2 لـ 1 , او من 3 لـ 2 , لكن واقعيا , عدد الـ features ممكن يكون ارقام ضخمة , ونكون عايزين نحول من 1000 لـ 100 مثلا , و ده هيكون بتكتيكات مختلفة , لكن معتمدة على نفس الافكار
- (مشاهدة فيديو تحويل الابعاد)

الـ Visualization

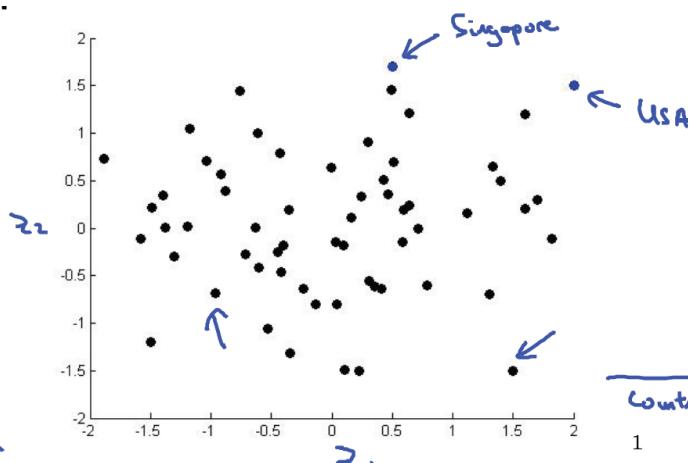
- وهو الهدف الثاني والهام من عملية الـ reduction
- اصلا من أسس عملية الـ ML هو فهم الداتا بشكل جيد , ولن تفهم الداتا مالم يتم فراحتها بصرريا جيدا , لأنه ليس باستطاعة انسان فهم جدول مكون من الف صف و الف عمود
- تعالي نشوف المثال ده

Data Visualization		x_1	x_2	x_3	x_4	x_5	Mean household income (thousands of US\$)	
Country		GDP (trillions of US\$)	Per capita GDP (thousands of intl. \$)	Human Development Index	Life expectancy	Poverty Index (Gini as percentage)		...
Canada	⇒	1.577	39.17	0.908	80.7	32.6	67.293	...
China		5.878	7.54	0.687	73	46.9	10.22	...
India		1.632	3.41	0.547	64.7	36.8	0.735	...
Russia		1.48	19.84	0.755	65.5	39.9	0.72	...
Singapore		0.223	56.69	0.866	80	42.5	67.1	...
USA		14.527	46.86	0.91	78.3	40.8	84.3	...

- لو عندي جدول بالشكل ده , وفيه خمسين عمود (يعني 50 feature) و عشرين صف (m) فمن المستحيل اني اقدر ارسم جراف من 50D
- فيكون الحل اني اعمل للابعاد , عشان نقل من 50 لـ 2 و بالتالي اقدر ارسمها

Country	$z_1 \leftarrow$	$z_2 \leftarrow$	$\mathbf{z} \in \mathbb{R}^2$
Canada	1.6	1.2	
China	1.7	0.3	Reduce data
India	1.6	0.2	from 500
Russia	1.4	0.5	to 2D
Singapore	0.5	1.7	
USA	2	1.5	

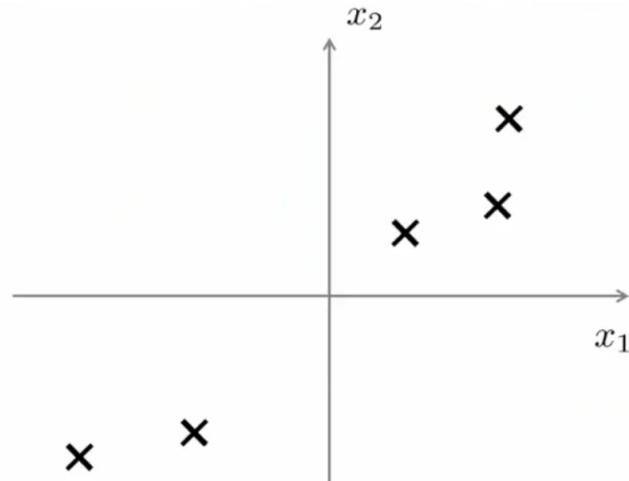
- فعمل بروسينج ، عشان يا اما احذف بعض العواميد المتكررة او اللي باهمية قليلة ، او اعمل دمج بين عمودين بعملية حسابية معينة ، لغاي لما اوصل لعمودين بس ، فممكنا ارسمهم كدة



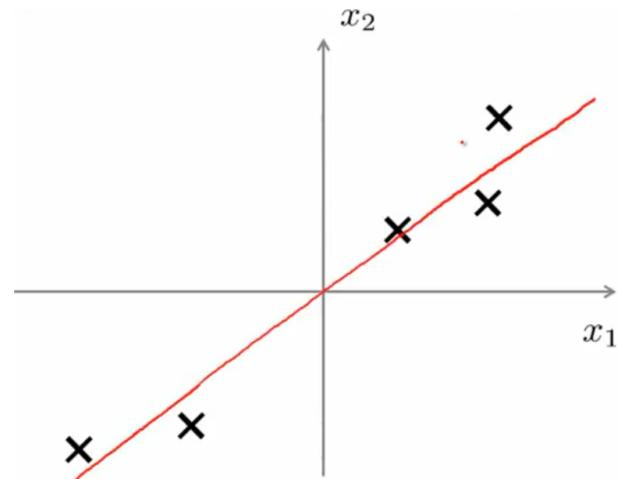
- فهنشوف z_1 في محور اكس ، بينما z_2 في محور واي ، والدول (اللي هي كانت m) هتنقى اسماء النقط عندي و الحقيقة ان المعضلة الاساسية في الخطوة ديه ، هي اختيار ايه العواميد اللي هتتحذف ، وايه اللي هنسبيه ، وايه اللي هندمجه ، وباي طريقة ، عشان تمون البيانات الجديدة المدمجة بالفعل معبرة عن البيانات الاصلية ، وبالتالي اقدر اعمل classification او regression

• principal component analysis

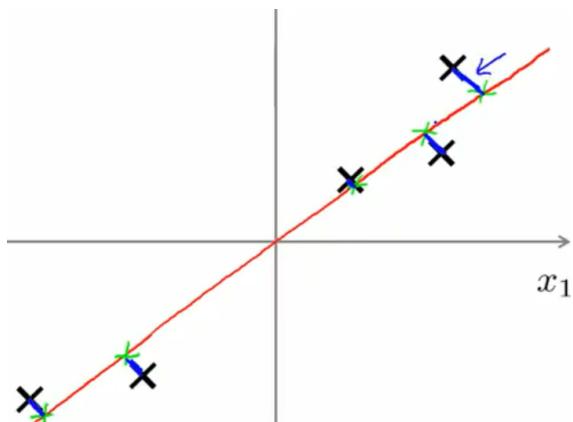
- عشان نفهم معناها ، تعالي نشوف المثال ده :
- لو عندي نقط زي كدة ، وعايز اعملها تقليل من 2D لـ 1D



ساعتها هرسم خط كدة يمثلهم بقدر الامكان

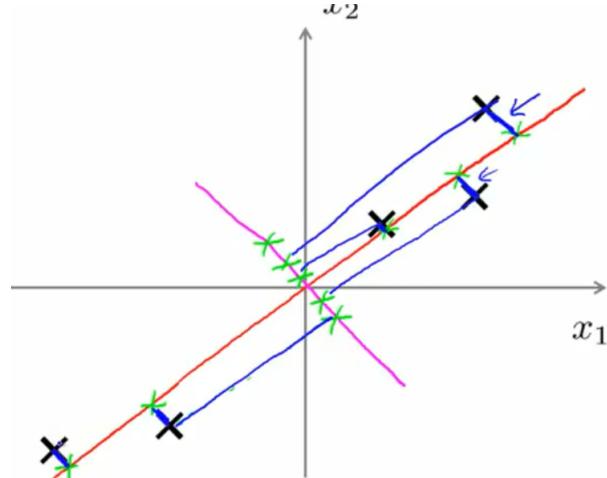


و كان النقط الخضرا ، هي تمثيل النقط السوداء على الخط الاحمر الـ best fit line ، بينما الخطوط الزرقاء هي المسافات بين القيمة الحقيقة و المتوقعة و اسمها projection error



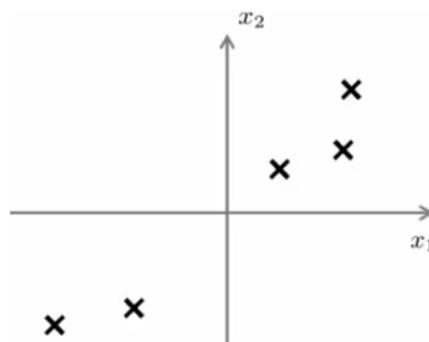
دور الـ PCA انها بتحاول تجييب افضل خط ، اللي يخلی المسافات الزرقاء اقل ما يكون

- طيب لو عملنا خط يمثّلهم بس يكون غلط تماماً (عشان نفهم الفكرة) ، مثلاً زي كدة :

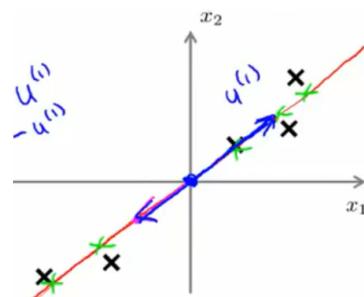


- الخط البدائي ، لو هعمله عشان يمثّلهم ، واضح انه غير معتبر تماماً ، وباعرف ده ، عن طريق لما بجيّب اسقاطات كل نقطة من النقط السوداء ، على الخط البدائي ، بالخطوط الزرقاء اللي واضح انها طويلة جداً
- فالعبرة عندنا بالخط اللي يعمل اقل مجموع للخطوط المتتساقطة عليه ، وطبعاً لو كان مجموع النقط ديه بصفر ، يعني كدة ان الخطوط اللي واقعة كلها باصفار ، يعني الخط الـ *best fit* مر بيهم كلهم

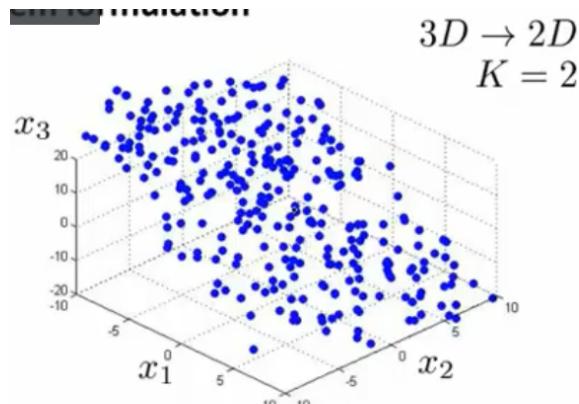
- وبالتالي انا لو عندي مجموعة من النقاط زي كدة (بعدين)



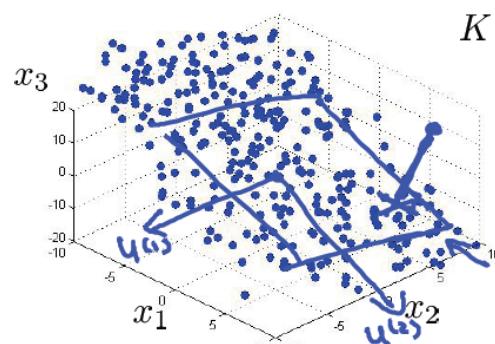
- هدف الـ PCA انها تجيّب فيكتور هنسميه u_1 والّي هيجّم النقاط مع بعض (التحولت بعد واحد)



- كذلك الحال، لما بيكون عندي بيانات من 3 بعد ، بحوالها لبعدين عن طريق ايجاد اتنين فيكتور



- بالشكل ده : هنجيب u_1 و u_2 وللي من خلاهم نقدر نرسم الشكل ده
- لاحظ ان هنا u_1 ماتشية مع x_2 بالضبط , بينما u_2 ماتشية مع x_1 و x_3 , بس ده مش دائما لكن ممكن في رسم تاني تكون العلاقات مختلفة

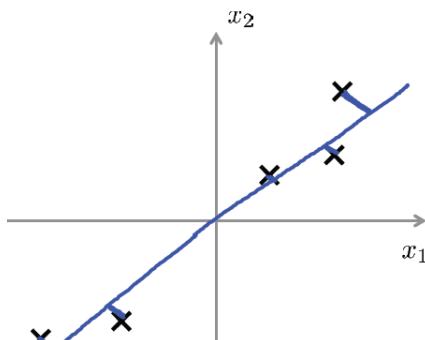


- اذن في الحالة العامة , لو عندي عدد ابعاد b عايز احولها لعدد ابعاد k هدور علي عدد k من الفيكتور
- وزي ما قلنا في الخط , ان هدف الـ PCA انها تجيب افضل خط اللي يقدر يقل جدا الابعاد بين باقي النقط و الخط بتاعنا , كذلك في المستوى الهدف هو الوصول للمستوى الامثل , اللي يخلي كل المسافات بين اي نقطة و المستوى ده اقل ما يكون

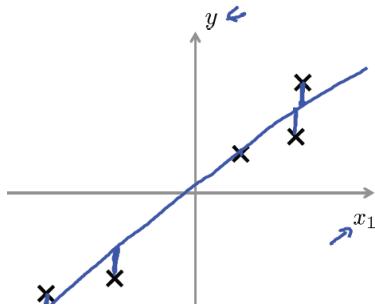
الـ Linear Regression والـ PCA •

- مع تشابه الشكلين , مش عايزين نتلغبط بينهم , ومننساش ان دول تكنيكين مختلفين تماما

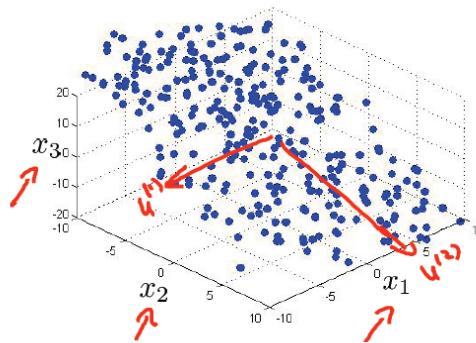
- الـ PCA بيحسب مسافات النقط اللي نازلة عمودي علي الخط



- بينما والـ Linear Regression بيحسب المسافة الرأسية بين النقط و الخط



- كمان الـ PCA عشان هو Unsupervised فمفيش قيمة للـ y يعني كل النقط زي بعض
- بينما الـ Linear Regression فيه قيمة للـ y لانه
- ده غير ان الـ PCA ثلثي الابعاد , مفيهوش قيمة للـ y لكن هي 3 اكسات بنفس المستوى , اما الـ Linear Regression بتكون فيه اتنين اكس والمستوى الراسى بيكون y



- خطوات تنفيذ الـ PCA

- أول خطوة هو عمل ما يسمى الـ mean normalization

■ و ديه معناها , اني عايز الارقام تبقى قريبة من بعضها

■ فأول خطوة اني همسك أول feature عندي و اجيب متوسط قيمه (المجموع على العدد)

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

■ بعدها هستبدل قيمة كل إكس بإكس ناقص المتوسط $x_j - \mu_j$.

■ كدة لو كانت الارقام متافرة او كبيرة كدة , فانا عملت تبسيط و تقريب ليها , وفي نفس الوقت حافظت على النسب بينهم من غير ما اغيرها

■ ولو عندي features بعيدة عن بعض , فلازم اقربهم من بعض , يعني لو عندي feature سعر البيت

■ اللي بيبيدا من 100 الف لـ 3 مليون , و feature تاني مساحة البيت اللي بيبيدا من 70 لـ 200

■ ساعتها اعمل العملية ديه :

$$x_j^{(i)} \leftarrow \frac{z_j^{(i)} - \bar{z}^{(i)}}{\sigma_z}$$

- و معناها ان قيمة كل x تساوي x ناقص المتوسط ، علي الرينج ، واللي هو اكبر قيمة ناقص اصغر قيمة
- فلو عندي قيمة x هي 100 الف دولار (سعر) ، والمتوسط 80 الف ، وارخص بيت هو 50 ، واغلي بيت 130 ، فهطرح الـ 100 ناقص 80 ، تساوي 20 الف ، بعدها اقسمها علي الرينج (80 الف) تبني قيمة ربع بس (0.25) فالارقام الضخمة قلت فورا
- فلما اعمل العملية ديه لكل feature فيهم ، هتلقي ارقام الاسعار قريبة من ارقام البيوت ، فاقدر ارسمهم و اقارن بينهم بسهولة

○ طيب بعد ما القيم قلت ، أجياب ما يسمى covariance matrix المصفوفة ديه مثـن محتاجين نفهم بالكامل معناها ايه ، بس يهمنا نعرف المتعلق بيها الرمز بتاعها هو Σ اللي هو شبه سميـش و اسمه Sigma قيمتها بالمعادلة :

$$\Sigma = \frac{1}{m} \sum_{i=1}^n (x^{(i)}) (x^{(i)})^T$$

حيث m عدد الصفوف ، و n عدد الاعمدة features

$$\underline{\Sigma} = (1/m) * \underline{X}^T * \underline{X}$$

- و ممكن تكتب في اوكتيف كدة
- أما فيكتور x فهو يعبر عن قيمة صف كامل (يعني كل الـ features الخاصة بدولة فرنسا مثلا) فهـيكون فيكتور قيمته n في 1

بعدـها هـقـوم بـحـساب الـ eigenvectors للمصفوفة عن طـرـيق استـخـدم الـ اـمـرـ المـباـشـرـ فيـ المـاتـلـابـ :

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

كلـمة Sigma هـنـا اللي هي Σ اللي هو شـبـهـ سـمـيـشـ و رـمـوزـ svdـ تـشـيرـ لـ : singular value decomposition مـمـكـنـ استـخـدمـ بـدـلـ دـالـةـ svdـ دـالـةـ تـانـيـةـ اسمـهـ eigـ فيـ المـاتـلـابـ . لكن svd اـفـضلـ مـتـسـاشـ ان Sigma اـبـعـادـها nـ فيـ nـ لـانـهاـ مـضـرـوبـ فيـكتـورـ xـ الليـ هو nـ فيـ 1ـ ،ـ فيـ التـرانـزـبوـسـ بـتـاعـهـ ،ـ فـهـيـكونـ nـ فيـ nـ

عـنـديـ منـ الدـالـةـ svdـ تـلـاتـ مـخـرـجـاتـ ،ـ هيـ Uـ ,ـ Sـ ,ـ Vـ وأـهـمـهـمـ الـ Uـ (ـلـاحـظـ اـنـناـ مـيـهـمـنـاشـ الـ عـلـمـيـةـ الحـاسـابـيـةـ تـمـتـ اـزـايـ ،ـ يـهـمـنـاـ بـسـ المـخـرـجـ وـ نـتـعـالـمـ مـعـاهـ اـزـايـ)

مـصـفـوـفـةـ الـ Uـ هـتـكـونـ مـصـفـوـفـةـ nـ فيـ nـ يـعـنـيـ بـنـفـسـ رـقـمـ عـدـدـ الـ اـعـمـودـ فيـ الـ مـدـخـلـاتـ xـ لوـ اـنـاـ عـاـيـزـ اـعـمـلـ تـقـلـيلـ لـعـدـدـ الصـفـوـفـ مـنـ nـ لـ kـ سـاعـتـهـاـ اـخـدـ اـولـ عـوـامـيدـ فيـ مـصـفـوـفـةـ Uـ بـعـدـ kـ

■ فلو عدد العواميد في البيانات الاصلية 50 , وعايز اقللها لعشرين , هتطلع الـ U مصفوفة 50 في 50 , ساعتها اخذ اول عشرين عمود بس , يعني المصفوفة الجديدة هتكون 50 في 20

$$U = \begin{bmatrix} | & | & | & | \\ u^{(1)} & u^{(2)} & u^{(3)} & \dots & u^{(k)} \\ | & | & | & | \end{bmatrix} \quad U \in \mathbb{R}^{n \times n}$$

$$u^{(1)}, \dots, u^{(k)}$$

■ وقتها قيمة الـ Z المطلوبة (البديل لـ X بعد التقليل) هتساوي

$$Z^i = U^T \cdot X^i$$

■ يعني مصفوفة Z ه تكون ترانزبوس دالة الـ U (الـ U هي اللي فوق بعد ما قللناها لـ k) مضروبة في مصفوفة X وه تكون كدة :

$$Z^i = \underbrace{\begin{bmatrix} | & | & | & | \\ u^{(1)} & u^{(2)} & u^{(3)} & \dots & u^{(k)} \\ | & | & | & | \end{bmatrix}}_{n \times k}^T \cdot \underbrace{X^i}_{k \times 1} = \underbrace{\begin{bmatrix} (u^{(1)})^T & & \\ \vdots & \ddots & \\ -(u^{(k)})^T & & \end{bmatrix}}_{k \times n} \cdot \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{k \times 1}$$

■ مع مراعاة ان :

- اول مصفوفة على الشمال اسمها U_{reduced} يعني المصفوفة U المقللة

- ابعاد U هي $n \cdot k$

- ابعاد U^T ه تكون $k \cdot n$

- ابعاد X^i هي $n \cdot 1$

- اذن Z لما تتضمن ه تكون $1 \cdot k$ وده يتوافق مع المطلوب , انا عايزين عدد صافوف k (الـ U هو العدد المطلوب بعد التقليل) وعمود واحد

- اذن ممكن الكود للخطوات الاخيرة يكون كدة :

$$[U, S, V] = \text{svd}(\text{Sigma});$$

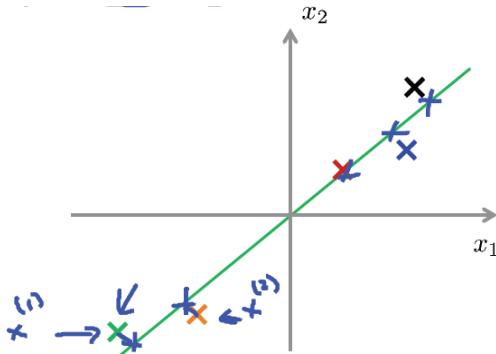
$$U_{\text{reduce}} = U(:, 1:k);$$

$$z = U_{\text{reduce}}' * x;$$

● التكبير :

- والمقصود بيه العملية العكسية لضغط البيانات اللي قلناه من شوية , ان كنا قبل كدة قلنا البيانات من 1000 عمود , لـ 100 عمود , فعايزين نعرف اي نقدر نكبر البيانات المضغوطة من 100 لـ 1000

- فلو ان عندي نقط مرسومة في بعدين زي كدة :



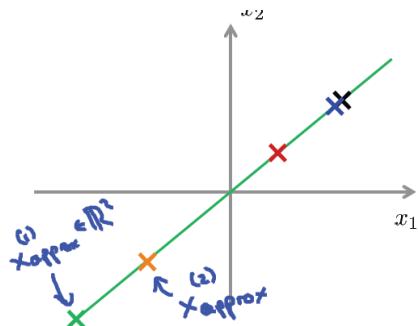
- فممكن احولها بعد واحد عن طريق المعادلة اللي خدناها ديه :
- تكون شكلها كدة :



- والعكس برضه , لو عندي بيانات مرسومة بخط واحد زي هاستخدم المعادلة ديه عشان احولها لبعدين :

$$\mathbf{X}^1_{\text{Approx}} = \mathbf{U}_{\text{reduced}} \cdot \mathbf{Z}^1$$

- لاحظ ان $\mathbf{U}_{\text{reduced}}$ فيكتور n^*k و إن الـ \mathbf{Z}^1 قيمته k^*1 هيساوي 1^*1
- وهترسم كدة :



- لاحظ ان النقط هنا واقعة علي الخط بالضبط , مش متباعدة شوية عن الخط , زي الرسم الاصلي , وده السبب اتنا بنقول ان الإكس approx , لأن ديه قيم تقريرية ليها , مش الحقيقية (مستحيل نجيب الحقيقة)

• تحديد قيمة K

- و تسمى number of principal components وآلية تحديدها هامة , لأن علي اساسها هاقدر اقول هي هتكون كام عمود بعد التقليل
- عشان نعرف هنحددها ازاي , تعالي الاول نعرف هو تكنيك الـ PCA بيعمل ايه
- الـ PCA بيحاول يقلل ما يسمى average squared projection error اللي هو بيتعمل بالقانون :

$$\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{\text{approx}}^{(i)}\|^2$$

- يعني مجموع مربعات الفروق بين اي قيمة X و القيمة المتوقعة ليها (اللي واقعة على الخط الـ best fit) مقسومة على m

○ كمان يهمني احسب قيمة الـ variation يعني تتنوعها من الاصفار واللي هتكون بالقانون

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

- الشرط بقى ان حاصل قسمة ده على ده يكون اقل او يساوى 0.01

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

- يعني من الاخر ، تكون قيمة محصلة الفروق اقل من 1% من التنوع variance عشان نضمن داتا قوية
- وتنقال بطريقة تانية ، ان 99% من البيانات رجعتلينا

“99% of variance is retained”

- وارد في احياناً تانية ان القيمة بدل ما تكون اقل من 0.01 ، تبقى اقل من 0.05 ، ساعتها هيبي محصلة الفروق 5% ، وبيفي النسبة اللي رجعتلينا 95% ، واضح ان ده كفائه اقل

- برضه ممكن يكون الرقم 0.1 ، يعني النسبة 10% ، اللي رجع 90% و ده اوحش

- في الحياة الطبيعية لأن كتير من البيانات بتكون مرتبطة بعض ، فممكن اعمل ضغط كبير للبيانات ، تظل نسبة الـ variance اللي هترجع كبيرة 95% مثلاً ، لكن لو كانت البيانات غير مرتبطة بعض ، فلما اقل البيانات ، الكفاءة بتقل

● طيب او اي احسب قيمة K :

- احنا هنجرب الاول قيمة $K=1$

- نجيب قيم $X^2, Z^1, Z^2, \dots, X^1$... وهكذا

Try PCA with $k = 1$

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

- بعدها نحسب قيمة النسبة و نشوف بتتساوي اقل من 0.01 ولا لا

Check if

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

- لو طلعت فعلاً بـ 0.01 بيفي هايل ، لو اكابر ، اخلي الـ $K=2$ و اجيب نفس القيم و احسب تاني

- لو طلعت اكابر من 0.01 ، ازود قيمة K لغاية لما اوصل للبعد المناسب

- لاحظ ان كل ما الـ K تقل يعني عدد العواميد المستخدم اقل ، يعني كفاءة اقل ، فلما تزيد بيفي كفاءة احسن ، بس

الرسم اصعب

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

، وهي :

● لاحظ ان فيه طريقة اسهل شوية لحساب القيمة الكبيرة ديه

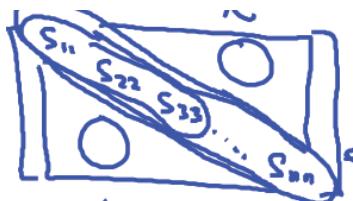
● لما باستخدم في اوكتيف دالة svd بيكون عندي تلات مخرجات U , S , V

● احنا قبل كدة استخدمنا بس الـ U بينما الـ S هت تكون مفيدة معانا هنا

● الـ S عبارة عن Diagonal Matrix يعني هي مربعة ، وكل قيمة باصفار ما عدا القطر زي كدة

$$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 & 0 \\ 0 & 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 11 \end{pmatrix}$$

● مصفوفة الـ S بتكون بالشكل ده :



○ ابعادها بتكون $n \times n$ و قيمها بتكون $S_{11}, S_{22}, S_{33}, S_{44}, S_{nn}$

○ فلو انا جمعت قيم الإسات ، من 1 لـ K (اللي هي عدد العواميد بعد الضغط) ، وقسمتها على مجموع الإسات من 1

لـ n (عدد العواميد الأصلية) ، وطرحتها من 1 هتدييني قيمة مستاوية لـ $\frac{\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2}$ بس بطريقة اسهل

$$1 - \frac{\sum_{i=1}^K S_{ii}}{\sum_{i=1}^n S_{ii}}$$

○ فلو عندي عدد الصفوف الاصلية $n=6$ و عايز اشوف انهي افضل K مناسبة ليها ، نقول الـ S هت تكون كدة

$$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 7 \end{pmatrix}$$

○ نفرض اولا قيمة K بـ 1 فهيا تكون الحساب كالتالي :

$$1 - \frac{2}{2+3+1+5+6+7}$$

◦ و نشوف القيمة كام , بعدها نقول الـ $K = 2$, هيكون الحساب

$$1 - \frac{2+3}{2+3+1+5+6+7}$$

◦ و نزود قيمة K تدريجيا

◦ لاحظ ان كل ما الـ K تزيد , كل ما البسط يزيد , ولأن المقام ثابت (عدد كل قيم الـ S) , فقيمة الكسر تزيد , فالقيمة الكلية (1 ناقص الكسر) تقل , لغاية لما اوصل لقيمة 0.01 اللي انا عايزها

$$\frac{\sum_{i=1}^k S_{ii}}{\sum_{i=1}^m S_{ii}}$$

◦ و بالتالي ممكن اعمل check على قيمة $\frac{\sum_{i=1}^k S_{ii}}{\sum_{i=1}^m S_{ii}}$ واشوف لو هي اكبر من او تساوي 0.99

$$\frac{\sum_{i=1}^k S_{ii}}{\sum_{i=1}^m S_{ii}} \geq 0.99$$

(99% of variance retained)

◦ و ساعتها اقدر اقول اني عرفت ارجع 99 % من البيانات

• تطبيق لـ PCA

◦ نتخيل ان عندنا supervised regression (يعني فيه اكس و واي) , مثلا صورة 100 بكسل في 100 بكسل , فكدة هيكون فيه 10 الاف feature للإكس , فيه قيمة للواي زي كدة $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$

◦ طيب كمية الـ features كبيرة جدا , فالـ regression هيكون بطئ , فيكون الحل , اني اعمل تقليل للبيانات عشان اعرف اتعامل معها بالخطوات ديه

■ او لا , اعمل تحويل من unsupervised لـ supervised يعني الغي قيم y دؤفات بحيث تكون كدة

$$x^{(1)}, x^{(2)}, \dots, x^{(m)} \in \mathbb{R}^{10000}$$

■ بعدها اطبق تكنيك الـ PCA عشان يتم تقليل الاعمدة من 10 الاف , لـ 100 مثلا , فهنخلي اسمها z

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

■ بعدها ارجع قيم الولايات , بس للزي , مش الاكس , بحيث الجدول عندي يكون بين z , y مش x

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

■ دلوقتي لو عايز اجرب الـ test data هي ه تكون بالـ x مش z , فلازم الاول احول بيانات الاختبار من الاكسات للزدات , منها اعمل تجريب , واقارن الولايات دلوقتي

■ متنساش ان عدد الصفوف متغيرش , الي اتغير هو عدد الاعمدة , يعني بدل ما كان 500 صف في 10 الاف عمود بقى 500 صف في 100 عمود

- عدم استخدام الـ PCA

- فيه اكتر من حالة يفضل عدم استخدام الـ PCA خلاها منها مثلا محاولة معالجة الـ OF
- تكنيك الـ PCA يقوم بتقليل عدد الـ features , وتقليل عددها يقلل من الـ OF , فممكن ناس تحاول تستخدمه لو لقينا OF في الجراف , و ه خطأ كبير
- والسبب ان تكنيك الـ PCA مش عارف قيمة y فلما هي عمل تقليل للعواميد , هو مش شايف y اساسا , فممكن جدا يضيع بيانات مهمة , و مش هيقدر يقلل الـ OF بشكل كامل
- فلو عندك مشكلة OF ساعتها استخدم طريقة الـ regulation

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

- الحالة الثانية هي استخدامها بدون داعي او سبب قوي
- يعني لو بصينا هنا , هنلاقي الخطوات المعتادة المطلوبة عشان نبني اي سистем ML وهي :

 - Get training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$
 - Run PCA to reduce $x^{(i)}$ in dimension to get $z^{(i)}$
 - 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 $\{(z_{test}^{(1)}, y_{test}^{(1)}), \dots, (z_{test}^{(m)}, y_{test}^{(m)})\}$

- الحصول على البيانات
- تطبيق الـ PCA
- تطبيق تكنيك الـ Logistic regression
- اعمل اختبار للبيانات على الـ test data

- مشكلة الكلام ده , انه بدا باستخدام الـ PCA من غير داعي , وده هيأخذ وقت , وهيقلل من كفاءة الناتج , يعني ممكن في الآخر تيجي تختبر الـ regression بتعاك على بيانات جديدة , متلاقيش نتيجة كويسبة , عشان انت اشتغلت على بيانات تم معالجتها و كفائفتها اقل
 - فالصح انك تلغى خطوة 2 , و تعمل الـ regression بتعاك على الاكسات مش الزدات , يعني على بيانات اصلية مش معمولها PCA
 - في حالة اقيمت المعالجة واقفة تماما , ومش عايزة تشتعل , او بطيئة جدا , ساعتها هتروح للـ PCA و خلاص
-