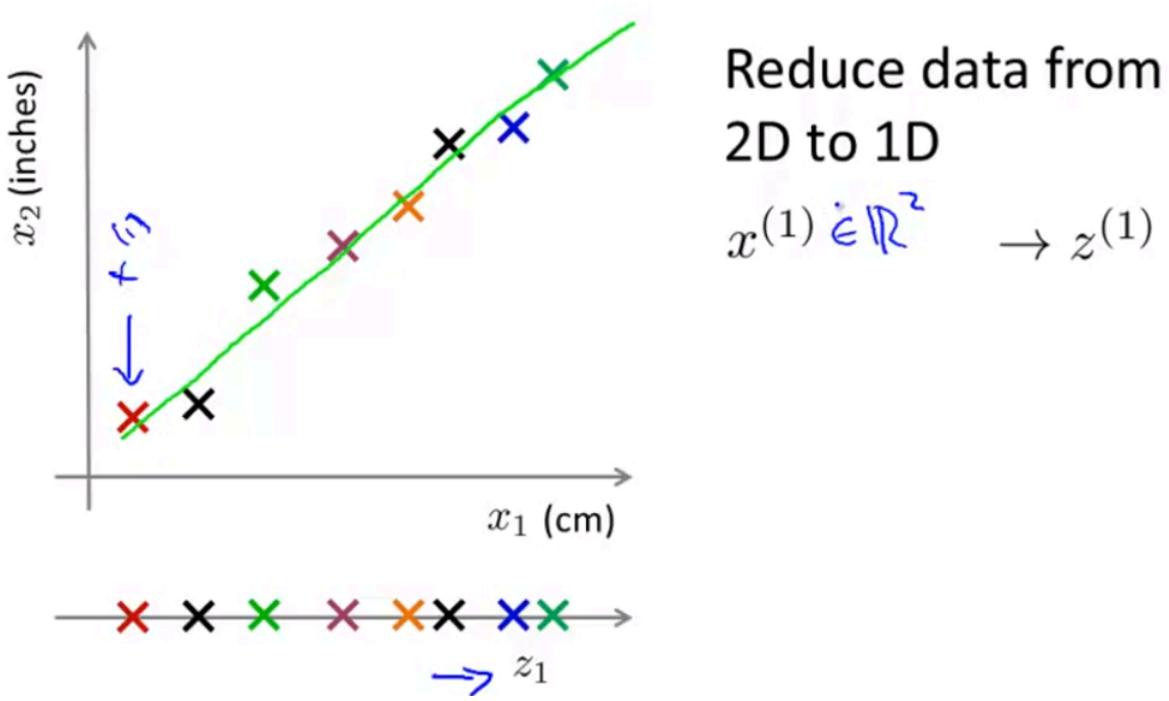
Dimensionality Reduction: è un altro algoritmo di apprendimento non supervisionato che ha lo scopo di ridurre il numero di features descrittive, ovvero rappresentare le informazioni utilizzando un minor numero di coordinare, passare da un piano N-dimensionale ad un piano (N-X)-dimensionale



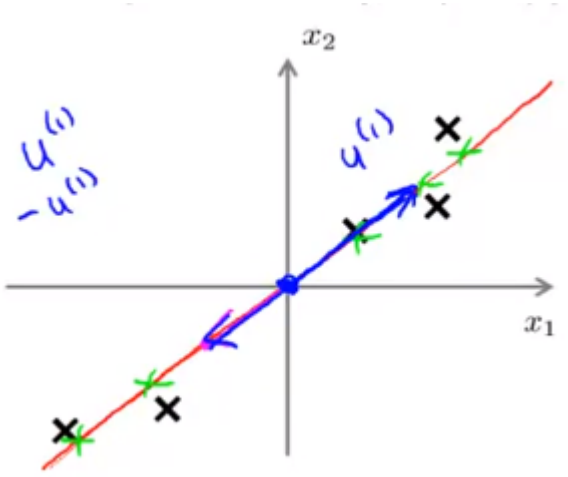
Può essere utile per comprimere i dati, per velocizzare gli algoritmi di apprendimento e per diminuire l' overfitting in alcuni casi (visto che più features usi più tendi all' overfitting). Un altro **IMPORTANTE** utilizzo e quello di sintetizzare le molteplici features di un problema in 2 o 3, cosi da permetterne la proiezione dei dati sugli assi cartesiani al fine di poter analizzare i dati e farsi un'idea sulla loro distribuzione.

Come funziona questo algoritmo (che si chiama PCA)?

Prima di applicare il PCA dobbiamo effettuare mean normalization e feature scaling.
A questo punto si tratta quindi di trovare una superficie di minore dimensione (passare da uno spazio tridimensionale (3 coordinate = 3 features) a uno bidimensionale) sul quale proiettare i dati minimizzando la distanza quadratica tra i punti e il posto in sulla nuova superficie dove verranno proiettati. A quel punto la nuova superficie verrà considerata il nuovo punto di riferimento.

Esempio:
Se ho dei dati su una spazio bidimensionale (solito piano cartesiano) e voglio avere quegli stessi dati su un'unica dimensione, devo trovare la retta che passa attraverso tutti questi dati

minimizzando le distanze e usare quella come fosse il piano cartesiano di riferimento.



Coordinate delle X prima
Xa = (x1 = 2,x2 = 2)

Coordinate delle X dopo:
Xa = (U = 2)

A questo punto potremmo iniziare a descrivere l' algoritmo ma prima dobbiamo effettuare del preprocessing sui dati:

Mean Normalization:

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

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

Feature Scaling:

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

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

Essendo matematicamente molto complesso il processo, ci limitiamo a capire quali funzioni di libreria utilizzare e come farlo. Lo strumento che ci serve è quello della “matrice di covarianza” (il cui simbolo è Sigma, ovvero Σ), che si calcola facendo:

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

nb: $(x^{(i)}) = Nx1$, $(x^{(i)})^T = 1xN$, $\Sigma = (NxN)$

Una volta trovata sigma la useremo per calcolare la “Singlular Value Decomposition” che in Matlab si chiama svd():

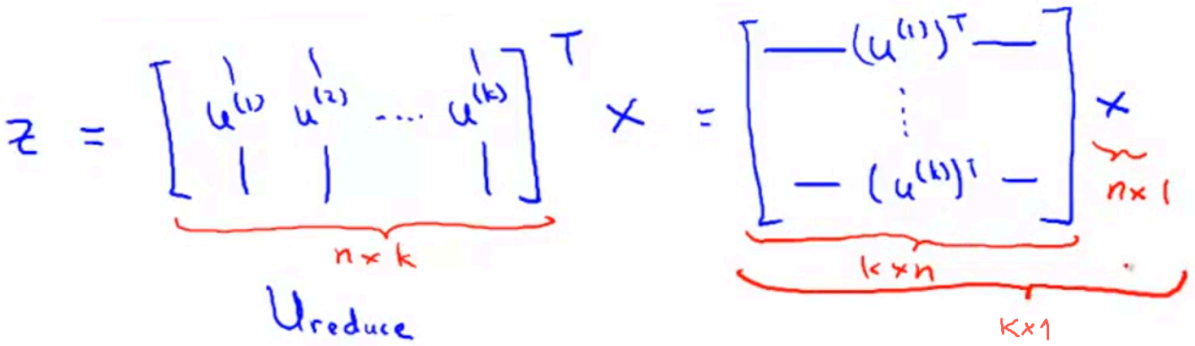
[U,S,V] = svd(sigma) → U = NxN

Ciò che ci interessa è il valore di U, che è una matrice nxn che che contiene tutti gli n vettori che danno la n-dimensionalità, se vogliamo scendere a K dimensioni ci basta raccogliere i primi K-elementi di questi vettori.

From `[U,S,V] = svd(Sigma)`, we get:

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

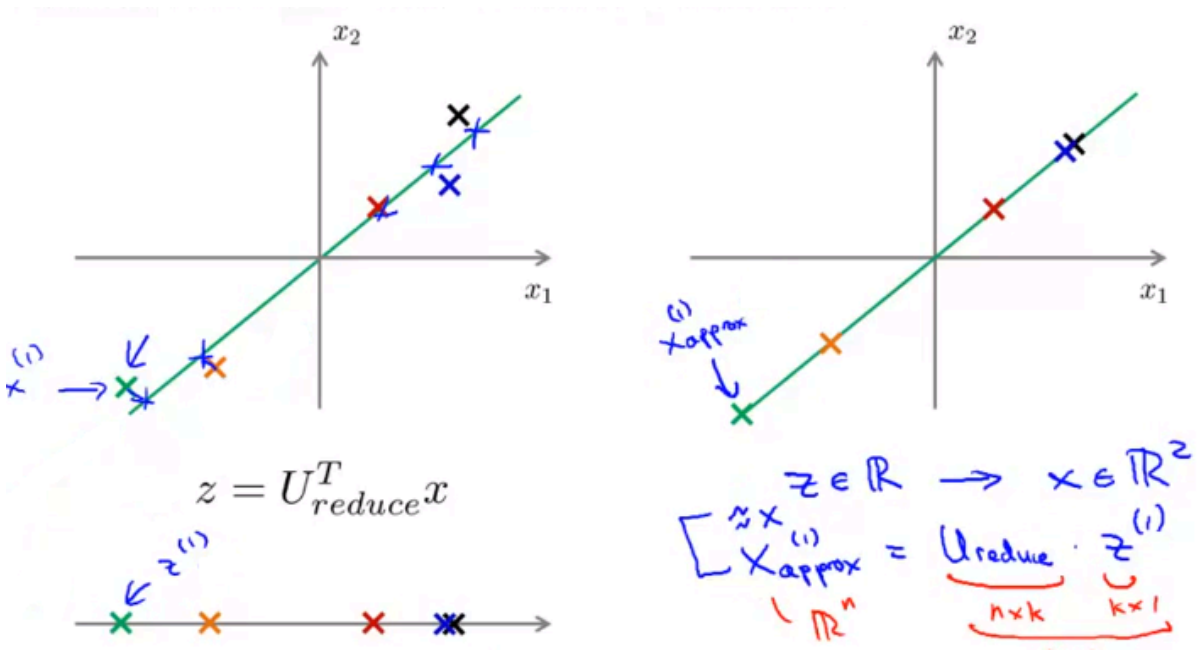
A questo punto possiamo trovare Z (ovvero le nuove X, proiettate sui nuovi assi) semplicemente moltiplicando il Ureduce (ovvero il sottoinsieme dei Ku elementi del vettore U) per i valori X del dataset.



Ricapitolando:

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

```
Per recuperare le X originali (approssimandole) a partire dalle z
trovate:
Xapprox = Ureduce * z
```



Quante dimensioni dovremmo usare? come scegliere correttamente il parametro k?

Esiste una funzione che stabilisce il grado di bontà di una certa k scelta:

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

A questo punto:

- 1) si parte da k = 1,
- 2) Si trova Ureduce, Z, e le Xapprox
- 3) si verifica con la formula appena descritta
- 4) si loopa incrementando k fino a che non si trova un risultato minore dello 0.01.

Trick di ottimizzazione:

Al punto 3 possiamo evitare di utilzzare la formula, piuttosto possiamo utilizzare il parametro S trovato al punto 2 con svd. La matrice S ha tutti elementi a 0 tranne nella diagonale. Lo stesso risultato della formula lo otteniamo facendo 1 - (la somma dei primi k elementi sulla diagonale di D / la somma degli n elementi sulla diagonale di S).

Choosing k (number of principal components)

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

Pick smallest value of k for which

Alt

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

(99% of variance retained)