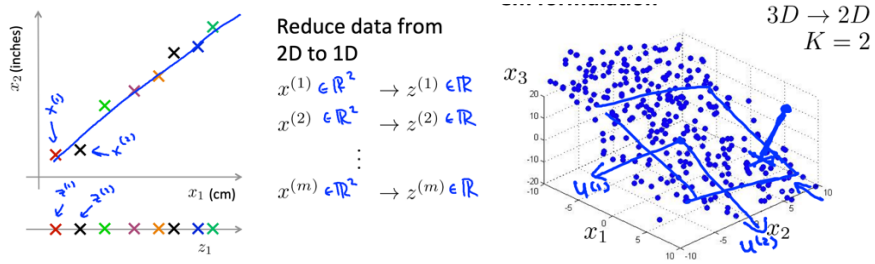
Dimensionality Reduction and PCA:
To reduce data from higher dimensions to a lower dimension.
Usually applicable to:

- ➔ Reduce Memory usage
- ➔ Speed up computation for data learning models
- ➔ Data Visualisation
  - Dimensionality Reduction to $z \in \mathbb{R}^2/\mathbb{R}^3$ allows us to better visualize the data graphically.
  - Hence allowing us to better understand it.

Principle Component Analysis:
Eg:




This is done through the use of 'Eigenvectors'.
For data of n features: $x \in \mathbb{R}^n \rightarrow k\ dimensions$
$we\ find\ k\ vectors: u^{(1)}, \ldots, u^{(k)} on\ to\ which\ we\ project\ data, while\ minimising\ projection\ error$

Data Preprocessing for PCA:

- ➔ **Mean Normalisation** this is critical to make the directional vectors be defined from the origin
- ➔ Feature scaling if some data ranges are massive as compared to others.

How it works:
Compute "covariance matrix": $\Sigma$

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

Then compute eigenvectors of the Covarience Matrix, using:
Eigan computation or svd-> Singular Value Decomposition
[U,S,V] = svm(sigma)
Then:
Index out
U_reduction = U(:,1:k)
z = U_reduction'x
z is the output compressed form of the PCA.
To uncompress the data:

$$x_{approx} = U_{reduce}.z$$

As long as x is relatively close to the projected line.

Choosing the k value:
Measuring the robustness of the PCA is done through

- ➔ Average Squared Projection error $\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)} - x_{approx}^{(i)}\right\|^2$
- ➔ Total Variation in the data $\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}\right\|^2$
- ➔ To choose k to be the smallest val st:

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

Eg threshold = 0.01
⇨ "99% of variance is retained"
Common good variances used are 95% or 90%

Hence to choose k:
PCA from k=1:m
Using S from SVD we can calculate

$$1 - \frac{\sum_{i=1}^{k} s_{ii}}{\sum_{i=1}^{n} s_{ii}} \leq threshold$$

Hence we only need to run SVD once.

Applying PCA

- ➔ For supervised learning speed up
  - Extract inputs:
  - $\left(x^{(1)}, y^{(1)}\right) \ldots (x^{(m)}, y^{(m)})$
  - $Unlabeled\ dataset\ x^{(1)} \ldots x^{(m)} \in \mathbb{R}^{10000}$
  - $=> z^{(1)}, \ldots z^{(m)} \in \mathbb{R}^{1000}$
  - $New\ Training\ set: \left(z^{(1)}, y^{(1)}\right) \ldots (z^{(m)}, y^{(m)})$
  - =>Feed to model

Bad use of PCA: Preventing Overfitting

Reduction of features causes the model to be less likely to overfit the dataset
However it works, but is not a good way to address overfitting.

Regularisation should instead be used as PCA could lose data variances that may be valuable in Error Analysis.

When to apply PCA?
Try running whatever learning algorithm using the raw dataset first, if that does not give the performance that you want, implement PCA and use the compressed form