~ Principal Component Analysis (PCA) Algorithm ~

Before applying PCA, there is a **data pre-processing** step which we should always do.

Given a training set of m unlabeled examples:

- Training set: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$

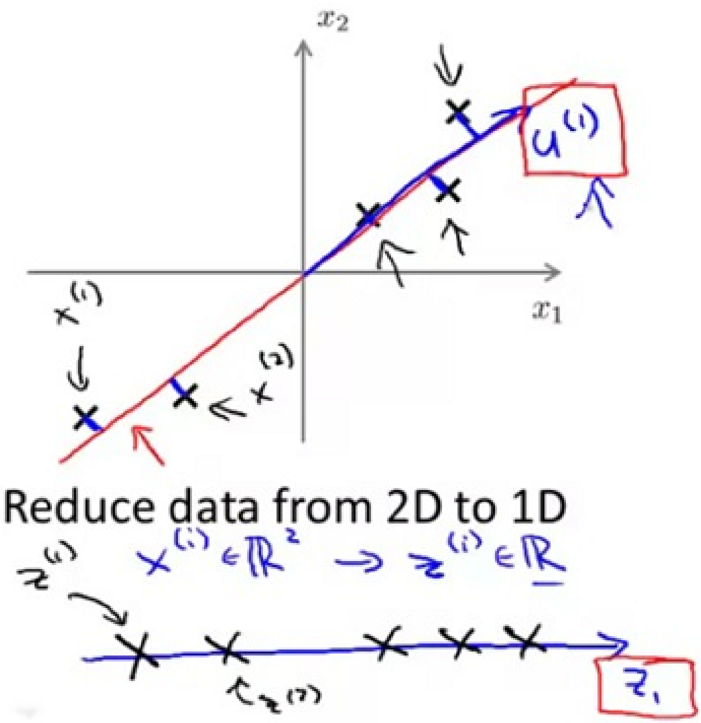
It's important to always perform Preprocessing - **feature scaling / mean normalization**.

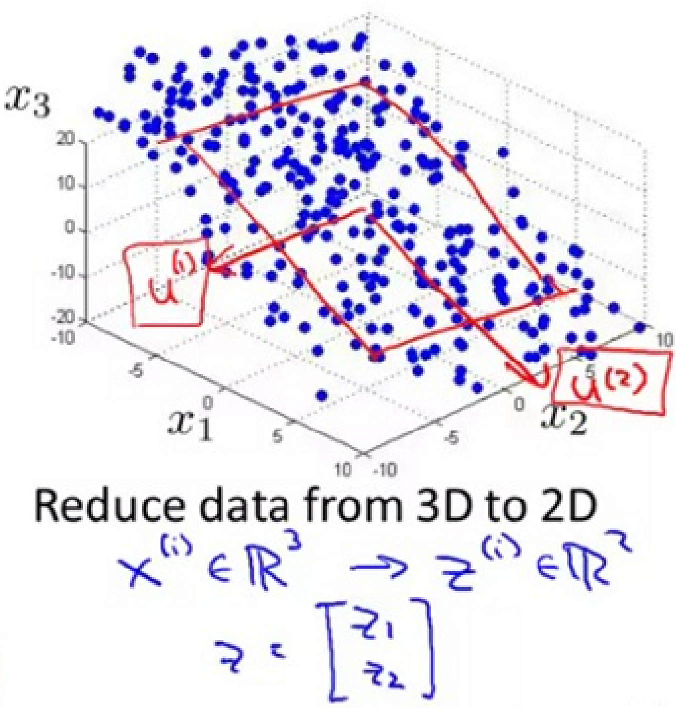
- **Mean Normalization:**
- $\mu_j = \sum_{i=1}^{m} x_j^{(i)}$
- Replace each $x_j^{(i)}$ with $x_j - \mu_j$. In other words, determine the mean of each feature set, and then for each feature subtract the mean from the value, so we re-scale the mean to be 0.
- **Feature Scaling:**
- If different features on different scales (e.g., x_1 = size of house, x_2 = number of bedrooms), scale features to have comparable range of values.
  - e.g. $x_j^{(i)}$ is set to $(x_j - \mu_j) / s_j$
  - Where s_j is some measure of the range, so could be:
    - Biggest / smallest
    - Standard deviation σ (more commonly)

Having done this sort of data pre-processing, here's what the PCA algorithm does:

We previously saw PCA does is, it tries to find a lower dimensional sub-space onto which to project the data, so as to **minimize sum of the squared projection errors**, and so what we wanted to do specifically is find a vector $u^{(1)}$, which specifies that direction or in the 2D case we want to find two vectors, $u^{(1)}$ and $u^{(2)}$, to define the surface onto which to project the data.

- **Need to compute the z vectors:** z vectors are the new, lower dimensionality feature vectors
- A mathematical derivation for the u vectors is very complicated
  - But once we've done it, the procedure to find each u vector is not that hard.





PCA Algorithm description:

Reduce data from n-dimensions to k-dimensions

- Compute "covariance matrix":

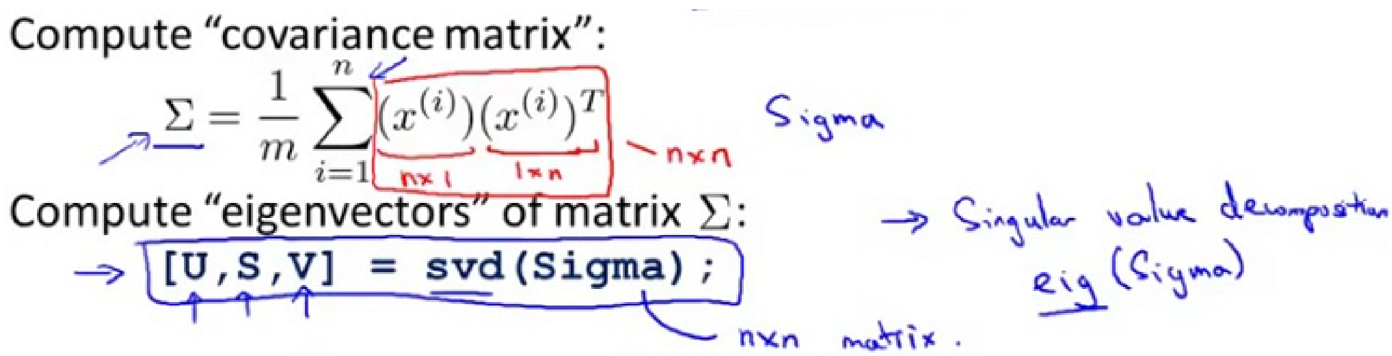
$$\Sigma = \frac{1}{m}\sum_{i=1}^{n}(x^{(i)})(x^{(i)})^T$$

This is commonly denoted as Σ (greek upper case sigma) - NOT summation symbol

- This is an $[n\ x\ n]$ matrix
  - Remember than $x^{(1)}$ is a $[n\ x\ 1]$ matrix

Compute "eigenvectors" of matrix Σ:

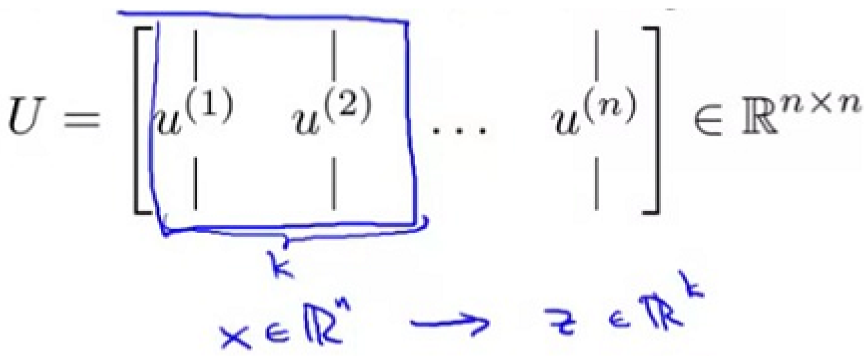
- On octave, the way we do that is to use the command: **[U,S,V] = svd(sigma)**
  - SVD stands for **singular value decomposition**.
    - More numerically stable than eig
    - **eig** = also gives a eigenvector



And what the SVD outputs three matrices, U, S, and V. The thing we really need out of the SVD is the u matrix.

- U matrix is also an $[n\ x\ n]$ matrix ($U \in \mathbb{R}^{n\ x\ n}$)
- Turns out the columns of U are the u vectors we want
  - So to reduce a system from n-dimensions to k-dimensions
  - Just take the first k-vectors from U (first k columns)

From **[U,S,V] = svd(sigma)**, we get:



Next we need to find some way to change x (which is n dimensional) to z (which is k dimensional) $x \in \mathbb{R}^n \rightarrow z \in \mathbb{R}^k$ (reduce the dimensionality).

- Take first k columns of the U matrix and stack in columns
  - $n\ x\ k$ matrix - call this U_{reduce}

We calculate z as follows:

$$x \in \mathbb{R}^n \rightarrow z \in \mathbb{R}^k$$

$$z^{(i)} = \underbrace{\begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(k)} \\ | & | & & | \end{bmatrix}}_{\substack{n \times k \\ U_{reduce}}}^T x^{(i)} = \underbrace{\underbrace{\begin{bmatrix} — (u^{(1)})^T — \\ \vdots \\ — (u^{(k)})^T — \end{bmatrix}}_{k \times n} \underbrace{x^{(i)}}_{n \times 1}}_{k \times 1}$$

$z \in \mathbb{R}^k$

- So $U_{reduce}^T * x^{(i)}$ which is $[k \ x \ n]$ * $[n \ x \ 1]$
- Generates a matrix which is $[k \ x \ 1] = z^{(i)}$

Principal Component Analysis (PCA) algorithm summary

After mean normalization (ensure every feature has zero mean) and optionally feature scaling:

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

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

- Calculate sigma (covariance matrix Σ)
  - In MATLAB or octave we can implement this as follows:
    - **sigma = (1 / m) * (X' * X);**
- Next we can apply SVD (Singular Value Decomposition) routine to calculate the eigenvectors U, S, V
  - Take k vectors from U:
  - **Ureduce = U(:, 1:k);**
- Finally calculate z:
  - **z = Ureduce' * x;**

Video Question: In PCA, we obtain $z \in \mathbb{R}^k$ from $x \in \mathbb{R}^n$ as follows:

$$z = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(k)} \\ | & | & & | \end{bmatrix}^T x = \begin{bmatrix} --- & (u^{(1)})^T & --- \\ --- & (u^{(2)})^T & --- \\ & \vdots & \\ --- & (u^{(k)})^T & --- \end{bmatrix} x$$

Which of the following is a correct expression for z_j?

- $z_j = (u^{(k)})^T x$
- $z_j = (u^{(j)})^T x_j$
- $z_j = (u^{(j)})^T x_k$

> $z_j = (u^{(j)})^T x$