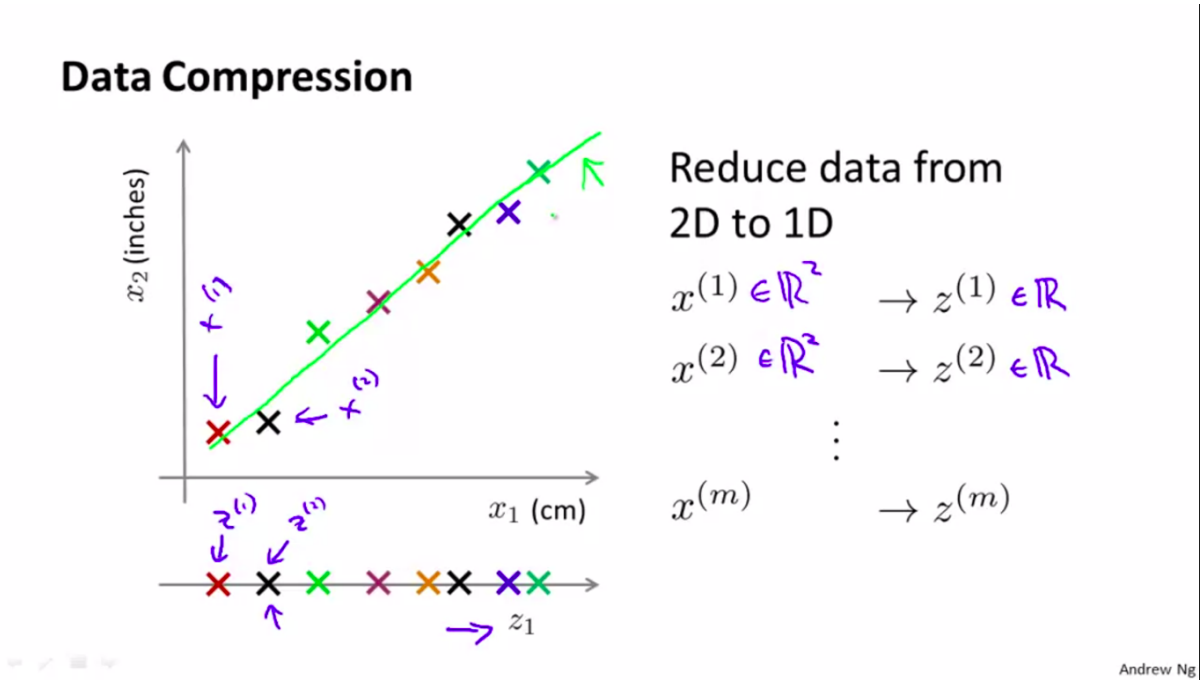
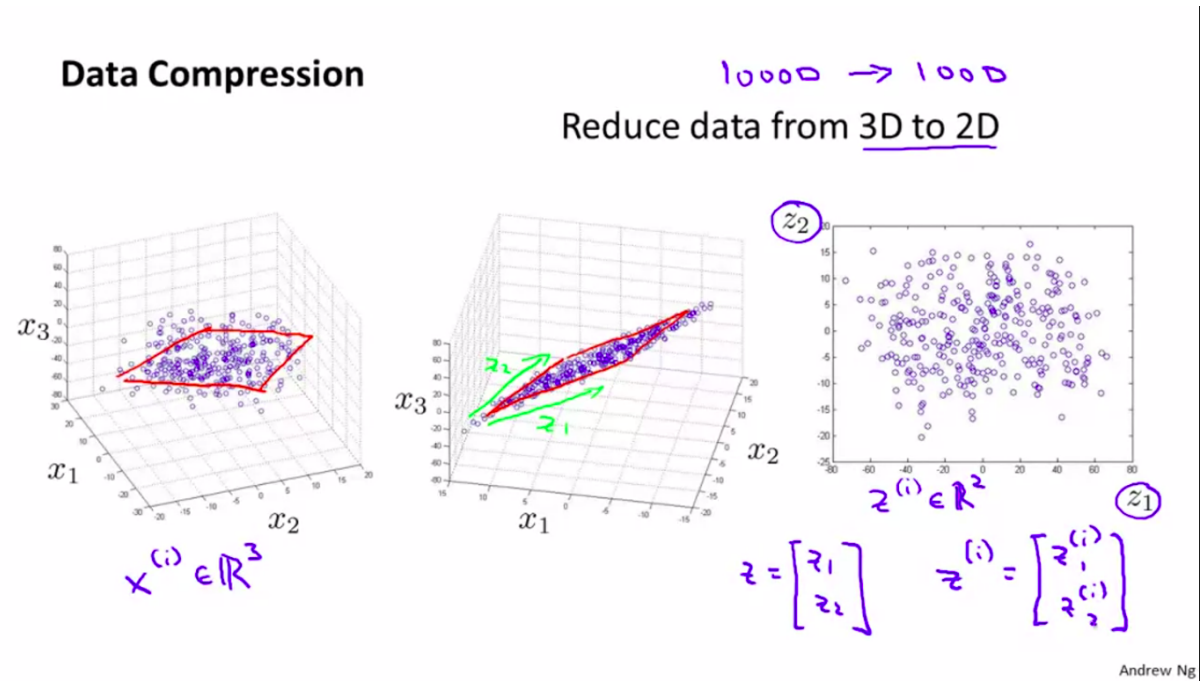
Week 8 - Lecture 2

Dimensionality Reduction

Motivation I: Data Compression





Motivation II: Visualization

Suppose we've collected a large dataset with many features like the following:

Data Visualization

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

Country	x_1 GDP (trillions of US$)	x_2 Per capita GDP (thousands of intl. $)	x_3 Human Develop-ment Index	x_4 Life expectancy	x_5 Poverty Index (Gini as percentage)	x_6 Mean household income (thousands of US$)	...
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	...
...	...	...	...	...	...	...	

[resources from en.wikipedia.org]

Andrew Ng

Suppose additionally that this dataset contais 50 features and that we want to visualize it somehow. It's difficult to visualize all the 50 features. In this context, we can represent each country in terms of new two new features obtained with dimensionality reduction:

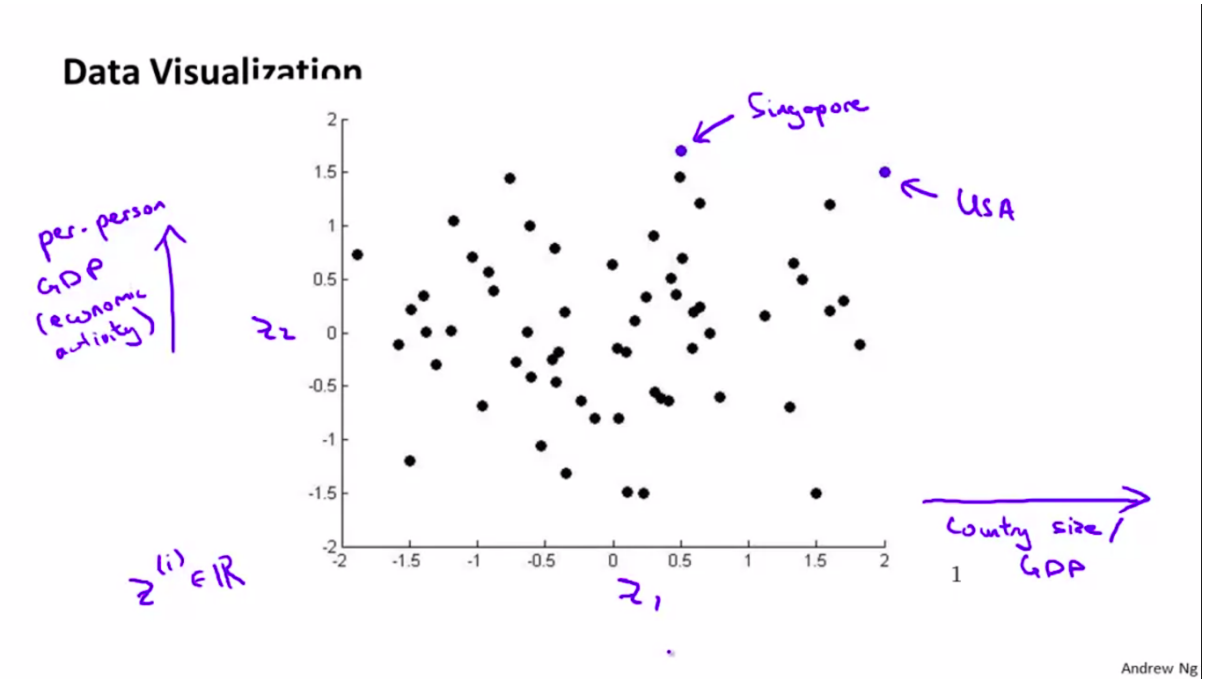
Data Visualization

$z^{(i)}$

Country	z_1	z_2
Canada	1.6	1.2
China	1.7	0.3
India	1.6	0.2
Russia	1.4	0.5
Singapore	0.5	1.7
USA	2	1.5
...	...	...

Andrew Ng

with these two features, we can visualize this dataset way more easily.

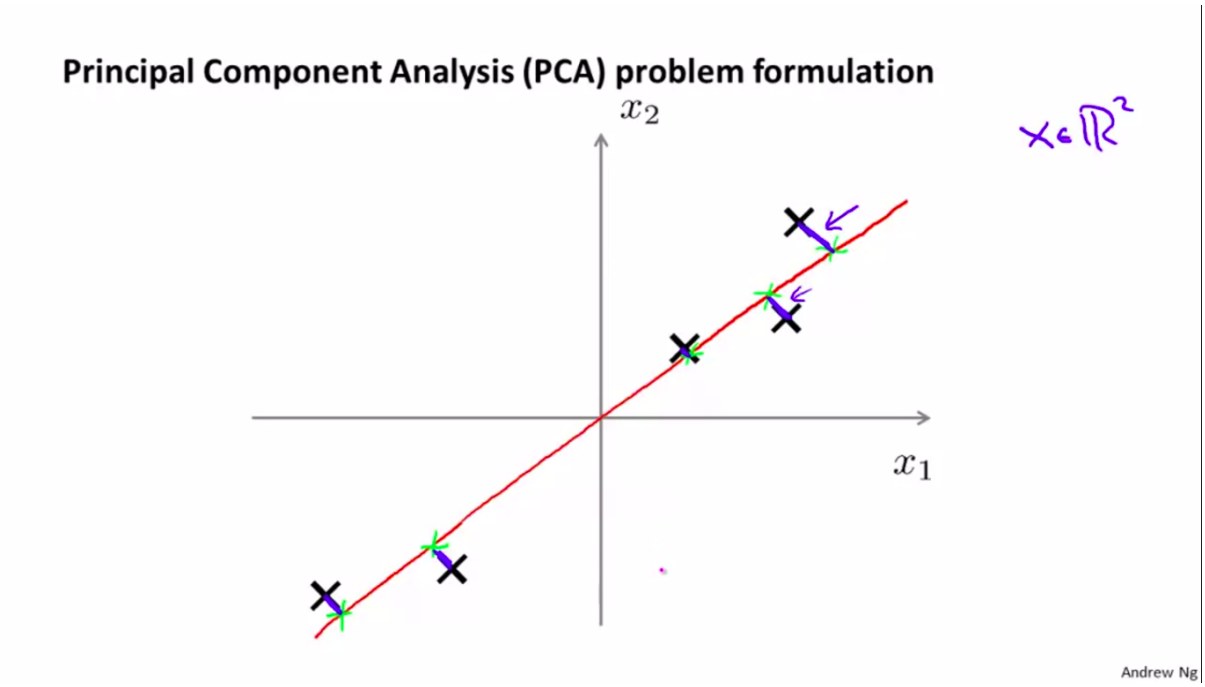


Principal Component Analysis

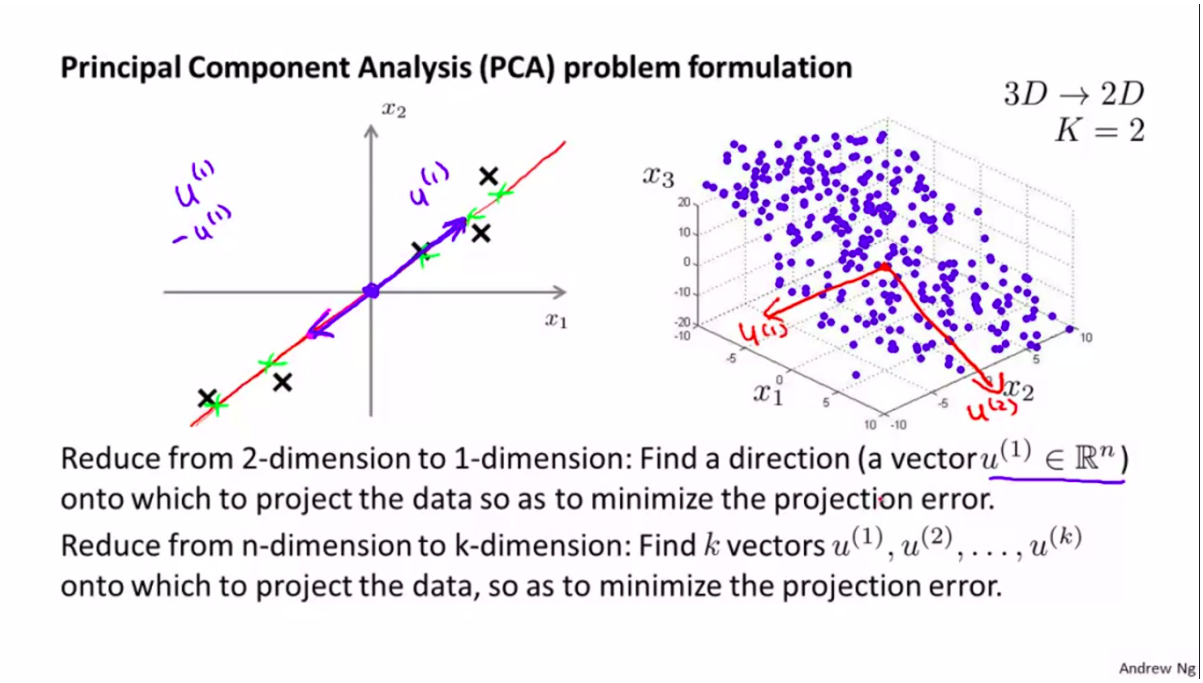
Problem Formulation

> **Note:** Before performing Principal Component Analysis, it is a good practice to perform mean normalization and feature scaling.

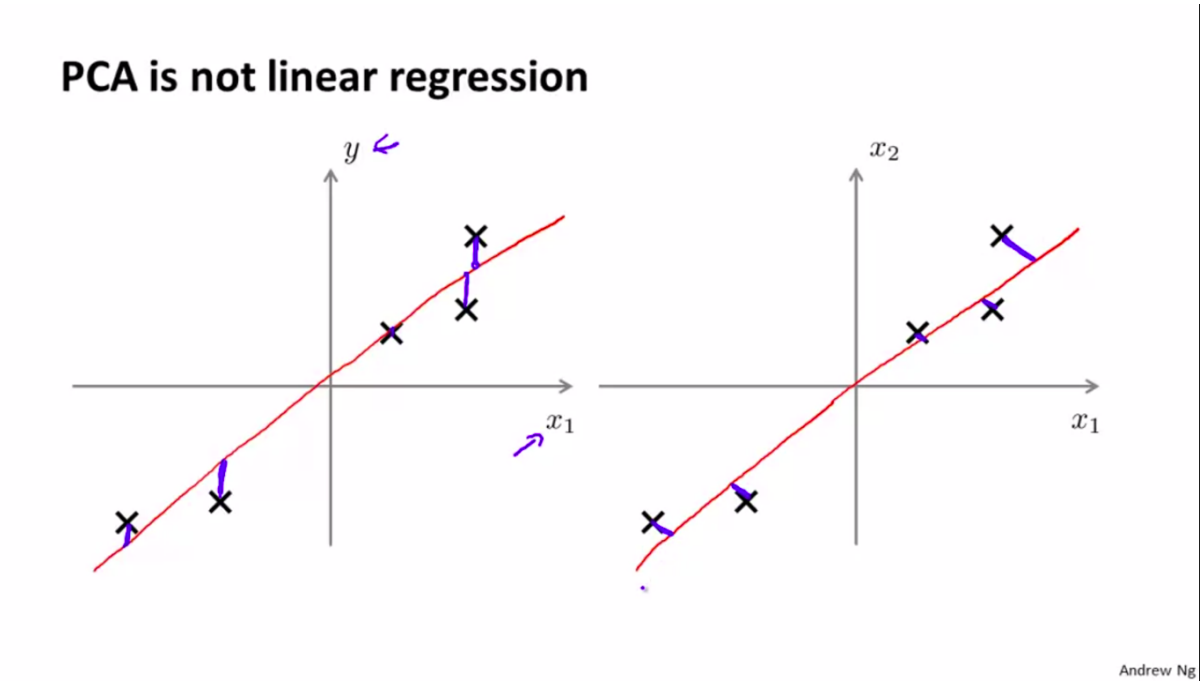
Suppose we want to reduce the dimmension of the data from $\mathbb{R}^2$ to $\mathbb{R}^1$, i.e., we want to find a good line onto project our data points. The red line in the figure below seems to be a good one.



More formally, the goal of PCA is:



p.s.: It is important to distinguish PCA from Linear Regression. In the figure below, on the left is an example of a Linear Regression problem (minimize the vertical distances from each point to the straight line). On the right, is an example of a PCA problem (minimize the orthogonal distances from each point to the straight line).



Algorithm

Data preprocessing

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

Preprocessing (feature scaling/mean normalization):

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

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

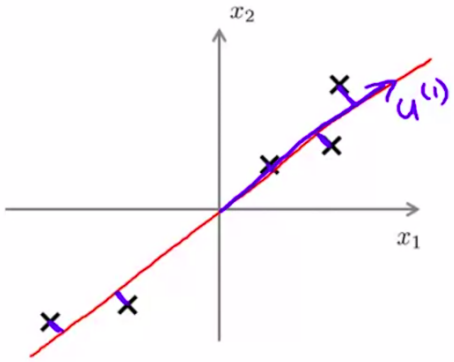
If different features on different scales (e.g., x_1 = size of house, x_2 = number of bedrooms), scale features to have comparable range of values.

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

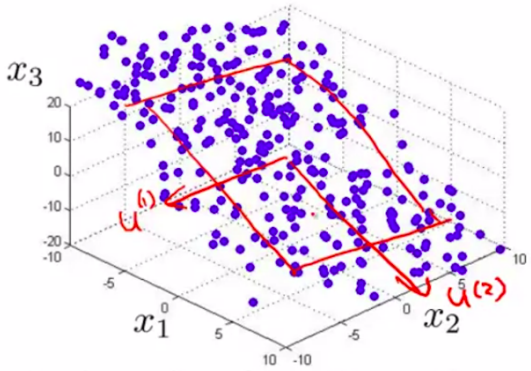
Andrew Ng

What PCA does is to find a lower dimensional subspace onto which to project the data, so as to minimize the squared projection errors, sum of the squared projection errors, like illustrated below.

Principal Component Analysis (PCA) algorithm



Reduce data from 2D to 1D



Reduce data from 3D to 2D

Andrew Ng

Principal Component Analysis (PCA) algorithm

Reduce data from n-dimensions to k-dimensions

Compute "covariance matrix":

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

$n \times 1$ $1 \times n$ $n \times n$ Sigma

Compute "eigenvectors" of matrix Σ: → Singular value decomposition

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

eig(Sigma)

$n \times n$ matrix.

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

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

k

Andrew Ng

Principal Component Analysis (PCA) algorithm

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

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

k

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

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

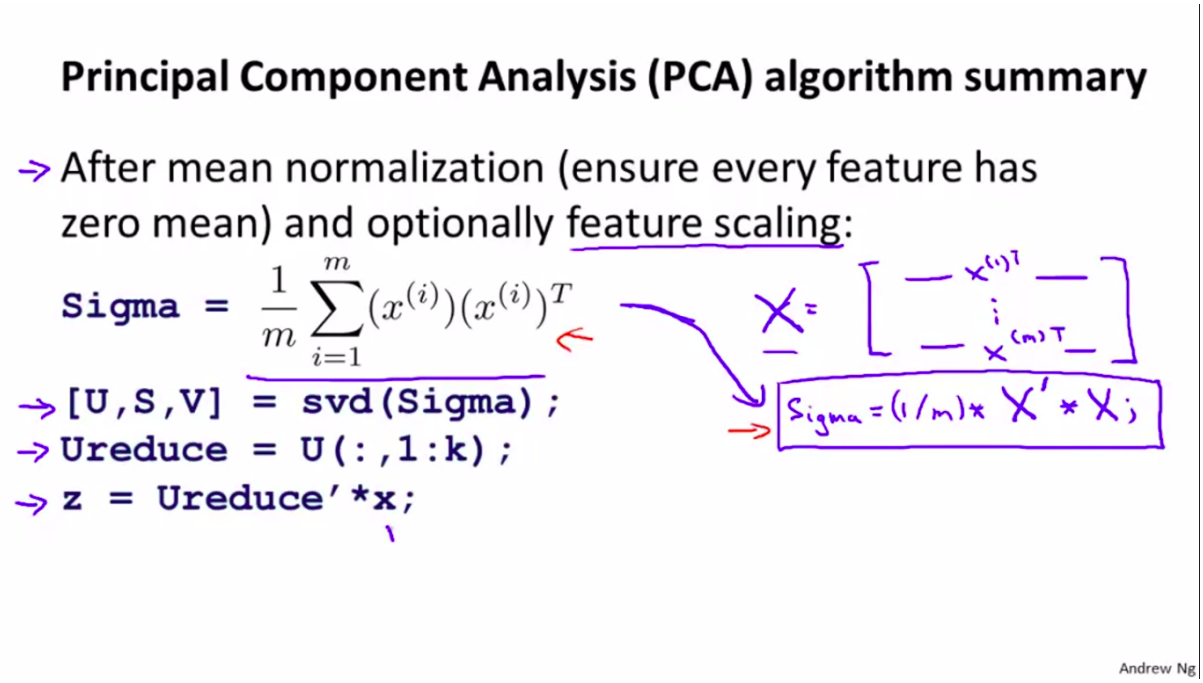
$z \in \mathbb{R}^k$

$n \times k$ $k \times n$ $n \times 1$ $k \times 1$

U_{reduce}

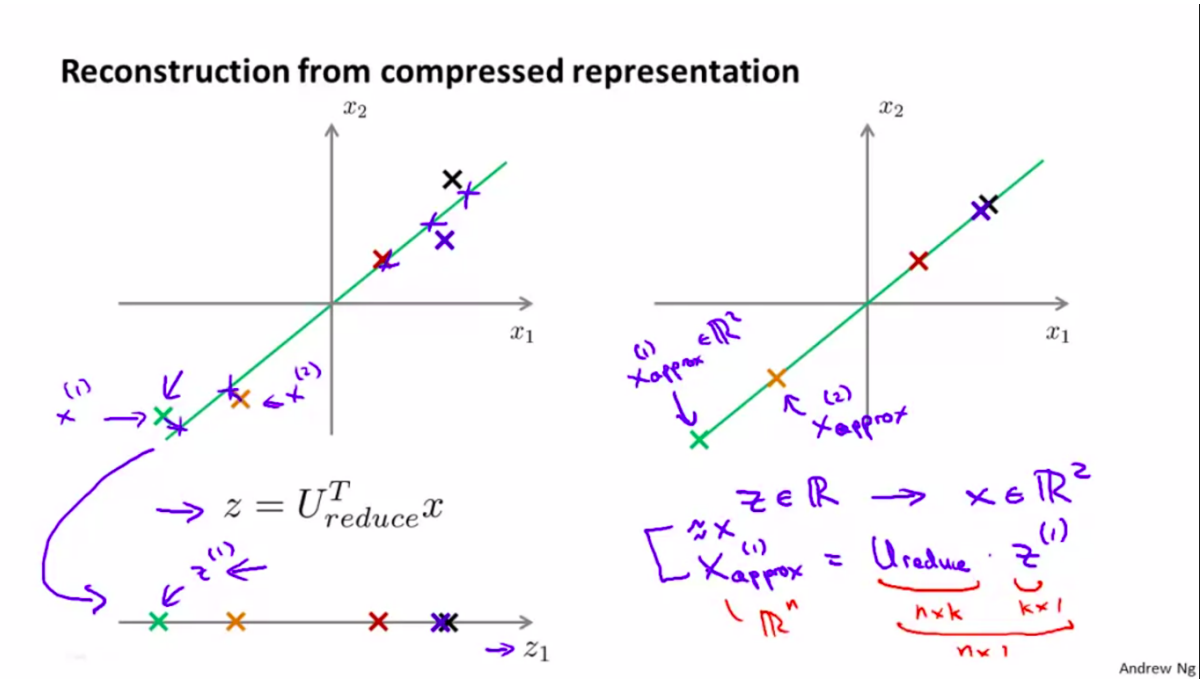
Andrew Ng

To summarize...



Applying PCA

Reconstruction from Compressed Representation



Note: It is not possible to get back the original values of $x^{(i)}$ if we are transforming from a lower dimension. The best we can do is to come up with an approximation, like ilustrated in the figure above.

Choosing the Number of Principal Components

Choosing k (number of principal components)

Average squared projection error: $\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - x_{approx}^{(i)}\|^2$

Total variation in the data: $\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2$

Typically, choose k to be smallest value so that

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

"99% of variance is retained"

Andrew Ng

Choosing k (number of principal components)

Algorithm:

Try PCA with $k = 1$, $k = 2$, $k = 3$, $k = 4$, ...

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

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

$k = 17$

→ [U, S, V] = svd(Sigma)

$S = \begin{bmatrix} s_{11} & & & \\ & s_{22} & & \\ & & s_{33} & \\ & & & \ddots \\ & & & & s_{nn} \end{bmatrix}$

For given k $\qquad k = 3$

$$1 - \frac{\sum_{i=1}^{k} s_{ii}}{\sum_{i=1}^{n} s_{ii}} \leq 0.01$$

$$\frac{\sum_{i=1}^{k} s_{ii}}{\sum_{i=1}^{n} s_{ii}} \geq 0.99$$

Andrew Ng

Advice for Applying PCA

Supervised learning speedup

$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$

$x^{(i)} \in \mathbb{R}^{10000}$

Extract inputs:

Unlabeled dataset: $x^{(1)}, x^{(2)}, \ldots, x^{(m)} \in \mathbb{R}^{10000}$

$\downarrow PCA$

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

New training set:

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

Application of PCA

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

Choose k by % of variance retained

- Visualization

$k = 2$ or $k = 3$

Bad use of PCA: To prevent overfitting

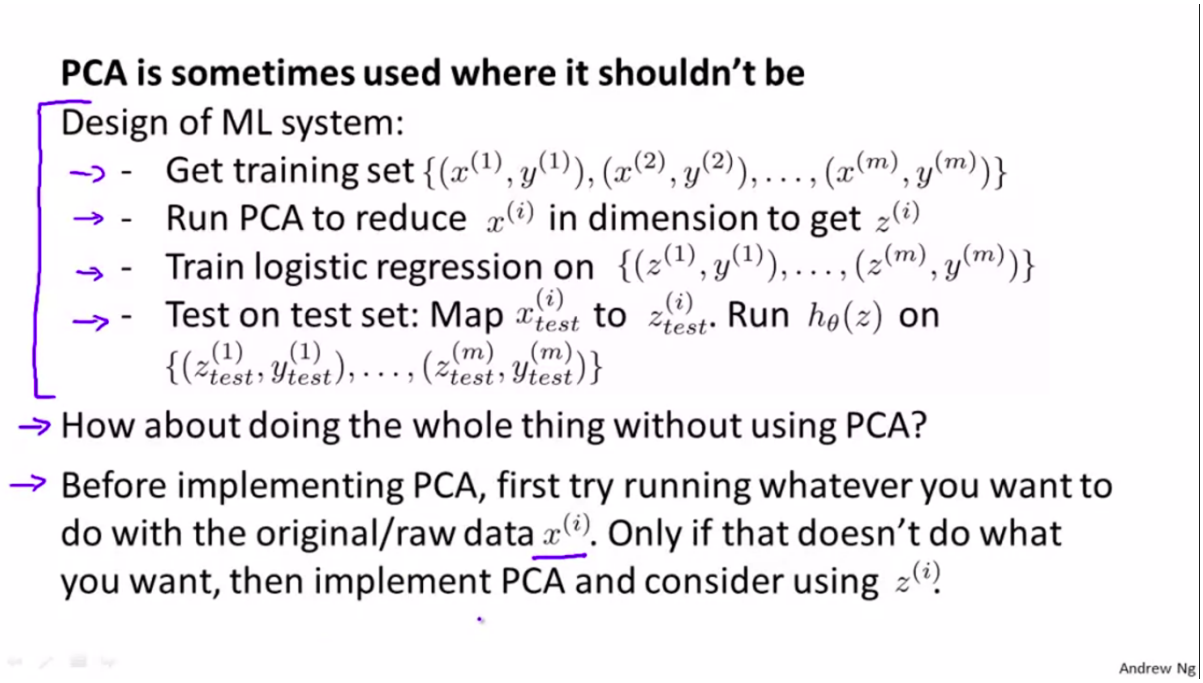
Use $z^{(i)}$ instead of $x^{(i)}$ to reduce the number of features to $k < n$. (k: 1000, n: 10000)

Thus, fewer features, less likely to overfit.

Bad!

This might work OK, but isn't a good way to address overfitting. Use regularization instead.

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



References

[1] Machine Learning - Stanford University (https://www.coursera.org/learn/machine-learning)