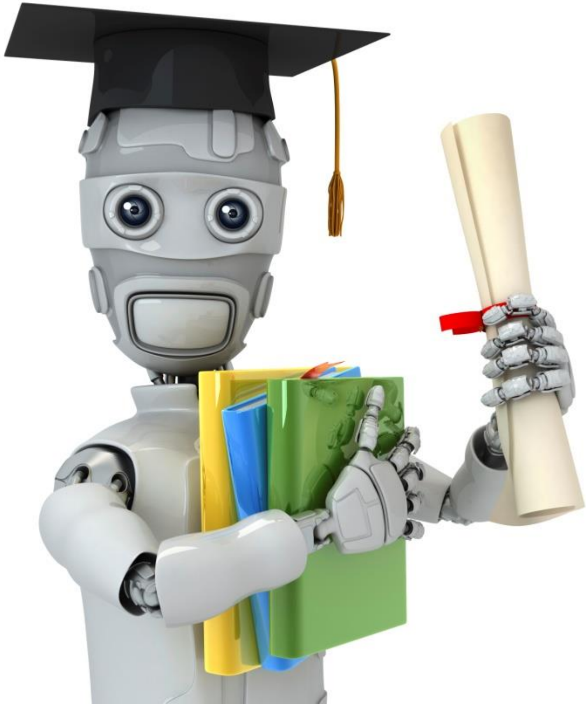
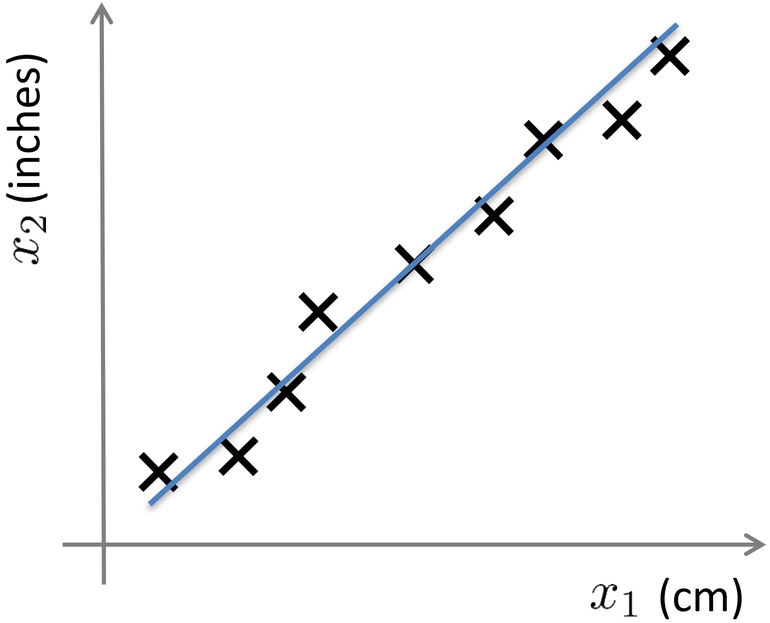


# Dimensionality Reduction

## Motivation I: Data Compression

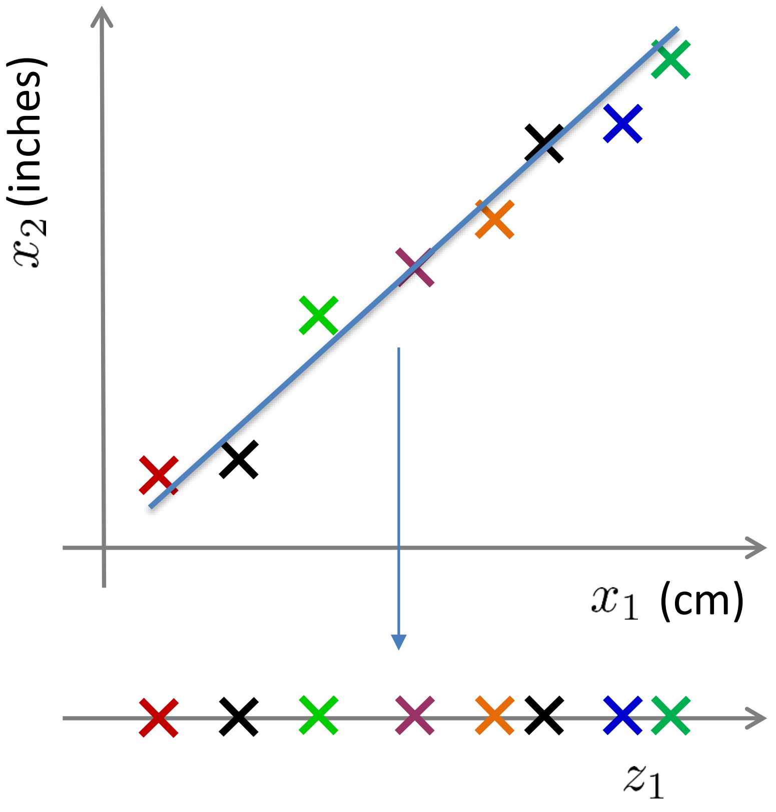
Machine Learning

# Data Compression




Reduce data from 2D to 1D

# Data Compression




Reduce data from 2D to 1D

$$x^{(1)} \rightarrow z^{(1)}$$

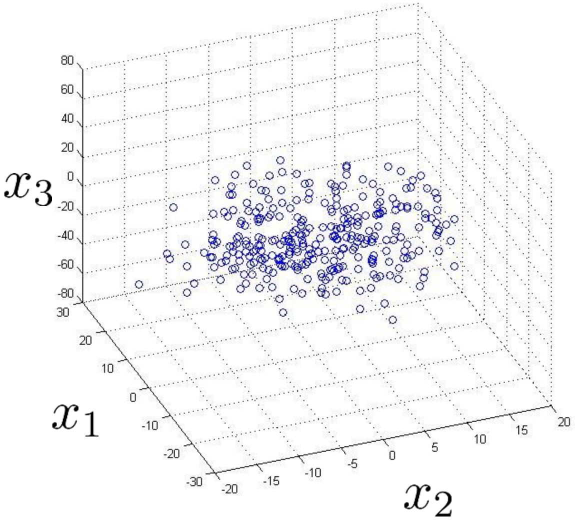
$$x^{(2)} \rightarrow z^{(2)}$$

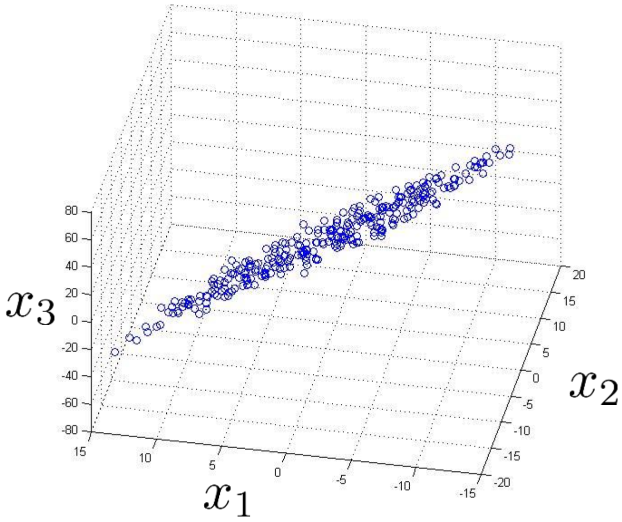
$$x^{(m)} \rightarrow z^{(m)}$$

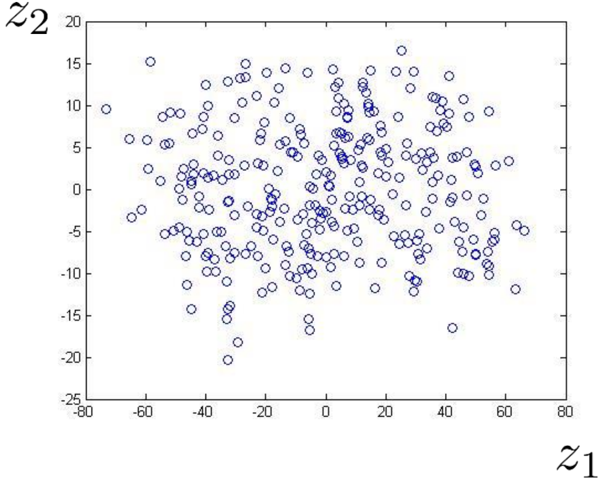
Andrew Ng

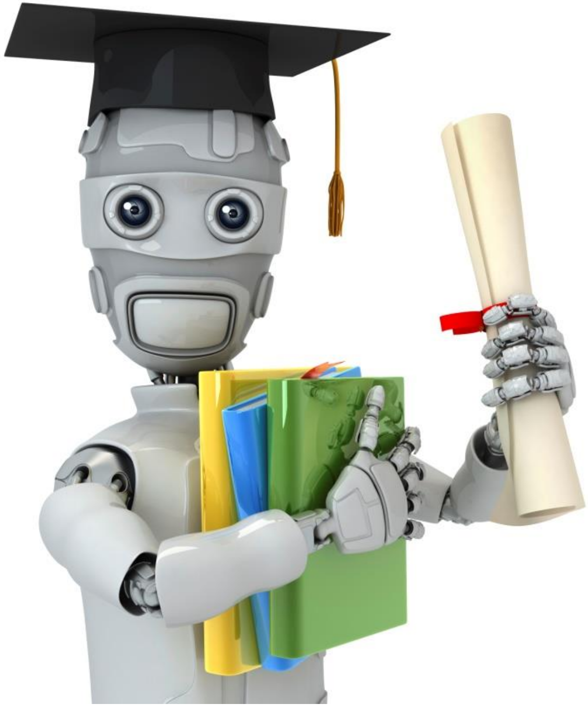
# Data Compression

Reduce data from 3D to 2D





Machine Learning

# Dimensionality Reduction

## Motivation II: Data Visualization

# Data Visualization

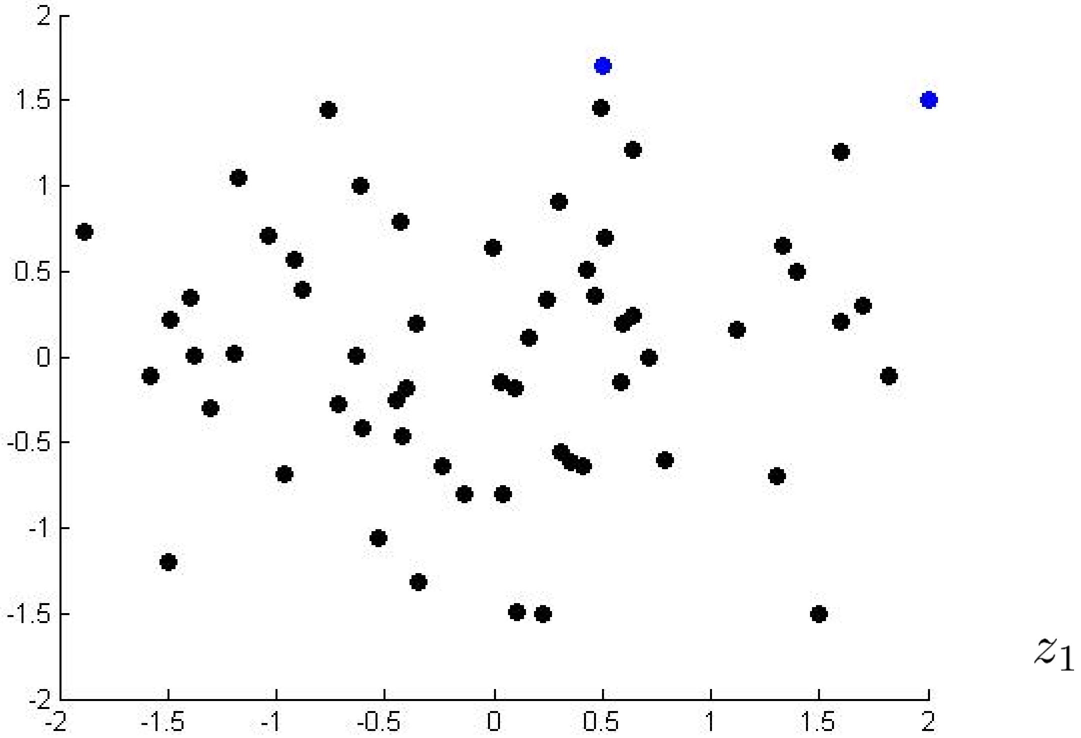
| Country | GDP (trillions of US$) | Per capita GDP (thousands of intl. $) | Human Develop-ment Index | Life expectancy | Poverty Index (Gini as percentage) | Mean household income (thousands of US$) | … |
|---|---|---|---|---|---|---|---|
| Canada | 1.577 | 39.17 | 0.908 | 80.7 | 32.6 | 67.293 | … |
| 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 | … |
| … | … | … | … | … | … | … | |

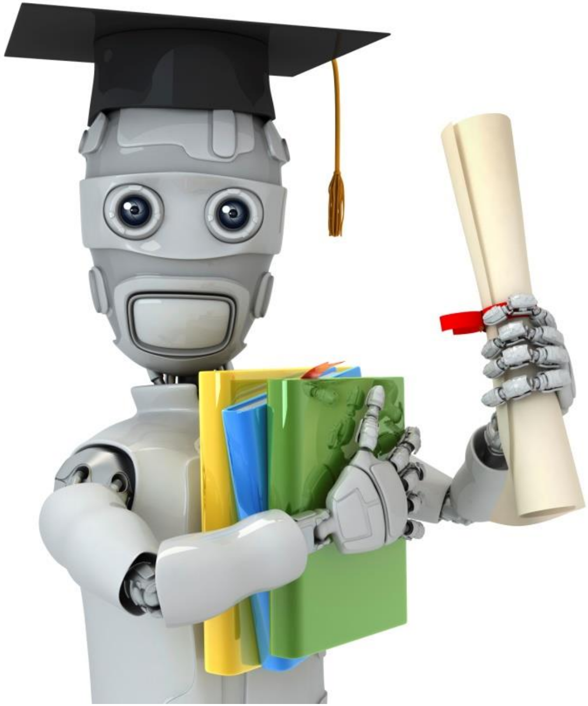
[resources from en.wikipedia.org]

# Data Visualization

| 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 |
| … | … | … |

# Data Visualization



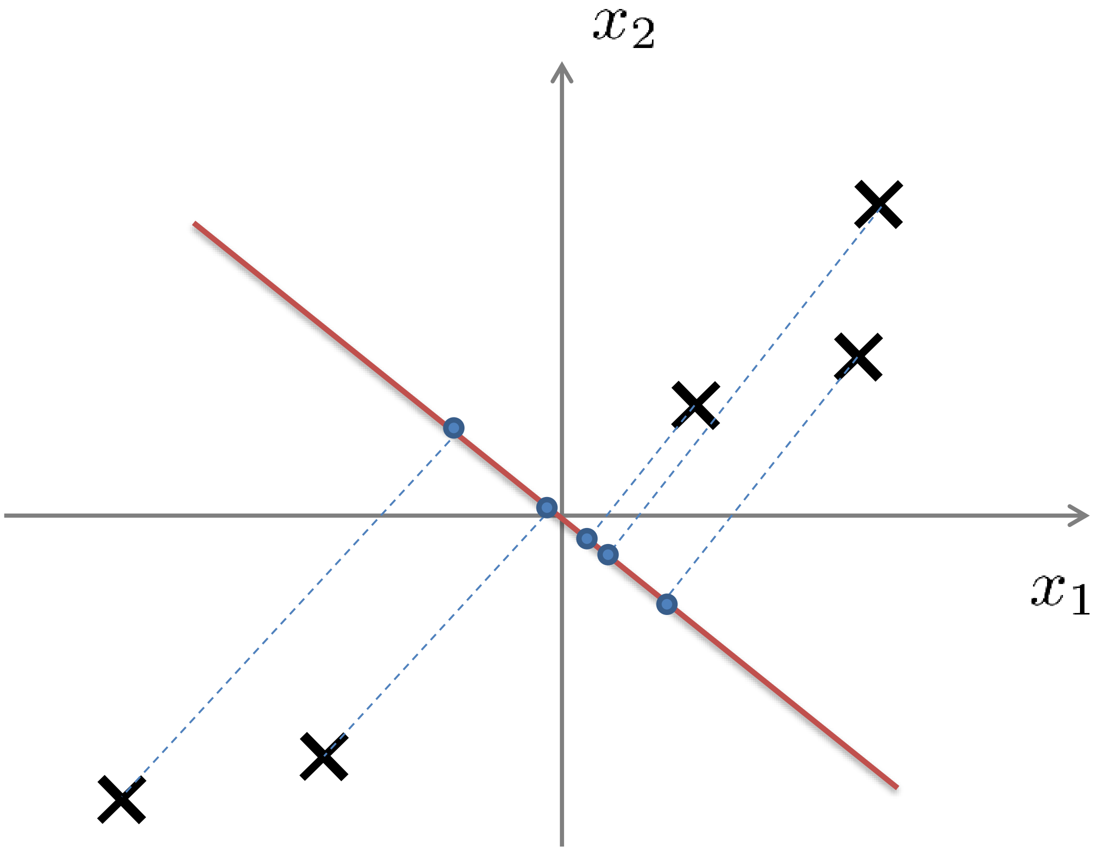


# Dimensionality Reduction

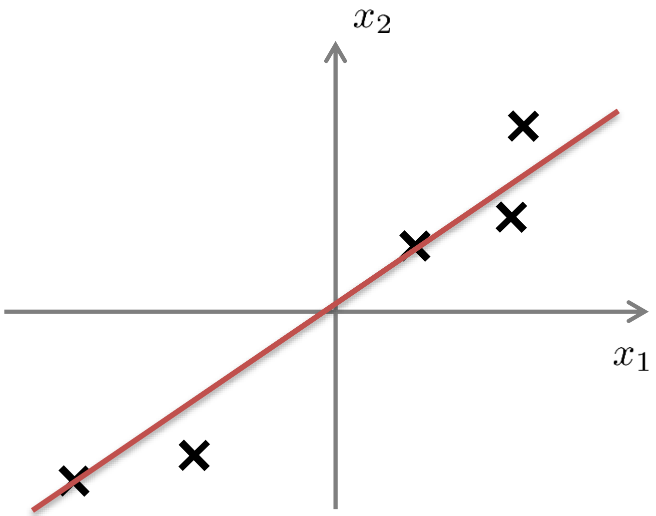
## Principal Component Analysis problem formulation

Machine Learning

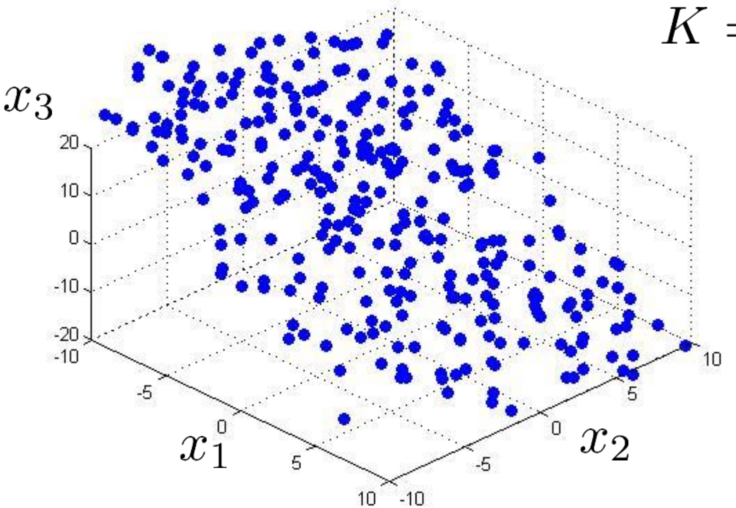
# Principal Component Analysis (PCA) problem formulation



# Principal Component Analysis (PCA) problem formulation



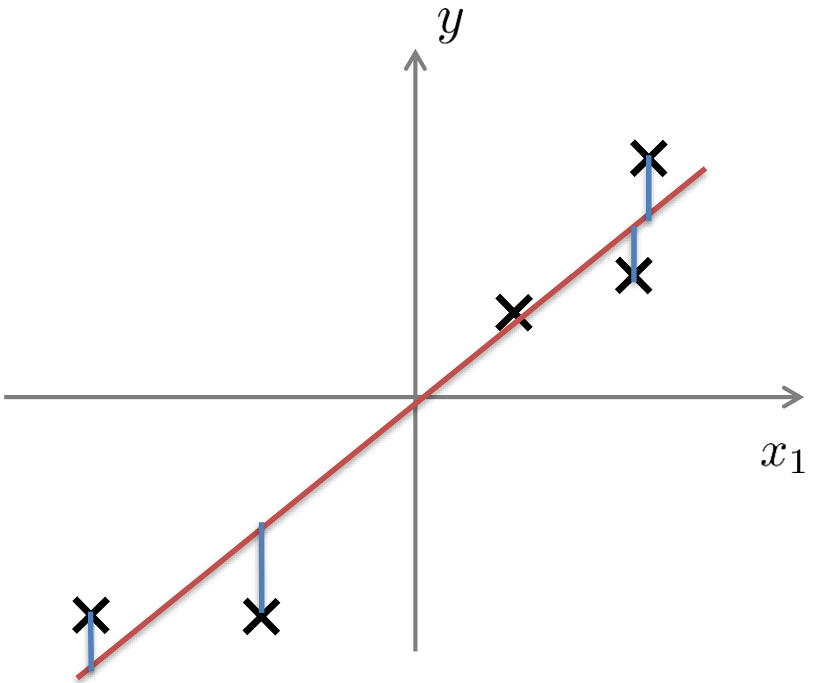

$$3D \rightarrow 2D$$
$$K = 2$$

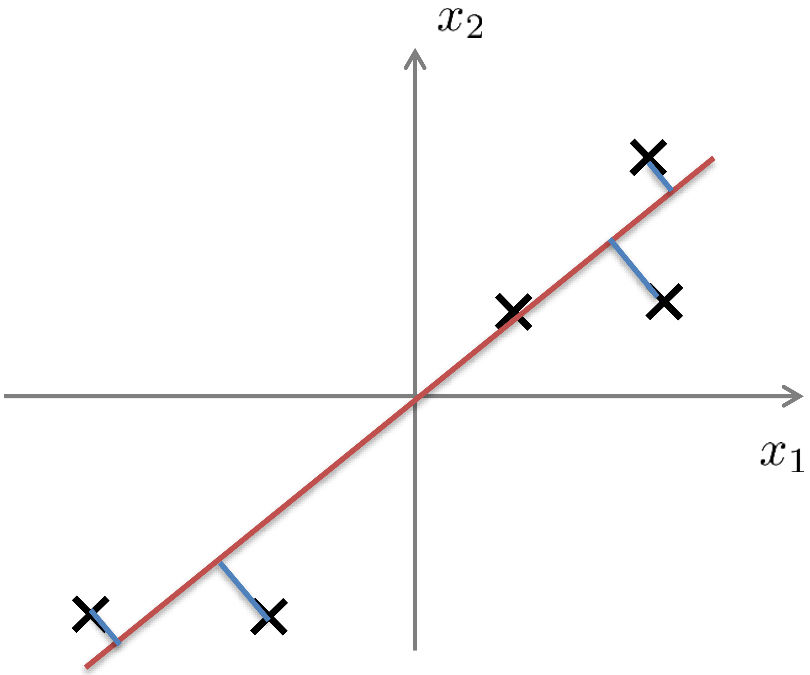



Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.

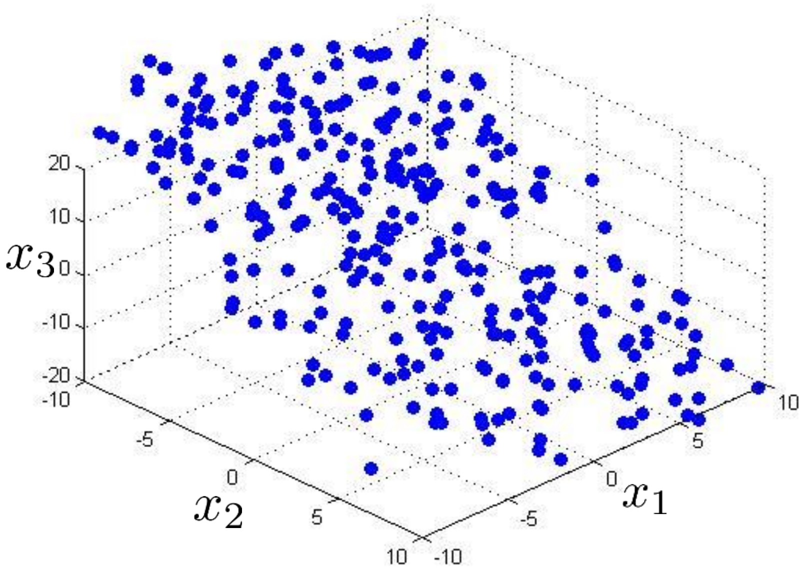
Reduce from n-dimension to k-dimension: Find $k$ vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data, so as to minimize the projection error.

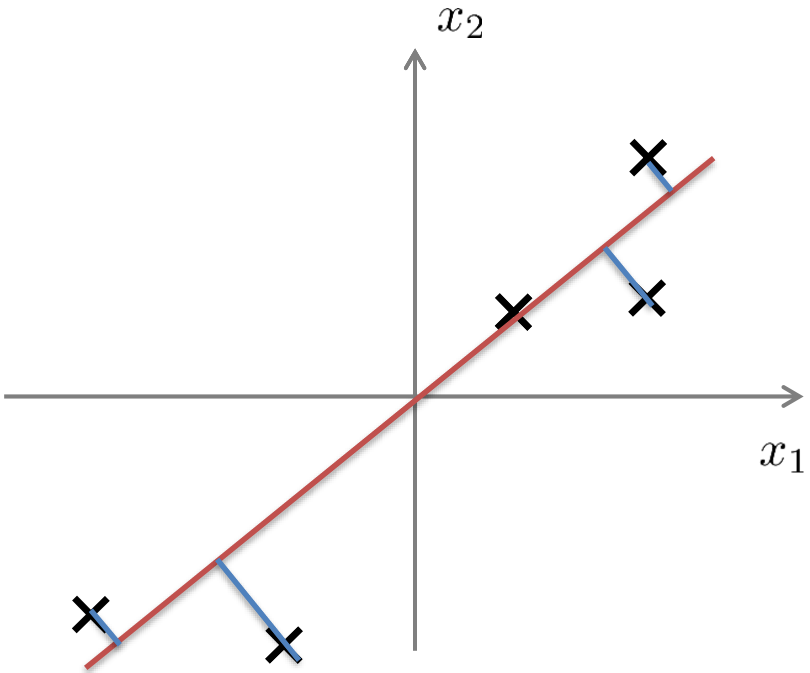
# PCA is not linear regression



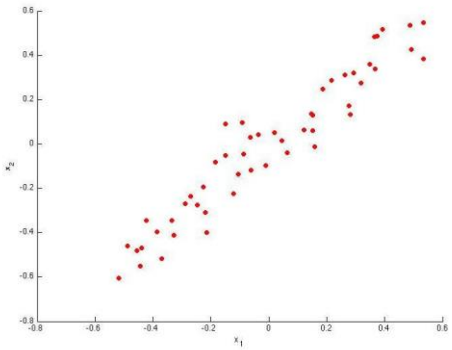



# PCA is not linear regression

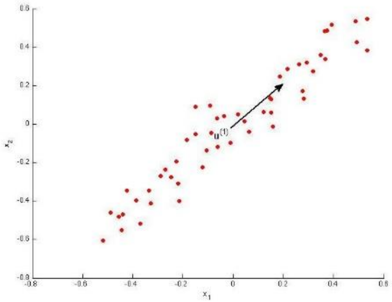


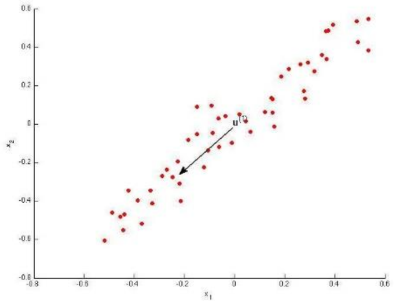
考虑以下二维数据集：



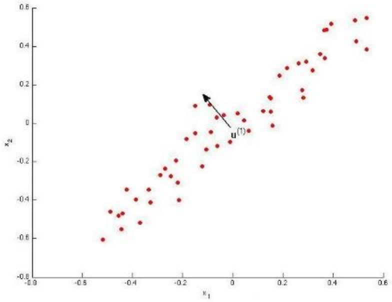
下列哪个图片对应的PCA可能返回的$u^{(1)}$（第一特征向量/第一主成分）的值？选出所有正确项



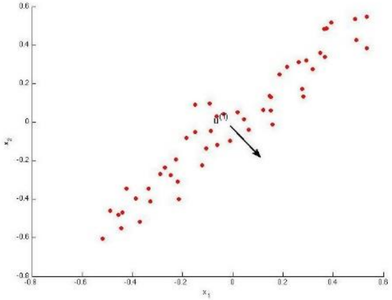

A




B

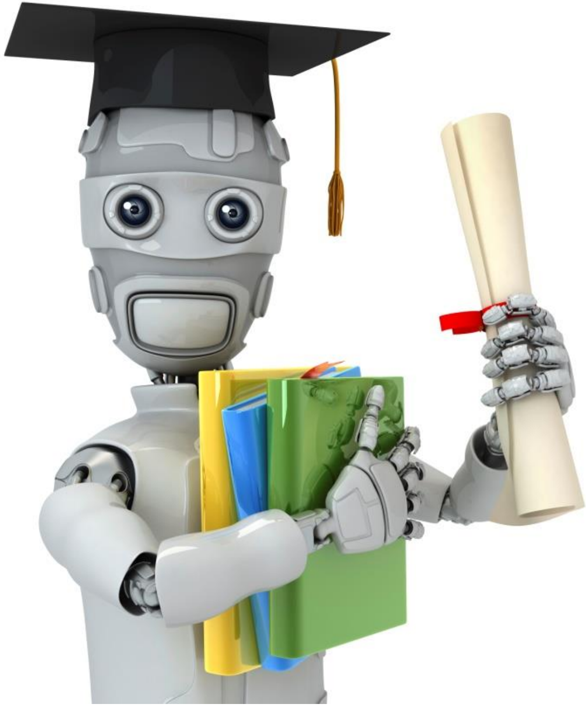



C




D

AB



Machine Learning

# Dimensionality Reduction

## Principal Component Analysis algorithm

## Data preprocessing

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

Preprocessing (feature scaling/mean normalization):

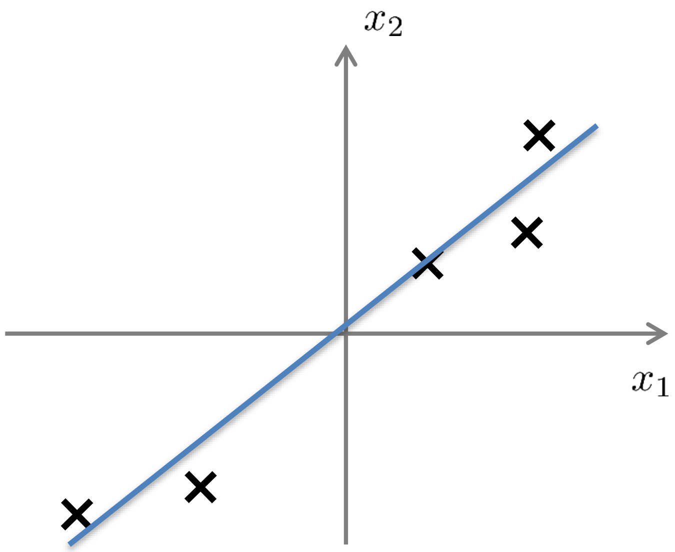
$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$$

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

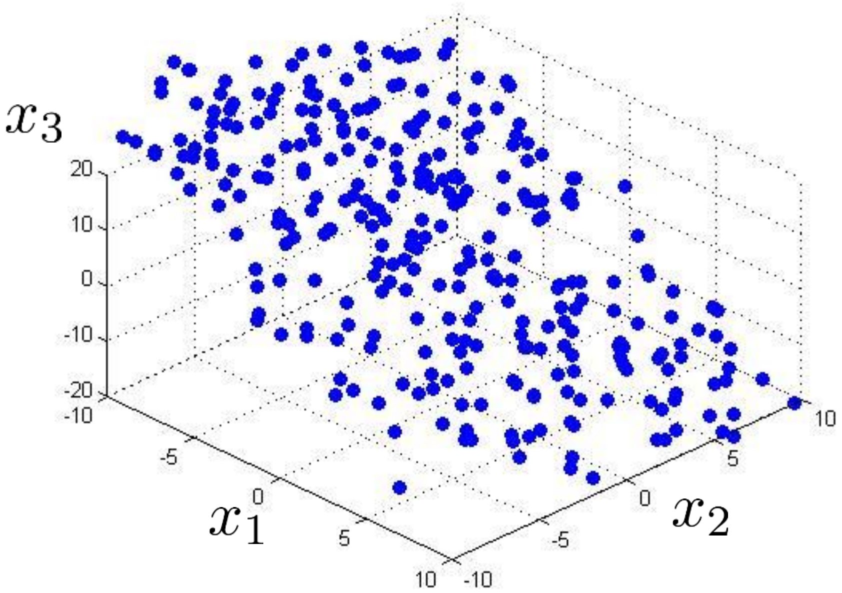
If different features on different scales (e.g., $x_1 =$ size of house, $x_2 =$ number of bedrooms), scale features to have comparable range of values.

$$x_j = \frac{x_j - \mu_j}{s_j}$$

# Principal Component Analysis (PCA) algorithm




Reduce data from 2D to 1D




Reduce data from 3D to 2D

# Principal Component Analysis (PCA) algorithm

Reduce data from $n$-dimensions to $k$-dimensions

Compute "covariance matrix":

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

Compute "eigenvectors" of matrix $\Sigma$:

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

$n * n$ *Matrix*

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

# Principal Component Analysis (PCA) algorithm

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

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

$$z^{(i)} = U_{reduce}^{T} * x^{(i)} =$$

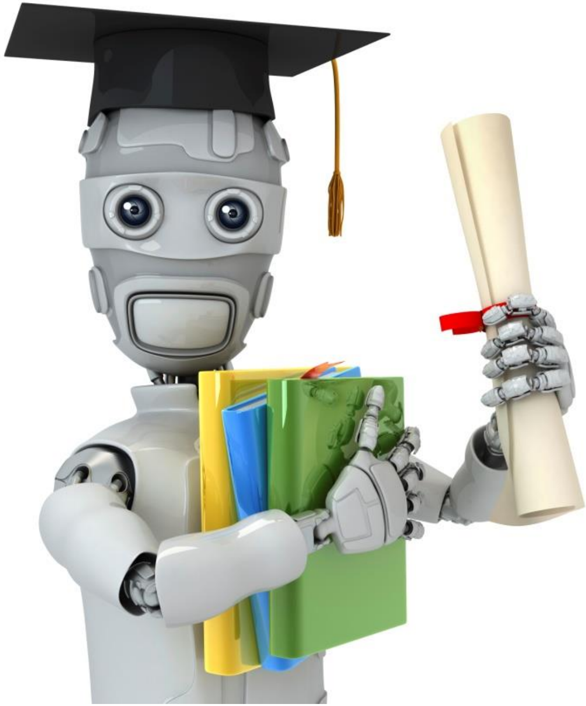
$$\underbrace{\begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(k)} \\ | & | & & | \end{bmatrix}^{T}}_{n * k\ Matrix} * \underset{\underset{n * 1\ Matrix}{\downarrow}}{x^{(i)}} = k * 1\ Matrix$$

# Principal Component Analysis (PCA) algorithm summary

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

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

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

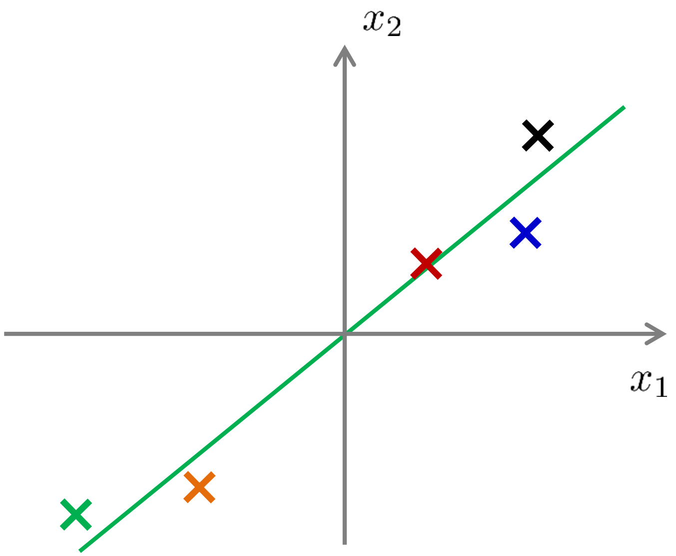


Machine Learning

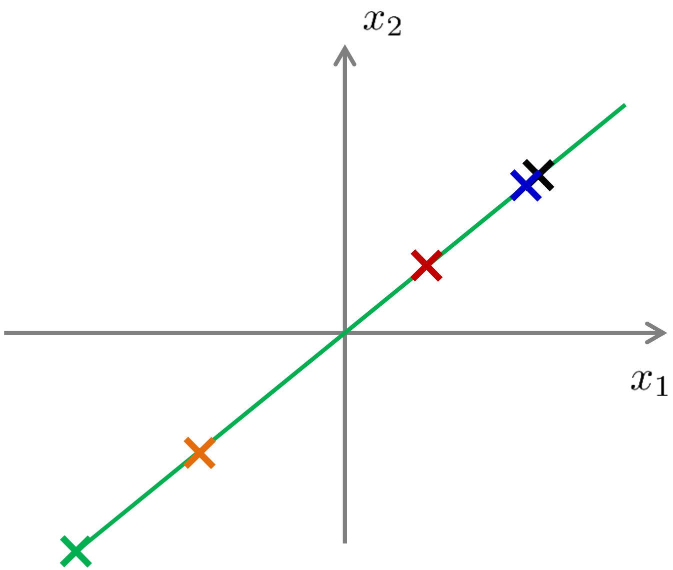
# Dimensionality Reduction

## Reconstruction from compressed representation

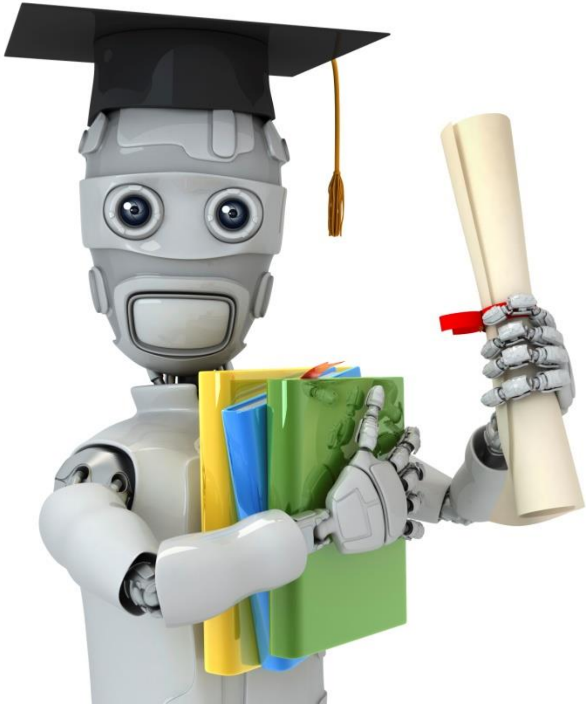
# Reconstruction from compressed representation




$$z = U_{reduce}^T x$$


$$x_{approx}^{(i)} = U_{reduce} * z^{(i)}$$



# Dimensionality Reduction

## Choosing the number of principal components

Machine Learning

## 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"

# Choosing $k$ (number of principal components)

Algorithm:

Try PCA with $k = 1$

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

Check if

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

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

$$S = \begin{bmatrix} S_{11} & 0 & 0 & 0 \\ 0 & S_{22} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & S_{nn} \end{bmatrix}$$

$$\Rightarrow \quad 1 - \frac{\Sigma_{i=1}^{k} s_{ii}}{\Sigma_{i=1}^{n} s_{ii}} \leq 1\%$$

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

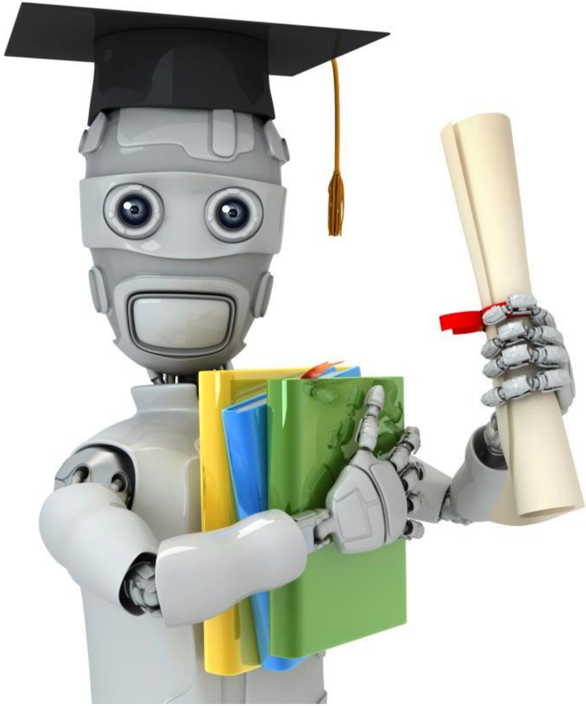
## Choosing $k$ (number of principal components)

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

Pick smallest value of $k$ for which

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

(99% of variance retained)



Machine Learning

# Dimensionality Reduction

## Advice for applying PCA

# Supervised learning speedup

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

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:

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

Note: Mapping $x^{(i)} \rightarrow z^{(i)}$ should be defined by running PCA only on the training set. This mapping can be applied as well to the examples $x_{cv}^{(i)}$ and $x_{test}^{(i)}$ in the cross validation and test sets.

# Application of PCA

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

- Visualization

## Bad use of PCA: To prevent overfitting

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

Thus, fewer features, less likely to overfit.

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

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

## PCA is sometimes used where it shouldn't be

Design of ML system:

- Get training set $\{(x^{(1)},y^{(1)}),(x^{(2)},y^{(2)}),\ldots,(x^{(m)},y^{(m)})\}$
- Run PCA to reduce $x^{(i)}$ in dimension to get $z^{(i)}$
- Train logistic regression on $\{(z^{(1)},y^{(1)}),\ldots,(z^{(m)},y^{(m)})\}$
- Test on test set: Map $x_{test}^{(i)}$ to $z_{test}^{(i)}$. Run $h_\theta(z)$ on $\{(z_{test}^{(1)},y_{test}^{(1)}),\ldots,(z_{test}^{(m)},y_{test}^{(m)})\}$

How about doing the whole thing without using PCA?

Before implementing PCA, first try running whatever you want to do with the original/raw data $x^{(i)}$. Only if that doesn't do what you want, then implement PCA and consider using $z^{(i)}$.