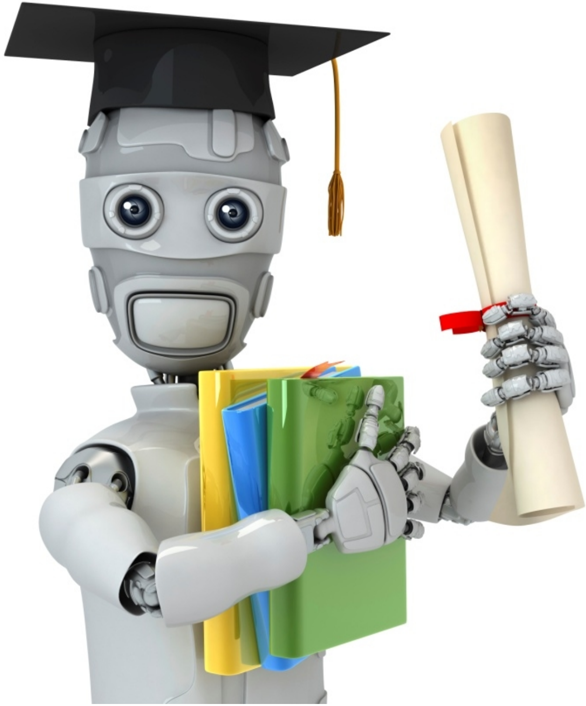
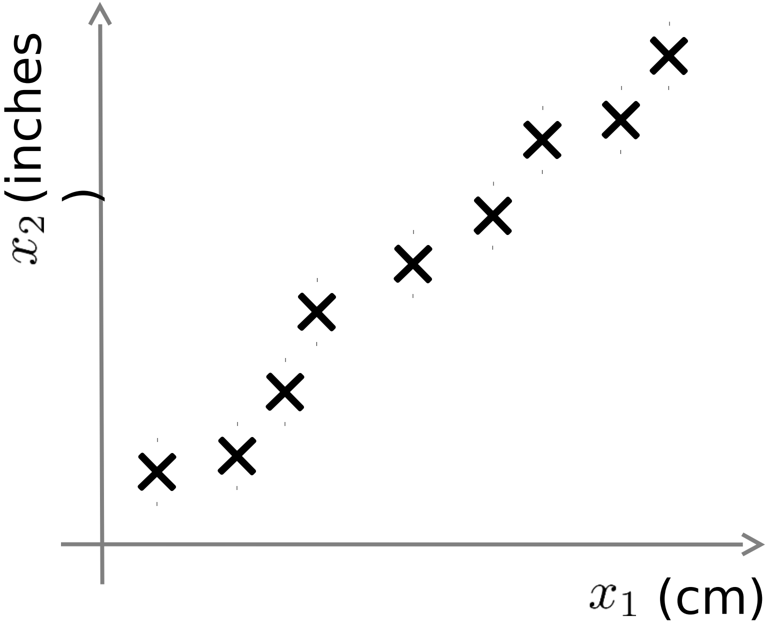


Machine Learning

# Dimensionality Reduction

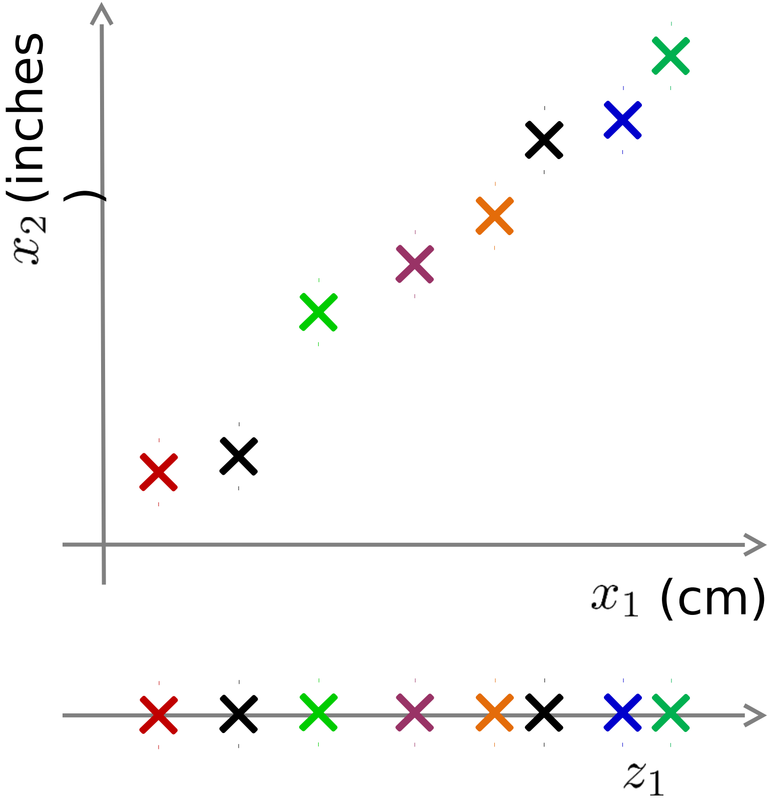
## Motivation I: Data Compression

# Data Compression




Reduce data from 2D to 1D

# Data Compression

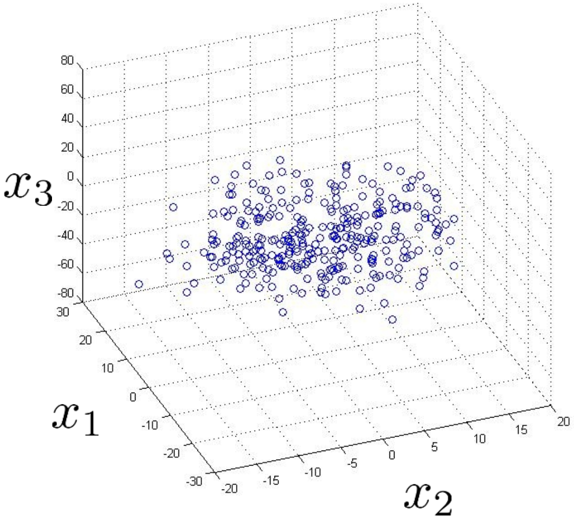


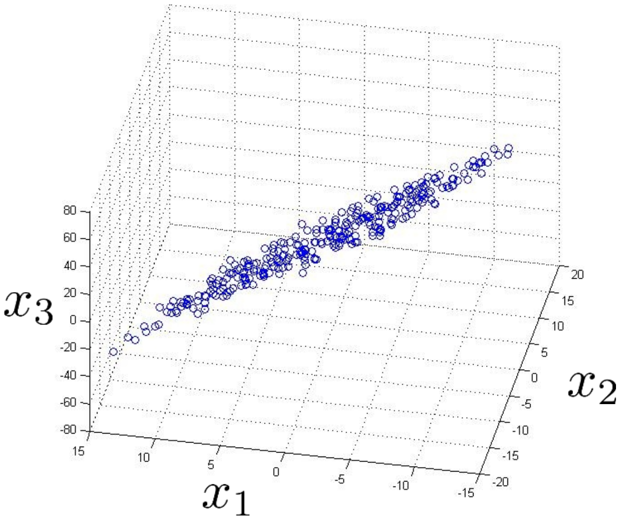

Reduce data from 2D to 1D

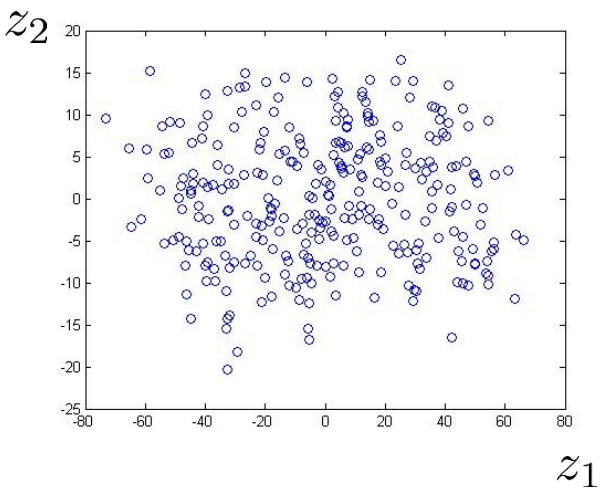
$$x^{(1)} \rightarrow z^{(1)}$$
$$x^{(2)} \rightarrow z^{(2)}$$
$$\vdots$$
$$x^{(m)} \rightarrow z^{(m)}$$

# Data Compression

Reduce data from 3D to 2D



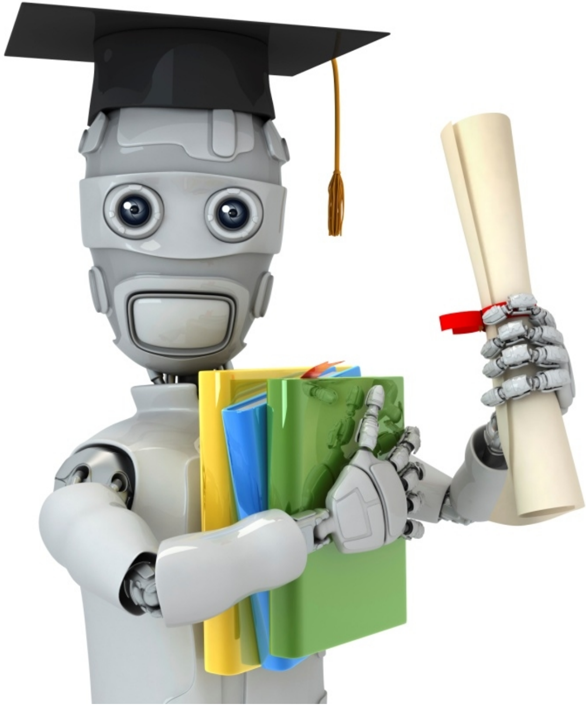



Suppose we apply dimensionality reduction to a dataset of m examples $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$, where $x^{(i)} \in \mathbb{R}^n$. As a result of this, we will get out:

- A lower dimensional dataset $\{z^{(1)}, z^{(2)}, \ldots, z^{(k)}\}$ of k examples where $k \leq n$.
- A lower dimensional dataset $\{z^{(1)}, z^{(2)}, \ldots, z^{(k)}\}$ of k examples where $k > n$.
- A lower dimensional dataset $\{z^{(1)}, z^{(2)}, \ldots, z^{(m)}\}$ of m examples where $z^{(i)} \in \mathbb{R}^k$ for some value of $k$ and $k \leq n$.

**Correct Response**

- A lower dimensional dataset $\{z^{(1)}, z^{(2)}, \ldots, z^{(m)}\}$ of m examples where $z^{(i)} \in \mathbb{R}^k$ for some value of $k$ and $k > n$.



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 | ... |
| ... | ... | ... | ... | ... | ... | ... | |

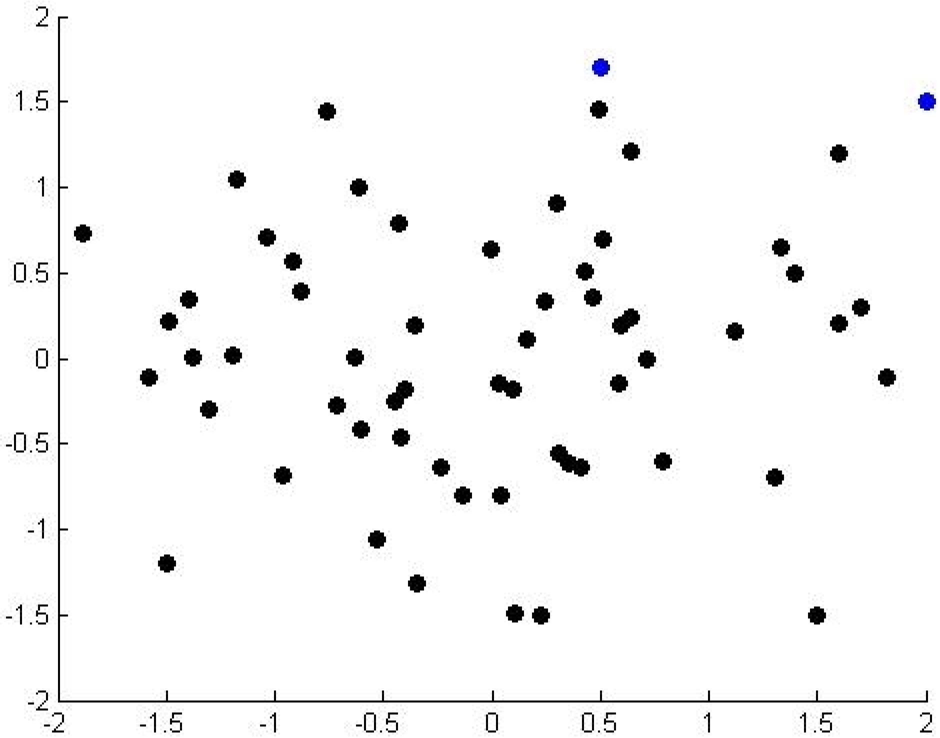
[resources from en.wikipedia.org]

Andrew Ng

# 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




1

Suppose you have a dataset $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ where $x^{(i)} \in \mathbb{R}^n$. In order to visualize it, we apply dimensionality reduction and get $\{z^{(1)}, z^{(2)}, \ldots, z^{(m)}\}$ where $z^{(i)} \in \mathbb{R}^k$ is k-dimensional. In a typical setting, which of the following would you expect to be true? Check all that apply.

- [ ] k > n

**Correct Response**

- [x] k ≤ n

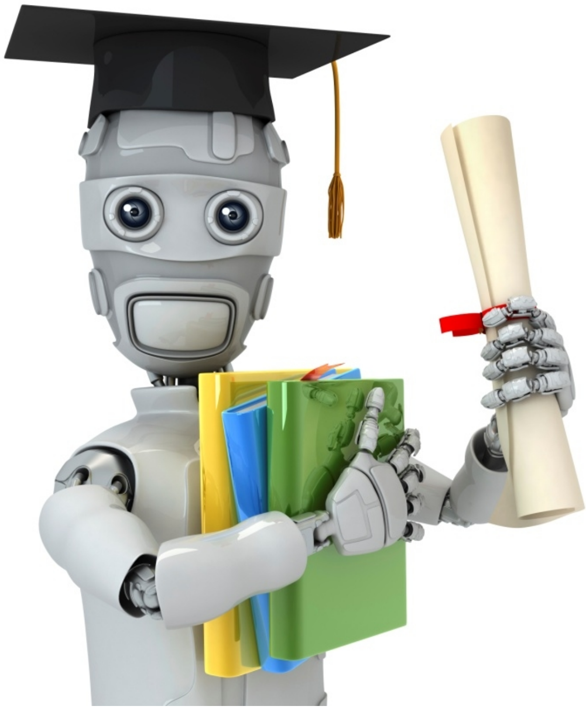
**Correct Response**

- [ ] k ≥ 4

**Correct Response**

- [x] k = 2 or k = 3 (since we can plot 2D or 3D data but don't have ways to visualize higher dimensional data)

**Correct Response**



# Dimensionality Reduction

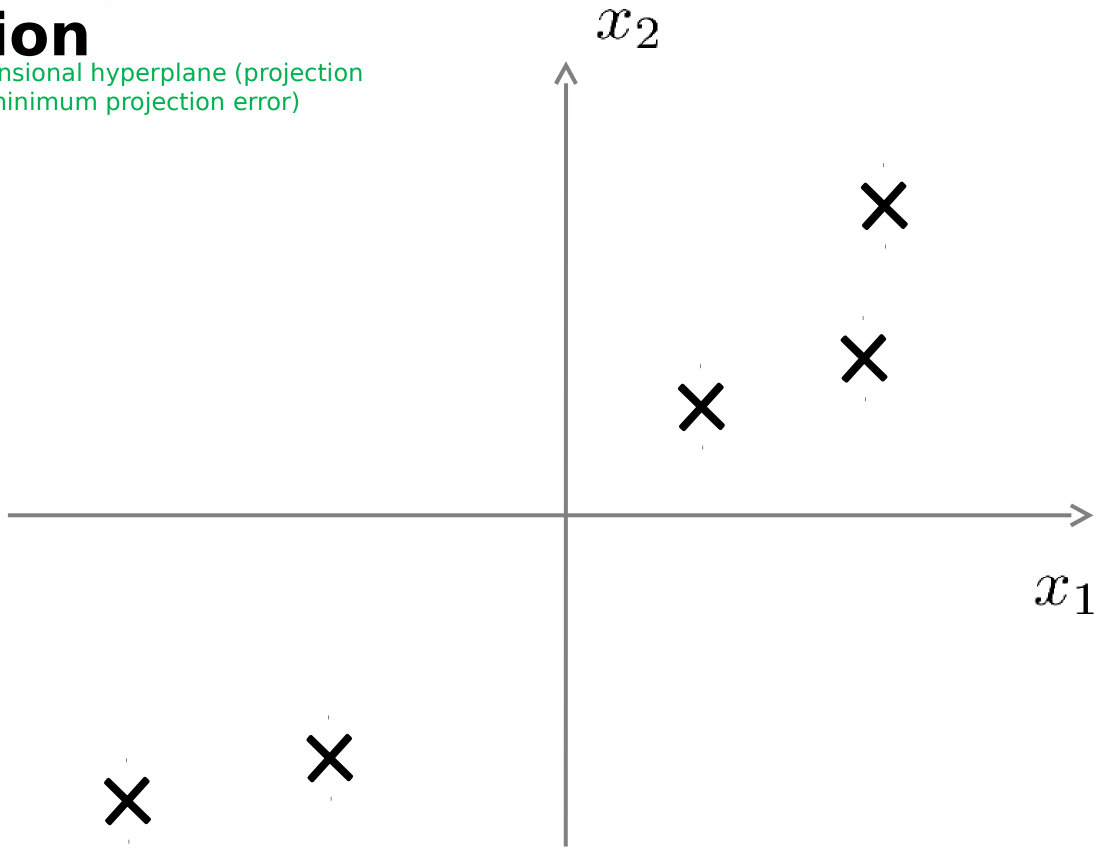
## Principal Component Analysis problem formulation

Machine Learning

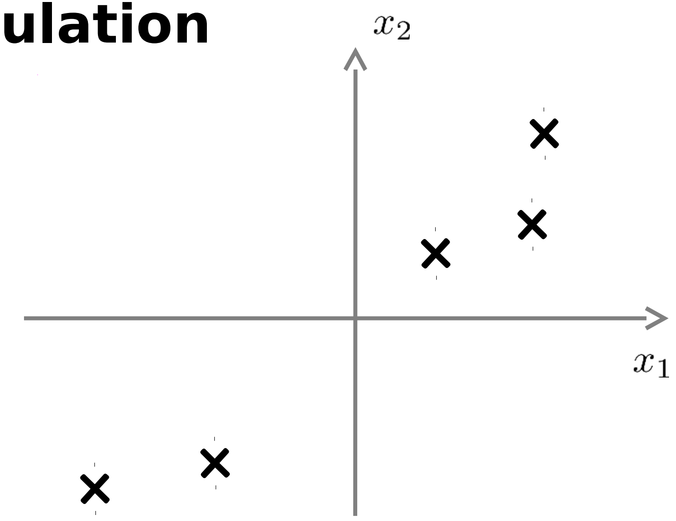
(PCA is the generalized algo. that is used for dimensionality reduction)

# Principal Component Analysis (PCA) problem formulation

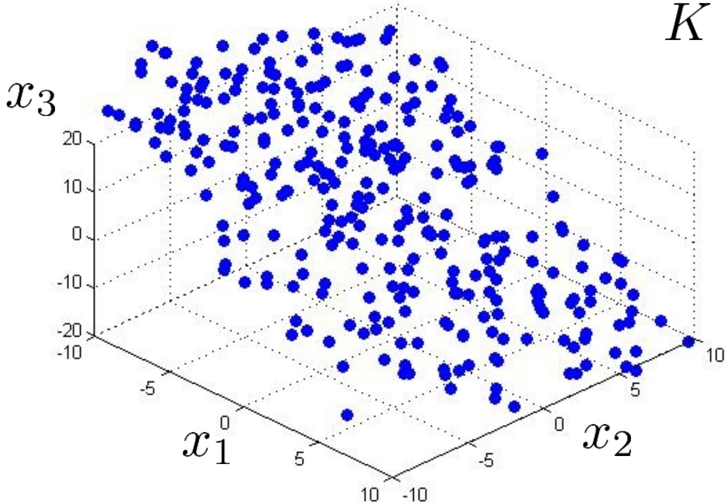
(PCA tries to find a low dimensional hyperplane (projection hyperplane) which has the minimum projection error)



# Principal Component Analysis (PCA) problem formulation




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

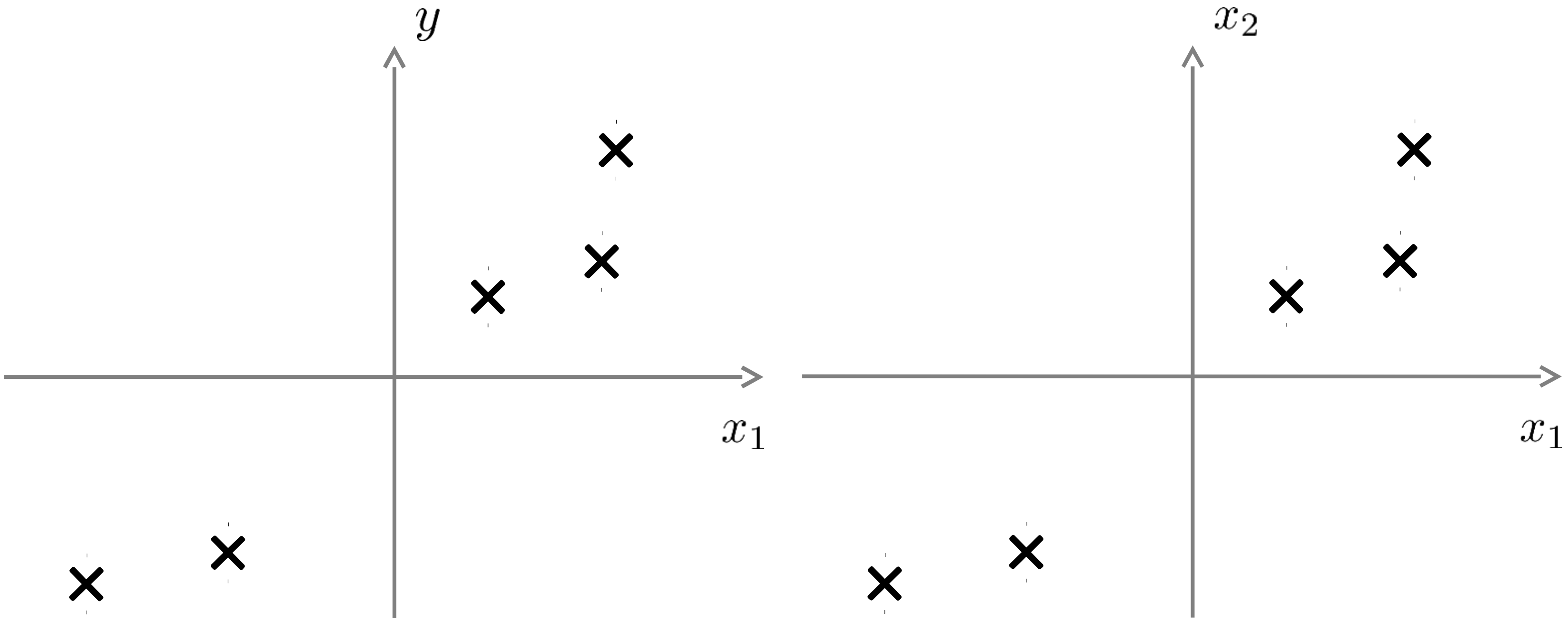



Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.

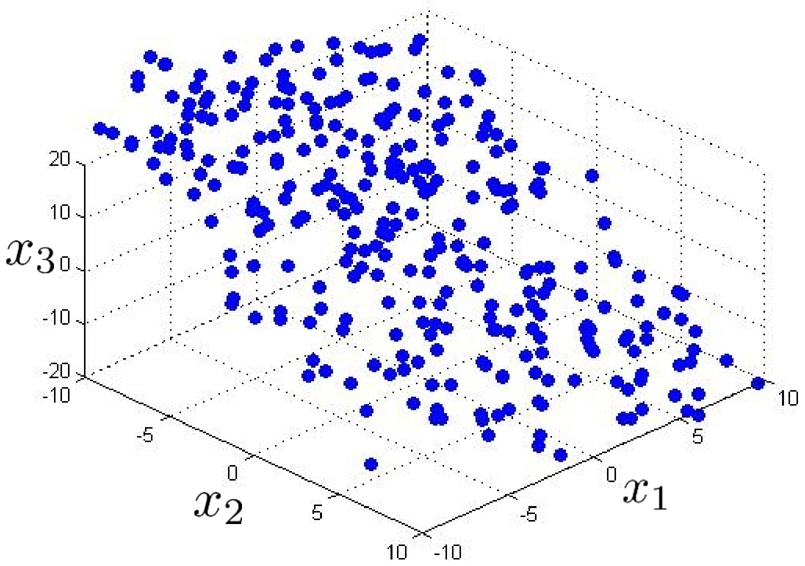
Reduce from n-dimension to k-dimension: Find $k$ vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data, so as to minimize the projection error.

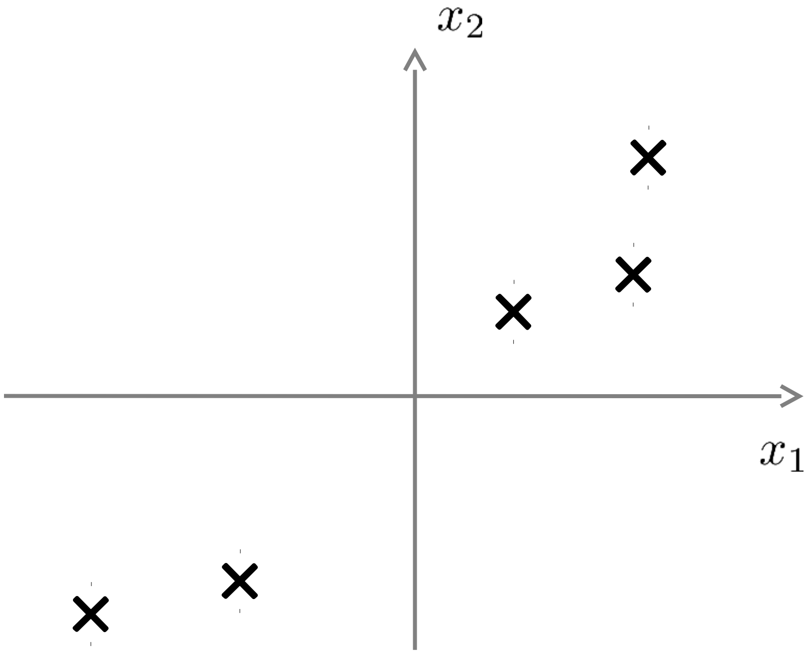
# PCA is not linear regression

(PCA tries to minimize (squared) projection error not MSE)



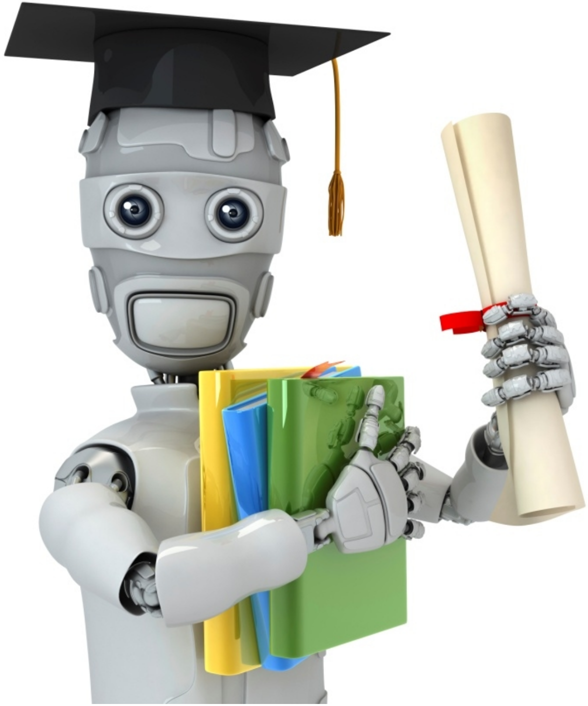
# PCA is not linear regression



# Dimensionality Reduction

Principal Component Analysis algorithm



Machine Learning

# Data preprocessing

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

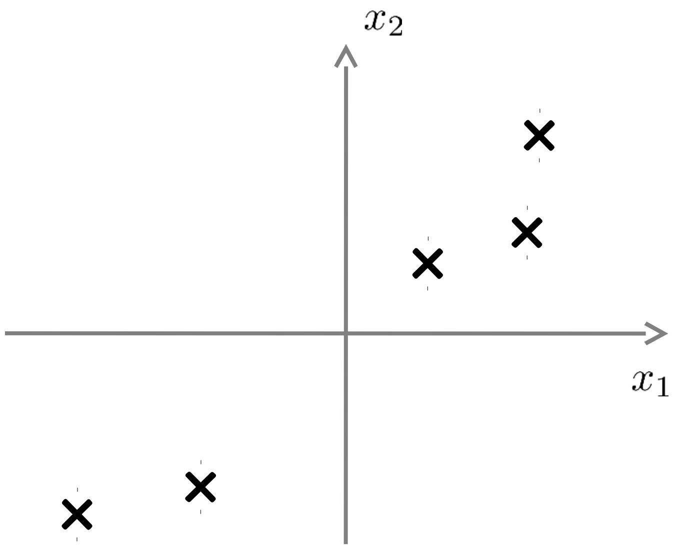
Preprocessing (feature scaling/mean normalization):

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

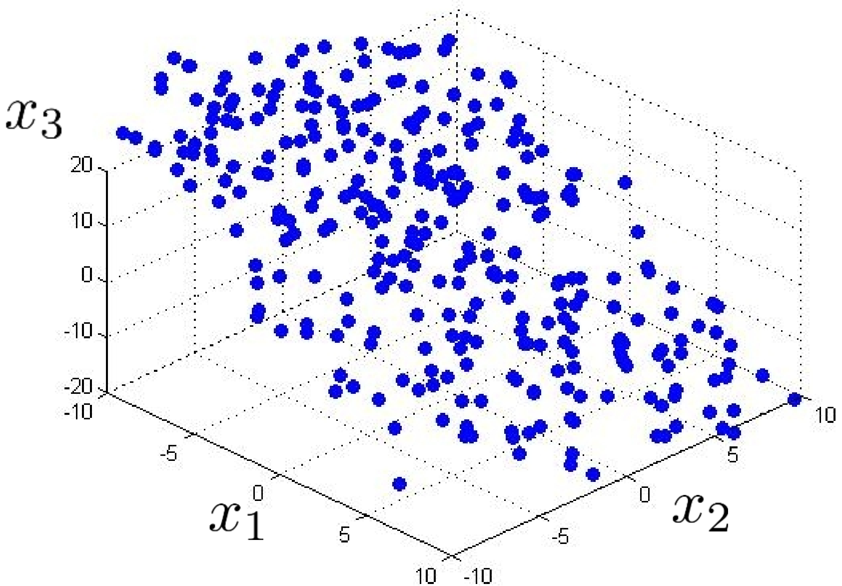
Replace each $x_j^{(i)}$ with $x_j - \mu_j$.

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.

# Principal Component Analysis (PCA) algorithm




Reduce data from 2D to 1D




Reduce data from 3D to 2D

Symbol *sigma* representing covariance matrix

# 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);
```

# Principal Component Analysis (PCA) algorithm

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

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

# 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;
```

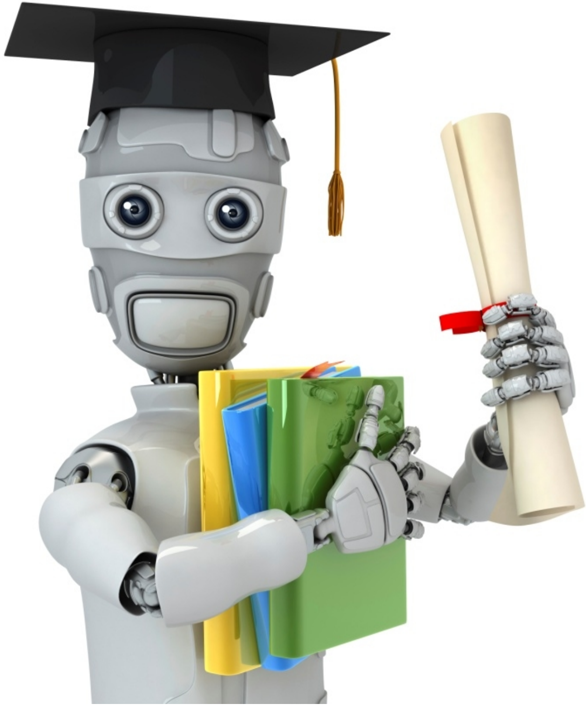
In PCA, we obtain $z \in \mathbb{R}^k$ from $x \in \mathbb{R}^n$ as follows:

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

Which of the following is a correct expression for $z_j$?

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

**Correct Response**

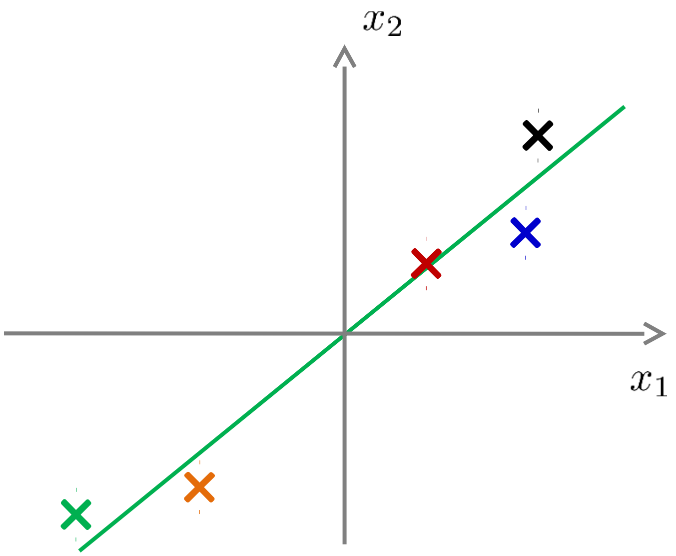


Machine Learning

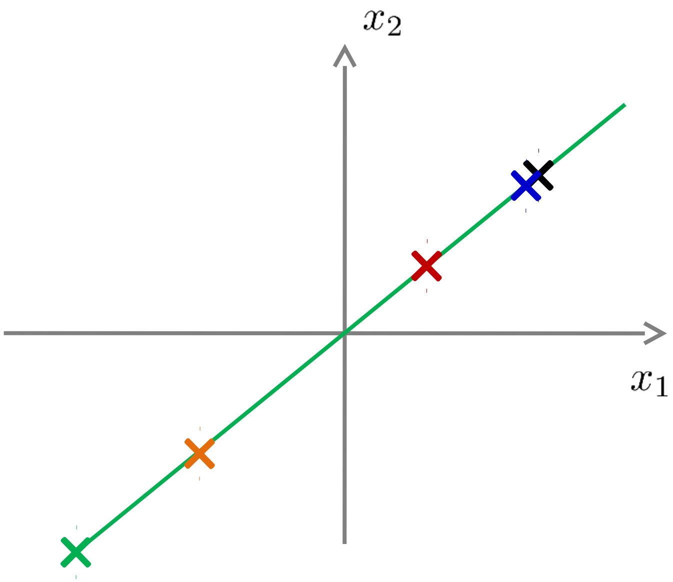
# Dimensionality Reduction

## Reconstruction from compressed representation

# Reconstruction from compressed representation




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


Andrew Ng

Suppose we run PCA with k = n, so that the dimension of the data is not reduced at all. (This is not useful in practice but is a good thought exercise.) Recall that the percent / fraction of variance retained is given by: $\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}}$ Which of the following will be true? Check all that apply.

- [x] $U_{\text{reduce}}$ will be an $n \times n$ matrix.

**Correct Response**

- [x] $x_{\text{approx}} = x$ for every example $x$.

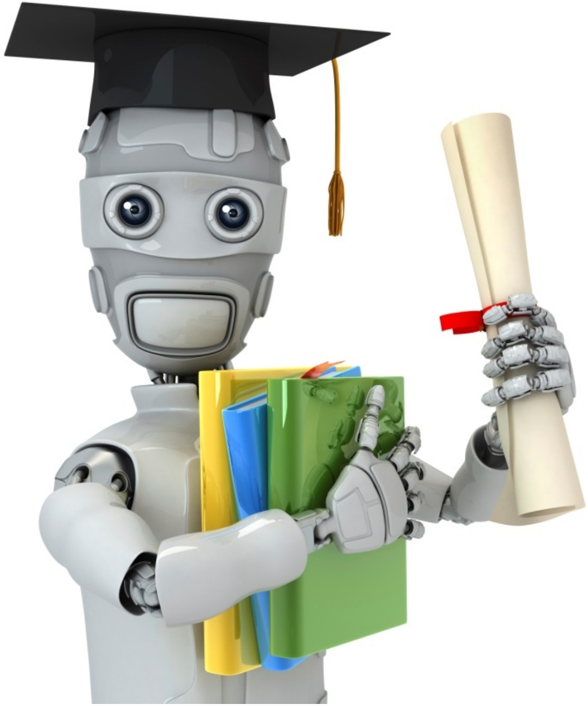
**Correct Response**

- [x] The percentage of variance retained will be 100%.

**Correct Response**

- [ ] We have that $\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} > 1$.

**Correct Response**



Machine Learning

# Dimensionality Reduction

## Choosing the number of principal components

# Choosing $k$ (number of principal components)

Average squared projection error:

Total variation in the data:

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

Check if

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

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

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

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

Pick smallest value of $k$ for which

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

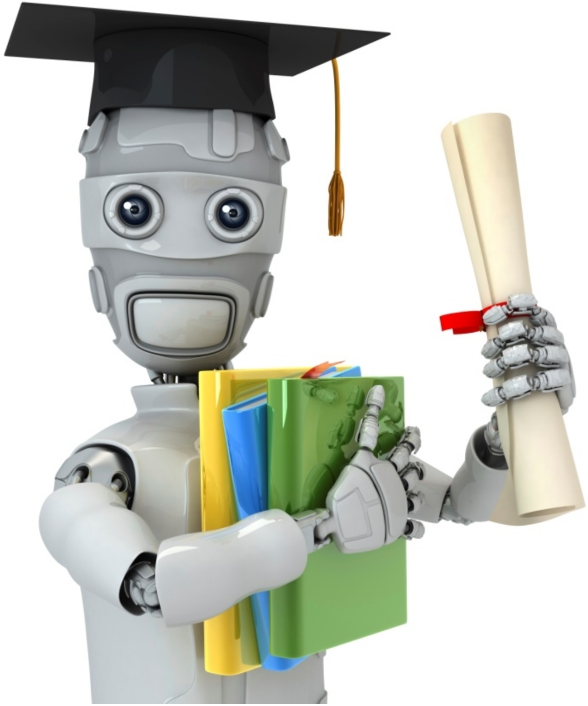
(99% of variance retained)

Previously, we said that PCA chooses a direction $u^{(1)}$ (or k directions $u^{(1)}, \dots, u^{(k)}$) onto which to project the data so as to minimize the (squared) projection error. Another way to say the same is that PCA tries to minimize:

- $\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2$
- $\frac{1}{m}\sum_{i=1}^{m}\|x_{\text{approx}}^{(i)}\|^2$
- $\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - x_{\text{approx}}^{(i)}\|^2$

Correct Response

- $\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} + x_{\text{approx}}^{(i)}\|^2$



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

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 $z^{(i)}$.

Which of the following are good / recommended applications of PCA? Select all that apply.

- [x] To compress the data so it takes up less computer memory / disk space

  **Correct Response**

- [x] To reduce the dimension of the input data so as to speed up a learning algorithm

  **Correct Response**

- [ ] Instead of using regularization, use PCA to reduce the number of features to reduce overfitting

  **Correct Response**

- [x] To visualize high-dimensional data (by choosing k = 2 or k = 3)

  **Correct Response**