Dimensionality Reduction

Motivation I:
Data Compression

Machine Learning

Data Compression

Reduce data from 2D to 1D

Data Compression

Reduce data from 2D to 1D

$x^{(1)} \in \mathbb{R}^2 \rightarrow z^{(1)} \in \mathbb{R}$

$x^{(2)} \in \mathbb{R}^2 \rightarrow z^{(2)} \in \mathbb{R}$

$\vdots$

$x^{(m)} \in \mathbb{R}^2 \rightarrow z^{(m)} \in \mathbb{R}$
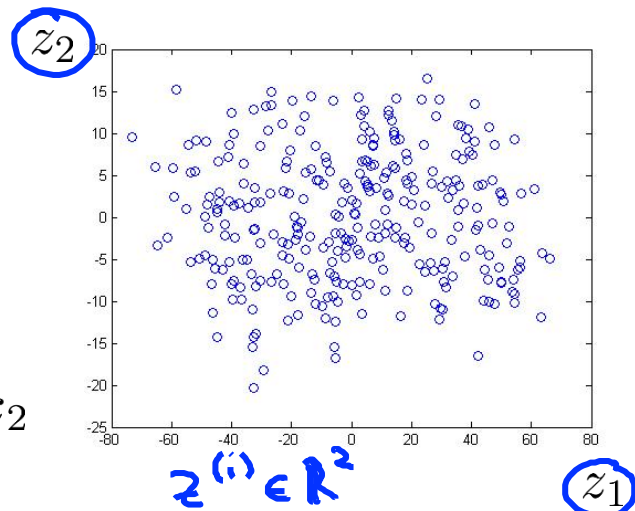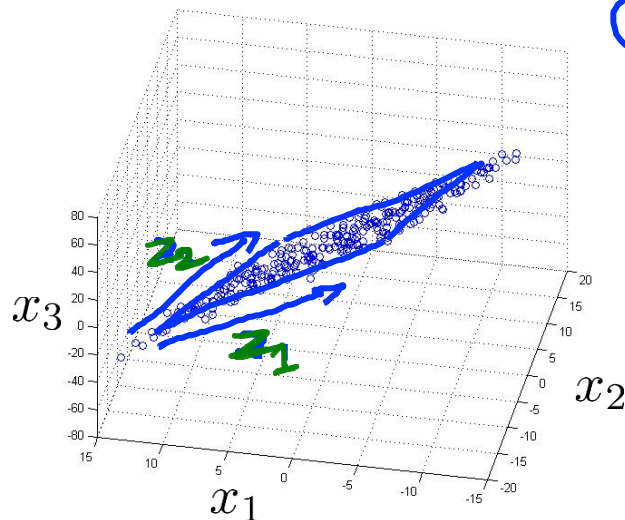
2-Dimension   1-Dimension

Andrew Ng

Data Compression

$10000 \rightarrow 1000$

Reduce data from 3D to 2D

$x^{(i)} \in \mathbb{R}^3$

3-D points

z_2

$z^{(i)} \in \mathbb{R}^2$

z_1

$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$

$z^{(i)} = \begin{bmatrix} z_1^{(i)} \\ z_2^{(i)} \end{bmatrix}$

Dimensionality Reduction

Motivation II:
Data Visualization

Machine Learning

Data Visualization

$x \in \mathbb{R}^{50}$ $x^{(i)} \in \mathbb{R}^{50}$

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

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
...	...	...

$z^{(i)} \in \mathbb{R}^2$

Reduce data from 50D to 2D

Data Visualization

per. person GDP (economic activity)

z_2

Singapore

USA

Country size / GDP

$z^{(i)} \in \mathbb{R}$

z_1

1

Andrew Ng

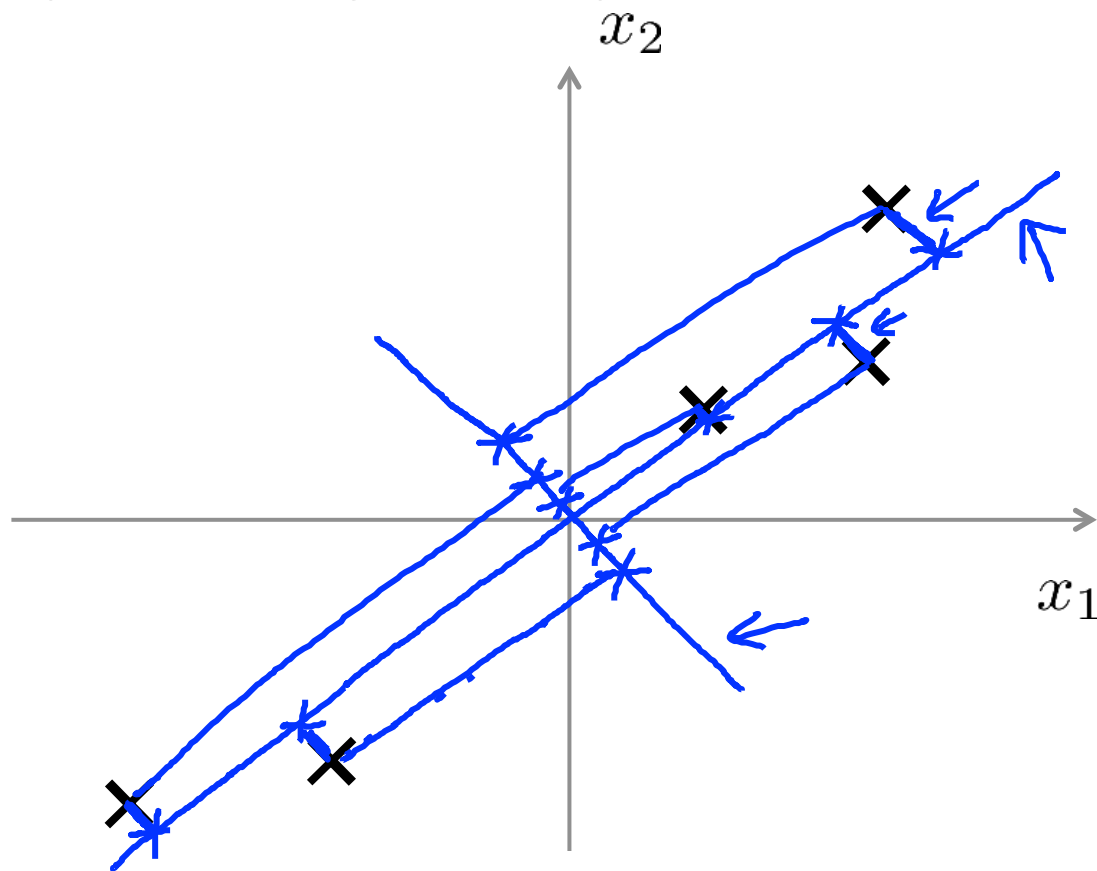# Dimensionality Reduction

Principal Component Analysis problem formulation

Machine Learning

Principal Component Analysis (PCA) problem formulation

Principal Component Analysis (PCA) problem formulation

$$3D \to 2D$$
$$K = 2$$

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

Andrew Ng

Dimensionality Reduction

Principal Component Analysis algorithm

Machine Learning

Data preprocessing

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

Preprocessing (feature scaling/mean normalization):

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$$
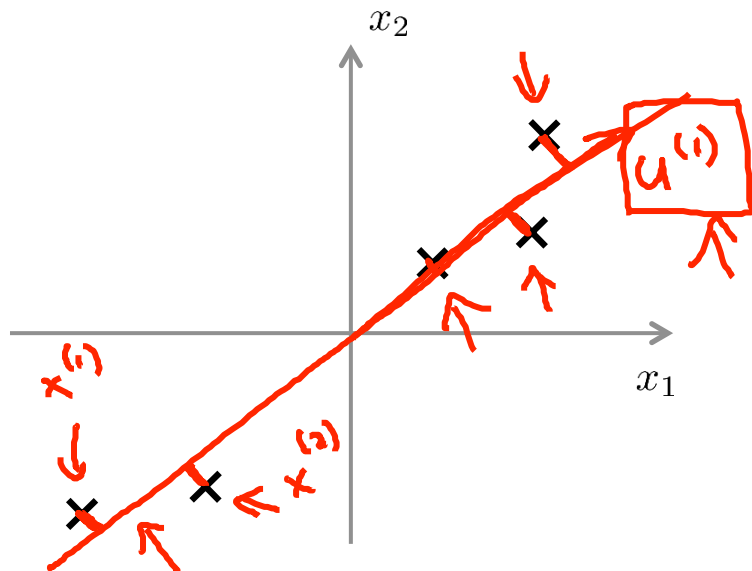
Replace each $x_j^{(i)}$ with $x_j - \mu_j$.

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.
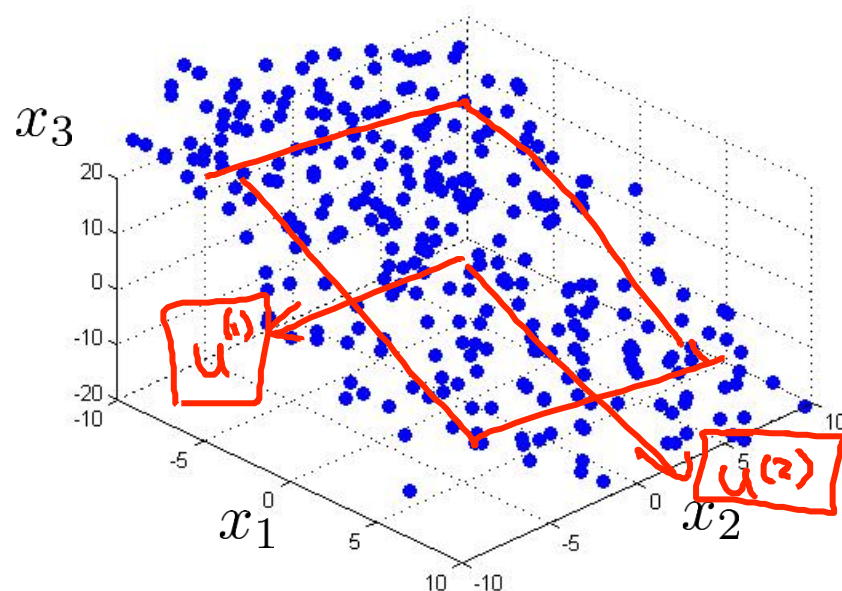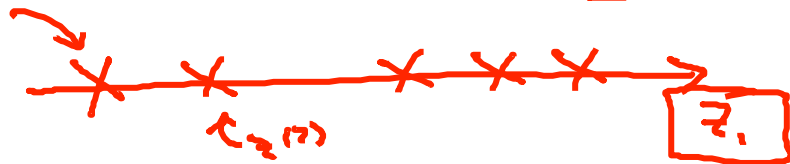
i^{th} data point of the jth feature

$x_j^{(i)}$ ⟵

$x_j^{(i)} - \mu_j$ ⟹ Mean of the jth feature

S_j ⟶ standard deviation of the jth feature

Principal Component Analysis (PCA) algorithm

Reduce data from 2D to 1D

$x^{(i)} \in \mathbb{R}^2 \rightarrow z^{(i)} \in \mathbb{R}$

Reduce data from 3D to 2D

$x^{(i)} \in \mathbb{R}^3 \rightarrow z^{(i)} \in \mathbb{R}^2$

$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$

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:

$$[\text{U},\text{S},\text{V}] = \text{svd}(\text{Sigma});$$

$\rightarrow$ Singular value decomposition

$eig(Sigma)$

$n \times n$ matrix

$$U = \begin{bmatrix} | & | & | & & | \\ u^{(1)} & u^{(2)} & u^{(3)} & \dots & u^{(n)} \\ | & | & | & & | \end{bmatrix}$$

k

$U \in \mathbb{R}^{n \times n}$

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

Principal Component Analysis (PCA) algorithm

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

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

$$\underbrace{}_{k}$$

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

$$z^{(i)} = \underbrace{\begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(k)} \\ | & | & & | \end{bmatrix}^T}_{n \times k} \quad x^{(i)} = \underbrace{\begin{bmatrix} - & (u^{(1)})^T & - \\ & \vdots & \\ - & (u^{(k)})^T & - \end{bmatrix}}_{k \times n} \begin{matrix} \downarrow \\ x^{(i)} \\ \sim \\ n \times 1 \end{matrix}$$

$$z \in \mathbb{R}^k \qquad \underbrace{}_{U_{reduce}} \qquad \underbrace{}_{k \times 1}$$

Principal Component Analysis (PCA) algorithm summary

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

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

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

$$X = \begin{bmatrix} \text{—} & x^{(1)T} & \text{—} \\ & \vdots & \\ \text{—} & x^{(m)T} & \text{—} \end{bmatrix}$$

$$\text{Sigma} = (1/m)* X'* X;$$

$$x \in \mathbb{R}^n \qquad \cancel{x_0 = 1}$$

Dimensionality Reduction

Reconstruction from compressed representation

Machine Learning

Reconstruction from compressed representation

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

$x_{approx}^{(1)} \in \mathbb{R}^2$

$x_{approx}^{(2)}$

$$z \in \mathbb{R} \longrightarrow x \in \mathbb{R}^2$$

$$x_{approx}^{(1)} \approx \underbrace{U_{reduce}}_{n \times k} \cdot \underbrace{z^{(1)}}_{k \times 1}$$
$$\mathbb{R}^n \qquad \underbrace{\qquad\qquad}_{n \times 1}$$

Andrew Ng
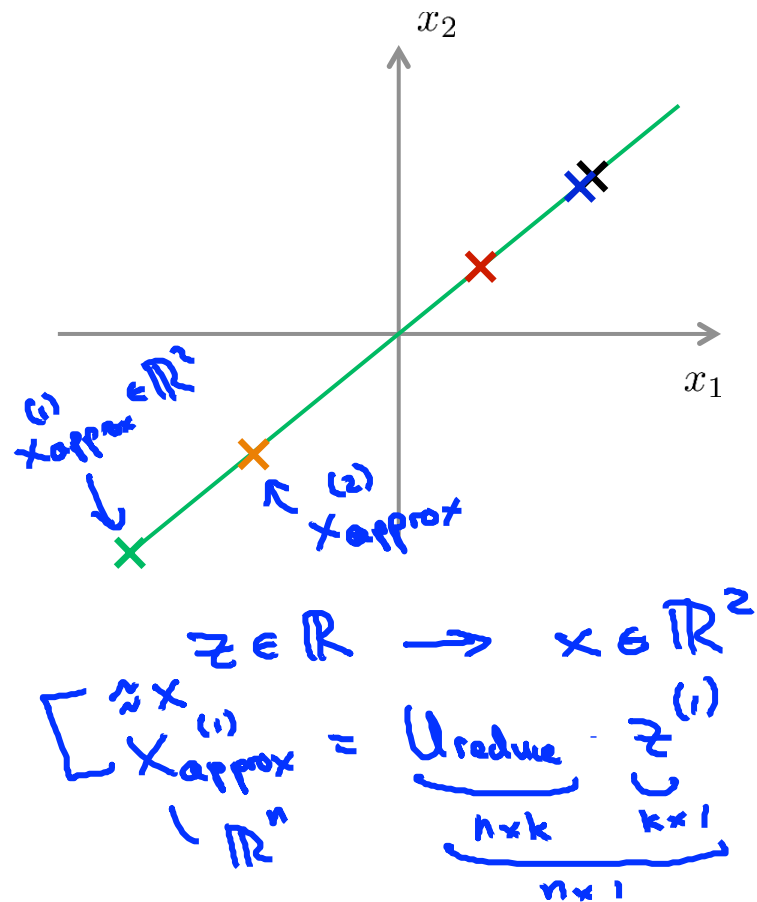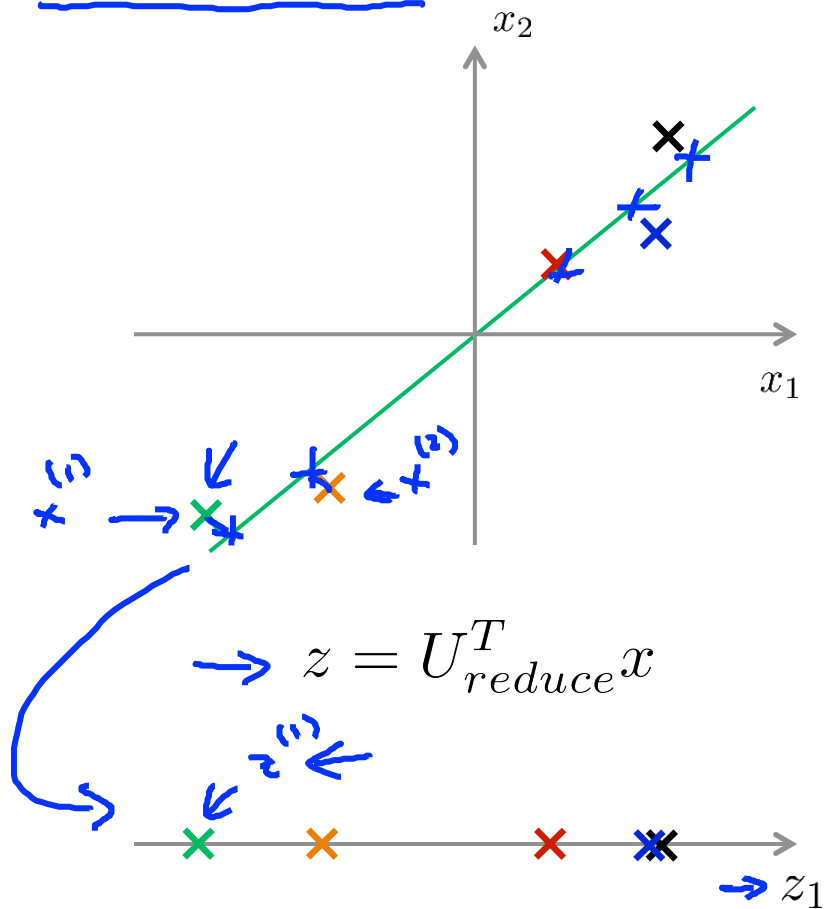
Dimensionality Reduction

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

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

$\to$

~~0.05~~ ~~5%~~

0.10 (10%)

$\to$ "99% of variance is retained"

95% to 90%

Choosing k (number of principal components)

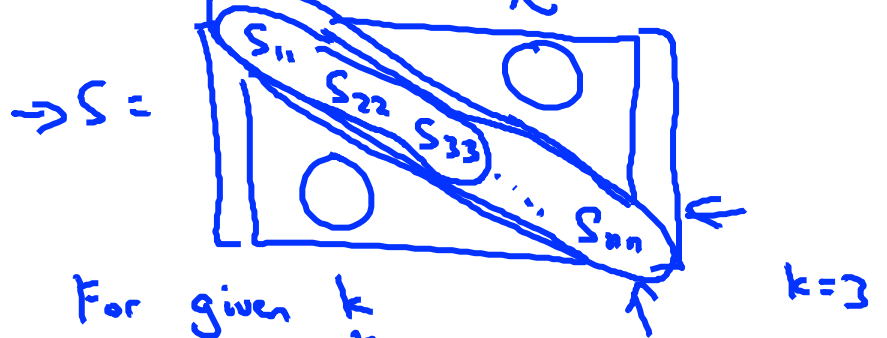Algorithm:

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

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

Check if

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

$k = 17$

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

$\rightarrow S =$

S_{11} S_{22} S_{33} S_{nn}

$k = 3$

For given k

$\rightarrow 1 - \dfrac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \leq 0.01$

$\rightarrow \dfrac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \geq 0.99$

Choosing k (number of principal components)

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

Pick smallest value of k for which

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

$k = 100$

(99% of variance retained)

Dimensionality Reduction

Advice for applying PCA

Machine Learning

Supervised learning speedup

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

$x^{(i)} \in \mathbb{R}^{10,000}$

Extract inputs:

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

$\downarrow PCA$

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

U_{reduce}

New training set:

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

x
$\downarrow$
z

$h_\theta(z) = \dfrac{1}{1 + e^{-\theta^T z}}$

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.
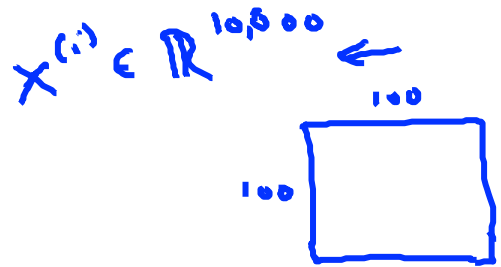
$x \rightarrow z$

Application of PCA

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

Choose k by % of variance retain

- 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 $k < n.$ — 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.

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

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 $z^{(i)}$.