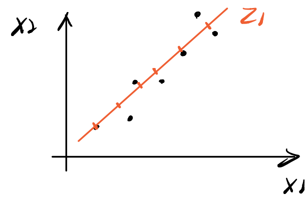
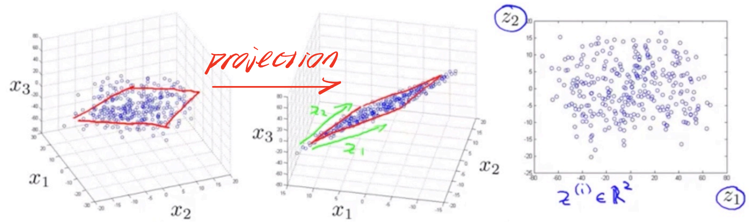
② Dimension reduction

application: ① Data compression   choose k by variation %



can reduce data from 2D to 1D.

$(x_1, x_2) \to z_1$



② Data visualization.

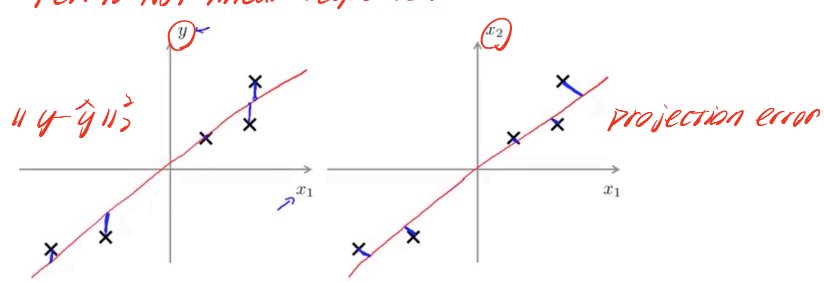
$\mathbb{R}^m \to \mathbb{R}^3$ or $\mathbb{R}^2$.

algorithm: Principle Component Analysis (PCA).

Reduce from n-dim to k-dim:

Find k vectors $u^{(1)}, u^{(2)} \dots u^{(k)}$ onto which to project the data, so as to minimize the projection error.

PCA is NOT linear regression



linear regression   PCA

$x \to y$ treat $x_1, x_2 \dots$ equally

Data preprocessing

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

Preprocessing (feature scaling/mean normalization):

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

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

② 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.

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

Principal Component Analysis (PCA) algorithm summary

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

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

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

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

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

First k columns

Singular Value Decomposition

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

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

α

"99% of variance is retained"

Algorithm:

Try PCA with $\cancel{k=1}$ $\cancel{k=2}$ $\cancel{k=3}$ $k=4$ ...

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

Check if

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

$k=17$

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

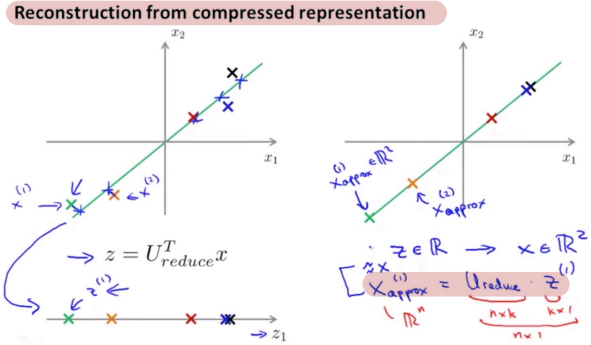
$\rightarrow S = \begin{bmatrix} S_{11} & & & \\ & S_{22} & & \\ & & S_{33} & \\ & & & \ddots \\ & & & & S_{nn} \end{bmatrix}$ $k=3$

For given k

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

$$\rightarrow \boxed{\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \ge 0.99}$$

Reconstruction from compressed representation



Supervised learning speedup

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

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

Extract inputs:

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

$\downarrow PCA$

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

$x \rightarrow z$

U_{reduce}

New training set:

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

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

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.

$x \rightarrow z$

Bad use of PCA: To prevent overfitting

$\rightarrow$ 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.

Don't know y : Might throw important information. Bad!

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

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

Misuse of PCA in ML : Use raw data first , try PCA whe some issues arise e.g. memory leak, speed ...