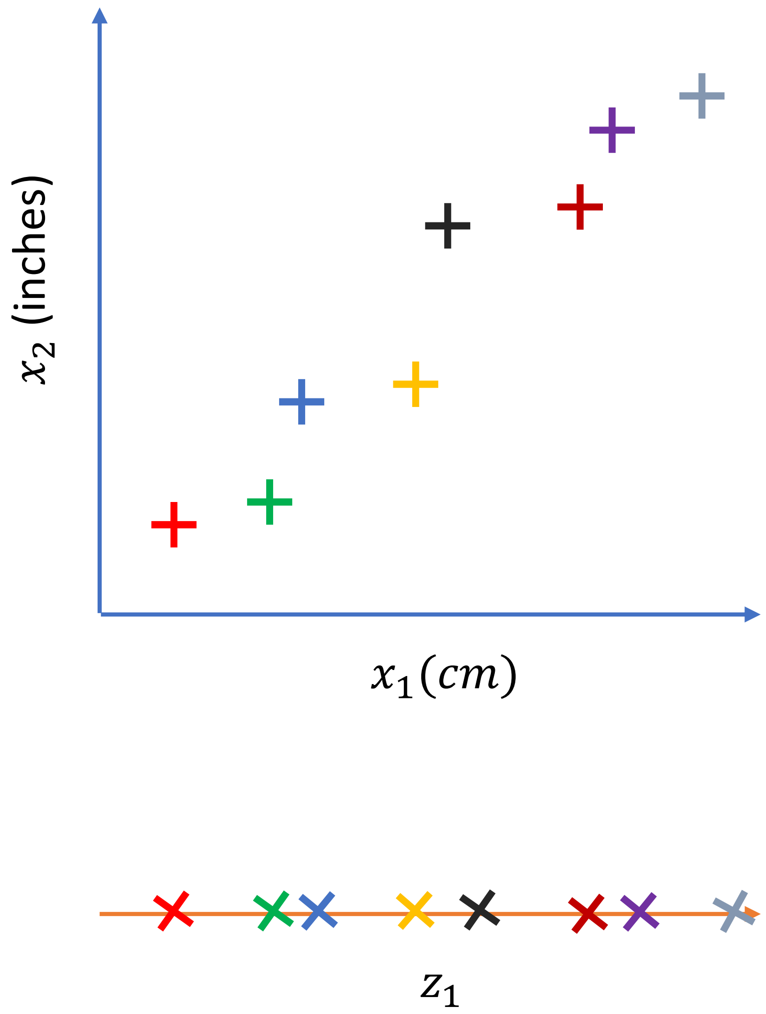
Dimensionality Reduction

PhD Abay Nussipbekov

Motivation 1: data compression



Reduce data from 2D to 1D

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

Motivation 2: Latent Variables

Measurable:

- Square footage
- Number of rooms
- School ranking
- Neighbourhood safety

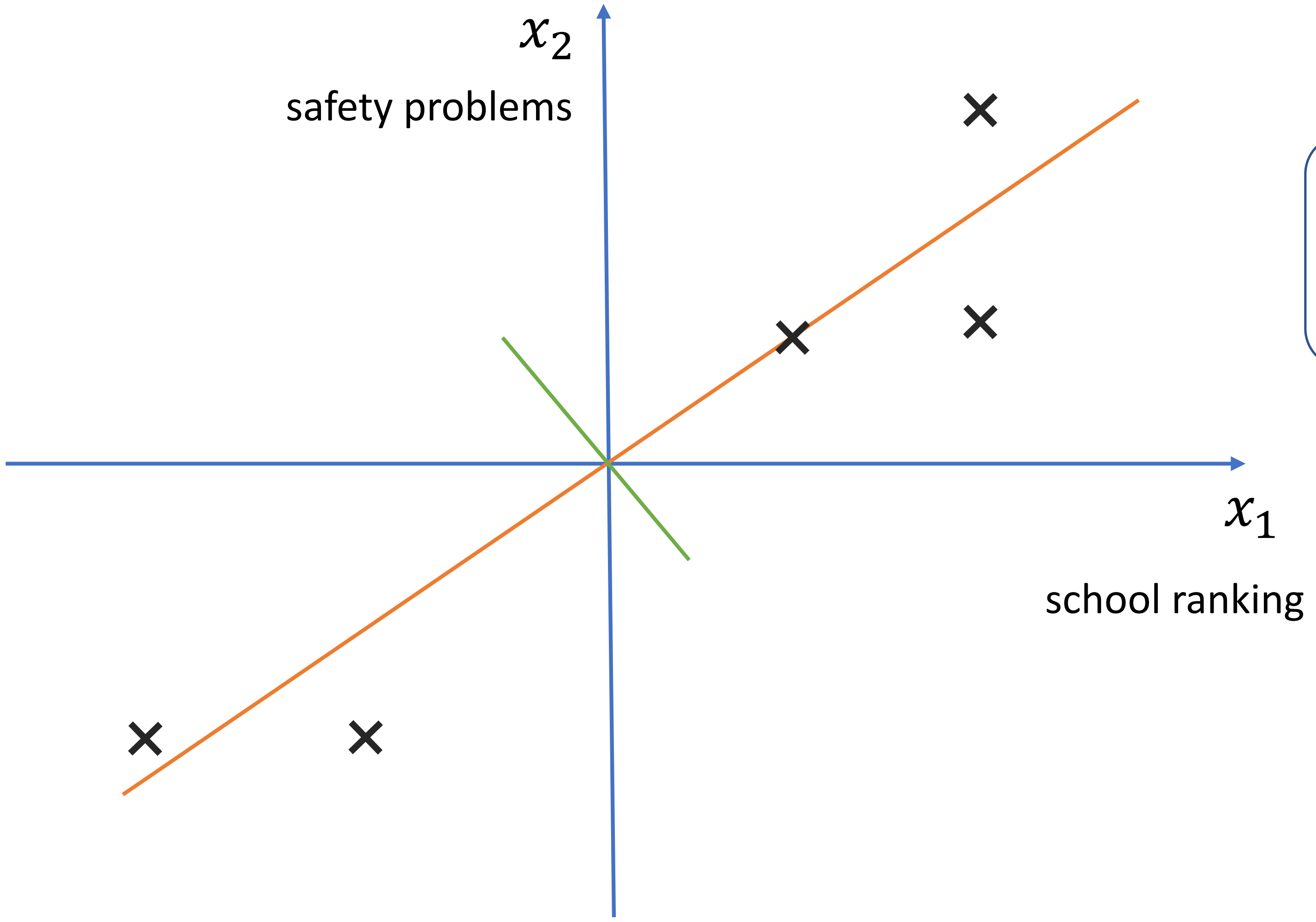
Latent:

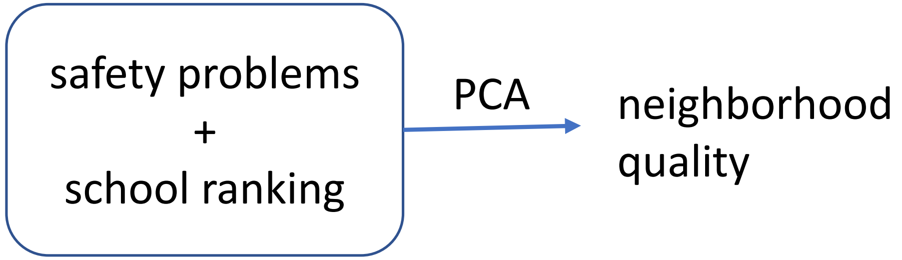
- Size
- Neighborhood

2 crucial things:

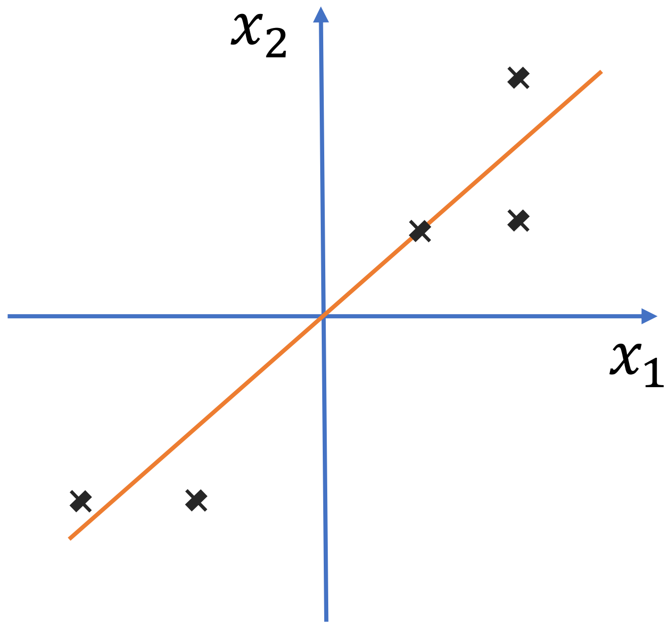
- many features but you hypothesize a smaller no. of features actually driving the patterns
- try making a composite feature that more directly probes the underlying phenomenon

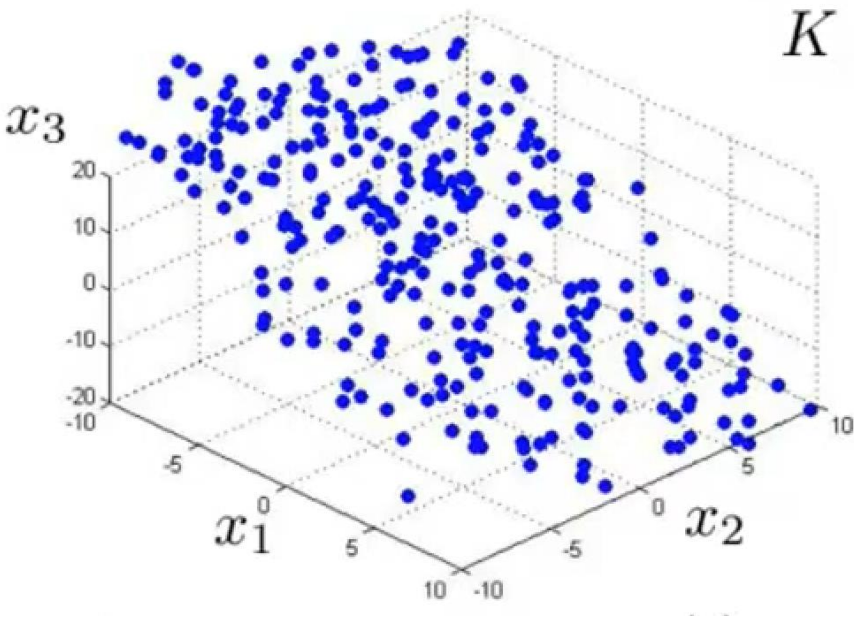
Principal component analysis





Principal component analysis (PCA) problem formulation

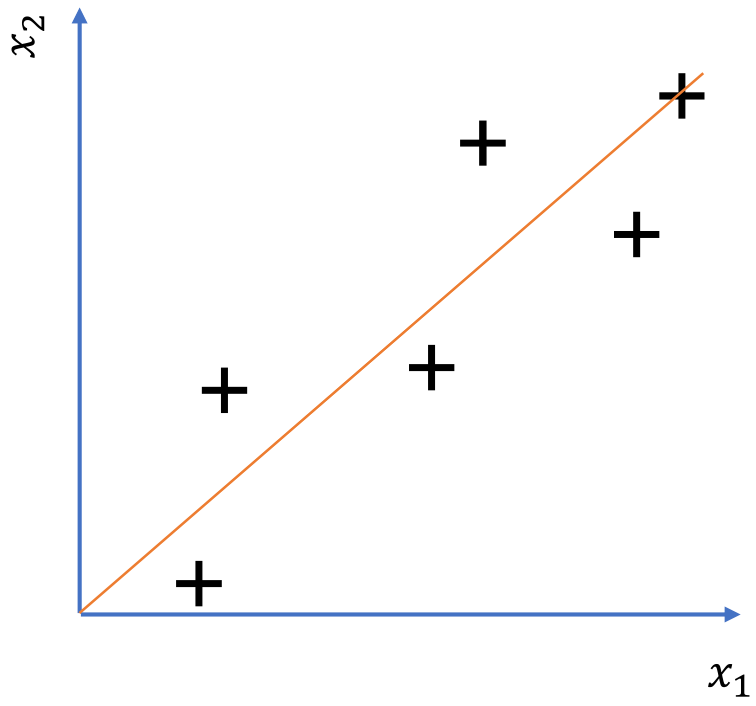


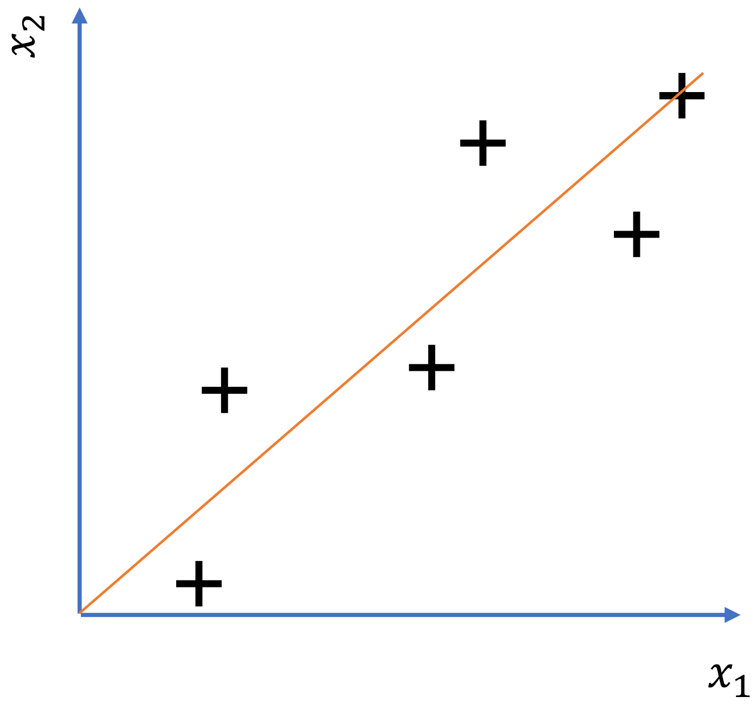


Reduce from 2-dimensional to 1-dimensional: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.

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

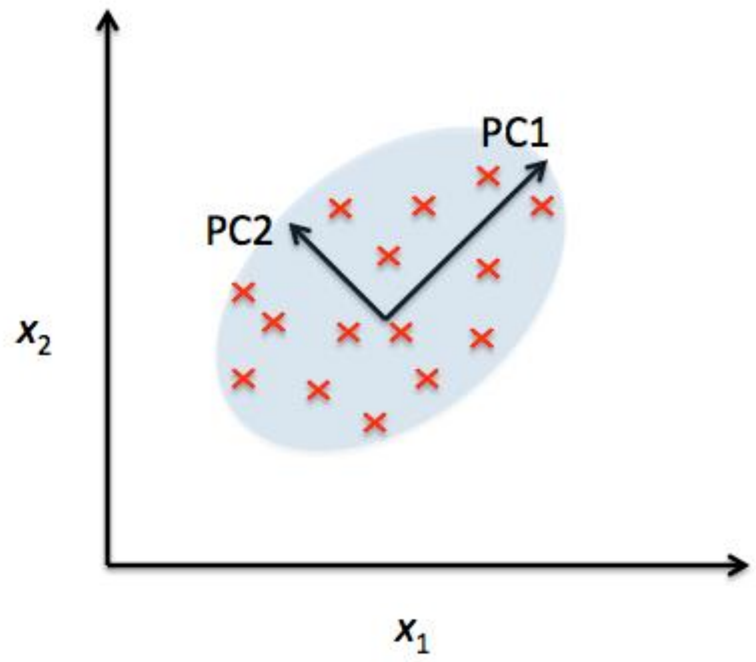
PCA is not a linear regression





Principal Component Analysis (PCA)

- Find the directions of maximum variance
- Project data onto the lower-dimensional space
- Original features: x1 and x2
- Principal components: **PC1** and **PC2**



Mapping to a low-dimensional space

- When we use PCA for dimensionality reduction, we construct a $n \times k$ transformation matrix **U**. We then map a sample vector x onto a new k-dimensional feature subspace (k << n)

$$\boldsymbol{x} = \left[x_1, x_2, \ldots, x_j\right], \boldsymbol{x} \in \mathbb{R}^{\boldsymbol{n}}$$

$$\downarrow \boldsymbol{xU} \quad \boldsymbol{U} \in \mathbb{R}^{n \times k}$$

$$z = [z_1, z_2, \ldots, z_k], \qquad \mathrm{z} \in \mathbb{R}^k$$

Principal components

- Transforming d-dimensional data to k dimensions
- First principal component will have the largest variance
- Second principal component will have next largest variance
- And so on...
- PCA sensitive to data scaling, so need to standardize features

PCA algorithm

1. Standardize the n-dimensional dataset.
2. Construct the covariance matrix.
3. Decompose the covariance matrix into its eigenvectors and eigenvalues.
4. Select k eigenvectors that correspond to the k largest eigenvalues, where k is the dimensionality of the new feature subspace ($k \leq n$).
5. Construct a projection matrix **U** from the "top" k eigenvectors.
6. Transform the n-dimensional input dataset **X** using the projection matrix **U** to obtain the new k-dimensional feature subspace.

PCA algorithm

- Compute “covariance matrix”:

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

- Compute “eigenvectors” of matrix Σ:

```
import numpy as np
cov_mat = np.cov(X_train_std.T)
eigen_vals, eigen_vecs = np.linalg.eig(cov_mat)

print('\nEigenvalues \n%s' % eigen_vals)
```

```
Eigenvalues
[ 4.8923083   2.46635032  1.42809973  1.01233462  0.84906459  0.60181514
  0.52251546  0.08414846  0.33051429  0.29595018  0.16831254  0.21432212
  0.2399553 ]
```

PCA algorithm

From `eigen_vals, eigen_vecs = np.linalg.eig(cov_mat)`, we get:

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

PCA algorithm summary

- After standardizing data:

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

```
eigen_vals, eigen_vecs = np.linalg.eig(cov_mat)
```

$U_{reduce}=U[:,1:k]$

$Z=U_{reduce}^{T}\times x$

Review / Definition of PCA

- Systemized way to transform input features into principal components
- Use principal components as a new features
- PCs are directions in data that maximize variance (minimize information loss) when you project down onto them
- More variance of data along a PC the higher that PC is ranked
- Second-most variance (without overlapping with first PC) – second PC

Choosing k (number of principal components)

- Typically, choose k to be smallest value so that

$$\frac{Average\ squared\ projection\ error}{Total\ variation\ in\ the\ data} \leq 0.01 \qquad (1\%)$$

"99% of data is retained"

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

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

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:

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

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 needed to store data
  - Speed up learning alrorithm
- Discovering latent variables
- Visualization

Bad use of PCA: to prevent overfitting

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

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 some algorithm on $\{(z^{(1)}, z^{(1)}), (z^{(2)}, z^{(2)}), \dots, (z^{(m)}, z^{(m)})\}$
  - Test on test set: Map $x_{test}^{(i)}$ to $z_{test}^{(i)}$.
- How about doing the whole thing without using PCA?
- Before running PCA, first try running with original data and only if that doesn’t work then apply PCA.