Dimensionality Reduction

Dimensionality Reduction

- reduce the number of features
- reducing the dimension of output, making it simple
- data compression
- visualization of data is simple
- complexity of the algorithm will be reduced
- it can help to overcome overfitting
- there will be information loss
- space and time complexity will be reduced

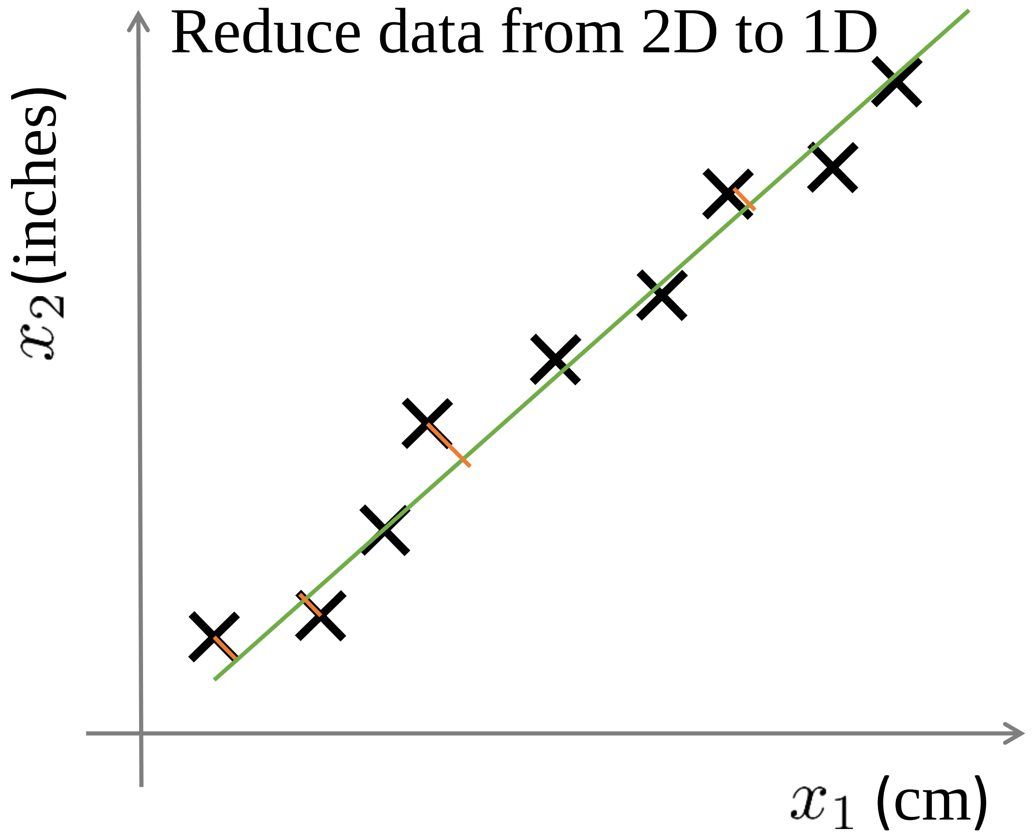
Need for Dimensionality Reduction

- In today’s life of technology, increasing amounts of data are produced and collected.
- Too much data can be a bad thing in machine learning because more features or dimensions can decrease a model’s accuracy (overfitting issue) and makes generalization of model difficult — this is known as the **curse of dimensionality**.

- **With dimensionality reduction** technique we are trying to reduce the complexity of a model and avoid overfitting.

- Two main categories of dimensionality reduction:
  - **Feature selection:** select a subset of the original features
  - **Feature extraction:** derive information from the feature set to construct a new feature subspace.

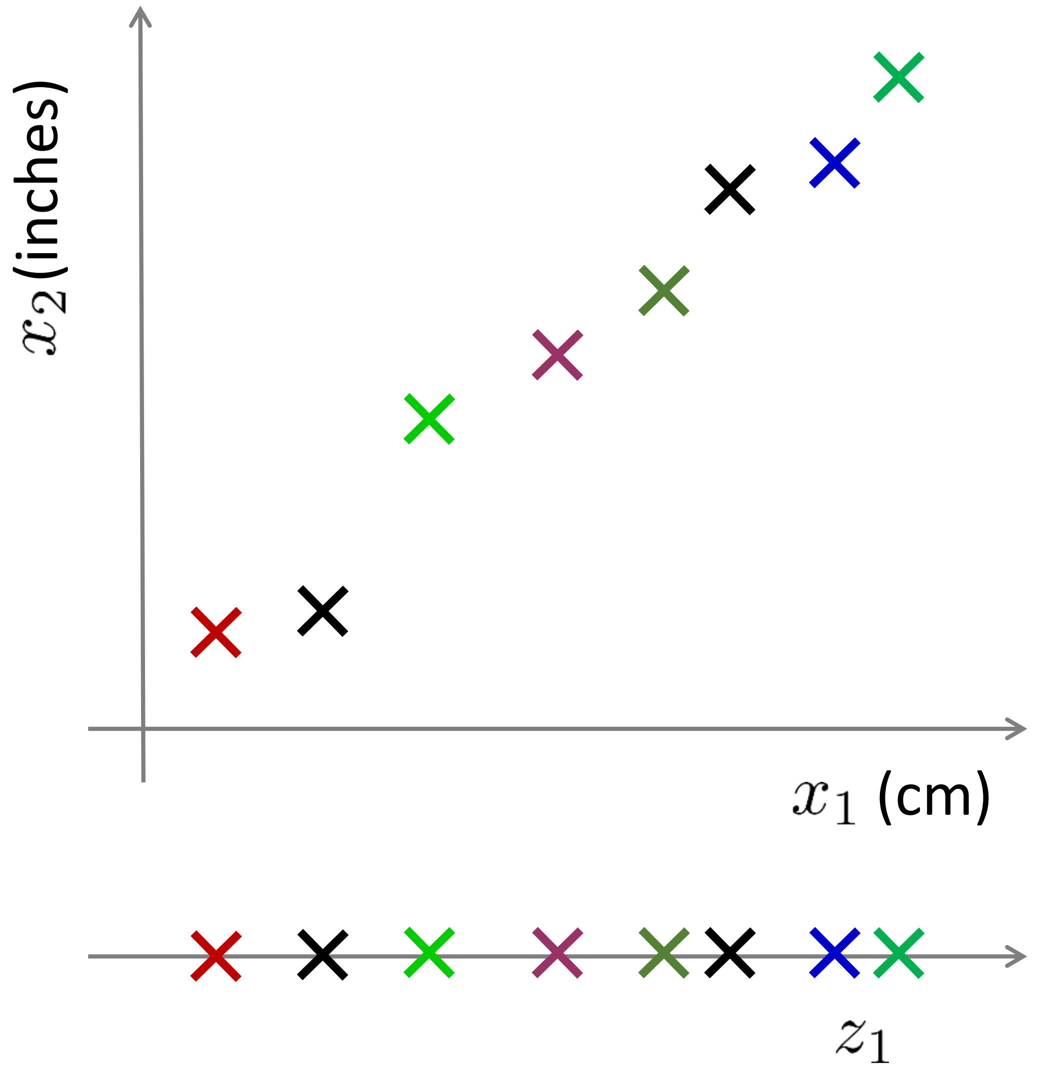
Data Compression



- For higher number of features or attributes, the visualization becomes harder and it also complicate the work of algorithm.
- It is observed that most of the features are correlated sometimes, and hence they are redundant.
- In such situation, dimensionality reduction algorithms come into play.
- It is the process of reducing the number of dimensions (i.e. random variables) under consideration, by obtaining a set of principal variables.

Transforming the data from a **high-dimensional** space into a **low-dimensional** space in such a way so that the **low-dimensional** representation retains some meaningful properties of the original data but reduces the memory and time required to store or process the transformed data.

Data Compression

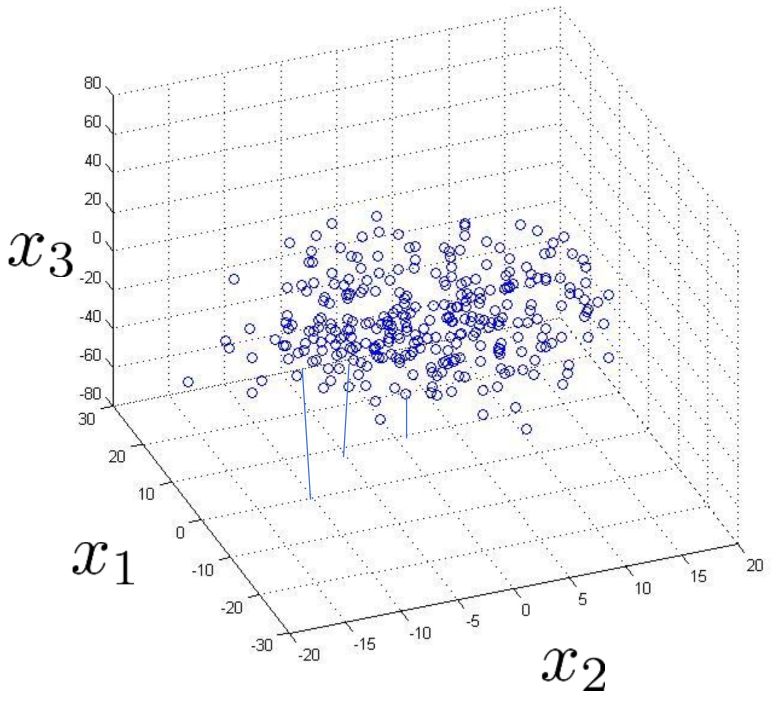


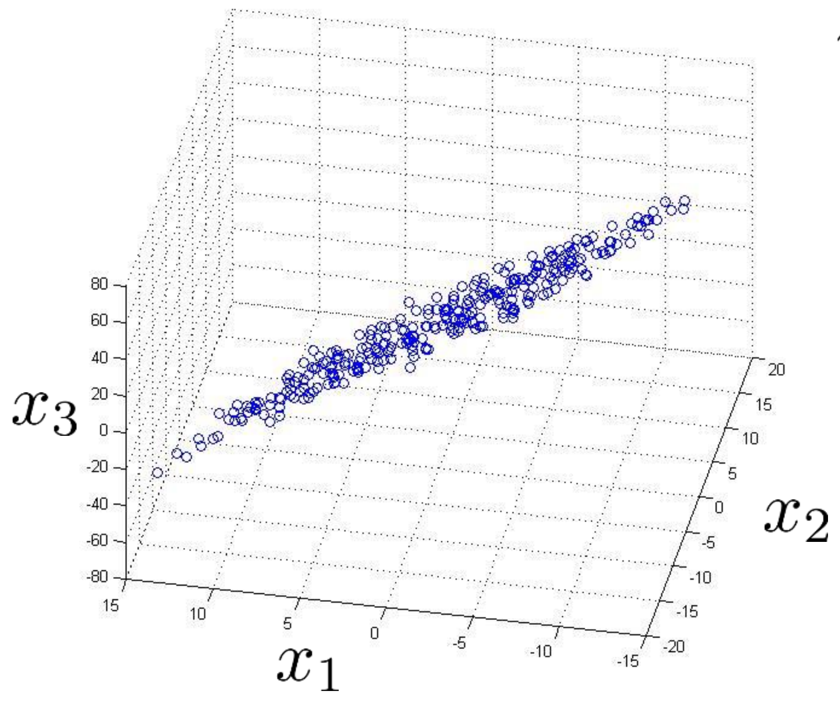
Reduce data from 2D to 1D

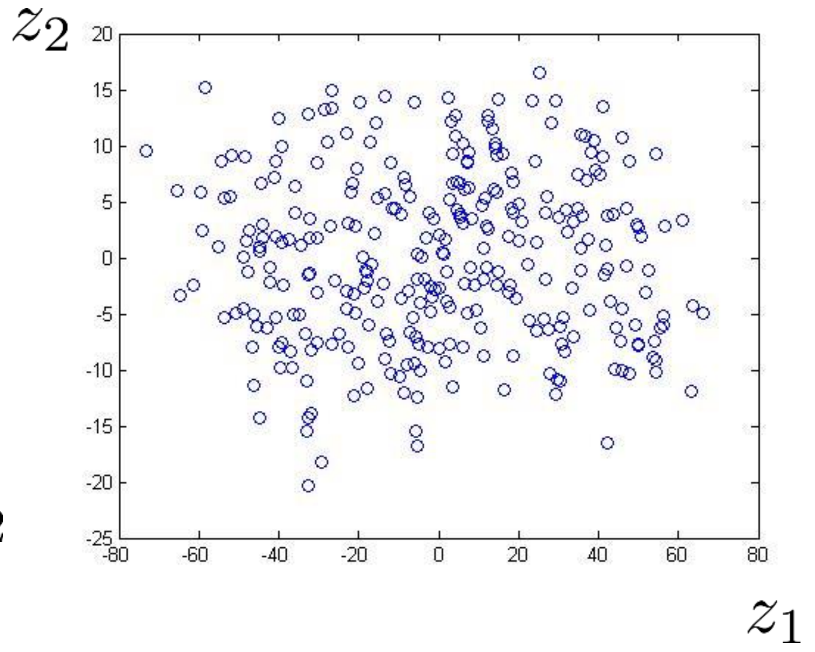
$$x^{(1)} \rightarrow z^{(1)}$$
$$x^{(2)} \rightarrow z^{(2)}$$
$$\vdots$$
$$x^{(m)} \rightarrow z^{(m)}$$

Data Compression

Reduce data from 3D to 2D







Data Visualization

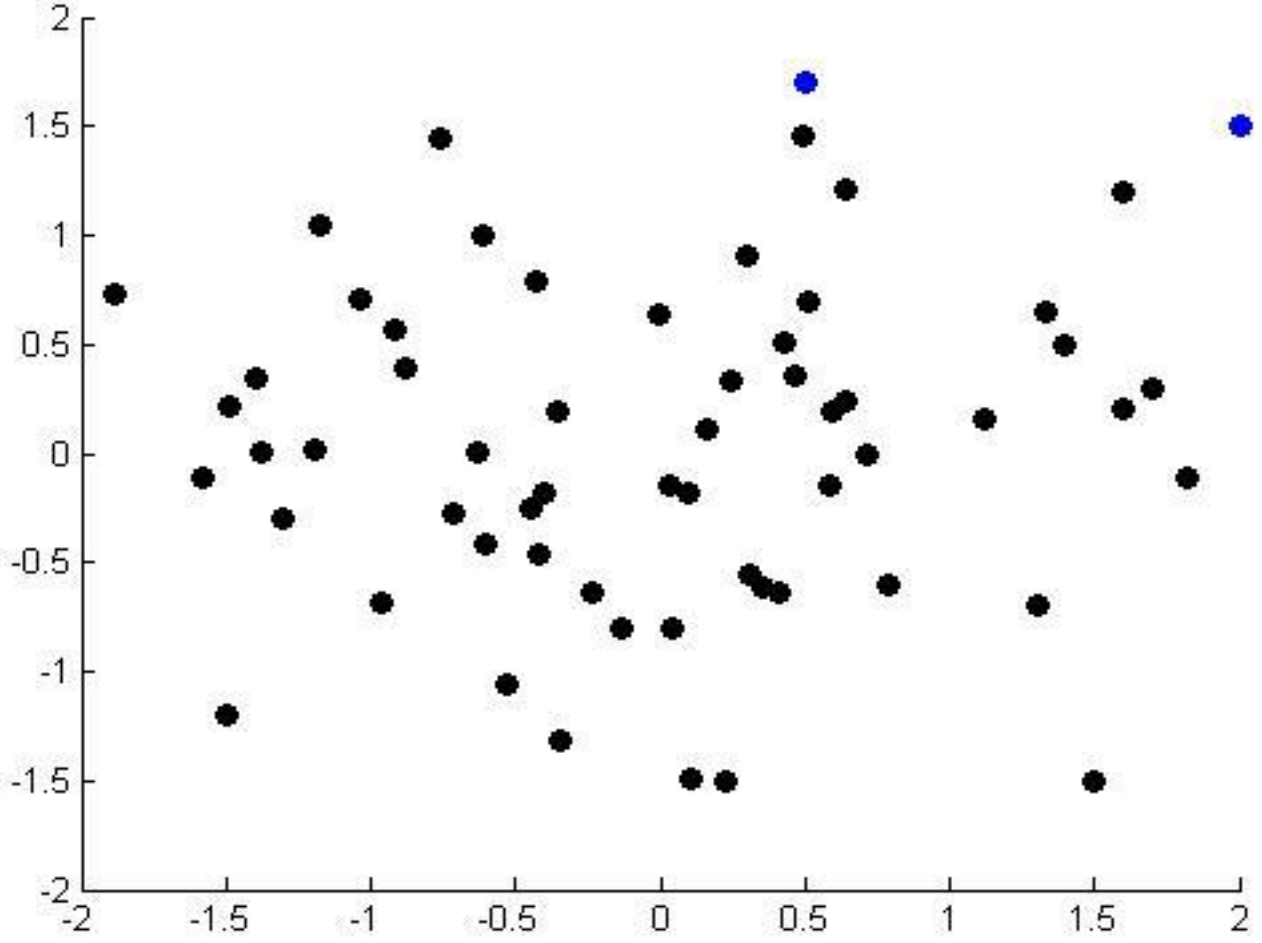
Country	GDP (trillions of US$)	Per capita GDP (thousands of intl. $)	Human Develop-ment Index	Life expectancy	Poverty Index (Gini as percentage)	Mean household income (thousands of US$)	…
Canada	1.577	39.17	0.908	80.7	32.6	67.293	…
China	5.878	7.54	0.687	73	46.9	10.22	…
India	1.632	3.41	0.547	64.7	36.8	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

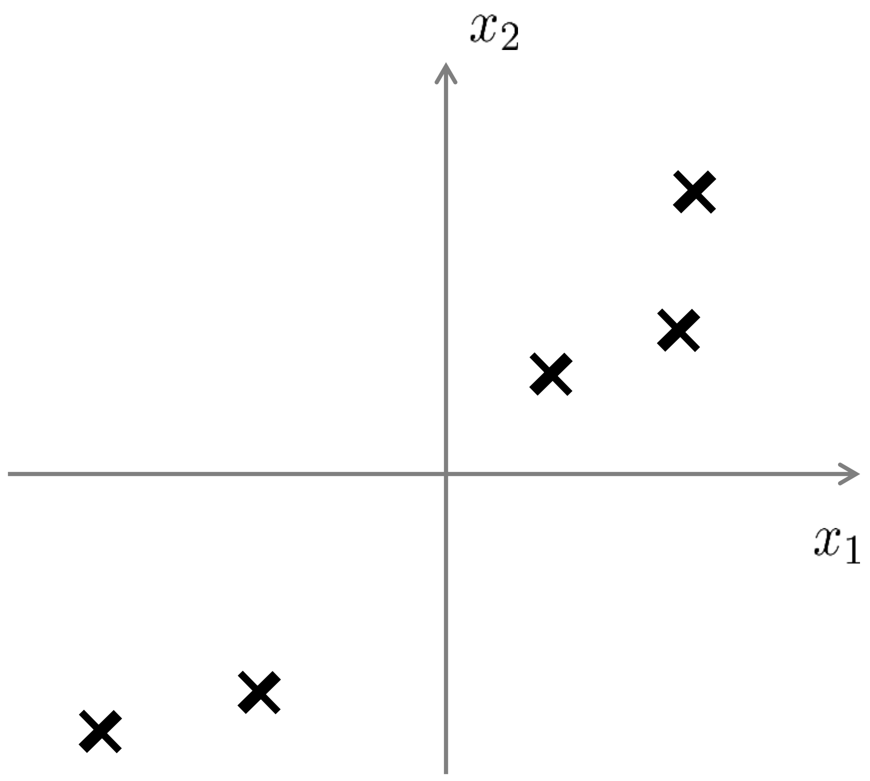


Principal Component Analysis (PCA)

This method was introduced by Karl Pearson in 1901.

It involves the following steps:

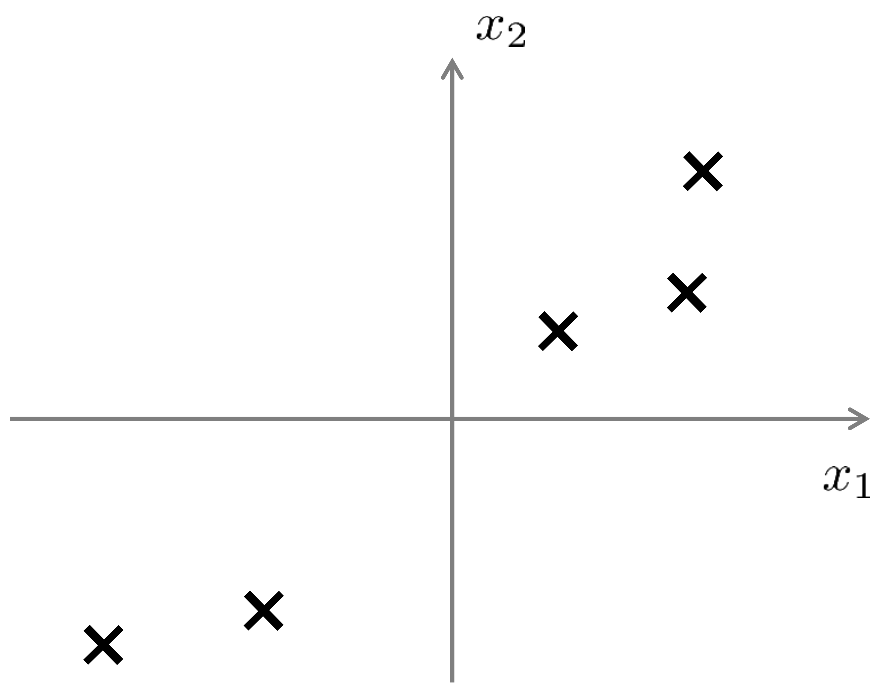
- Standardize the *d*-dimensional dataset.
- Construction of the covariance matrix for the given data.
- Decompose the covariance matrix into its eigenvectors and eigenvalues.
- Sort the eigenvalues by decreasing order to rank the corresponding eigenvectors.
- Select *k* eigenvectors which correspond to the *k* largest eigenvalues, where *k* is the dimensionality of the new feature subspace ($k \leq d$)
- Construct a projection matrix ***W*** from the "top" *k* eigenvectors.
- Transform the *d*-dimensional input dataset ***X*** using the projection matrix ***W*** to obtain the new *k*-dimensional feature subspace.



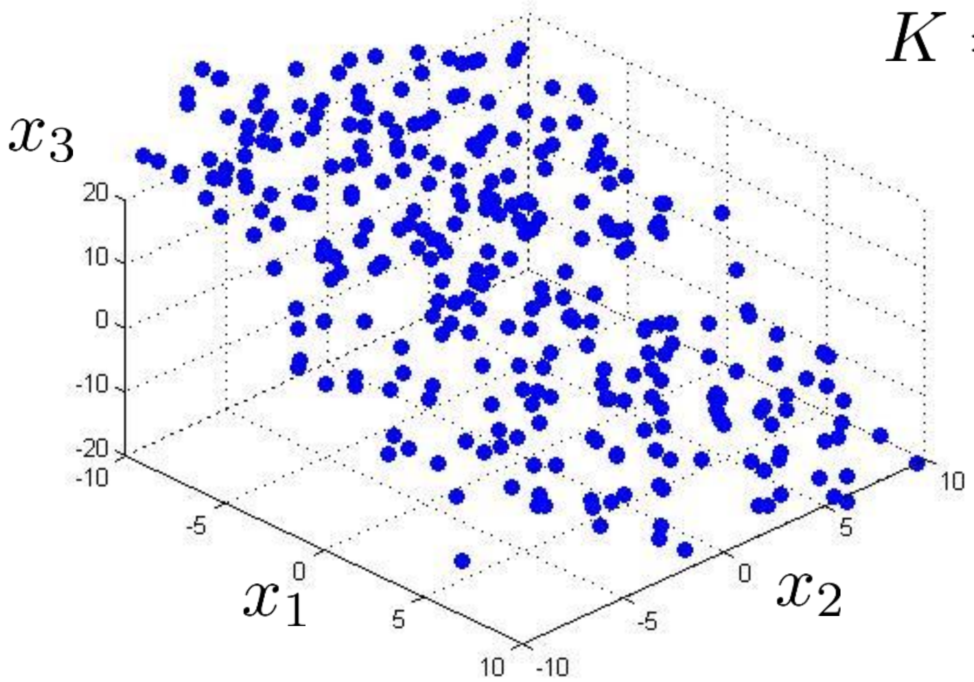
(PCA tries to find a low dimensional hyperplane (projection hyperplane) which has the minimum projection error)

Principal Component Analysis (**PCA**) is an unsupervised linear transformation technique that is widely used across different fields, most prominently for feature extraction and **dimensionality reduction**

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.

- **The orthogonal axes (principal components) of the new subspace can be interpreted as the directions of maximum variance given the constraint that the new feature axes are orthogonal to each other.**

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.

Note: The PCA is highly sensitive to data scaling therefore, there is a need to standardize the features *prior* to PCA. This ensures equal importance to all features.

Principal Component Analysis (PCA) algorithm

Reduce data from n-dimensions to k-dimensions

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

Compute "covariance matrix":

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

A **covariance matrix** is a square matrix that gives covariance between each pair of elements of a given random vector. It is symmetric and positive semi-definite in nature and its main diagonal contains variances (i.e., the covariance of each element with itself).

Compute "eigenvectors" of matrix Σ:

It generalizes the notion of variance to multiple dimensions.
Example: variation in a collection of random points in two-dimensional space cannot be characterized fully by a single number, nor would the variances in the x and y directions contain all of the necessary information; a [2* 2] matrix would be necessary to fully characterize the two-dimensional variation.

- The **eigenvectors** (principal components) determine the directions of the new feature space, and the **eigenvalues** determine their magnitude

Principal Component Analysis (PCA) algorithm

From eigen vectors, we get:

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

$$\curvearrowright$$

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

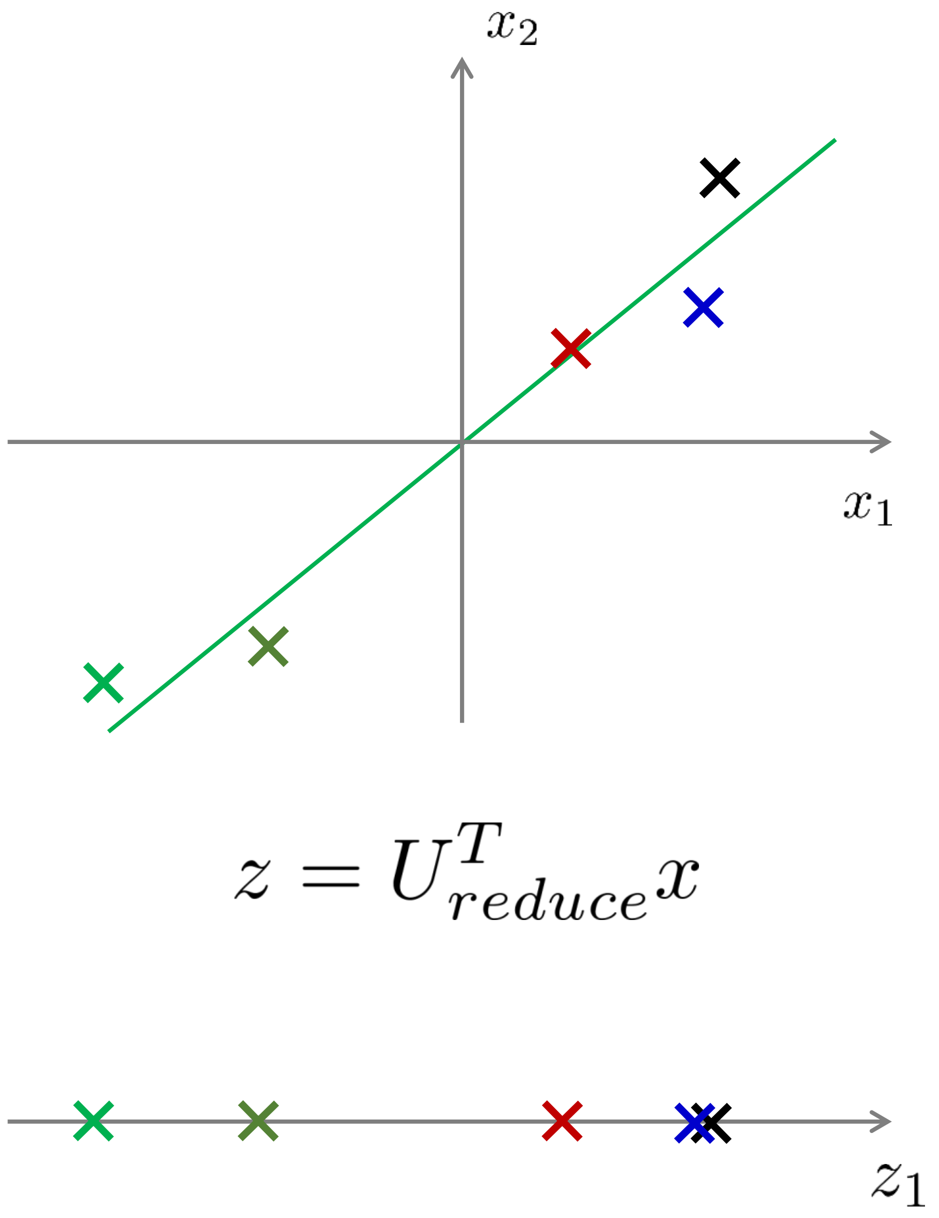
Z(i) = U * X(i)

$\curvearrowright$ n * 1

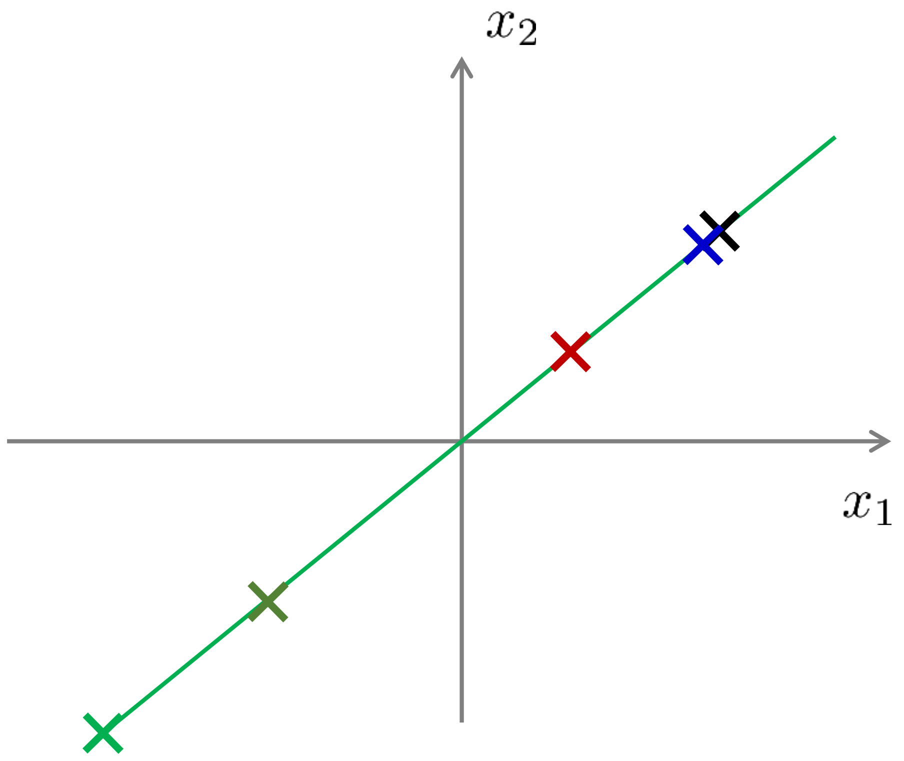
K * n

$\curvearrowright$ K * 1

Reconstruction from compressed representation



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



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”

Numerical:

Given data = { 2, 3, 4, 5, 6, 7 ; 1, 5, 3, 6, 7, 8 }.Compute the principal component using PCA Algorithm.

$$\begin{bmatrix} 2 \\ 1 \end{bmatrix} \begin{bmatrix} 3 \\ 5 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} \begin{bmatrix} 5 \\ 6 \end{bmatrix} \begin{bmatrix} 6 \\ 7 \end{bmatrix} \begin{bmatrix} 7 \\ 8 \end{bmatrix}$$

Mean vector (μ)
= ((2 + 3 + 4 + 5 + 6 + 7) / 6, (1 + 5 + 3 + 6 + 7 + 8) / 6)
= (4.5, 5)

Subtract mean vector (μ) from the given feature vectors.

$$\begin{bmatrix} -2.5 \\ -4 \end{bmatrix} \begin{bmatrix} -1.5 \\ 0 \end{bmatrix} \begin{bmatrix} -0.5 \\ -2 \end{bmatrix} \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \begin{bmatrix} 1.5 \\ 2 \end{bmatrix} \begin{bmatrix} 2.5 \\ 3 \end{bmatrix}$$

$$\boxed{\text{Covariance Matrix} = \frac{\Sigma (x_i - \mu)(x_i - \mu)^t}{n}}$$

$$m_1 = (x_1 - \mu)(x_1 - \mu)^t = \begin{bmatrix} -2.5 \\ -4 \end{bmatrix} \begin{bmatrix} -2.5 & -4 \end{bmatrix} = \begin{bmatrix} 6.25 & 10 \\ 10 & 16 \end{bmatrix}$$

$$m_2 = (x_2 - \mu)(x_2 - \mu)^t = \begin{bmatrix} -1.5 \\ 0 \end{bmatrix} \begin{bmatrix} -1.5 & 0 \end{bmatrix} = \begin{bmatrix} 2.25 & 0 \\ 0 & 0 \end{bmatrix}$$

$$m_3 = (x_3 - \mu)(x_3 - \mu)^t = \begin{bmatrix} -0.5 \\ -2 \end{bmatrix} \begin{bmatrix} -0.5 & -2 \end{bmatrix} = \begin{bmatrix} 0.25 & 1 \\ 1 & 4 \end{bmatrix}$$

$$m_4 = (x_4 - \mu)(x_4 - \mu)^t = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \begin{bmatrix} 0.5 & 1 \end{bmatrix} = \begin{bmatrix} 0.25 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

$$m_5 = (x_5 - \mu)(x_5 - \mu)^t = \begin{bmatrix} 1.5 \\ 2 \end{bmatrix} \begin{bmatrix} 1.5 & 2 \end{bmatrix} = \begin{bmatrix} 2.25 & 3 \\ 3 & 4 \end{bmatrix}$$

$$m_6 = (x_6 - \mu)(x_6 - \mu)^t = \begin{bmatrix} 2.5 \\ 3 \end{bmatrix} \begin{bmatrix} 2.5 & 3 \end{bmatrix} = \begin{bmatrix} 6.25 & 7.5 \\ 7.5 & 9 \end{bmatrix}$$

Now, Covariance matrix

$= (m_1 + m_2 + m_3 + m_4 + m_5 + m_6) / 6$

$$\text{Covariance Matrix} = \frac{1}{6}\begin{bmatrix} 17.5 & 22 \\ 22 & 34 \end{bmatrix}$$

$$\text{Covariance Matrix} = \begin{bmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{bmatrix}$$

λ is an Eigen value for a matrix M if it is a solution of the characteristic equation $|M - \lambda I| = 0$.

$$\begin{vmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{vmatrix} - \begin{vmatrix} \lambda & 0 \\ 0 & \lambda \end{vmatrix} = 0$$

$$\begin{vmatrix} 2.92 - \lambda & 3.67 \\ 3.67 & 5.67 - \lambda \end{vmatrix} = 0$$

$(2.92 - \lambda)(5.67 - \lambda) - (3.67 \times 3.67) = 0$
$16.56 - 2.92\lambda - 5.67\lambda + \lambda^2 - 13.47 = 0$
$\lambda^2 - 8.59\lambda + 3.09 = 0$

Thus, two Eigen values are $\lambda_1 = 8.22$ and $\lambda_2 = 0.38$.

$$\begin{bmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{bmatrix} \begin{bmatrix} X1 \\ X2 \end{bmatrix} = 8.22 \begin{bmatrix} X1 \\ X2 \end{bmatrix}$$

$2.92X_1 + 3.67X_2 = 8.22X_1$
$3.67X_1 + 5.67X_2 = 8.22X_2$

$5.3X_1 = 3.67X_2$ ………(1)
$3.67X_1 = 2.55X_2$ ………(2)

From (1) and (2), **$X_1 = 0.69X_2$**

Eigen Vector : $\begin{bmatrix} X1 \\ X2 \end{bmatrix} = \begin{bmatrix} 2.55 \\ 3.67 \end{bmatrix}$

Principal Component : $\begin{bmatrix} X1 \\ X2 \end{bmatrix} = \begin{bmatrix} 2.55 \\ 3.67 \end{bmatrix}$

Application of PCA

- Data Compression
- Reduce memory/disk needed to store data
- Speed up learning algorithm
- Model will have fewer degrees of freedom that reduces the likelihood of overfitting.
- The model will generalize more easily to new data.
- It also helps to remove redundant features and thus, promote the important variables.
- Data Visualization
- Popular applications of PCA: exploratory data analyses and de-noising of signals in stock market trading, and the analysis of genome data and gene expression levels in the field of bioinformatics.

Anomaly detection

Meaning of Anomaly Detection

- Find rare events/observations
- Something different
- Not easily classified
- Abnormal
- Peculiar
- Deviation from common rule/method
- Can be linked with structural defects, errors or frauds, medical problems, and malfunctioning equipment
- Machine learning algorithm helps in enhancing the speed of detection (anomaly)

"In machine learning, anomaly detection is referred to the identification of observations/events/items that do not conform to an expected pattern or to other items present in a dataset"

Raise suspicions by differing significantly from the majority of the data.

Anomalies aren't categorically good or bad, they're just deviations from the expected value for a metric at a given point in time.

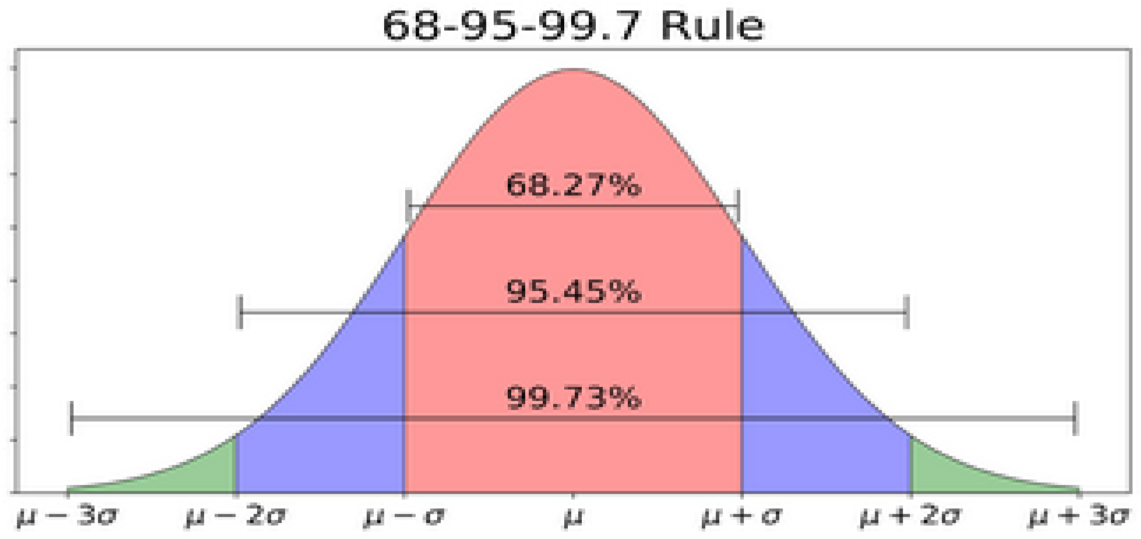
Application of Anomaly Detection

- Data cleaning
- Fraud detection
- Systems health monitoring
- Event detection in sensor networks
- Ecosystem disturbances
- Banking Security
- Surveillance
- predictive maintenance

How to Detect Anomalous Data Point

- Data Visualization (can provide important information)
- For two dimensional graph it becomes easy to find anomalous data point that is located outside the typical distribution
- However, this approach is not going to work for high-dimensional data
- For a collection of data points, we will typically have a certain distribution (e.g. a Gaussian distribution).
- To detect anomalies, the computation of the probability distribution p(x) is required from the data points.
- For a new example, x the value of p(x) is compared with a threshold ε. If $p(x) < \varepsilon$, it is considered as an anomaly because normal examples tend to have a large p(x) while anomalous examples tend to have a small p(x).

Normal Distribution Significance



•**Product** of two Normal Distribution results into a Normal Distribution
•**The Sum** of two Normal Distributions is a Normal Distribution
•**Convolution** of two Normal Distribution is also a Normal Distribution
•**Fourier Transformation** of a Normal Distribution is also Normal

•**68.27%** of data lies within 1 standard deviation of the mean
•**95.45%** of data lies within 2 standard deviations of the mean
•**99.73%** of data lies within 3 standard deviations of the mean
•Curve is **symmetric** around the Mean
•the **Mean, Median, and Mode are all the same.**
•It retains the normal shape throughout, unlike other probability distributions that change their properties after a transformation
•**Bell-shaped curve**

Example: Anomaly detection

Fraud Detection based on Transactions:

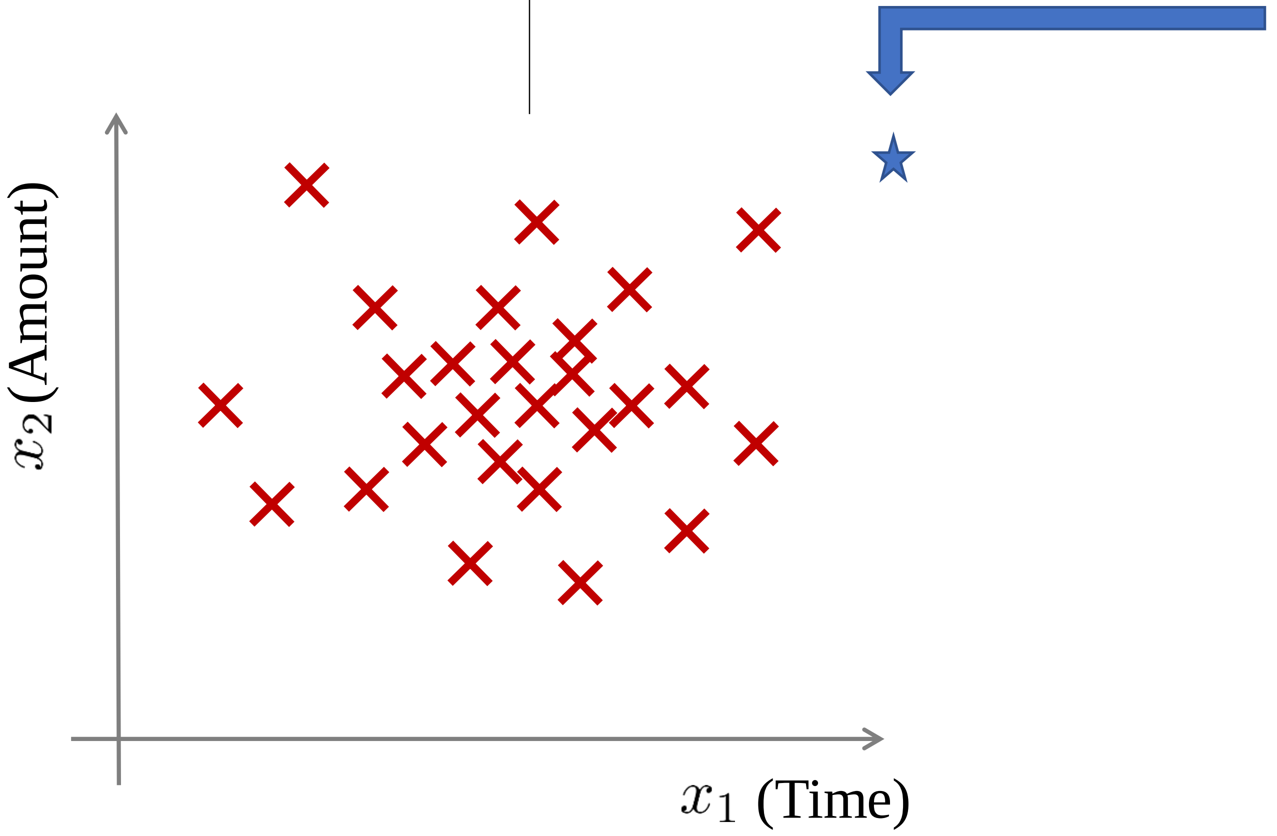
x_1 = Time

x_2 = Amount

$\dots$

Dataset: $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

New transaction: x_{test}

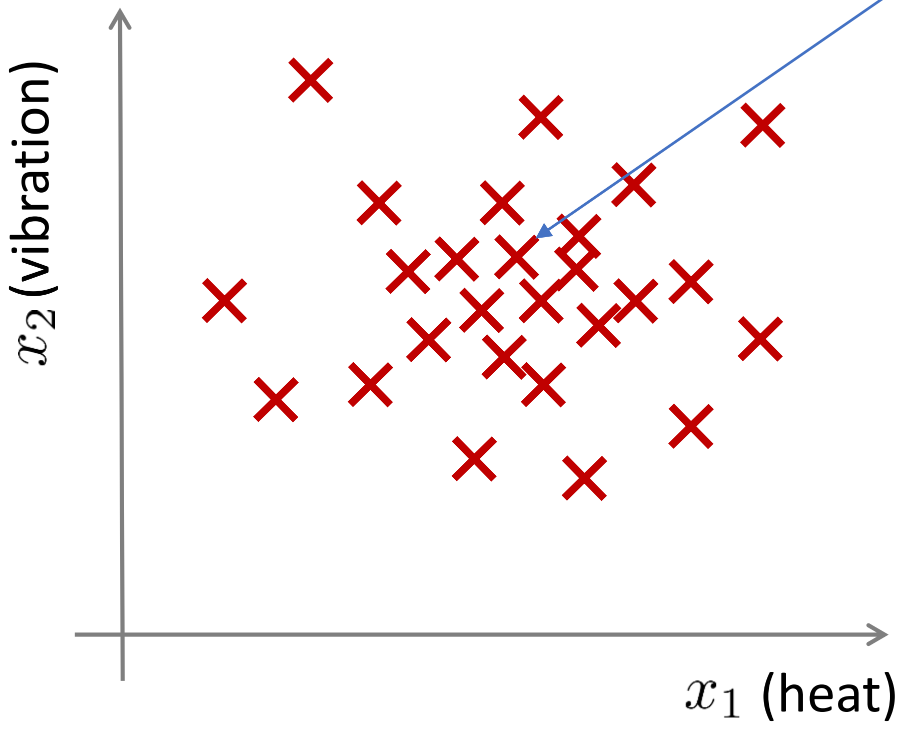


Density estimation

(Probability is high at the centre)

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

Is x_{test} anomalous?



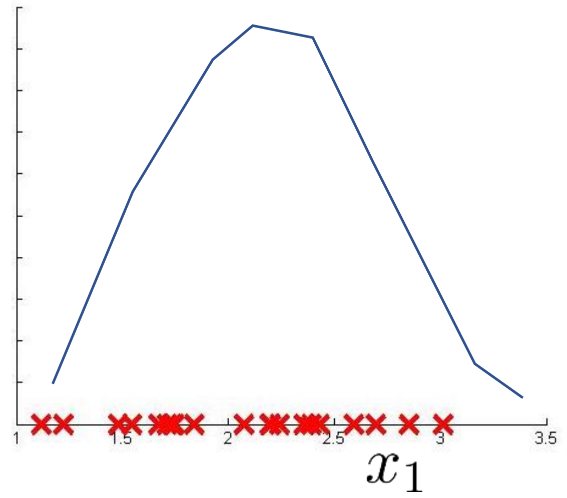
$p(x) < \varepsilon$

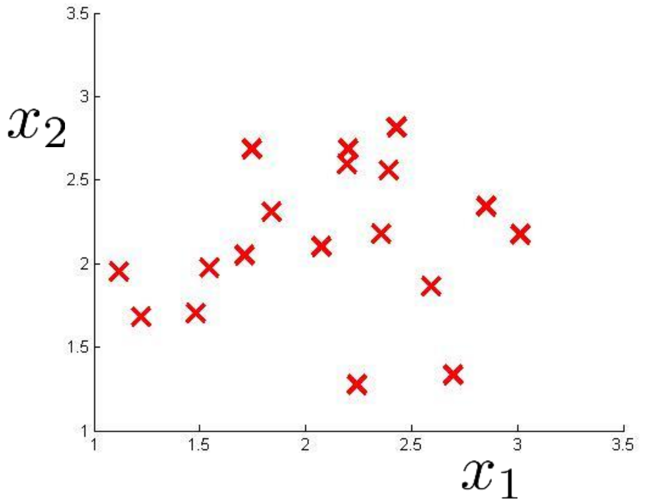
Declare anomaly

Error analysis for anomaly detection

Want $p(x)$ large for normal examples x .
$p(x)$ small for anomalous examples x.

Most common problem:
$p(x)$ is comparable (say, both large) for normal and anomalous examples





Anomaly detection example

Fraud detection:

$x^{(i)}$ = features of user i 's activities

Model $p(x)$ from data.

Identify unusual users by checking which have $p(x) < \varepsilon$

Manufacturing

Monitoring computers in a data center.

$x^{(i)}$ = features of machine i

x_1 = memory use, x_2 = number of disk accesses/sec,

x_3 = CPU load, x_4 = CPU load/network traffic.

…

Gaussian/Normal Distribution

- For a credit card transaction dataset, the fraudulent transactions are an anomaly as the number of fraud cases is very few in comparison to normal transactions in a large dataset.
- The goal is to identify observations that are statistically different from the rest of the observations.
- To achieve this goal, gaussian models are used that fits “k” gaussians to the data.
- Then, we find the distribution parameters i.e., mean and variance for each cluster.
- For each data point, we calculate the probabilities of belonging to each of the clusters.

Mathematical Interpretation

- For one-dimensional model

$$p(x) = \sum_{i=1}^{K} \phi_i \mathcal{N}(x \mid \mu_i, \sigma_i)$$

$$\mathcal{N}(x \mid \mu_i, \sigma_i) = \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left(-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right)$$

- For multi-dimensional model

$$p(\vec{x}) = \sum_{i=1}^{K} \phi_i \mathcal{N}(\vec{x} \mid \vec{\mu}_i, \Sigma_i)$$

$\longrightarrow$ Covariance matrix

- For a given new data point, algorithm finds its distance from every distribution & compute probability of that point belonging to each cluster. If the probability is very low that’s an indication of the data point being an anomaly.

Multivariate Gaussian Distribution

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

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)$$

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

Anomaly detection algorithm for Univariate Gaussian Model

1. Choose features x_i that you think might be indicative of anomalous examples.
2. Fit parameters $\mu_1, \ldots, \mu_n, \sigma_1^2, \ldots, \sigma_n^2$

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

$$\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)} - \mu_j)^2$$

3. Given new example x, compute $p(x)$:

$$p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

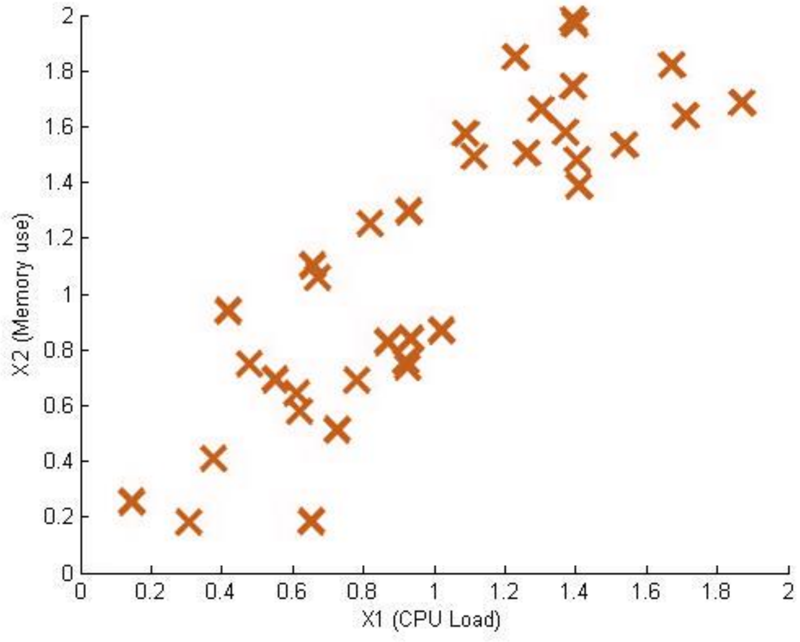
Anomaly if $p(x) < \varepsilon$

Anomaly detection with the multivariate Gaussian

1. Fit model $p(x)$ by setting

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

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



2. Given a new example x, compute

$$p(x) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

Flag an anomaly if $p(x) < \varepsilon$

Original model vs. Multivariate Gaussian

Original model	Multivariate Gaussian
$p(x_1;\mu_1,\sigma_1^2)\times\cdots\times p(x_n;\mu_n,\sigma_n^2)$	$p(x;\mu,\Sigma)=\frac{1}{(2\pi)^{\frac{n}{2}}\lvert\Sigma\rvert^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$
Manually create features to capture anomalies where x_1, x_2 take unusual combinations of values.	Automatically captures correlations between features
Computationally cheaper (alternatively, scales better to large n)	Computationally more expensive
OK even if m (training set size) is small	Must have $m > n$, or else Σ is non-invertible.

Anomaly detection	**vs.**	**Supervised learning**
Very small number of positive examples ($y = 1$). (0-20 is common). Large number of negative ($y = 0$) examples.		Large number of positive and negative examples.
Many different "types" of anomalies. Hard for any algorithm to learn from positive examples what the anomalies look like; future anomalies may look nothing like any of the anomalous examples we've seen so far.		Enough positive examples for algorithm to get a sense of what positive examples are like, future positive examples likely to be similar to ones in training set.

Anomaly detection vs. Supervised learning

Anomaly detection

- Fraud detection
- Manufacturing (e.g. aircraft engines)
- Monitoring machines in a data center

⋮

Supervised learning

- Email spam classification
- Weather prediction (sunny/rainy/etc).
- Cancer classification

⋮