

# CS685: DATA MINING DATA REDUCTION

Arnab Bhattacharya  
arnabb@cse.iitk.ac.in

Computer Science and Engineering,  
Indian Institute of Technology, Kanpur  
<http://web.cse.iitk.ac.in/~cs685/>

1<sup>st</sup> semester, 2018-19  
Mon, Thu 1030-1145 at RM101

# Data Reduction

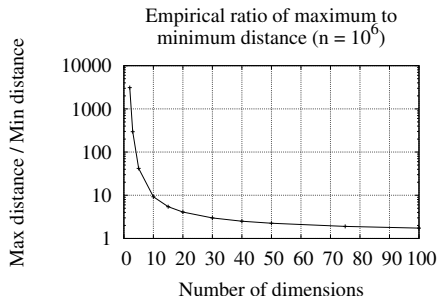
- *Data reduction* decreases data
- Benefits of data reduction

- *Data reduction* decreases data
- Benefits of data reduction
  - Simpler model
    - Less number of rules
    - Less complex rules, i.e., involving less number of attributes
  - Faster algorithms
  - Easier visualization
  - Avoids *overfitting*

- *Data reduction* decreases data
- Benefits of data reduction
  - Simpler model
    - Less number of rules
    - Less complex rules, i.e., involving less number of attributes
  - Faster algorithms
  - Easier visualization
  - Avoids *overfitting*
- Important ways of data reduction
  - Dimensionality reduction
  - Numerosity reduction
  - Data discretization
  - Data modeling
  - Feature selection

# Dimensionality Reduction

- Dimensionality reduction reduces the number of dimensions
- New dimensions are generally different from original ones
- **Curse of dimensionality**
  - Data becomes too sparse as dimensions increase
  - Data is mostly at the boundaries
  - Classification: Not enough data to create good models or methods
  - Clustering: Density becomes irrelevant and distance between points becomes similar



# Singular Value Decomposition (SVD)

- **Singular value decomposition** is factorization of a matrix

$$A = U\Sigma V^T$$

- If  $A$  is of size  $m \times n$ , then  $U$  is  $m \times m$ ,  $V$  is  $n \times n$  and  $\Sigma$  is  $m \times n$
- Columns of  $U$  are *eigenvectors* of  $AA^T$ 
  - **Left singular vectors**
  - $UU^T = I_m$  (orthonormal)
- Columns of  $V$  are *eigenvectors* of  $A^T A$ 
  - **Right singular vectors**
  - $V^T V = I_n$  (orthonormal)
- $\sigma_{ii}$  are the **singular** values
  - $\Sigma$  is *diagonal*
  - Singular values are *positive square roots of eigenvalues* of  $AA^T$  or  $A^T A$
- $\sigma_{11} \geq \sigma_{22} \geq \dots \geq \sigma_{nn}$  (assuming  $n$  singular values)

# Transformation using SVD

- Transformed data

$$T = AV = U\Sigma$$

- $V$  is called **SVD transform matrix**
- Essentially,  $T$  is just a rotation of  $A$
- Dimensionality of  $T$  is  $n$
- $n$  different basis vectors than the original space
- Columns of  $V$  give the basis vectors in rotated space

# Transformation using SVD

- Transformed data

$$T = AV = U\Sigma$$

- $V$  is called **SVD transform matrix**
- Essentially,  $T$  is just a rotation of  $A$
- Dimensionality of  $T$  is  $n$
- $n$  different basis vectors than the original space
- Columns of  $V$  give the basis vectors in rotated space
- $V$  shows how each *object* can be represented as a linear combination of other objects
- $U$  shows how each *dimension* can be represented as a linear combination of other dimensions



# Transformation using SVD

- Transformed data

$$T = AV = U\Sigma$$

- $V$  is called **SVD transform matrix**
- Essentially,  $T$  is just a rotation of  $A$
- Dimensionality of  $T$  is  $n$
- $n$  different basis vectors than the original space
- Columns of  $V$  give the basis vectors in rotated space
- $V$  shows how each *object* can be represented as a linear combination of other objects
- $U$  shows how each *dimension* can be represented as a linear combination of other dimensions
- Lengths of vectors are preserved

$$||\vec{a}_i||_2 = ||\vec{t}_i||_2$$

# SVD of Real Symmetric Matrix

- $A$  is real symmetric of size  $n \times n$
- $A = A^T$
- $U = V$  since  $A^T A = A A^T = A^2$

$$A = Q \Sigma Q^T$$

- $Q$  is of size  $n \times n$  and contains eigenvectors of  $A^2$
- This is called **spectral decomposition** of  $A$
- $\Sigma$  contains  $n$  singular values
- Eigenvectors of  $A =$  eigenvectors of  $A^2$
- Eigenvalues of  $A =$  square root of eigenvalues of  $A^2$
- Eigenvalues of  $A =$  singular values of  $A$

# Example

$$A \begin{bmatrix} 2 & 4 & 1 \\ 1 & 3 & 0 \\ 5 & 2 & 1 \\ 0 & 0 & 7 \\ 3 & 3 & 3 \end{bmatrix} = U \begin{bmatrix} -0.41 & 0.29 & 0.49 & -0.41 & -0.56 \\ -0.23 & 0.27 & 0.48 & 0.77 & 0.18 \\ -0.48 & 0.36 & -0.71 & 0.23 & -0.25 \\ -0.47 & -0.83 & 0.02 & 0.18 & -0.19 \\ -0.55 & 0.05 & 0.01 & -0.37 & 0.73 \end{bmatrix} \\ \times \Sigma \begin{bmatrix} 9.30 & 0 & 0 \\ 0 & 6.47 & 0 \\ 0 & 0 & 2.91 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \times V^T \begin{bmatrix} -0.55 & 0.44 & -0.70 \\ -0.53 & 0.45 & 0.70 \\ -0.63 & -0.77 & 0.01 \end{bmatrix}^T$$

# Transformed Data

$$\begin{aligned} T = AV = U\Sigma &= \begin{bmatrix} 2 & 4 & 1 \\ 1 & 3 & 0 \\ 5 & 2 & 1 \\ 0 & 0 & 7 \\ 3 & 3 & 3 \end{bmatrix} \times \begin{bmatrix} -0.55 & 0.44 & -0.70 \\ -0.53 & 0.45 & 0.70 \\ -0.63 & -0.77 & 0.01 \end{bmatrix} \\ &= \begin{bmatrix} -3.89 & 1.93 & 1.44 \\ -2.16 & 1.80 & 1.42 \\ -4.47 & 2.36 & -2.08 \\ -4.45 & -5.39 & 0.08 \\ -5.18 & 0.38 & 0.05 \end{bmatrix} \\ \text{Lengths} &= [4.58, 3.16, 5.47, 7.00, 5.19] \end{aligned}$$

# Compact Form

$$A \begin{bmatrix} 2 & 4 & 1 \\ 1 & 3 & 0 \\ 5 & 2 & 1 \\ 0 & 0 & 7 \\ 3 & 3 & 3 \end{bmatrix} = U \begin{bmatrix} -0.41 & 0.29 & 0.49 \\ -0.23 & 0.27 & 0.48 \\ -0.48 & 0.36 & -0.71 \\ -0.47 & -0.83 & 0.02 \\ -0.55 & 0.05 & 0.01 \end{bmatrix} \\ \times \Sigma \begin{bmatrix} 9.30 & 0 & 0 \\ 0 & 6.47 & 0 \\ 0 & 0 & 2.91 \end{bmatrix} \times V^T \begin{bmatrix} -0.55 & 0.44 & -0.70 \\ -0.53 & 0.45 & 0.70 \\ -0.63 & -0.77 & 0.01 \end{bmatrix}^T$$

- If  $A$  is of size  $m \times n$ , then  $U$  is  $m \times n$ ,  $V$  is  $n \times n$  and  $\Sigma$  is  $n \times n$
- Works because there at most  $n$  non-zero singular values in  $\Sigma$

# Dimensionality Reduction using SVD

$$A = U\Sigma V^T = \sum_{i=1}^n (u_i \sigma_{ii} v_i^T)$$

- Use only  $k$  dimensions
- Retain first  $k$  columns for  $U$  and  $V$  and first  $k$  values for  $\Sigma$
- First  $k$  columns of  $V$  give the basis vectors in reduced space

$$A_k \approx \sum_{i=1}^k (u_i \sigma_{ii} v_i^T) = U_{1\dots k} \Sigma_{1\dots k} V_{1\dots k}^T$$

$$T_k \approx AV_{1\dots k}$$

# Reduced Dimensionality

$$\begin{aligned} A \approx A_k &= U_k \begin{bmatrix} -0.41 & 0.29 \\ -0.23 & 0.27 \\ -0.48 & 0.36 \\ -0.47 & -0.83 \\ -0.55 & 0.05 \end{bmatrix} \times \Sigma \begin{bmatrix} 9.30 & 0 \\ 0 & 6.47 \end{bmatrix} \times V^T \begin{bmatrix} -0.55 & 0.44 \\ -0.53 & 0.45 \\ -0.63 & -0.77 \end{bmatrix}^T \\ &= \begin{bmatrix} 3.01 & 2.97 & 0.98 \\ 2.00 & 1.98 & -0.01 \\ 3.52 & 3.48 & 1.02 \\ 0.05 & -0.05 & 6.99 \\ 3.03 & 2.96 & 2.99 \end{bmatrix} \\ T \approx T_k &= AV_k = U_k \Sigma_k = \begin{bmatrix} -3.89 & 1.93 \\ -2.16 & 1.80 \\ -4.47 & 2.36 \\ -4.45 & -5.39 \\ -5.18 & 0.38 \end{bmatrix} \end{aligned}$$

Reduced Lengths = [4.34, 2.82, 5.06, 6.99, 5.19]

Length Ratios = [0.95, 0.89, 0.92, 1.00, 1.00]

# Best Approximation

- **Frobenius norm** of a matrix  $C$  of size  $n \times m$  is

$$\|C\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^m C_{ij}^2}$$



# Best Approximation

- **Frobenius norm** of a matrix  $C$  of size  $n \times m$  is

$$\|C\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^m C_{ij}^2}$$

- Consider any rank- $k$  approximation  $A_k$  of  $A$
- SVD produces  $A_k^*$  that minimizes the Frobenius norm of the difference
  - Best in terms of sum squared error

$$A_k^* = \arg \min_{A_k: \text{rank}=k} \|A - A_k\|_F$$

# How Many Dimensions to Retain?

# How Many Dimensions to Retain?

- There is no easy answer

# How Many Dimensions to Retain?

- There is no easy answer
- Concept of **energy** of a dataset
- Total energy is sum of squares of singular values (aka **spread** or variance)

$$E = \sum_{i=1}^n \sigma_{ii}^2$$

- Retain  $k$  dimensions such that  $p\%$  of the energy is retained

$$E_k = \sum_{i=1}^k \sigma_{ii}^2$$

$$E_k/E \geq p$$

# How Many Dimensions to Retain?

- There is no easy answer
- Concept of **energy** of a dataset
- Total energy is sum of squares of singular values (aka **spread** or variance)

$$E = \sum_{i=1}^n \sigma_{ii}^2$$

- Retain  $k$  dimensions such that  $p\%$  of the energy is retained

$$E_k = \sum_{i=1}^k \sigma_{ii}^2$$

$$E_k/E \geq p$$

- Generally,  $p$  is between 80 % to 95 %

# How Many Dimensions to Retain?

- There is no easy answer
- Concept of **energy** of a dataset
- Total energy is sum of squares of singular values (aka **spread** or variance)

$$E = \sum_{i=1}^n \sigma_{ii}^2$$

- Retain  $k$  dimensions such that  $p\%$  of the energy is retained

$$E_k = \sum_{i=1}^k \sigma_{ii}^2$$

$$E_k/E \geq p$$

- Generally,  $p$  is between 80 % to 95 %
- In the above example,  $k = 1$  (resp.  $k = 2$ ) retains 63 % (resp. 94 %) of the energy

# How Many Dimensions to Retain?

- There is no easy answer
- Concept of **energy** of a dataset
- Total energy is sum of squares of singular values (aka **spread** or variance)

$$E = \sum_{i=1}^n \sigma_{ii}^2$$

- Retain  $k$  dimensions such that  $p\%$  of the energy is retained

$$E_k = \sum_{i=1}^k \sigma_{ii}^2$$

$$E_k/E \geq p$$

- Generally,  $p$  is between 80 % to 95 %
- In the above example,  $k = 1$  (resp.  $k = 2$ ) retains 63 % (resp. 94 %) of the energy
- Running time:  $O(m.n.r)$  for  $A$  of size  $m \times n$  and rank  $r$

# Principal Component Analysis (PCA)

- Way of identifying patterns in data
  - How input basis vectors are correlated for the given data
- A transformation from a set of (possibly) correlated axes to another set of uncorrelated axes
- Orthogonal linear transformation (i.e., rotation)
- New axes are **principal components**
- First principal component produces projections that are best in the squared error sense
- Optimal least squares solution



# Algorithm

- Mean center the data (optional)
- Compute the *covariance matrix* of the dimensions
- Find eigenvectors of covariance matrix
- Sort eigenvectors in decreasing order of eigenvalues
- Project onto eigenvectors in order

# Algorithm

- Mean center the data (optional)
- Compute the *covariance matrix* of the dimensions
- Find eigenvectors of covariance matrix
- Sort eigenvectors in decreasing order of eigenvalues
- Project onto eigenvectors in order
- Assume data matrix is  $B$  of size  $m \times n$
- For each dimension, compute mean  $\mu_i$
- Mean center  $B$  by subtracting  $\mu_i$  from each column  $i$  to get  $A$
- Compute covariance matrix  $C$  of size  $n \times n$ 
  - If mean centered,  $C = A^T A$
- Find eigenvectors and corresponding eigenvalues  $(V, E)$  of  $C$
- Sort eigenvalues such that  $e_1 \geq e_2 \geq \dots \geq e_n$
- Project step-by-step onto the principal components  $\vec{v}_1, \vec{v}_2, \dots$ , etc.

## Example

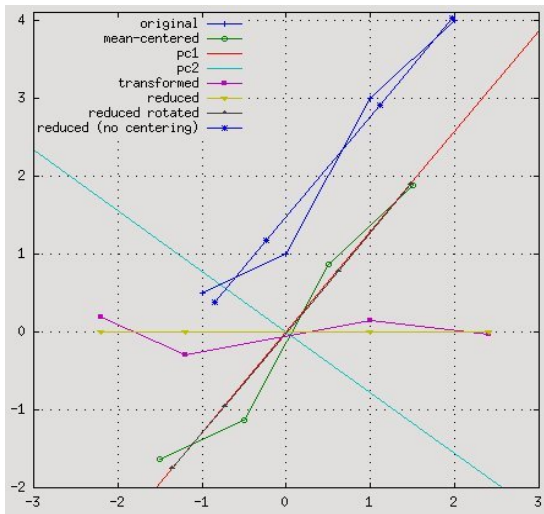
$$B = \begin{bmatrix} 2 & 4 \\ 1 & 3 \\ 0 & 1 \\ -1 & 0.5 \end{bmatrix}; \mu(B) = \begin{bmatrix} 0.500 & 2.125 \end{bmatrix}$$

$$\therefore A = \begin{bmatrix} 1.5 & 1.875 \\ 0.5 & 0.875 \\ -0.5 & -1.125 \\ -1.5 & -1.625 \end{bmatrix} \text{ and, } C = A^T A = \begin{bmatrix} 5.000 & 6.250 \\ 6.250 & 8.187 \end{bmatrix}$$

$$\text{Eigenvectors } V = \begin{bmatrix} 0.613 & -0.789 \\ 0.789 & 0.613 \end{bmatrix}; \text{ eigenvalues } E = \begin{bmatrix} 13.043 \\ 0.143 \end{bmatrix}$$

$$\text{Transformed data } T = AV = \begin{bmatrix} 2.400 & -0.034 \\ 0.997 & 0.142 \\ -1.195 & -0.295 \\ -2.203 & 0.187 \end{bmatrix}$$

# Visual Example



# Properties

- Also known as Karhunen-Loève transform (KLT)

# Properties

- Also known as **Karhunen-Loève transform (KLT)**
- Works for  $L_2$  distances only as others are not invariant to rotation

# Properties

- Also known as **Karhunen-Loève transform (KLT)**
- Works for  $L_2$  distances only as others are not invariant to rotation
- Mean-centering
  - Easier way to compute covariance:  $A^T A$  is covariance matrix
  - Allows use of SVD to compute PCA

# Properties

- Also known as **Karhunen-Loève transform (KLT)**
- Works for  $L_2$  distances only as others are not invariant to rotation
- Mean-centering
  - Easier way to compute covariance:  $A^T A$  is covariance matrix
  - Allows use of SVD to compute PCA
- Can be done using SVD
  - Eigenvector matrix  $V$  of  $C$  is really the SVD transform matrix  $V$  for  $A$
  - Different from  $SVD$  of  $B$  though



# Properties

- Also known as **Karhunen-Loève transform (KLT)**
- Works for  $L_2$  distances only as others are not invariant to rotation
- Mean-centering
  - Easier way to compute covariance:  $A^T A$  is covariance matrix
  - Allows use of SVD to compute PCA
- Can be done using SVD
  - Eigenvector matrix  $V$  of  $C$  is really the SVD transform matrix  $V$  for  $A$
  - Different from SVD of  $B$  though
- How many dimensions to retain?
  - Based on energy (similar to SVD)
  - Total energy is sum of eigenvalues  $e_i$
  - Retain  $k$  dimensions such that 80 – 95 % of the energy is retained

- Also known as **Karhunen-Loève transform (KLT)**
- Works for  $L_2$  distances only as others are not invariant to rotation
- Mean-centering
  - Easier way to compute covariance:  $A^T A$  is covariance matrix
  - Allows use of SVD to compute PCA
- Can be done using SVD
  - Eigenvector matrix  $V$  of  $C$  is really the SVD transform matrix  $V$  for  $A$
  - Different from SVD of  $B$  though
- How many dimensions to retain?
  - Based on energy (similar to SVD)
  - Total energy is sum of eigenvalues  $e_i$
  - Retain  $k$  dimensions such that 80 – 95 % of the energy is retained
  - In the above example,  $k = 1$  retains 98.91 % of the energy

- Also known as **Karhunen-Loève transform (KLT)**
- Works for  $L_2$  distances only as others are not invariant to rotation
- Mean-centering
  - Easier way to compute covariance:  $A^T A$  is covariance matrix
  - Allows use of SVD to compute PCA
- Can be done using SVD
  - Eigenvector matrix  $V$  of  $C$  is really the SVD transform matrix  $V$  for  $A$
  - Different from SVD of  $B$  though
- How many dimensions to retain?
  - Based on energy (similar to SVD)
  - Total energy is sum of eigenvalues  $e_i$
  - Retain  $k$  dimensions such that 80 – 95 % of the energy is retained
  - In the above example,  $k = 1$  retains 98.91 % of the energy
- Running time:  $O(mn^2 + n^3)$  for  $A$  of size  $m \times n$

# Numerosity Reduction

- Numerosity reduction reduces the *volume* of data

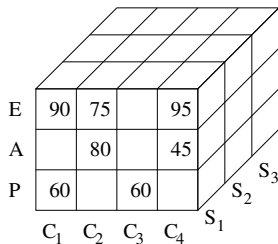
# Numerosity Reduction

- Numerosity reduction reduces the *volume* of data
- Reduction in number of data objects
- Compression
- Modeling
- Discretization

# Aggregation

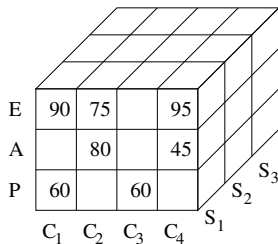
- Considers a set of data objects having some similar attribute(s)
- **Aggregates** some other attribute(s) into single value(s)
- Example: sum, average
- Benefits of aggregation
  - Aggregate value has less variability
  - Absorbs individual errors
  - Reduces noise

# Data Cube



- For multi-dimensional datasets, aggregation can happen along different dimensions
- Data cubes are essentially multi-dimensional arrays
- Each cell or face or lower dimensional surface represents a certain *projection* operation

# Data Cube



- For multi-dimensional datasets, aggregation can happen along different dimensions
- Data cubes are essentially multi-dimensional arrays
- Each cell or face or lower dimensional surface represents a certain *projection* operation
- Aggregation can also happen along different resolutions in each dimension



# Sampling

- A **sample** is a representative if it has (approximately) the same property of interest as the full dataset
- Sampling approaches

# Sampling

- A **sample** is a representative if it has (approximately) the same property of interest as the full dataset
- Sampling approaches
- Sampling **without replacement** (SRSWOR): produces **population**
- Sampling **with replacement** (SRSWR): can be picked more than once

# Sampling

- A **sample** is a representative if it has (approximately) the same property of interest as the full dataset
- Sampling approaches
- Sampling **without replacement** (SRSWOR): produces **population**
- Sampling **with replacement** (SRSWR): can be picked more than once
- Simple *random sampling*
- For different types of objects, **stratified sampling**
  - Picks equal or representative number of objects from each group

# Sampling

- A **sample** is a representative if it has (approximately) the same property of interest as the full dataset
- Sampling approaches
- Sampling **without replacement** (SRSWOR): produces **population**
- Sampling **with replacement** (SRSWR): can be picked more than once
- Simple *random sampling*
- For different types of objects, **stratified sampling**
  - Picks equal or representative number of objects from each group
- Sample size
  - Sample should have enough data to capture variability

# Sampling

- A **sample** is a representative if it has (approximately) the same property of interest as the full dataset
- Sampling approaches
- Sampling **without replacement** (SRSWOR): produces **population**
- Sampling **with replacement** (SRSWR): can be picked more than once
- Simple *random sampling*
- For different types of objects, **stratified sampling**
  - Picks equal or representative number of objects from each group
- Sample size
  - Sample should have enough data to capture variability
- **Progressive sampling** or **adaptive sampling**
  - Start with a small sample size
  - Keep on increasing till it is acceptable

# Histograms

- Method of discretizing the data
- Mostly useful for one-dimensional data
- *Equi-width histograms*: bins are equally spaced apart
- *Equi-height (equi-depth) histograms*: each bin has the same height

# Histograms

- Method of discretizing the data
- Mostly useful for one-dimensional data
- *Equi-width histograms*: bins are equally spaced apart
- *Equi-height (equi-depth) histograms*: each bin has the same height
- **MaxDiff histograms**
  - Values are first sorted
  - To get  $b$  bins, the largest  $b - 1$  differences are made bin boundaries

# Data Summarization

- **Central tendency** measures
  - Mean: may be weighted
  - Median: “middle” value
  - Mode: dataset may be *unimodal* or *multimodal*
  - Midrange:



# Data Summarization

- **Central tendency** measures
  - Mean: may be weighted
  - Median: “middle” value
  - Mode: dataset may be *unimodal* or *multimodal*
  - Midrange: average of largest and smallest value

# Data Summarization

- **Central tendency** measures
  - Mean: may be weighted
  - Median: “middle” value
  - Mode: dataset may be *unimodal* or *multimodal*
  - Midrange: average of largest and smallest value
- **Dispersion** measures

# Data Summarization

- **Central tendency** measures
  - Mean: may be weighted
  - Median: “middle” value
  - Mode: dataset may be *unimodal* or *multimodal*
  - Midrange: average of largest and smallest value
- **Dispersion** measures
  - Variance
  - Standard deviation
  - Range
  - *Percentile (quartile)*
  - **Five-number summary:**

# Data Summarization

- **Central tendency** measures
  - Mean: may be weighted
  - Median: “middle” value
  - Mode: dataset may be *unimodal* or *multimodal*
  - Midrange: average of largest and smallest value
- **Dispersion** measures
  - Variance
  - Standard deviation
  - Range
  - *Percentile (quartile)*
  - **Five-number summary**: minimum, first quartile, median, third quartile, maximum
  - **Box plot**:

# Data Summarization

- **Central tendency** measures
  - Mean: may be weighted
  - Median: “middle” value
  - Mode: dataset may be *unimodal* or *multimodal*
  - Midrange: average of largest and smallest value
- **Dispersion** measures
  - Variance
  - Standard deviation
  - Range
  - *Percentile (quartile)*
  - **Five-number summary**: minimum, first quartile, median, third quartile, maximum
  - **Box plot**: plot of five values

# Data Summarization (contd.)

- **Distributive** measures

# Data Summarization (contd.)

- **Distributive** measures
  - Can be computed by partitioning the dataset

# Data Summarization (contd.)

- **Distributive** measures
  - Can be computed by partitioning the dataset
  - Example: mean
- **Holistic** measures



# Data Summarization (contd.)

- **Distributive** measures
  - Can be computed by partitioning the dataset
  - Example: mean
- **Holistic** measures
  - Cannot be computed by partitioning the dataset

# Data Summarization (contd.)

- **Distributive** measures
  - Can be computed by partitioning the dataset
  - Example: mean
- **Holistic** measures
  - Cannot be computed by partitioning the dataset
  - Example: median

# Data Summarization (contd.)

- **Distributive** measures
  - Can be computed by partitioning the dataset
  - Example: mean
- **Holistic** measures
  - Cannot be computed by partitioning the dataset
  - Example: median
- Graphical measures help in **data visualization**

- Histogram:

# Data Visualization

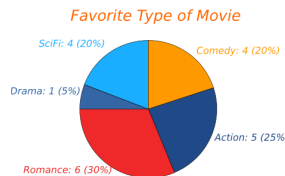
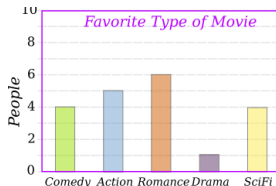
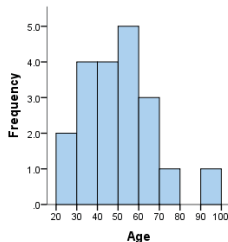
- **Histogram**: frequency versus grouped values
- **Bar chart**:

# Data Visualization

- **Histogram**: frequency versus grouped values
- **Bar chart**: histograms where bins are categorical
- **Pie chart**:

# Data Visualization

- **Histogram**: frequency versus grouped values
- **Bar chart**: histograms where bins are categorical
- **Pie chart**: relative frequencies shown as sectors in a circle



# Data Plots

- Quantile plot:

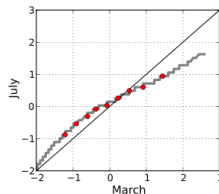
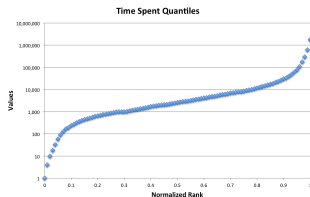


# Data Plots

- **Quantile plot:** quantiles against value
- **Quantile-quantile plot (q-q plot):**

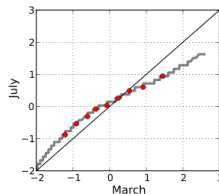
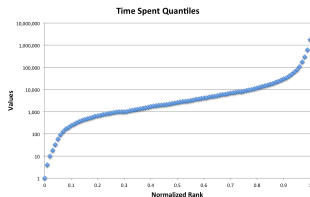
# Data Plots

- **Quantile plot:** quantiles against value
- **Quantile-quantile plot (q-q plot):** quantiles against quantiles



# Data Plots

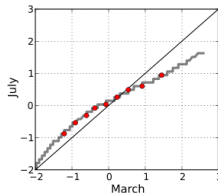
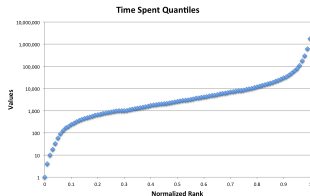
- **Quantile plot:** quantiles against value
- **Quantile-quantile plot (q-q plot):** quantiles against quantiles



- **Scatter plot:**

# Data Plots

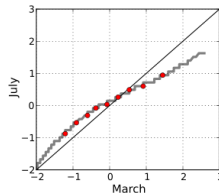
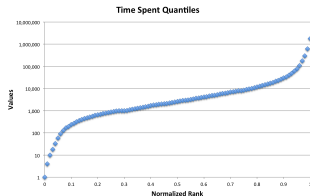
- **Quantile plot:** quantiles against value
- **Quantile-quantile plot (q-q plot):** quantiles against quantiles



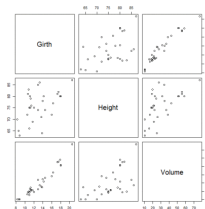
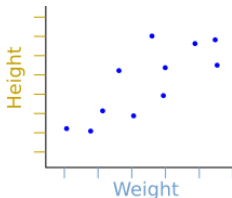
- **Scatter plot:** values of one variable against another
- **Scatter plot matrix:**

# Data Plots

- **Quantile plot:** quantiles against value
- **Quantile-quantile plot (q-q plot):** quantiles against quantiles



- **Scatter plot:** values of one variable against another
- **Scatter plot matrix:**  $n^2$  scatter plots for  $n$  variables



# Regression

- Aim is to describe how the **response** variable  $Y$  is generated based on a set of  $k$  **predictors**, denoted by  $X$
- Produces a model of  $Y$

# Regression

- Aim is to describe how the **response** variable  $Y$  is generated based on a set of  $k$  **predictors**, denoted by  $X$
- Produces a model of  $Y$
- Can predict missing values
- The general form of regression model is

$$Y = f(X) + \varepsilon$$

- The function  $f$  can be chosen using domain knowledge
- For *linear regression*,  $f$  is linear
- $\varepsilon$  encodes the *error* (includes *noise*) terms associated with each observation

# Linear Regression

- The function  $f$  is chosen to be linear
- The response variable depends only linearly with the predictor variables
- The general form of linear regression model is

$$Y = XW + \varepsilon$$

- $W$  are the **regression coefficients** or *weights* on the predictors
- Sizes of matrices



# Linear Regression

- The function  $f$  is chosen to be linear
- The response variable depends only linearly with the predictor variables
- The general form of linear regression model is

$$Y = XW + \varepsilon$$

- $W$  are the **regression coefficients** or *weights* on the predictors
- Sizes of matrices
  - $Y : n \times 1$
  - $X : n \times k$
  - $W : k \times 1$
  - $\varepsilon : n \times 1$

# Linear Regression

- The function  $f$  is chosen to be linear
- The response variable depends only linearly with the predictor variables
- The general form of linear regression model is

$$Y = XW + \varepsilon$$

- $W$  are the **regression coefficients** or *weights* on the predictors
- Sizes of matrices
  - $Y : n \times 1$
  - $X : n \times k$
  - $W : k \times 1$
  - $\varepsilon : n \times 1$
- $W$  can be solved through *least squares* method

# Feature Selection

- **Feature selection** or **attribute selection** is different from dimensionality reduction, as it retains original attributes
- Aims to remove redundant and irrelevant features

# Feature Selection

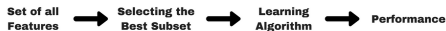
- **Feature selection** or **attribute selection** is different from dimensionality reduction, as it retains original attributes
- Aims to remove redundant and irrelevant features
- *Feature weighting*: Variant of feature selection
  - SVM does it naturally

# Feature Selection Methods

- Three main approaches

# Feature Selection Methods

- Three main approaches
- **Filter**
  - Features chosen before the algorithm
  - These use criteria that are independent of the algorithm
  - Entropy, correlation. etc.

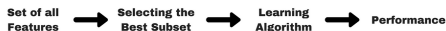


# Feature Selection Methods

- Three main approaches

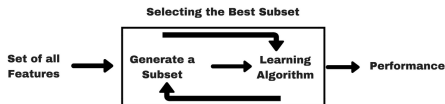
- **Filter**

- Features chosen before the algorithm
- These use criteria that are independent of the algorithm
- Entropy, correlation. etc.



- **Wrapper**

- Feature selection is targeted to the algorithm that is used
- Training error is reduced

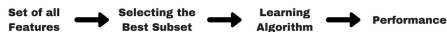


# Feature Selection Methods

- Three main approaches

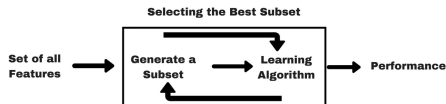
- **Filter**

- Features chosen before the algorithm
- These use criteria that are independent of the algorithm
- Entropy, correlation. etc.



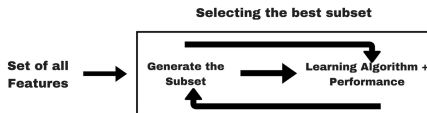
- **Wrapper**

- Feature selection is targeted to the algorithm that is used
- Training error is reduced



- **Embedded**

- Algorithm has built-in feature selection strategy
- Example: decision trees. LASSO regression





# General Strategy

- Total number of possibilities is

# General Strategy

- Total number of possibilities is  $2^n - 1$  ( $-1$ )
- Some feature selection algorithms do it using a single shot
  - Example: **Relief** algorithm

# General Strategy

- Total number of possibilities is  $2^n - 1$  ( $-1$ )
- Some feature selection algorithms do it using a single shot
  - Example: **Relief** algorithm
- Otherwise, systematically add a feature to the empty set
- Or, systematically delete a feature from the full set

# General Strategy

- Total number of possibilities is  $2^n - 1$  ( $-1$ )
- Some feature selection algorithms do it using a single shot
  - Example: **Relief** algorithm
- Otherwise, systematically add a feature to the empty set
- Or, systematically delete a feature from the full set
- Greedy approach

# General Strategy

- Total number of possibilities is  $2^n - 1$  ( $-1$ )
- Some feature selection algorithms do it using a single shot
  - Example: **Relief** algorithm
- Otherwise, systematically add a feature to the empty set
- Or, systematically delete a feature from the full set
- Greedy approach
- Define some error criterion
  - Example: **mutual information**  $I(X, Y) = KL(p(x, y) | p(x)p(y))$

# General Strategy

- Total number of possibilities is  $2^n - 1$  ( $-1$ )
- Some feature selection algorithms do it using a single shot
  - Example: **Relief** algorithm
- Otherwise, systematically add a feature to the empty set
- Or, systematically delete a feature from the full set
- Greedy approach
- Define some error criterion
  - Example: **mutual information**  $I(X, Y) = KL(p(x, y) | p(x)p(y))$
- Start from empty set of features
- Add the feature that decreases the error the most
- Stop when there is no significant decrease or there is an increase

# General Strategy

- Total number of possibilities is  $2^n - 1$  ( $-1$ )
- Some feature selection algorithms do it using a single shot
  - Example: **Relief** algorithm
- Otherwise, systematically add a feature to the empty set
- Or, systematically delete a feature from the full set
- Greedy approach
- Define some error criterion
  - Example: **mutual information**  $I(X, Y) = KL(p(x, y) | p(x)p(y))$
- Start from empty set of features
- Add the feature that decreases the error the most
- Stop when there is no significant decrease or there is an increase
- Start from full set of features
- Remove the feature that increases the error the most
- Stop when there is no significant increase or there is a decrease

# Relief Algorithm

- Computes a *feature score* for every feature
- Features with low score can be thresholded away
- They may also be used as feature weights



# Relief Algorithm

- Computes a *feature score* for every feature
- Features with low score can be thresholded away
- They may also be used as feature weights
- Feature score is based on similarity with the *nearest neighbor*

# Relief Algorithm

- Computes a *feature score* for every feature
- Features with low score can be thresholded away
- They may also be used as feature weights
- Feature score is based on similarity with the *nearest neighbor*
- Each attribute range is normalised to  $[0, 1]$

# Relief Algorithm

- Computes a *feature score* for every feature
- Features with low score can be thresholded away
- They may also be used as feature weights
- Feature score is based on similarity with the *nearest neighbor*
- Each attribute range is normalised to  $[0, 1]$
- For feature  $i$ , initially the score is  $s_i = 0$
- Pick a random object
- Find its *nearest hit* from the same class and *nearest miss* from the other class

# Relief Algorithm

- Computes a *feature score* for every feature
- Features with low score can be thresholded away
- They may also be used as feature weights
- Feature score is based on similarity with the *nearest neighbor*
- Each attribute range is normalised to  $[0, 1]$
- For feature  $i$ , initially the score is  $s_i = 0$
- Pick a random object
- Find its *nearest hit* from the same class and *nearest miss* from the other class
- Update the score of feature  $i$  as

$$s_i = s_i - \text{diff}_i(\text{nearest-hit}) + \text{diff}_i(\text{nearest-miss})$$

- Difference function is Euclidean ( $L_1$  can also be used)

# Relief Algorithm

- Computes a *feature score* for every feature
- Features with low score can be thresholded away
- They may also be used as feature weights
- Feature score is based on similarity with the *nearest neighbor*
- Each attribute range is normalised to  $[0, 1]$
- For feature  $i$ , initially the score is  $s_i = 0$
- Pick a random object
- Find its *nearest hit* from the same class and *nearest miss* from the other class
- Update the score of feature  $i$  as

$$s_i = s_i - \text{diff}_i(\text{nearest-hit}) + \text{diff}_i(\text{nearest-miss})$$

- Difference function is Euclidean ( $L_1$  can also be used)
- Done for  $m$  random objects

# Relief Algorithm

- Computes a *feature score* for every feature
- Features with low score can be thresholded away
- They may also be used as feature weights
- Feature score is based on similarity with the *nearest neighbor*
- Each attribute range is normalised to  $[0, 1]$
- For feature  $i$ , initially the score is  $s_i = 0$
- Pick a random object
- Find its *nearest hit* from the same class and *nearest miss* from the other class
- Update the score of feature  $i$  as

$$s_i = s_i - \text{diff}_i(\text{nearest-hit}) + \text{diff}_i(\text{nearest-miss})$$

- Difference function is Euclidean ( $L_1$  can also be used)
- Done for  $m$  random objects
- All features with score less than threshold  $\tau$  are removed

# Variants of Relief Algorithm

- $k$  nearest neighbors

# Variants of Relief Algorithm

- $k$  nearest neighbors
- If nearest neighbor has a missing value for the attribute, difference



# Variants of Relief Algorithm

- $k$  nearest neighbors
- If nearest neighbor has a missing value for the attribute, difference
  - can be ignored

# Variants of Relief Algorithm

- $k$  nearest neighbors
- If nearest neighbor has a missing value for the attribute, difference
  - can be ignored
  - is set to  $1 - 1/\text{\#values-of-attribute}$

# Variants of Relief Algorithm

- $k$  nearest neighbors
- If nearest neighbor has a missing value for the attribute, difference
  - can be ignored
  - is set to  $1 - 1/\# \text{values-of-attribute}$
  - is estimated as  $1 - P(\text{value}|\text{class}(\text{nearest-neighbor}))$

# Variants of Relief Algorithm

- $k$  nearest neighbors
- If nearest neighbor has a missing value for the attribute, difference
  - can be ignored
  - is set to  $1 - 1/\# \text{values-of-attribute}$
  - is estimated as  $1 - P(\text{value}|\text{class}(\text{nearest-neighbor}))$
- For multiple classes

# Variants of Relief Algorithm

- $k$  nearest neighbors
- If nearest neighbor has a missing value for the attribute, difference
  - can be ignored
  - is set to  $1 - 1/\# \text{values-of-attribute}$
  - is estimated as  $1 - P(\text{value}|\text{class}(\text{nearest-neighbor}))$
- For multiple classes
  - Nearest miss can be taken from any other class

# Variants of Relief Algorithm

- $k$  nearest neighbors
- If nearest neighbor has a missing value for the attribute, difference
  - can be ignored
  - is set to  $1 - 1/\# \text{values-of-attribute}$
  - is estimated as  $1 - P(\text{value}|\text{class}(\text{nearest-neighbor}))$
- For multiple classes
  - Nearest miss can be taken from any other class
  - Nearest miss is an average of nearest misses from every other class