

Data Preprocessing

Agenda

- Why data preprocessing?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Why Data Preprocessing?

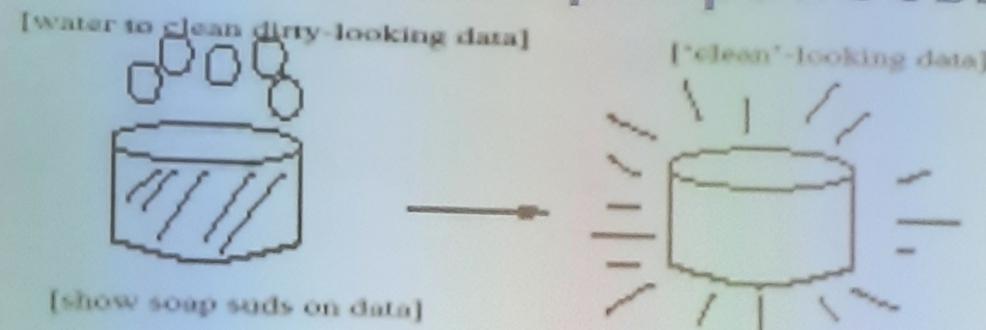
- Data in the real world is dirty
 - incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
 - noisy: containing errors or outliers
 - inconsistent: containing discrepancies in codes or names
- No quality data, no quality mining results!
 - Quality decisions must be based on quality data
 - Data warehouse needs consistent integration of quality data
- A multi-dimensional measure of data quality:
 - A well-accepted multi-dimensional view:
 - accuracy, completeness, consistency, timeliness, believability, value added, interpretability, accessibility
 - Broad categories:
 - intrinsic, contextual, representational, and accessibility.

Major Tasks in Data Preprocessing

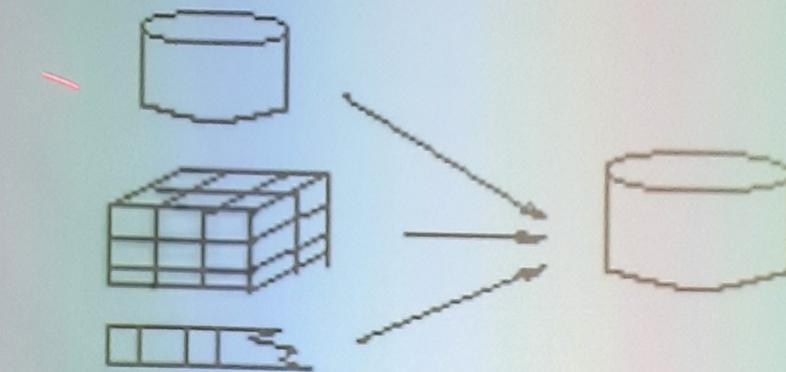
- Data cleaning
 - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
- Data integration
 - Integration of multiple databases, data cubes, files, or notes
- Data transformation
 - Normalization (scaling to a specific range)
 - Aggregation
- Data reduction
 - Obtains reduced representation in volume but produces the same or similar analytical results
 - Data discretization: with particular importance, especially for numerical data
 - Data aggregation, dimensionality reduction, data compression, generalization

Forms of data preprocessing

Data Cleaning



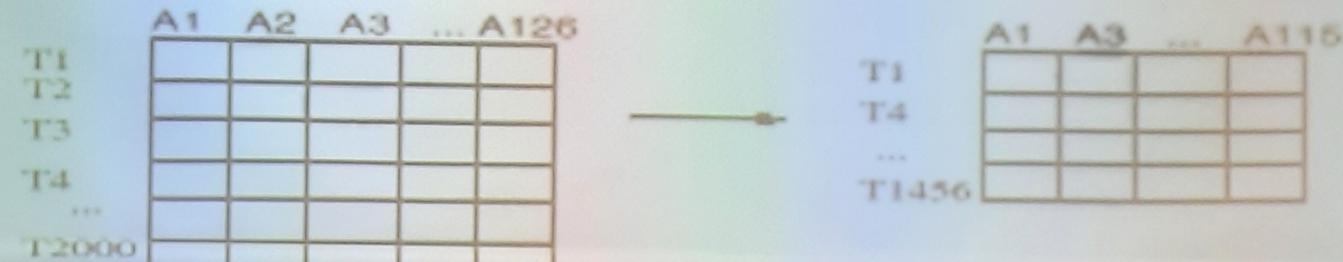
Data Integration



Data Transformation

-2, 32, 100, 59, 48 → -0.02, 0.32, 1.00, 0.59, 0.48

Data Reduction



Missing Data

- Data is not always available
 - E.g., many tuples have no recorded value for several attributes, such as customer income in sales data
- Missing data may be due to
 - equipment malfunction
 - inconsistent with other recorded data and thus deleted
 - data not entered due to misunderstanding
 - certain data may not be considered important at the time of entry
 - not register history or changes of the data
- Missing data may need to be inferred

How to Handle Missing Data?

- Ignore the tuple: usually done when class label is missing (assuming the task is classification—not effective in certain cases)
- Fill in the missing value manually: tedious + infeasible?
- Use a global constant to fill in the missing value: e.g., “unknown”, a new class?!
- Use the attribute mean to fill in the missing value
- Use the attribute mean for all samples of the same class to fill in the missing value: smarter
- Use the most probable value to fill in the missing value: inference-based such as regression, Bayesian formula, decision tree

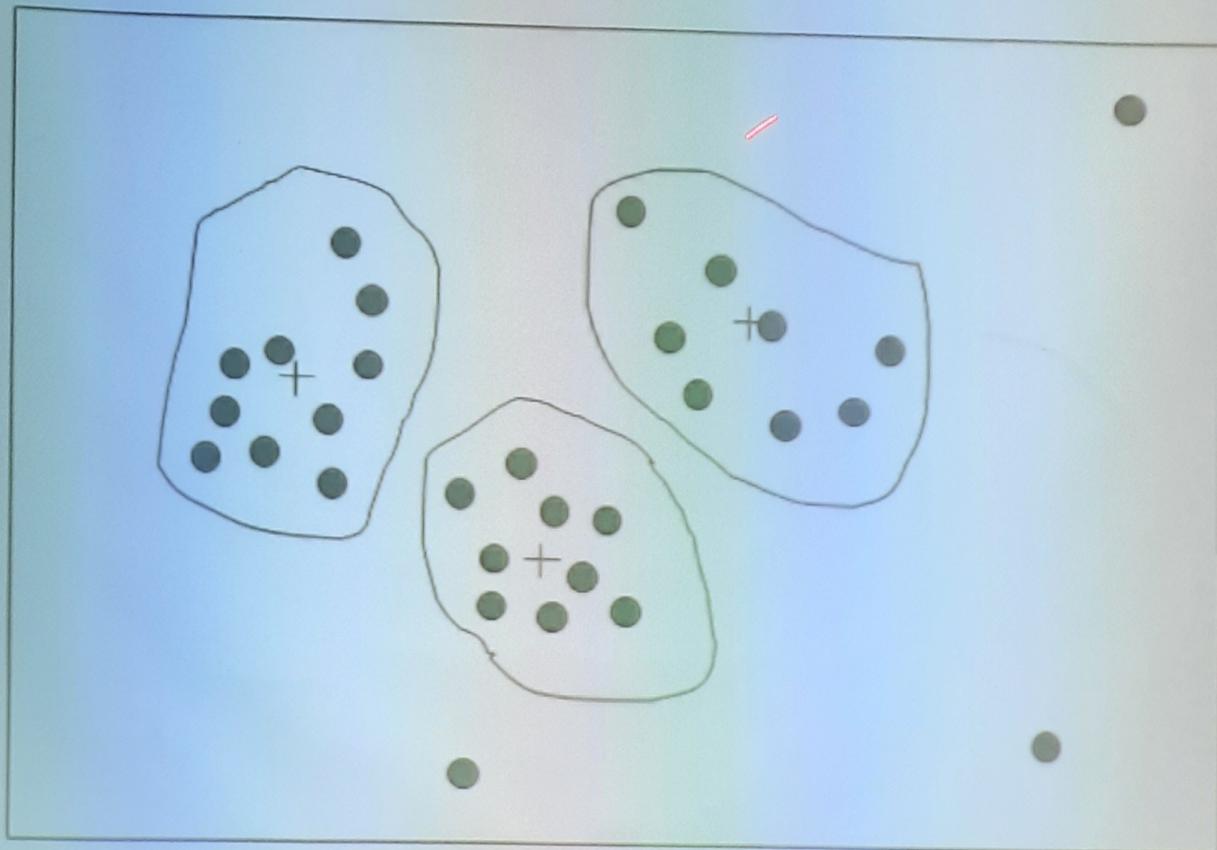
Noisy Data

- Q: What is noise?
- A: Random error in a measured variable.
- Incorrect attribute values may be due to
 - faulty data collection instruments
 - data entry problems
 - data transmission problems
 - technology limitation
 - inconsistency in naming convention
- Other data problems which requires data cleaning
 - duplicate records
 - incomplete data
 - inconsistent data

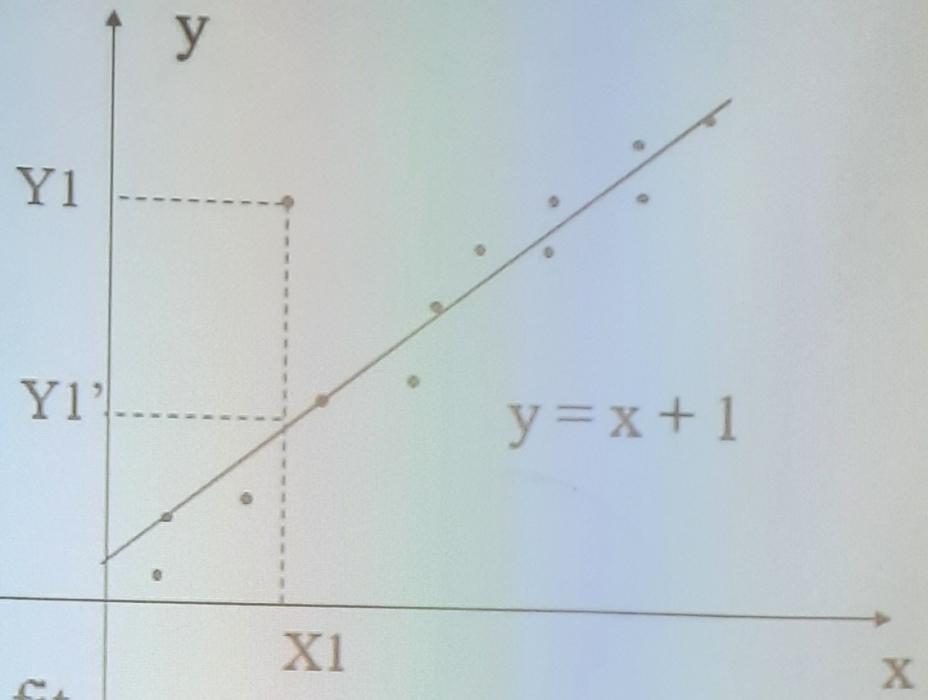
How to Handle Noisy Data?

- Binning method:
 - first sort data and partition into (equi-depth) bins
 - then one can smooth by bin means, smooth by bin median, smooth by bin boundaries, etc.
 - used also for discretization (discussed later)
- Clustering
 - detect and remove outliers
- Semi-automated method: combined computer and human inspection
 - detect suspicious values and check manually
- Regression
 - smooth by fitting the data into regression functions

Cluster Analysis



Regression



- Linear regression (best line to fit two variables)
- Multiple linear regression (more than two variables, fit to a multidimensional surface)

How to Handle Inconsistent Data?

- Manual correction using external references
- Semi-automatic using various tools
 - To detect violation of known functional dependencies and data constraints
 - To correct redundant data

Data Integration

- Data integration:
 - combines data from multiple sources into a coherent store
- Schema integration
 - integrate metadata from different sources
 - Entity identification problem: identify real world entities from multiple data sources, e.g., A.cust-id \equiv B.cust-#
- Detecting and resolving data value conflicts
 - for the same real world entity, attribute values from different sources are different
 - possible reasons: different representations, different scales, e.g., metric vs. British units, different currency

Handling Redundant Data in Data Integration

- Redundant data occur often when integrating multiple DBs
 - The same attribute may have different names in different databases
 - One attribute may be a “derived” attribute in another table, e.g., annual revenue
- Redundant data may be able to be detected by correlational analysis

$$r_{A,B} = \frac{\sum(A - \bar{A})(B - \bar{B})}{(n-1)\sigma_A\sigma_B}$$

- Careful integration can help reduce/avoid redundancies and inconsistencies and improve mining speed and quality

Data Transformation

- Smoothing: remove noise from data (binning, clustering, regression)
- Aggregation: summarization, data cube construction
- Generalization: concept hierarchy climbing
- Normalization: scaled to fall within a small, specified range
 - min-max normalization
 - z-score normalization
 - normalization by decimal scaling
- Attribute/feature construction
 - New attributes constructed from the given ones

Data Transformation: Normalization

Particularly useful for classification (NNs, distance measurements, nn classification, etc)

- min-max normalization

$$v' = \frac{v - \text{min}_A}{\text{max}_A - \text{min}_A} (\text{new_max}_A - \text{new_min}_A) + \text{new_min}_A$$

- z-score normalization

$$v' = \frac{v - \text{mean}_A}{\text{stand_dev}_A}$$

- normalization by decimal scaling

$$v' = \frac{v}{10^j} \quad \text{Where } j \text{ is the smallest integer such that } \text{Max}(|v'|) < 1$$

Agenda

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Data Reduction

- Problem:
Data Warehouse may store terabytes of data:
Complex data analysis/mining may take a very
long time to run on the complete data set
- Solution?
 - Data reduction...

Data Reduction

- Obtains a reduced representation of the data set that is much smaller in volume but yet produces the same (or almost the same) analytical results
- Data reduction strategies
 - Data cube aggregation
 - Dimensionality reduction
 - Data compression
 - Numerosity reduction
 - Discretization and concept hierarchy generation

Data Cube Aggregation

- Multiple levels of aggregation in data cubes
 - Further reduce the size of data to deal with
- Reference appropriate levels
 - Use the smallest representation capable to solve the task
- Queries regarding aggregated information should be answered using data cube, when possible

Dimensionality Reduction

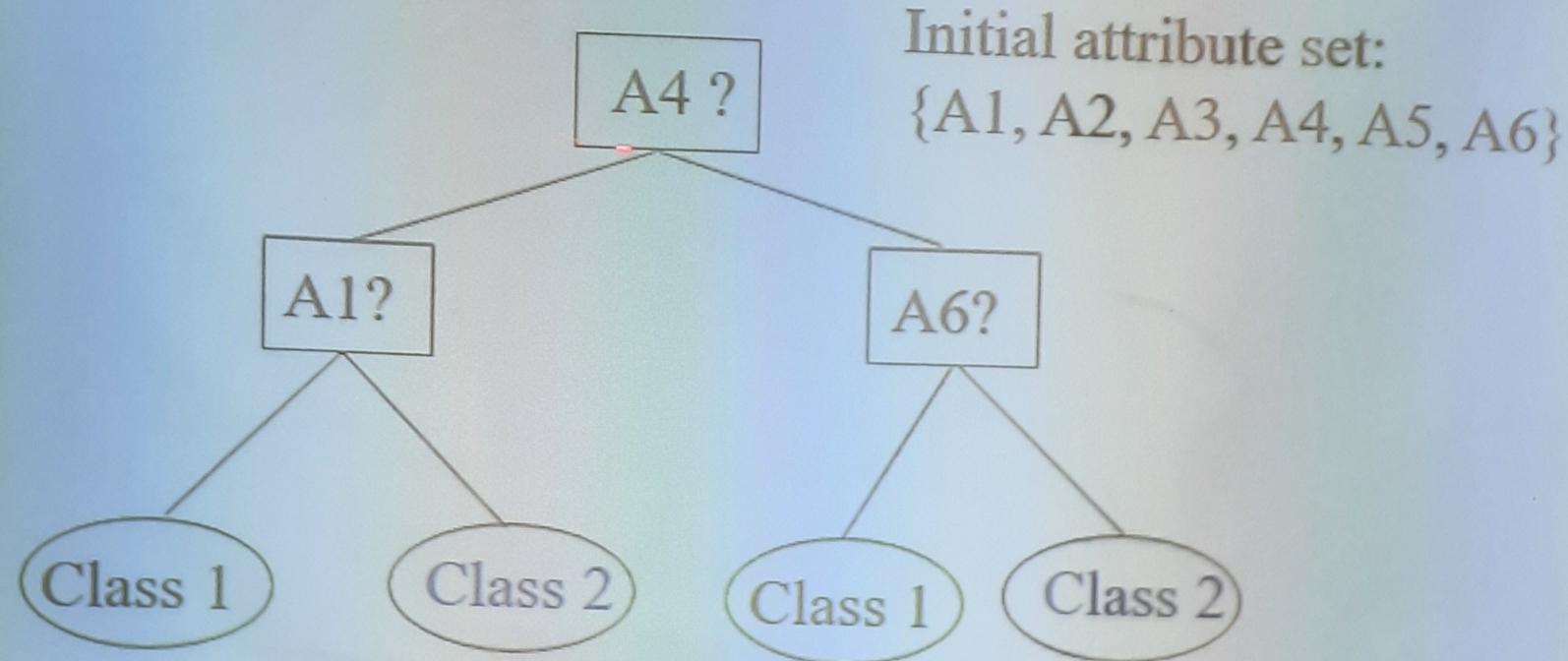
- Problem: Feature selection (i.e., attribute subset selection):
 - Select a minimum set of features such that the probability distribution of different classes given the values for those features is as close as possible to the original distribution given the values of all features
 - Nice side-effect: reduces # of attributes in the discovered patterns (which are now easier to understand)
- Solution: Heuristic methods (due to exponential # of choices) usually greedy:
 - step-wise forward selection
 - step-wise backward elimination
 - combining forward selection and backward elimination
 - decision-tree induction

Example of Decision Tree Induction

nonleaf nodes: tests

branches: outcomes of tests

leaf nodes: class prediction

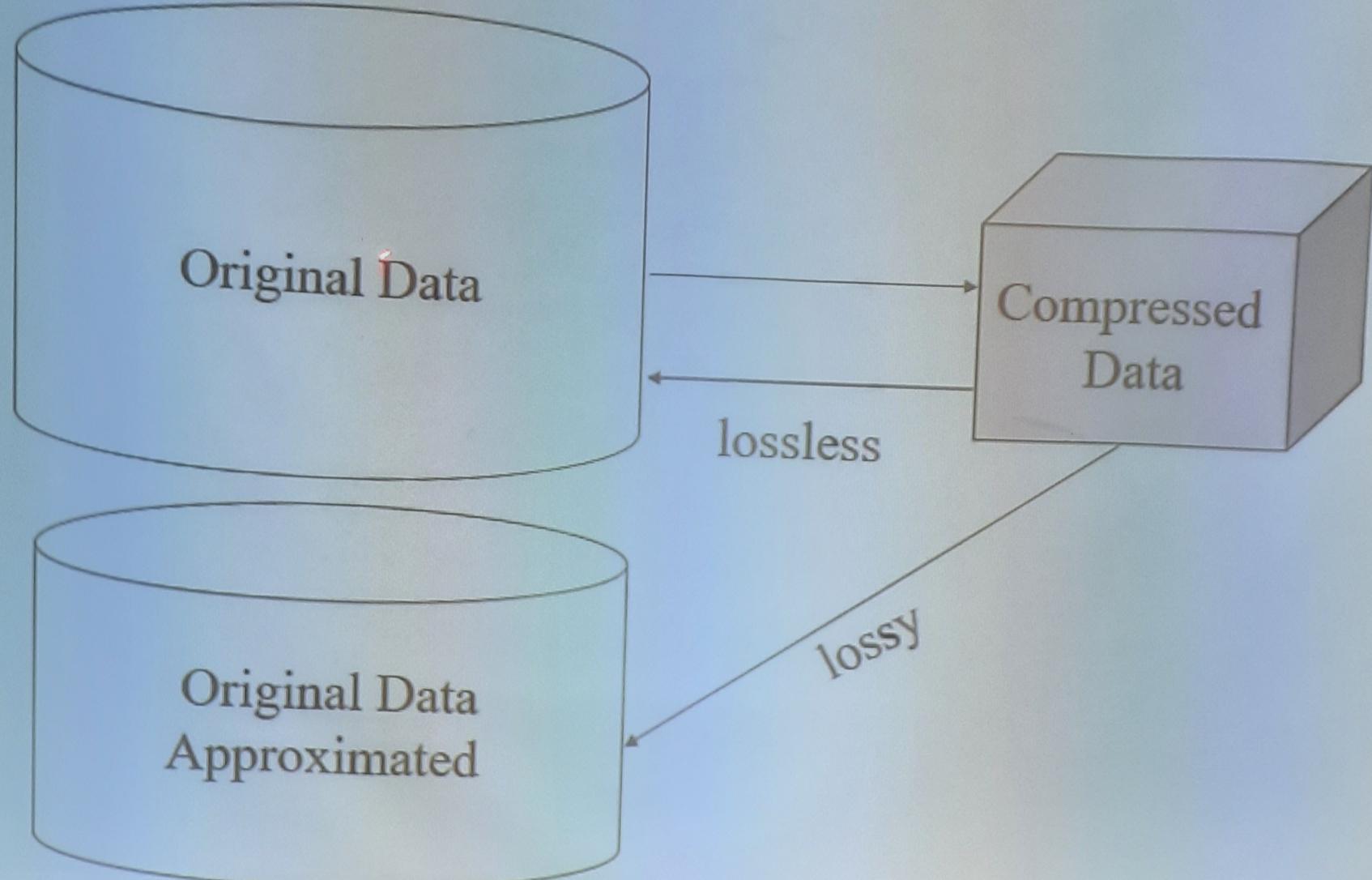


-----> Reduced attribute set: {A1, A4, A6}

Data Compression

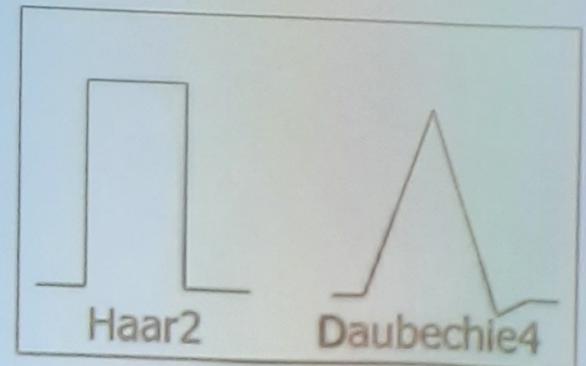
- String compression
 - There are extensive theories and well-tuned algorithms
 - Typically lossless
 - But only limited manipulation is possible without expansion
- Audio/video, image compression
 - Typically lossy compression, with progressive refinement
 - Sometimes small fragments of signal can be reconstructed without reconstructing the whole
- Time sequence is not audio
 - Typically short and vary slowly with time

Data Compression



Wavelet Transforms

- Discrete wavelet transform (DWT): linear signal processing
- Compressed approximation: store only a small fraction of the strongest of the wavelet coefficients
- Similar to discrete Fourier transform (DFT), but better lossy compression, localized in space (conserves local details)
- Method (hierarchical pyramid algorithm):
 - Length, L , must be an integer power of 2 (padding with 0s, when necessary)
 - Each transform has 2 functions:
 - smoothing (e.g., sum, weighted avg.), weighted difference
 - Applies to pairs of data, resulting in two sets of data of length $L/2$
 - Applies the two functions recursively, until reaches the desired length



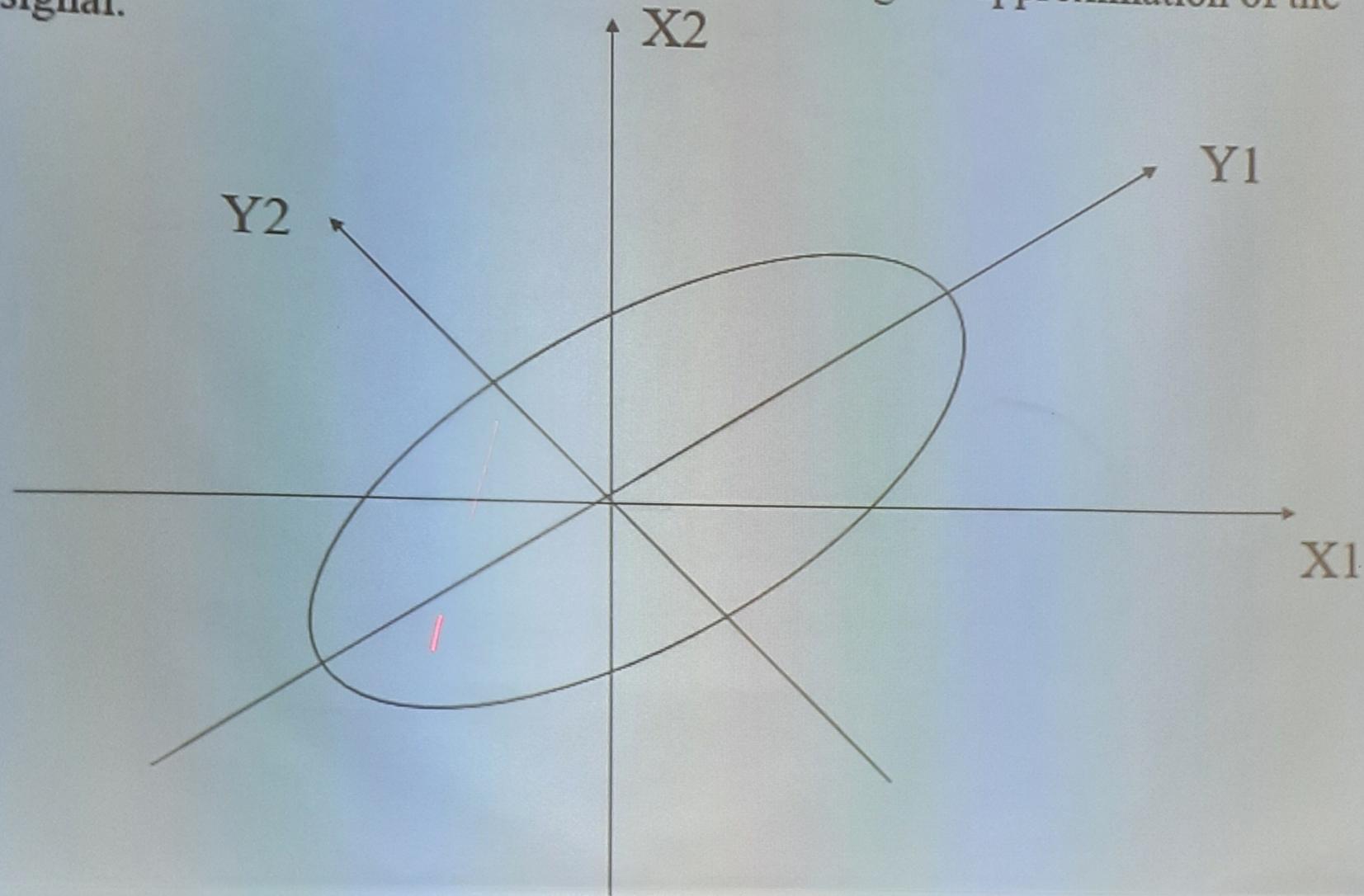
Principal Component Analysis (PCA)

Karhunen-Loeve (K-L) method

- Given N data vectors from k -dimensions, find $c \leq k$ orthogonal vectors that can be best used to represent data
 - The original data set is reduced (projected) to one consisting of N data vectors on c principal components (reduced dimensions)
- Each data vector is a linear combination of the c principal component vectors
- Works for ordered and unordered attributes
- Used when the number of dimensions is large

Principal Component Analysis

- The principal components (new set of axes) give important information about variance.
- Using the strongest components one can reconstruct a good approximation of the original signal.



Numerosity Reduction

- Parametric methods
 - Assume the data fits some model, estimate model parameters, store only the parameters, and discard the data (except possible outliers)
 - E.g.: Log-linear models: obtain value at a point in m-D space as the product on appropriate marginal subspaces
- Non-parametric methods
 - Do not assume models
 - Major families: histograms, clustering, sampling

Histograms

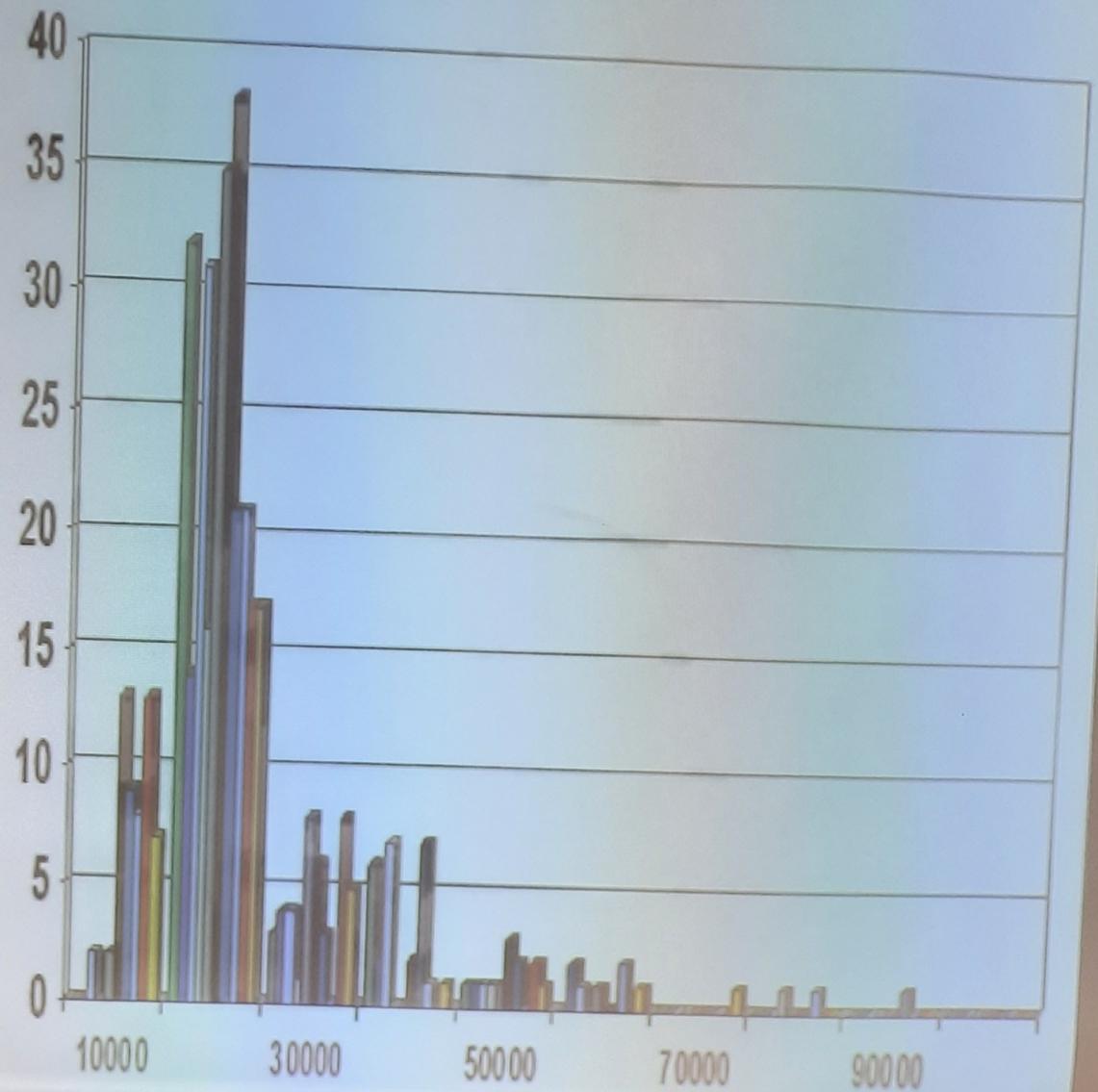
Approximate data distributions

Divide data into buckets and store average (sum) for each bucket

A bucket represents an attribute-value/frequency pair

Can be constructed optimally in one dimension using dynamic programming

Related to quantization problems.



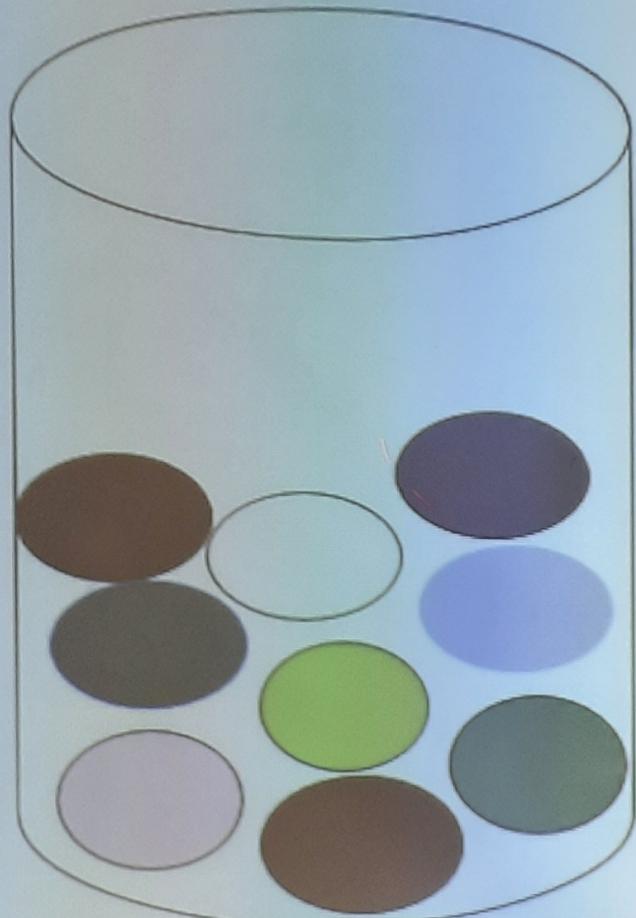
Clustering

- Partition data set into clusters, and store cluster representation only
- Quality of clusters measured by their diameter (max distance between any two objects in the cluster) or centroid distance (avg. distance of each cluster object from its centroid)
- Can be very effective if data is clustered but not if data is “smeared”
- Can have hierarchical clustering (possibly stored in multi-dimensional index tree structures (B+-tree, R-tree, quad-tree, etc))
- There are many choices of clustering definitions and clustering algorithms (further details later)

Sampling

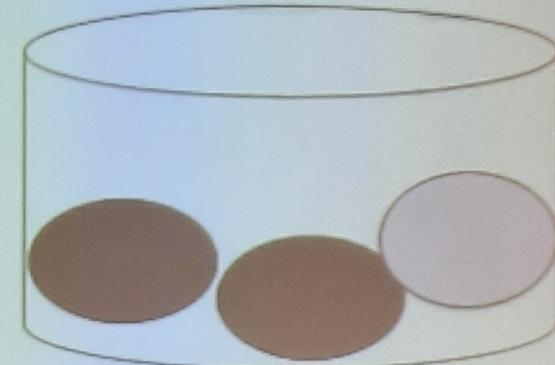
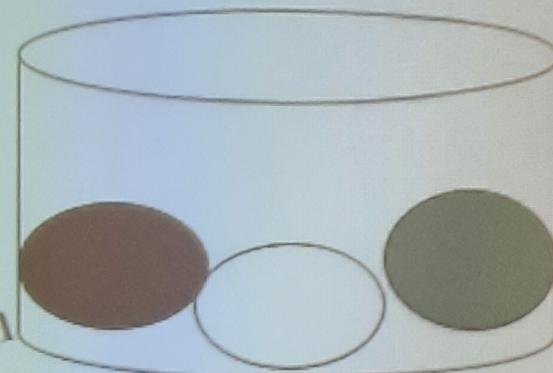
- Allow a mining algorithm to run in complexity that is potentially sub-linear to the size of the data
- Cost of sampling: proportional to the size of the sample, increases linearly with the number of dimensions
- Choose a representative subset of the data
 - Simple random sampling may have very poor performance in the presence of skew
- Develop adaptive sampling methods
 - Stratified sampling:
 - Approximate the percentage of each class (or subpopulation of interest) in the overall database
 - Used in conjunction with skewed data
- Sampling may not reduce database I/Os (page at a time).
- Sampling: natural choice for progressive refinement of a reduced data set.

Sampling



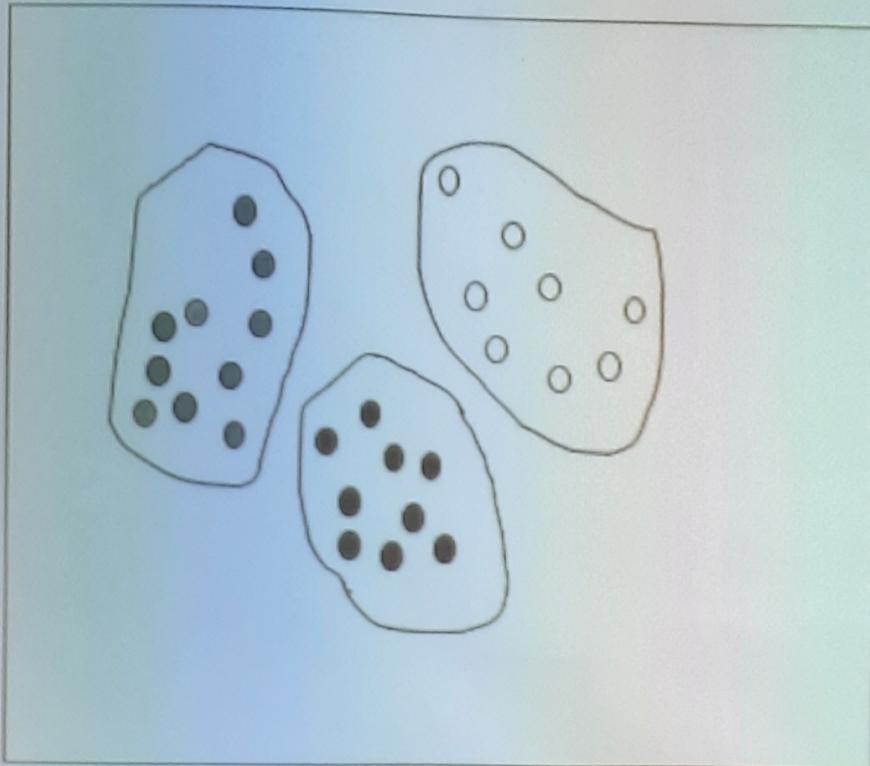
SRSWOR
(simple random
sample without
replacement)

SRSWR

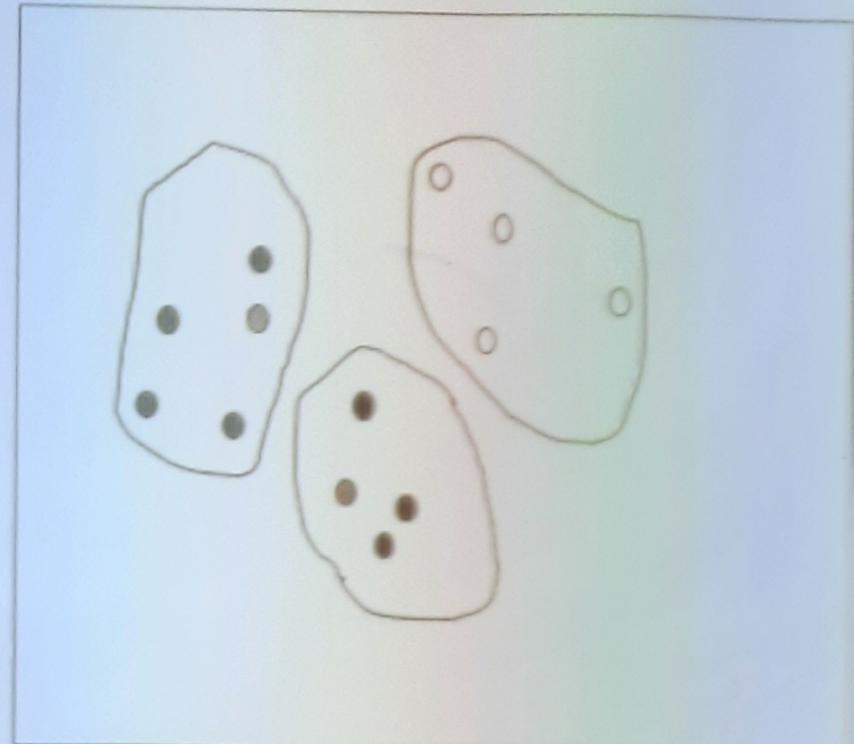


Sampling

Raw Data



Cluster/Stratified Sample



Hierarchical Reduction

- Use multi-resolution structure with different degrees of reduction
- Hierarchical clustering is often performed but tends to define partitions of data sets rather than “clusters”
- Parametric methods are usually not amenable to hierarchical representation
- Hierarchical aggregation
 - An index tree hierarchically divides a data set into partitions by value range of some attributes
 - Each partition can be considered as a bucket
 - Thus an index tree with aggregates stored at each node is a hierarchical histogram

Agenda

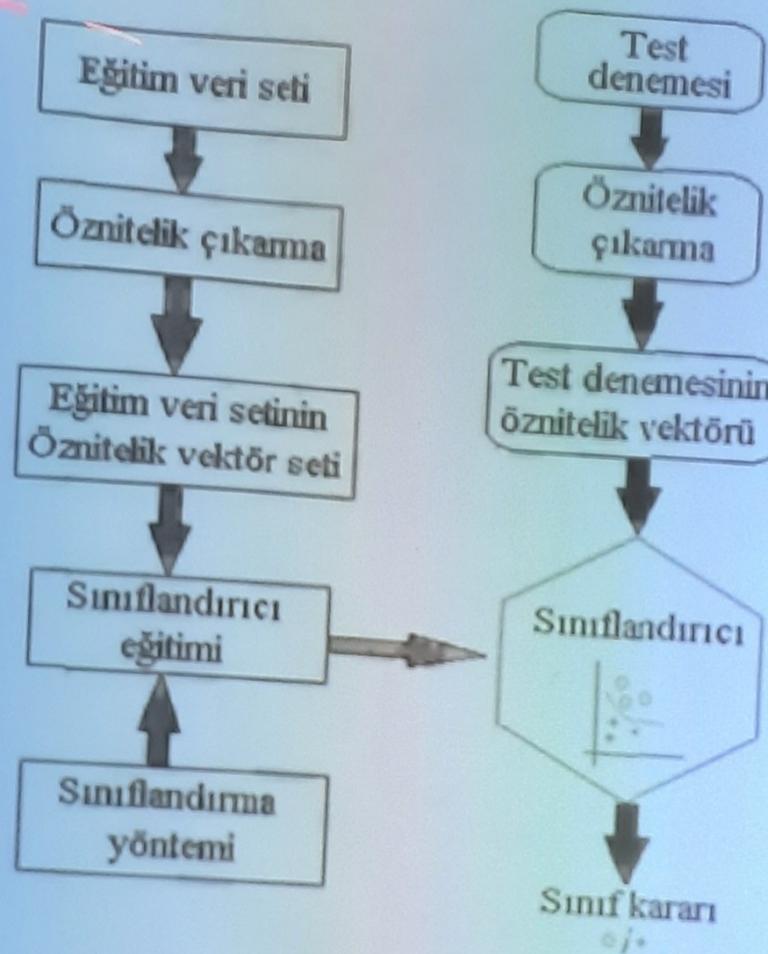
- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Sınıflandırma Algoritmalarındaki Performans Değerlendirme Kriterleri

Veri işleme ve modellemede amaç, analiz edilerek bilgi çıkarılması zor olan büyük veri yiğinlarını analiz ederek anlamlı, gizli ve faydalı olabilecek bilgi çıkarmak; bu bilgileri içerisinde barındıran bir model oluşturarak yeni gelecek bir veri nesnesi hakkında yorum yapmayı ve bu veri hakkında tahminde bulunmayı sağlamaktır. Üzerinde çalışılan veri kümesinden çıkarılan bilgi bir doğruluk derecesine sahip olup deterministik bir bilgi değildir. Oluşturulan modellerin başarım derecelerini belirleyen doğruluk, kesinlik, duyarlılık ve f-ölçütü gibi kriterler kullanılarak kullanılan algoritmaların başarıları değerlendirilir.

Sınıflandırma Algoritmalarının Karşılaştırılmasında Önemli Hususlar

Veri önisleme, parametre seçimi ve test kümesi seçimi veri işleme ve modelleme uygulamasında ortaya çıkacak olan modelin başarısını etkiler. Dolayısı ile yapılan karşılaştırma sonuçları büyük ölçüde uygulamacıya bağlıdır.



Her türlü probleminin her zaman eniyi sınıflandırma sonucunu veren bir sınıflandırmayı yöntemi yoktur. Farklı problemler içinsınıflandırma yöntemlerinin performansları farklılık gösteremektedir. Ancak, öznitelik verisinin genel dağılımının bulunduğu bir takım öngörülerde bulunabilmek mümkündür.

Parametre Seçimi

Veri işleme ve modellemede kullanılan farklı algoritmaların farklı parametreleri olabilir. Örneğin yapay sinir ağlarında gizli nöron sayısı, karar ağaçlarındaki budama işleminin parametreleri, algoritmaların kullanacağı parametrik değerleri belirler. Bu parametreler algoritmadan algoritmaya değişebilir, ya da kullanılan veri işleme ve modellemede araç programlarında farklı olabilir. Bunların seçimi oluşacak olan modelin başarımını etkileyecektir.

Test Kümesinin Seçimi

Model oluşturulurken kullanılan öğrenme ve test kümelerinin belirlenmesinin de modelin başarımı üzerinde etkisi vardır. Eldeki verinin öğrenme kümesi ve test kümesi olarak ayrılmasında farklı metodlar kullanılabilir. Kullanılan veri madenciliği programında bu işlem için farklı seçenekler bulunabilir. Öğrenme kümesi ve test kümesi farklı dosyalardan programa verilebileceği gibi, programın bir veri dosyasını belirtilen bir oranda test kümesi olarak kullanması ya da nfold metodu ile programın veri kümesini n sayıdaki parçalara ayırarak sırayla her parçayı test kümesi olarak kullanması sağlanabilir.

Model Başarım Ölçütleri

Model başarımını değerlendirirken kullanılan temel kavramlar hata oranı, kesinlik, duyarlılık ve F-ölçütüdür. Modelin başarısı, doğru sınıfa atanan örnek sayısı ve yanlış sınıfa atılan örnek sayısı nicelikleriyle alakalıdır. Test sonucunda ulaşılan sonuçların başarım bilgileri karışıklık matrisi ile ifade edilebilir. Karışıklık matrisinde satırlar test kümesindeki örneklerle ait gerçek sayıları, kolonlar ise modelin tahminlemesini ifade eder.

		Öngörülen Sınıf	
		Sınıf=1	Sınıf=0
Doğru Sınıf	Sınıf=1	a	b
	Sınıf=0	c	d

a: TP (True Pozitif)

b: FN (False Negatif)

c: FP (False Pozitif)

d: TN (True Negatif)

Doğruluk - Hata oranı

Model başarımının ölçülmesinde kullanılan en popüler ve basit yöntem, modele ait doğruluk oranıdır. Doğru sınıflandırılmış örnek sayısının ($TP + TN$), toplam örnek sayısına ($TP+TN+FP+FN$) oranıdır. Hata oranı ise bu değerin 1'e tamlayanıdır. Diğer bir ifadeyle yanlış sınıflandırılmış örnek sayısının ($FP+FN$), toplam örnek sayısına ($TP+TN+FP+FN$) oranıdır.

$$Doğruluk = \frac{TP + TN}{TP + FP + FN + TN}$$

$$Hata\ Oranı = \frac{FP + FN}{TP + FP + FN + TN}$$

Kesinlik

Kesinlik, sınıfı 1 olarak tahminlenmiş True Pozitif örnek sayısının, sınıfı 1 olarak tahminlenmiş tüm örnek sayısına oranıdır

$$Kesinlik = \frac{TP}{TP + FP}$$

F-Ölçütü

Kesinlik ve duyarlılık ölçütleri tek başına anlamlı bir karşılaştırma sonucu çıkarmamıza yeterli değildir. Her iki ölçütü beraber değerlendirmek daha doğru sonuçlar verir. Bunun için F-ölçütü tanımlanmıştır. F-ölçütü, kesinlik ve duyarlılığın harmonik ortalamasıdır.

$$F - \text{Ölçütü} = \frac{2 \times \text{Duyarlılık} \times \text{Kesinlik}}{\text{Duyarlılık} + \text{Kesinlik}}$$

Kesinlik

Kesinlik, sınıfı 1 olarak tahminlenmiş True Pozitif örnek sayısının, sınıfı 1 olarak tahminlenmiş tüm örnek sayısına oranıdır

$$Kesinlik = \frac{TP}{TP + FP}$$

Duyarlılık

Doğru sınıflandırılmış pozitif örnek sayısının toplam pozitif örnek sayısına oranıdır.

$$Duyarlılık = \frac{TP}{TP + FN}$$

F-Ölçütü

Kesinlik ve duyarlılık ölçütleri tek başına anlamlı bir karşılaştırma sonucu çıkarmamıza yeterli değildir. Her iki ölçütü beraber değerlendirmek daha doğru sonuçlar verir. Bunun için F-ölçütü tanımlanmıştır. F-ölçütü, kesinlik ve duyarlılığın harmonik ortalamasıdır.

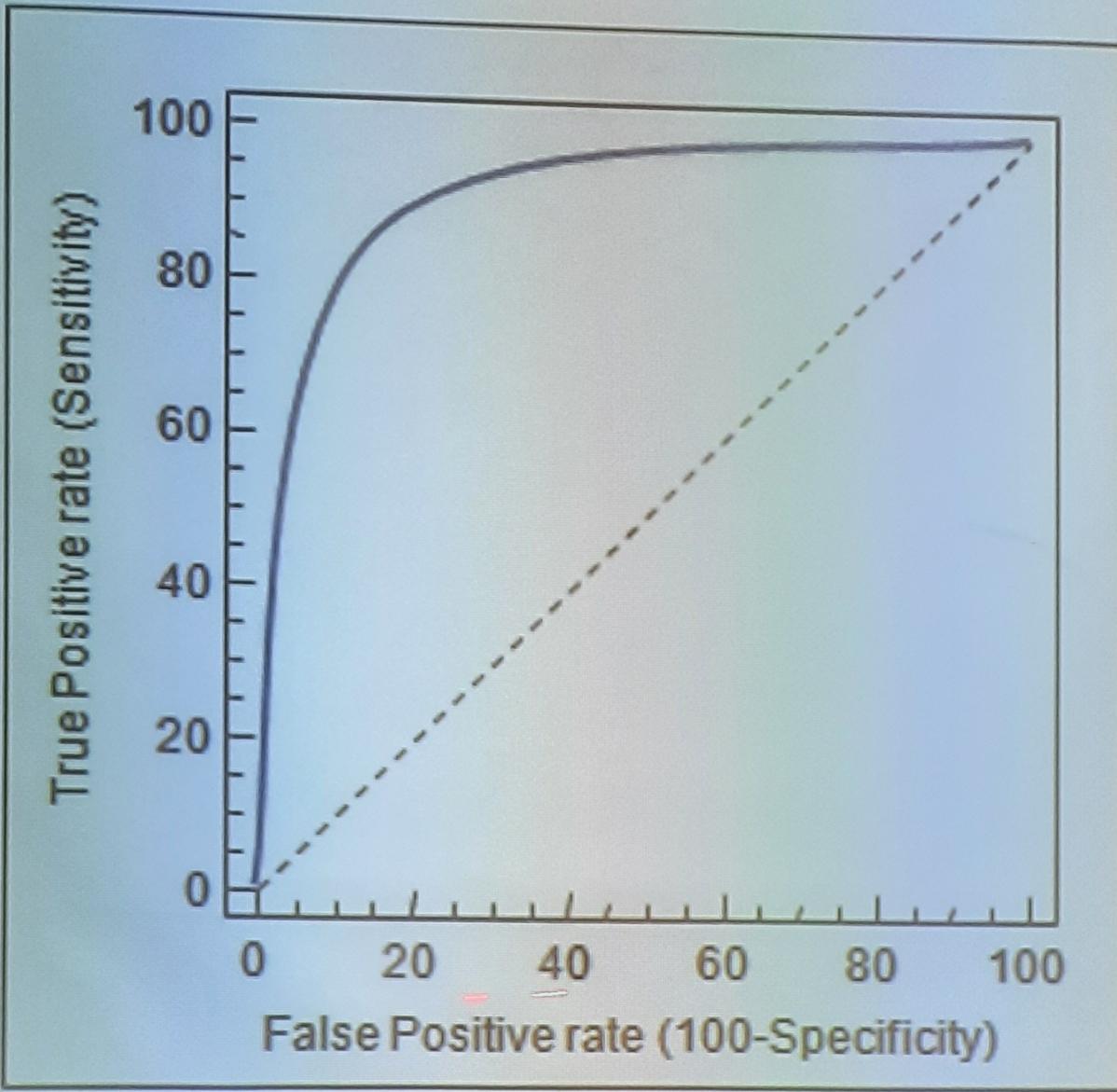
$$F - \text{Ölçütü} = \frac{2 \times \text{Duyarlılık} \times \text{Kesinlik}}{\text{Duyarlılık} + \text{Kesinlik}}$$

ROC Eğrisi (ROC Curve)

ROC eğrisi ve eksiksiz bir hassasiyet / özgüllük raporu oluşturmayı sağlar.

ROC eğrisinde, gerçek pozitif oran (Hassasiyet), bir parametrenin farklı kesme noktaları için yanlış pozitif oranı (100-Özgüllük) işlevinde çizilir. ROC eğrisindeki her nokta belirli bir karar eşigine karşılık gelen bir duyarlılık / özgüllük çifti temsil eder. ROC eğrisi altındaki alan (AUC), bir parametrenin iki grubun ne kadar iyi ayırt edilebildiğinin bir ölçüsüdür.

Örneğin iki ayrı tanı üzerinden gidelim. Hastalığa sahip bir popülasyondaki, hastalığı olmayan diğer popülasyondaki belirli bir testin sonuçlarını değerlendirirken, nadiren iki grup arasında mükemmel bir ayrim gözlemlenir. Aslında, test sonuçlarının dağılımı, aşağıdaki şekilde görüldüğü gibi çakışacaktır.



ROC eğrisi

Bir Alıcı Çalışma Karakteristiği(Receiver Operating Characteristic)(ROC) eğrisinde, gerçek pozitif oranı (Hassasiyet), farklı kesme noktaları için yanlış pozitif oranın (100-Özgüllük) fonksiyonunda çizilir. ROC eğrisindeki her nokta belirli bir karar eşigine karşılık gelen bir duyarlılık / özgüllük çifti temsil eder. Mükemmel ayrımcılıkla (iki dağılımda çakışma olmaz) yapılan bir test, sol üst köşeden geçen bir ROC eğrisine sahiptir (% 100 hassasiyet,% 100 özgüllük). Bu nedenle, ROC eğrisinin sol üst köşeye yaklaştıkça, testin genel doğruluğu artar (Zweig & Campbell, 1993).

ROC eğrisi

Bir Alıcı Çalışma Karakteristiği(Receiver Operating Characteristic)(ROC) eğrisinde, gerçek pozitif oranı (Hassasiyet), farklı kesme noktaları için yanlış pozitif oranın (100-Özgüllük) fonksiyonunda çizilir. ROC eğrisindeki her nokta belirli bir karar eşigine karşılık gelen bir duyarlılık / özgüllük çifti temsil eder. Mükemmel ayrımcılıkla (iki dağılımda çakışma olmaz) yapılan bir test, sol üst köşeden geçen bir ROC eğrisine sahiptir (% 100 hassasiyet,% 100 özgüllük). Bu nedenle, ROC eğrisinin sol üst köşeye yaklaştıkça, testin genel doğruluğu artar (Zweig & Campbell, 1993).

ROC eğrisi

Bir Alıcı Çalışma Karakteristiği(Receiver Operating Characteristic)(ROC) eğrisinde, gerçek pozitif oranı (Hassasiyet), farklı kesme noktaları için yanlış pozitif oranın (100-Özgüllük) fonksiyonunda çizilir. ROC eğrisindeki her nokta belirli bir karar eşigine karşılık gelen bir duyarlılık / özgüllük çifti temsil eder. Mükemmel ayrımcılıkla (iki dağılımda çakışma olmaz) yapılan bir test, sol üst köşeden geçen bir ROC eğrisine sahiptir (% 100 hassasiyet,% 100 özgüllük). Bu nedenle, ROC eğrisinin sol üst köşeye yaklaştıkça, testin genel doğruluğu artar (Zweig & Campbell, 1993).