Thus, we say that $\mathcal{C}$ contains complete information regarding its corresponding frequent itemsets. On the other hand, $\mathcal{M}$ registers only the support of the maximal itemsets. It usually does not contain the complete support information regarding its corresponding frequent itemsets. We illustrate these concepts with Example 6.2.

Example 6.2 Closed and maximal frequent itemsets. Suppose that a transaction database has only two transactions: $\{\langle a_1, a_2, \ldots, a_{100}\rangle; \langle a_1, a_2, \ldots, a_{50}\rangle\}$. Let the minimum support count threshold be $min\_sup = 1$. We find two closed frequent itemsets and their support counts, that is, $\mathcal{C} = \{\{a_1, a_2, \ldots, a_{100}\} : 1; \{a_1, a_2, \ldots, a_{50}\} : 2\}$. There is only one maximal frequent itemset: $\mathcal{M} = \{\{a_1, a_2, \ldots, a_{100}\} : 1\}$. Notice that we cannot include $\{a_1, a_2, \ldots, a_{50}\}$ as a maximal frequent itemset because it has a frequent superset, $\{a_1, a_2, \ldots, a_{100}\}$. Compare this to the preceding where we determined that there are $2^{100} - 1$ frequent itemsets, which are too many to be enumerated!

The set of closed frequent itemsets contains complete information regarding the frequent itemsets. For example, from $\mathcal{C}$, we can derive, say, (1) $\{a_2, a_{45} : 2\}$ since $\{a_2, a_{45}\}$ is a sub-itemset of the itemset $\{a_1, a_2, \ldots, a_{50} : 2\}$; and (2) $\{a_8, a_{55} : 1\}$ since $\{a_8, a_{55}\}$ is not a sub-itemset of the previous itemset but of the itemset $\{a_1, a_2, \ldots, a_{100} : 1\}$. However, from the maximal frequent itemset, we can only assert that both itemsets ($\{a_2, a_{45}\}$ and $\{a_8, a_{55}\}$) are frequent, but we cannot assert their actual support counts. ■

6.2 Frequent Itemset Mining Methods

In this section, you will learn methods for mining the simplest form of frequent patterns such as those discussed for market basket analysis in Section 6.1.1. We begin by presenting **Apriori**, the basic algorithm for finding frequent itemsets (Section 6.2.1). In Section 6.2.2, we look at how to generate strong association rules from frequent itemsets. Section 6.2.3 describes several variations to the Apriori algorithm for improved efficiency and scalability. Section 6.2.4 presents pattern-growth methods for mining frequent itemsets that confine the subsequent search space to only the data sets containing the current frequent itemsets. Section 6.2.5 presents methods for mining frequent itemsets that take advantage of the vertical data format.

6.2.1 Apriori Algorithm: Finding Frequent Itemsets by Confined Candidate Generation

Apriori is a seminal algorithm proposed by R. Agrawal and R. Srikant in 1994 for mining frequent itemsets for Boolean association rules [AS94b]. The name of the algorithm is based on the fact that the algorithm uses *prior knowledge* of frequent itemset properties, as we shall see later. Apriori employs an iterative approach known as a *level-wise* search, where k-itemsets are used to explore $(k+1)$-itemsets. First, the set of frequent 1-itemsets is found by scanning the database to accumulate the count for each item, and

collecting those items that satisfy minimum support. The resulting set is denoted by L_1. Next, L_1 is used to find L_2, the set of frequent 2-itemsets, which is used to find L_3, and so on, until no more frequent k-itemsets can be found. The finding of each L_k requires one full scan of the database.

To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the **Apriori property** is used to reduce the search space.

Apriori property: *All nonempty subsets of a frequent itemset must also be frequent.*

The Apriori property is based on the following observation. By definition, if an itemset I does not satisfy the minimum support threshold, *min_sup*, then I is not frequent, that is, $P(I) < min\_sup$. If an item A is added to the itemset I, then the resulting itemset (i.e., $I \cup A$) cannot occur more frequently than I. Therefore, $I \cup A$ is not frequent either, that is, $P(I \cup A) < min\_sup$.

This property belongs to a special category of properties called **antimonotonicity** in the sense that *if a set cannot pass a test, all of its supersets will fail the same test as well.* It is called *antimonotonicity* because the property is monotonic in the context of failing a test.[6]

"*How is the Apriori property used in the algorithm?*" To understand this, let us look at how L_{k-1} is used to find L_k for $k \geq 2$. A two-step process is followed, consisting of **join** and **prune** actions.

1. **The join step**: To find L_k, a set of **candidate** k-itemsets is generated by joining L_{k-1} with itself. This set of candidates is denoted C_k. Let l_1 and l_2 be itemsets in L_{k-1}. The notation $l_i[j]$ refers to the jth item in l_i (e.g., $l_1[k-2]$ refers to the second to the last item in l_1). For efficient implementation, Apriori assumes that items within a transaction or itemset are sorted in lexicographic order. For the $(k-1)$-itemset, l_i, this means that the items are sorted such that $l_i[1] < l_i[2] < \cdots < l_i[k-1]$. The join, $L_{k-1} \bowtie L_{k-1}$, is performed, where members of L_{k-1} are joinable if their first $(k-2)$ items are in common. That is, members l_1 and l_2 of L_{k-1} are joined if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \cdots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$. The condition $l_1[k-1] < l_2[k-1]$ simply ensures that no duplicates are generated. The resulting itemset formed by joining l_1 and l_2 is $\{l_1[1], l_1[2], \ldots, l_1[k-2], l_1[k-1], l_2[k-1]\}$.
2. **The prune step**: C_k is a superset of L_k, that is, its members may or may not be frequent, but all of the frequent k-itemsets are included in C_k. A database scan to determine the count of each candidate in C_k would result in the determination of L_k (i.e., all candidates having a count no less than the minimum support count are frequent by definition, and therefore belong to L_k). C_k, however, can be huge, and so this could involve heavy computation. To reduce the size of C_k, the Apriori property

[6]The Apriori property has many applications. For example, it can also be used to prune search during data cube computation (Chapter 5).

is used as follows. Any $(k-1)$-itemset that is not frequent cannot be a subset of a frequent k-itemset. Hence, if any $(k-1)$-subset of a candidate k-itemset is not in L_{k-1}, then the candidate cannot be frequent either and so can be removed from C_k. This **subset testing** can be done quickly by maintaining a hash tree of all frequent itemsets.

Example 6.3 **Apriori.** Let's look at a concrete example, based on the *AllElectronics* transaction database, D, of Table 6.1. There are nine transactions in this database, that is, $|D| = 9$. We use Figure 6.2 to illustrate the Apriori algorithm for finding frequent itemsets in D.

1. In the first iteration of the algorithm, each item is a member of the set of candidate 1-itemsets, C_1. The algorithm simply scans all of the transactions to count the number of occurrences of each item.
2. Suppose that the minimum support count required is 2, that is, $min\_sup = 2$. (Here, we are referring to *absolute* support because we are using a support count. The corresponding relative support is $2/9 = 22\%$.) The set of frequent 1-itemsets, L_1, can then be determined. It consists of the candidate 1-itemsets satisfying minimum support. In our example, all of the candidates in C_1 satisfy minimum support.
3. To discover the set of frequent 2-itemsets, L_2, the algorithm uses the join $L_1 \bowtie L_1$ to generate a candidate set of 2-itemsets, C_2.[7] C_2 consists of $\binom{|L_1|}{2}$ 2-itemsets. Note that no candidates are removed from C_2 during the prune step because each subset of the candidates is also frequent.

Table 6.1 Transactional Data for an *AllElectronics* Branch

TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

[7] $L_1 \bowtie L_1$ is equivalent to $L_1 \times L_1$, since the definition of $L_k \bowtie L_k$ requires the two joining itemsets to share $k-1 = 0$ items.

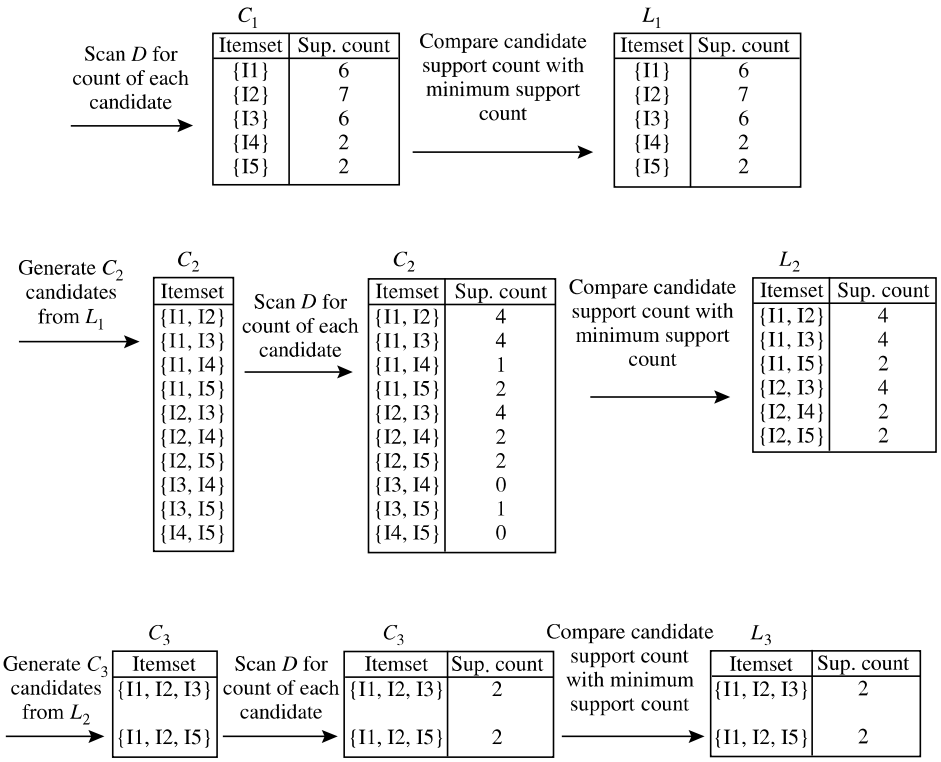


Figure 6.2 Generation of the candidate itemsets and frequent itemsets, where the minimum support count is 2.

4. Next, the transactions in D are scanned and the support count of each candidate itemset in C_2 is accumulated, as shown in the middle table of the second row in Figure 6.2.
5. The set of frequent 2-itemsets, L_2, is then determined, consisting of those candidate 2-itemsets in C_2 having minimum support.
6. The generation of the set of the candidate 3-itemsets, C_3, is detailed in Figure 6.3. From the join step, we first get $C_3 = L_2 \bowtie L_2 = \{\{I1, I2, I3\}, \{I1, I2, I5\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\}\}$. Based on the Apriori property that all subsets of a frequent itemset must also be frequent, we can determine that the four latter candidates cannot possibly be frequent. We therefore remove them from C_3, thereby saving the effort of unnecessarily obtaining their counts during the subsequent scan of D to determine L_3. Note that when given a candidate k-itemset, we only need to check if its $(k-1)$-subsets are frequent since the Apriori algorithm uses a level-wise

(a) Join: $C_3 = L_2 \bowtie L_2 = \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\}$
$\bowtie \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\}$
$= \{\{I1, I2, I3\}, \{I1, I2, I5\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\}\}$.

(b) Prune using the Apriori property: All nonempty subsets of a frequent itemset must also be frequent. Do any of the candidates have a subset that is not frequent?

- The 2-item subsets of {I1, I2, I3} are {I1, I2}, {I1, I3}, and {I2, I3}. All 2-item subsets of {I1, I2, I3} are members of L_2. Therefore, keep {I1, I2, I3} in C_3.
- The 2-item subsets of {I1, I2, I5} are {I1, I2}, {I1, I5}, and {I2, I5}. All 2-item subsets of {I1, I2, I5} are members of L_2. Therefore, keep {I1, I2, I5} in C_3.
- The 2-item subsets of {I1, I3, I5} are {I1, I3}, {I1, I5}, and {I3, I5}. {I3, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I1, I3, I5} from C_3.
- The 2-item subsets of {I2, I3, I4} are {I2, I3}, {I2, I4}, and {I3, I4}. {I3, I4} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I3, I4} from C_3.
- The 2-item subsets of {I2, I3, I5} are {I2, I3}, {I2, I5}, and {I3, I5}. {I3, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I3, I5} from C_3.
- The 2-item subsets of {I2, I4, I5} are {I2, I4}, {I2, I5}, and {I4, I5}. {I4, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I4, I5} from C_3.

(c) Therefore, $C_3 = \{\{I1, I2, I3\}, \{I1, I2, I5\}\}$ after pruning.

Figure 6.3 Generation and pruning of candidate 3-itemsets, C_3, from L_2 using the Apriori property.

search strategy. The resulting pruned version of C_3 is shown in the first table of the bottom row of Figure 6.2.

7. The transactions in D are scanned to determine L_3, consisting of those candidate 3-itemsets in C_3 having minimum support (Figure 6.2).
8. The algorithm uses $L_3 \bowtie L_3$ to generate a candidate set of 4-itemsets, C_4. Although the join results in $\{\{I1, I2, I3, I5\}\}$, itemset $\{I1, I2, I3, I5\}$ is pruned because its subset $\{I2, I3, I5\}$ is not frequent. Thus, $C_4 = \phi$, and the algorithm terminates, having found all of the frequent itemsets. ■

Figure 6.4 shows pseudocode for the Apriori algorithm and its related procedures. Step 1 of Apriori finds the frequent 1-itemsets, L_1. In steps 2 through 10, L_{k-1} is used to generate candidates C_k to find L_k for $k \geq 2$. The apriori_gen procedure generates the candidates and then uses the Apriori property to eliminate those having a subset that is not frequent (step 3). This procedure is described later. Once all of the candidates have been generated, the database is scanned (step 4). For each transaction, a subset function is used to find all subsets of the transaction that are candidates (step 5), and the count for each of these candidates is accumulated (steps 6 and 7). Finally, all the candidates satisfying the minimum support (step 9) form the set of frequent itemsets, L (step 11).

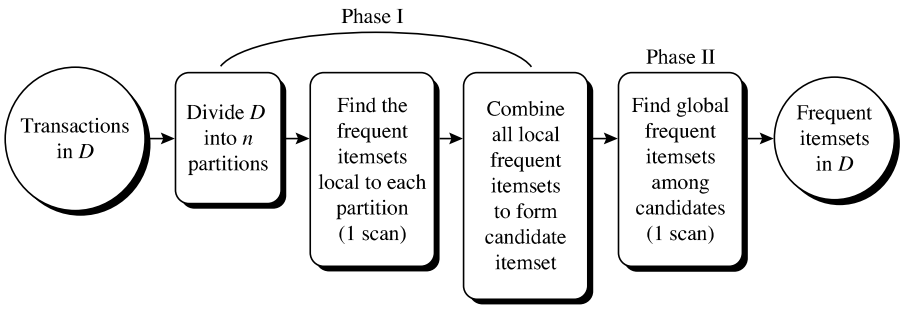


Figure 6.6 Mining by partitioning the data.

a second scan of D is conducted in which the actual support of each candidate is assessed to determine the global frequent itemsets. Partition size and the number of partitions are set so that each partition can fit into main memory and therefore be read only once in each phase.

Sampling (mining on a subset of the given data): The basic idea of the sampling approach is to pick a random sample S of the given data D, and then search for frequent itemsets in S instead of D. In this way, we trade off some degree of accuracy against efficiency. The S sample size is such that the search for frequent itemsets in S can be done in main memory, and so only one scan of the transactions in S is required overall. Because we are searching for frequent itemsets in S rather than in D, it is possible that we will miss some of the global frequent itemsets.

To reduce this possibility, we use a lower support threshold than minimum support to find the frequent itemsets local to S (denoted L^S). The rest of the database is then used to compute the actual frequencies of each itemset in L^S. A mechanism is used to determine whether all the global frequent itemsets are included in L^S. If L^S actually contains all the frequent itemsets in D, then only one scan of D is required. Otherwise, a second pass can be done to find the frequent itemsets that were missed in the first pass. The sampling approach is especially beneficial when efficiency is of utmost importance such as in computationally intensive applications that must be run frequently.

Dynamic itemset counting (adding candidate itemsets at different points during a scan): A dynamic itemset counting technique was proposed in which the database is partitioned into blocks marked by start points. In this variation, new candidate itemsets can be added at any start point, unlike in Apriori, which determines new candidate itemsets only immediately before each complete database scan. The technique uses the count-so-far as the lower bound of the actual count. If the count-so-far passes the minimum support, the itemset is added into the frequent itemset collection and can be used to generate longer candidates. This leads to fewer database scans than with Apriori for finding all the frequent itemsets.

Other variations are discussed in the next chapter.

6.2.4 A Pattern-Growth Approach for Mining Frequent Itemsets

As we have seen, in many cases the Apriori candidate generate-and-test method significantly reduces the size of candidate sets, leading to good performance gain. However, it can suffer from two nontrivial costs:

- *It may still need to generate a huge number of candidate sets.* For example, if there are 10^4 frequent 1-itemsets, the Apriori algorithm will need to generate more than 10^7 candidate 2-itemsets.
- *It may need to repeatedly scan the whole database and check a large set of candidates by pattern matching.* It is costly to go over each transaction in the database to determine the support of the candidate itemsets.

"Can we design a method that mines the complete set of frequent itemsets without such a costly candidate generation process?" An interesting method in this attempt is called **frequent pattern growth,** or simply **FP-growth**, which adopts a *divide-and-conquer* strategy as follows. First, it compresses the database representing frequent items into a **frequent pattern tree,** or **FP-tree**, which retains the itemset association information. It then divides the compressed database into a set of *conditional databases* (a special kind of projected database), each associated with one frequent item or "pattern fragment," and mines each database separately. For each "pattern fragment," only its associated data sets need to be examined. Therefore, this approach may substantially reduce the size of the data sets to be searched, along with the "growth" of patterns being examined. You will see how it works in Example 6.5.

Example 6.5 **FP-growth (finding frequent itemsets without candidate generation).** We reexamine the mining of transaction database, D, of Table 6.1 in Example 6.3 using the frequent pattern growth approach.

The first scan of the database is the same as Apriori, which derives the set of frequent items (1-itemsets) and their support counts (frequencies). Let the minimum support count be 2. The set of frequent items is sorted in the order of descending support count. This resulting set or *list* is denoted by L. Thus, we have $L = \{\{\text{I2: 7}\}, \{\text{I1: 6}\}, \{\text{I3: 6}\}, \{\text{I4: 2}\}, \{\text{I5: 2}\}\}$.

An FP-tree is then constructed as follows. First, create the root of the tree, labeled with "null." Scan database D a second time. The items in each transaction are processed in L order (i.e., sorted according to descending support count), and a branch is created for each transaction. For example, the scan of the first transaction, "T100: I1, I2, I5," which contains three items (I2, I1, I5 in L order), leads to the construction of the first branch of the tree with three nodes, ⟨I2: 1⟩, ⟨I1: 1⟩, and ⟨I5: 1⟩, where I2 is linked as a child to the root, I1 is linked to I2, and I5 is linked to I1. The second transaction, T200, contains the items I2 and I4 in L order, which would result in a branch where I2 is linked to the root and I4 is linked to I2. However, this branch would share a common **prefix,** I2, with the existing path for T100. Therefore, we instead increment the count of the I2 node by 1, and create a new node, ⟨I4: 1⟩, which is linked as a child to ⟨I2: 2⟩. In general,

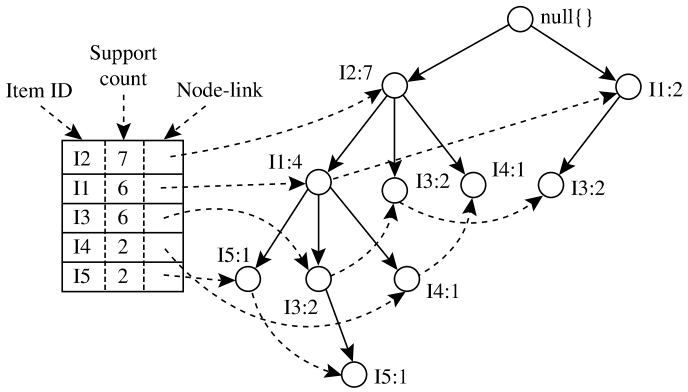


Figure 6.7 An FP-tree registers compressed, frequent pattern information.

when considering the branch to be added for a transaction, the count of each node along a common prefix is incremented by 1, and nodes for the items following the prefix are created and linked accordingly.

To facilitate tree traversal, an item header table is built so that each item points to its occurrences in the tree via a chain of **node-links**. The tree obtained after scanning all the transactions is shown in Figure 6.7 with the associated node-links. In this way, the problem of mining frequent patterns in databases is transformed into that of mining the FP-tree.

The FP-tree is mined as follows. Start from each frequent length-1 pattern (as an initial **suffix pattern**), construct its **conditional pattern base** (a "sub-database," which consists of the set of *prefix paths* in the FP-tree co-occurring with the suffix pattern), then construct its (*conditional*) FP-tree, and perform mining recursively on the tree. The pattern growth is achieved by the concatenation of the suffix pattern with the frequent patterns generated from a conditional FP-tree.

Mining of the FP-tree is summarized in Table 6.2 and detailed as follows. We first consider I5, which is the last item in L, rather than the first. The reason for starting at the end of the list will become apparent as we explain the FP-tree mining process. I5 occurs in two FP-tree branches of Figure 6.7. (The occurrences of I5 can easily be found by following its chain of node-links.) The paths formed by these branches are ⟨I2, I1, I5: 1⟩ and ⟨I2, I1, I3, I5: 1⟩. Therefore, considering I5 as a suffix, its corresponding two prefix paths are ⟨I2, I1: 1⟩ and ⟨I2, I1, I3: 1⟩, which form its conditional pattern base. Using this conditional pattern base as a transaction database, we build an I5-conditional FP-tree, which contains only a single path, ⟨I2: 2, I1: 2⟩; I3 is not included because its support count of 1 is less than the minimum support count. The single path generates all the combinations of frequent patterns: {I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}.

For I4, its two prefix paths form the conditional pattern base, {{I2 I1: 1}, {I2: 1}}, which generates a single-node conditional FP-tree, ⟨I2: 2⟩, and derives one frequent pattern, {I2, I4: 2}.

Table 6.2 Mining the FP-Tree by Creating Conditional (Sub-)Pattern Bases

Item	*Conditional Pattern Base*	*Conditional FP-tree*	*Frequent Patterns Generated*
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}	⟨I2: 2, I1: 2⟩	{I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}
I4	{{I2, I1: 1}, {I2: 1}}	⟨I2: 2⟩	{I2, I4: 2}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}	⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩	{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}
I1	{{I2: 4}}	⟨I2: 4⟩	{I2, I1: 4}

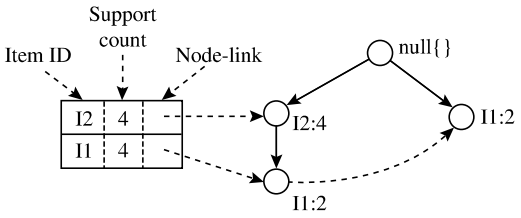


Figure 6.8 The conditional FP-tree associated with the conditional node I3.

Similar to the preceding analysis, I3's conditional pattern base is {{I2, I1: 2}, {I2: 2}, {I1: 2}}. Its conditional FP-tree has two branches, ⟨I2: 4, I1: 2⟩ and ⟨I1: 2⟩, as shown in Figure 6.8, which generates the set of patterns {{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}}. Finally, I1's conditional pattern base is {{I2: 4}}, with an FP-tree that contains only one node, ⟨I2: 4⟩, which generates one frequent pattern, {I2, I1: 4}. This mining process is summarized in Figure 6.9. ■

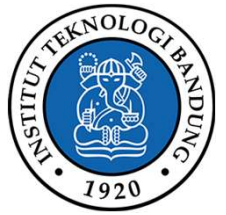
The FP-growth method transforms the problem of finding long frequent patterns into searching for shorter ones in much smaller conditional databases recursively and then concatenating the suffix. It uses the least frequent items as a suffix, offering good selectivity. The method substantially reduces the search costs.

When the database is large, it is sometimes unrealistic to construct a main memory-based FP-tree. An interesting alternative is to first partition the database into a set of projected databases, and then construct an FP-tree and mine it in each projected database. This process can be recursively applied to any projected database if its FP-tree still cannot fit in main memory.

A study of the FP-growth method performance shows that it is efficient and scalable for mining both long and short frequent patterns, and is about an order of magnitude faster than the Apriori algorithm.

6.2.5 Mining Frequent Itemsets Using the Vertical Data Format

Both the Apriori and FP-growth methods mine frequent patterns from a set of transactions in *TID-itemset* format (i.e., {*TID* : *itemset*}), where *TID* is a transaction ID and *itemset* is the set of items bought in transaction *TID*. This is known as the **horizontal data format**. Alternatively, data can be presented in *item-TID_set* format



Klasifikasi #01

IF4041 – Semester I 2020/2021

Slide diadopsi sumber:
Sumber: Jiawei Han, Micheline Kamber, and Jian Pei,

Data Mining: Concepts & Techniques , 3rd Ed.

Ikhtisar

- Pengantar
- K-Nearest Neighbor
- Decision Tree
- Naïve Bayes Classifier
- Metrik Evaluasi untuk Klasifikasi
- Cross Validation
- Metode Ensemble

Pengantar

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by **labels** indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Prediction Problems: Classification vs. Numeric Prediction

- Classification
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Numeric Prediction
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
  - Credit/loan approval:
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is

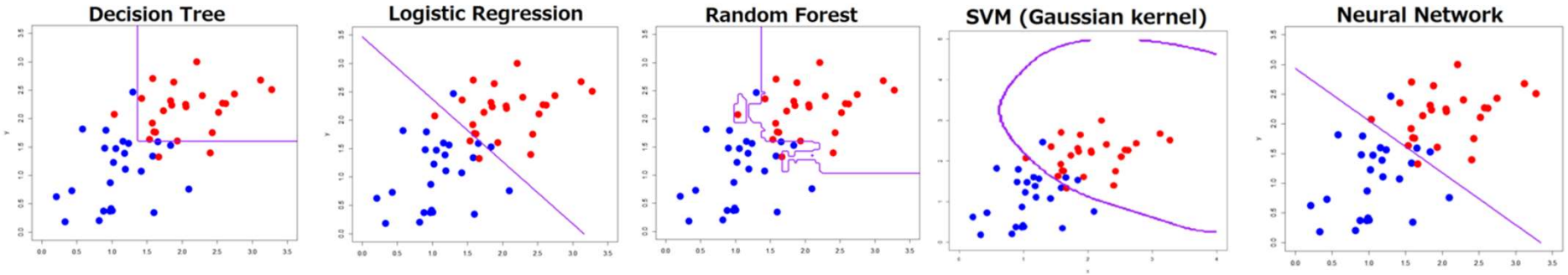
Classification—A Two-Step Process

- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set (otherwise overfitting)
  - If the accuracy is acceptable, use the model to classify new data
- Note: If *the test set* is used to select models, it is called validation (test) set

Pengantar

- **Pengertian:** sebuah tugas yang ***mengidentifikasi kelas*** suatu data/pengamatan, dari domain kelas yang diberikan. Secara teknis, klasifier bertugas untuk membuat ***decision boundary***.

- **Contoh tugas klasifikasi:**
  - ✓ Membedakan email spam *vs.* bukan spam
  - ✓ Mengklasifikasikan sentimen dari suatu teks
  - ✓ OCR (*optical character recognition*), khususnya setelah memisahkan teks menjadi per karakter
  - ✓ Fitur login menggunakan wajah
  - ✓ Mendeteksi *fraud* dalam suatu transaksi

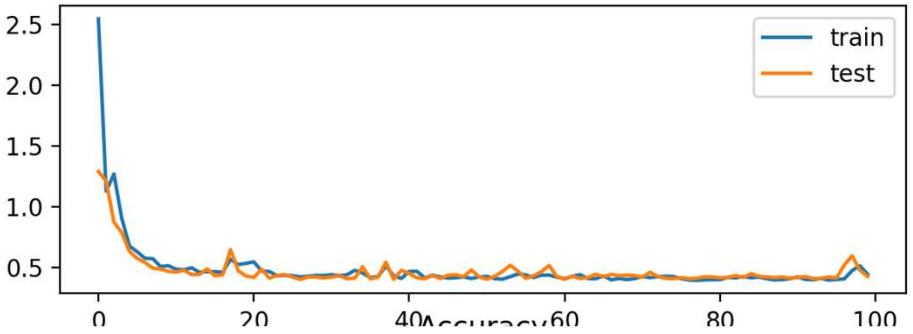


Alur Klasifikasi

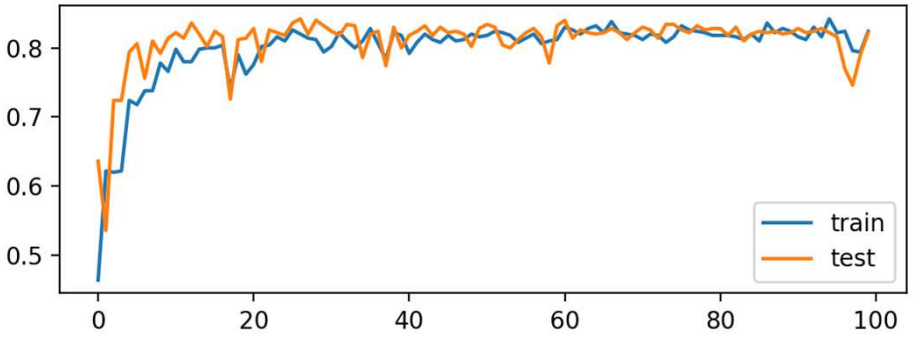
- (i) data training dan (ii) testing
- (i) data training, (ii) validation, (iii) testing

model klasifier

loss



akurasi



Contoh hasil pelatihan dari model klasifier: grafik akurasi dan loss

Contoh Klasifikasi Menggunakan Tools

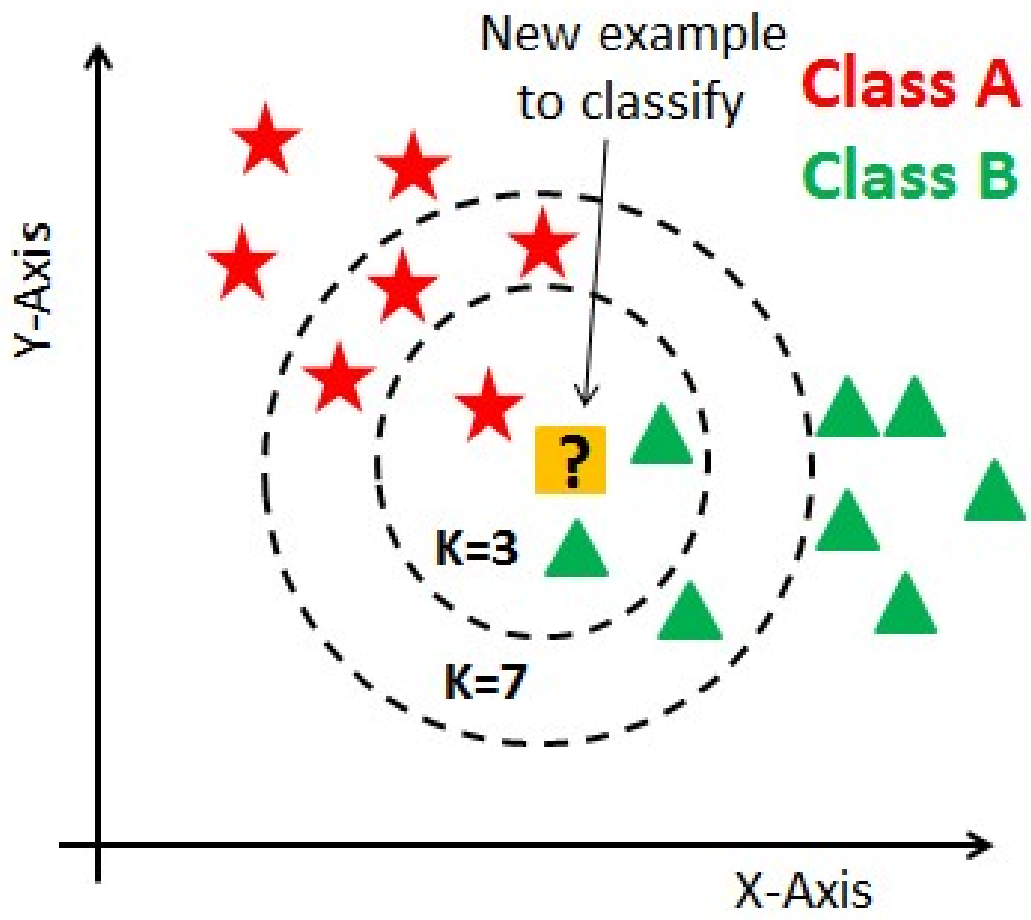
```
from sklearn import neighbors #import untuk KNN
from sklearn import tree #import untuk decision tree

MODEL = ['knn', 'svm', 'dt']; MODEL = MODEL[0]
if MODEL == 'knn':
    model = neighbors.KNeighborsClassifier(n_neighbors=5) #mendefinisikan model
    model.fit(X_train, y_train) #mentraining model
    hasil_prediksi = model.predict(X_test) #mengetes model
elif MODEL == 'dt':
    model = tree.DecisionTreeClassifier() #mendefinisikan model
    model.fit(X_train, y_train) #mentraining model
    hasil_prediksi = model.predict(X_test) #mengetes mode
```

- Sangat mudah bukan?

K-Nearest Neighbor

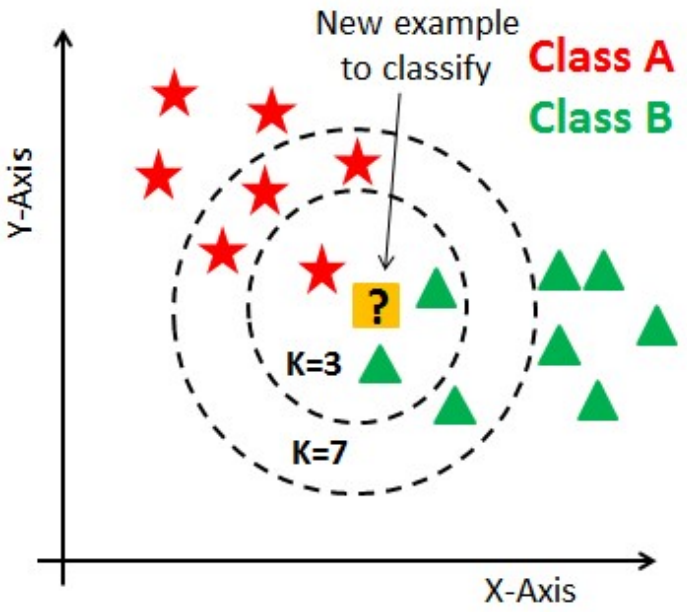
Cara Kerja KNN



- K = 3, '*new example*'
  → kelas B
- K = 7, '*new example*'
  → kelas ?

Cara Kerja KNN

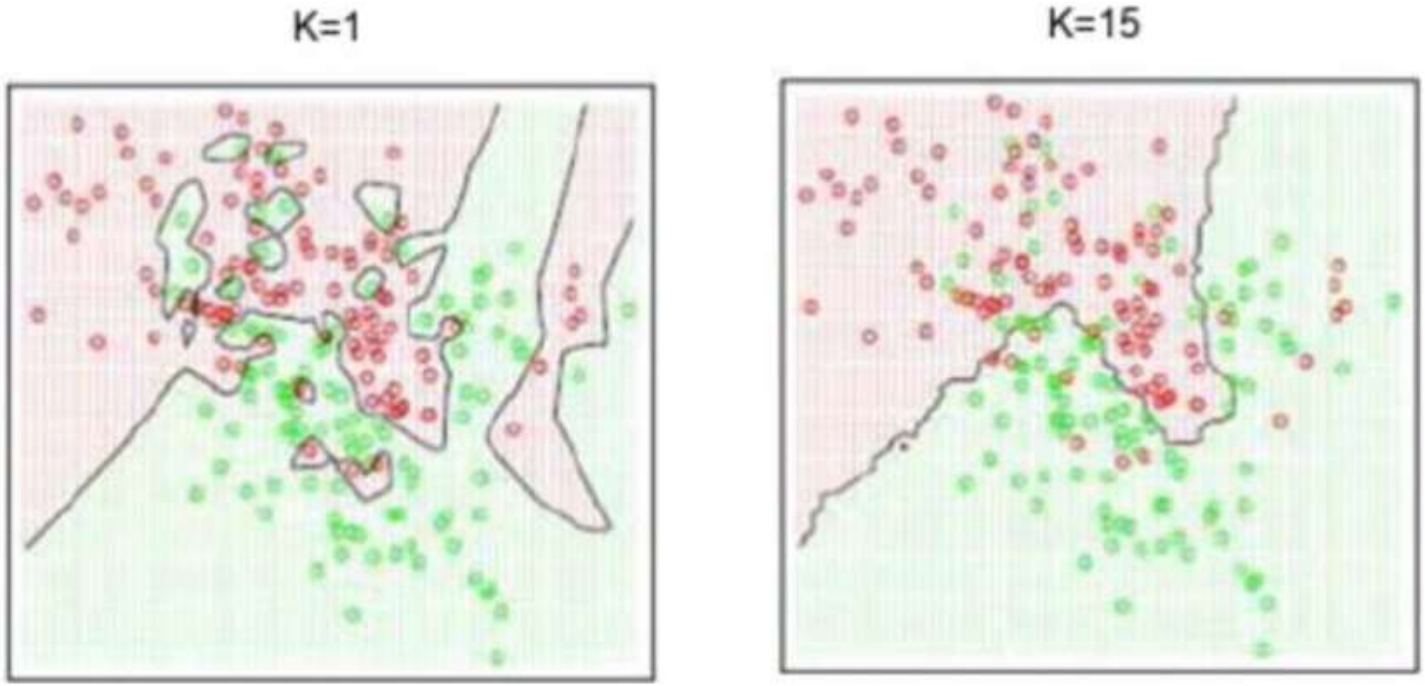
a. **Input**: set data training beserta labelnya, data testing, nilai *k*
b. Untuk data testing *i*, hitung jaraknya ke setiap data training
c. Tentukan *k*-data training yang jaraknya paling dekat dengan data testing *i*
d. Periksa label dari *k* data tersebut
e. Tentukan label yang frekuensinya paling banyak
f. Masukan data testing *i* tersebut ke kelas dengan frekuensi yang paling banyak
g. Ulangi langkah (3) sampai (6) untuk semua data testing yang lainnya.



Jarak yang lazim digunakan: Euclidean distance

$$Euc(X,Y) = \sqrt{\sum_{i=1}^{n}(X_i - Y_i)^2}$$

Efek Nilai *k* pada KNN

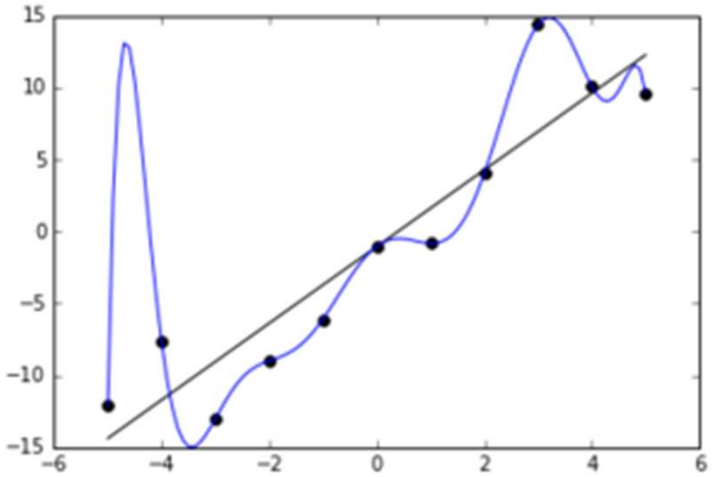


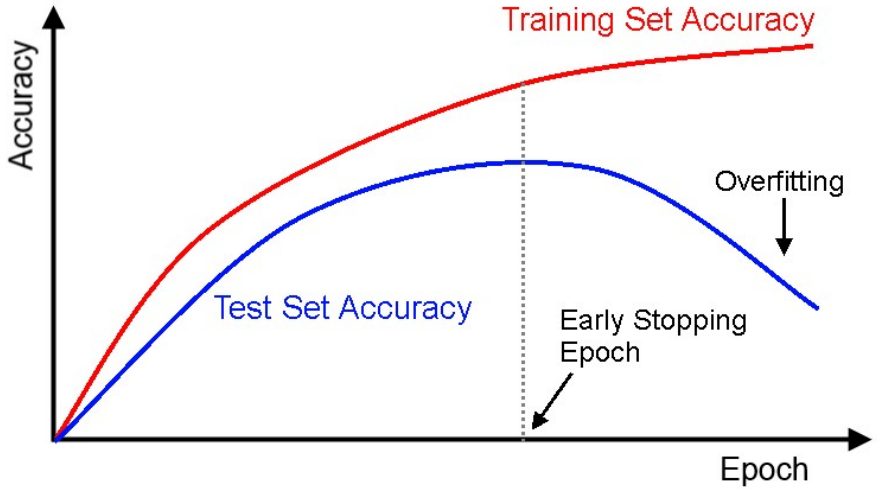
Efek nilai *k* terhadap hasil dar KNN

- Nilai *k* yang kecil cenderung membuat model *overfitting* (*less general*)
- Sebaliknya, nilai *k* yang besar cenderung membuat model yang general, dengan konsekuensi beberapa objek *'miss classified'*.

Konsep Overfitting

- **Pengertian**: memiliki kinerja bagus pada data training, tetapi kinerja akan turun drastis pada data testing
- Ilustrasi paling mudah adalah dengan model regresi (kiri) dan metode iteratif (kanan) berikut:





- *Trade-off* dari *overfitting* adalah ***generalization***.

Decision Tree

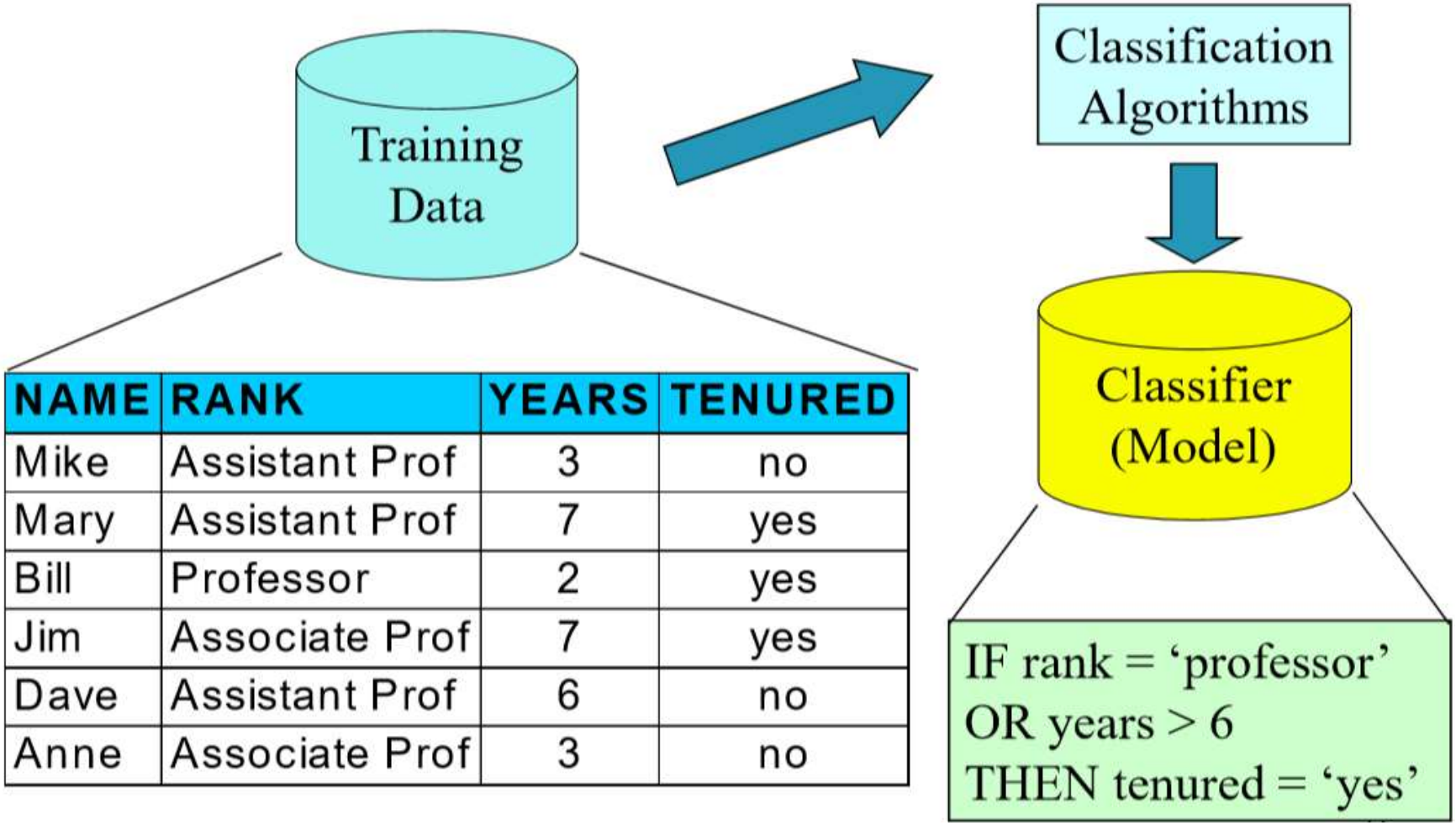
Pengantar Decision Tree

a. Salah satu metode klasifikasi yang paling populer, karena mudah untuk diinterpretasi oleh manusia, yakni berupa "**aturan-aturan dasar**", atau mirip dengan "***if-else***".

b. **Ilustrasi**: dari tabel ini, cobalah buat "aturan-aturan dasar" untuk membedakan "tenured professor" dan "bukan tenured professor"!

NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

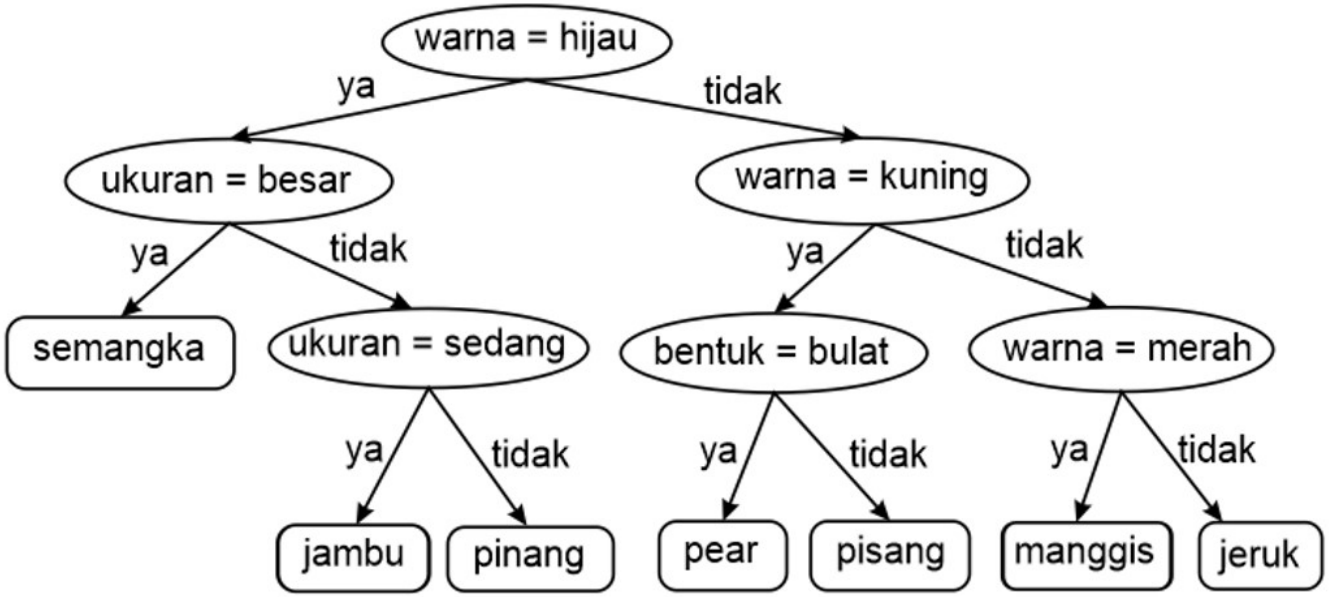
Pengantar Decision Tree



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Pengantar Decision Tree

- Seiring dengan tugas yang lebih kompleks, diperlukan aturan yang kompleks pula. Hal ini dapat distrukturkan menjadi sebuah “pohon keputusan”, seperti ilustrasi di bawah ini.



- Bagaimana cara memilih simpul (*node*) yang tepat? Decision tree adalah jawabannya.

Cara Kerja Decision Tree

- Diberikan sebuah tabel data $\boldsymbol{D}$ (‘main’ *vs* ‘tidak main’) sebagai berikut, kita akan memilih **atribut paling bagus** untuk dijadikan sebagai **simpul pemecah pertama**.

Cuaca	Temperatur	Kelembaban	Angin	Keputusan
x1	x2	x3	x4	Y
Cerah	Panas	Tinggi	kecil	tidak
Cerah	Panas	Tinggi	besar	tidak
Mendung	Panas	Tinggi	kecil	Ya
Hujan	Sedang	Tinggi	kecil	Ya
Hujan	Dingin	Normal	kecil	Ya
Hujan	Dingin	Normal	besar	Tidak
Mendung	Dingin	Normal	besar	Ya
Cerah	Sedang	Tinggi	kecil	Tidak
Cerah	Dingin	Normal	kecil	Ya
Hujan	Sedang	Normal	Kecil	Ya
Cerah	Sedang	Normal	Besar	Ya
Mendung	Sedang	Tinggi	Besar	Ya
Mendung	Panas	Normal	Kecil	Ya
Hujan	Sedang	Tinggi	Besar	Tidak

- Salah satu metodenya = ***information gain*.**

Cara Kerja Decision Tree

- Diberikan tabel data $\boldsymbol{D}$, banyaknya informasi yang dibutuhkan untuk pengklasifikasian data $\boldsymbol{D}$ tesebut adalah sebanyak nilai *entropy-nya,* yang dirumuskan dengan:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

Dengan:

$p_i = probabilitas\ kelas_i$

$m = banyaknya\ kelas$

- Informasi yang dibutuhkan setelah memilih atribut A untuk memecah data $\boldsymbol{D}$ menjadi v bagian:

$$Info_A(D) = -\sum_{j=1}^{v} \frac{D_j}{D} \times info(D_j)$$

- Informasi gain = selisih kedua nilai tersebut di atas

$$Gain(A) = Info(D) - Info_A(D)$$

- **Rule:** pilih atribut A yang memiliki nilai $Gain$ yang paling besar.

Cara Kerja Decision Tree

- Contoh perhitungan

Cuaca	Temperatur	Kelembaban	Angin	Keputusan
x1	x2	x3	x4	Y
Cerah	Panas	Tinggi	kecil	tidak
Cerah	Panas	Tinggi	besar	tidak
Mendung	Panas	Tinggi	kecil	Ya
Hujan	Sedang	Tinggi	kecil	Ya
Hujan	Dingin	Normal	kecil	Ya
Hujan	Dingin	Normal	besar	Tidak
Mendung	Dingin	Normal	besar	Ya
Cerah	Sedang	Tinggi	kecil	Tidak
Cerah	Dingin	Normal	kecil	Ya
Hujan	Sedang	Normal	Kecil	Ya
Cerah	Sedang	Normal	Besar	Ya
Mendung	Sedang	Tinggi	Besar	Ya
Mendung	Panas	Normal	Kecil	Ya
Hujan	Sedang	Tinggi	Besar	Tidak

- 9 ‘main’ , 5 ‘tidak’ dari total 14 baris data
- 5 cuaca cerah = 2 ‘main’ , 3 ‘tidak’
- 4 cuaca mendung = 4 ‘main’, ‘0’ tidak
- 5 cuaca hujan = 3 ‘main’, 2 ‘tidak’

- Entropy awal

$$Info(D) = Info(9,5)$$

$$Info(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0{,}940$$

- Entropy setelah memilih atribut cuaca

$$Info_{cuaca}(D) = \frac{5}{14}info(2,3) + \frac{4}{14}info(4,0) + \frac{5}{14}info(3,2) = 0{,}693$$

- Informasi gain:

$$Gain(cuaca) = 0{,}940 - 0{,}693 = \mathbf{0{,}247}$$

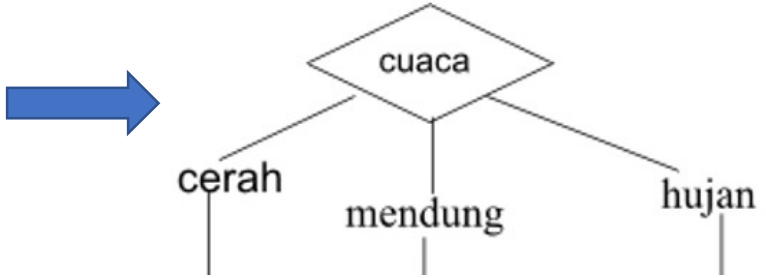
Dengan cara yang sama, akan didapatkan:

- ✓ $Gain(temp.) = 0{,}029$
- ✓ $Gain(kelembapan) = 0{,}152$
- ✓ $Gain(angin) = 0{,}048$

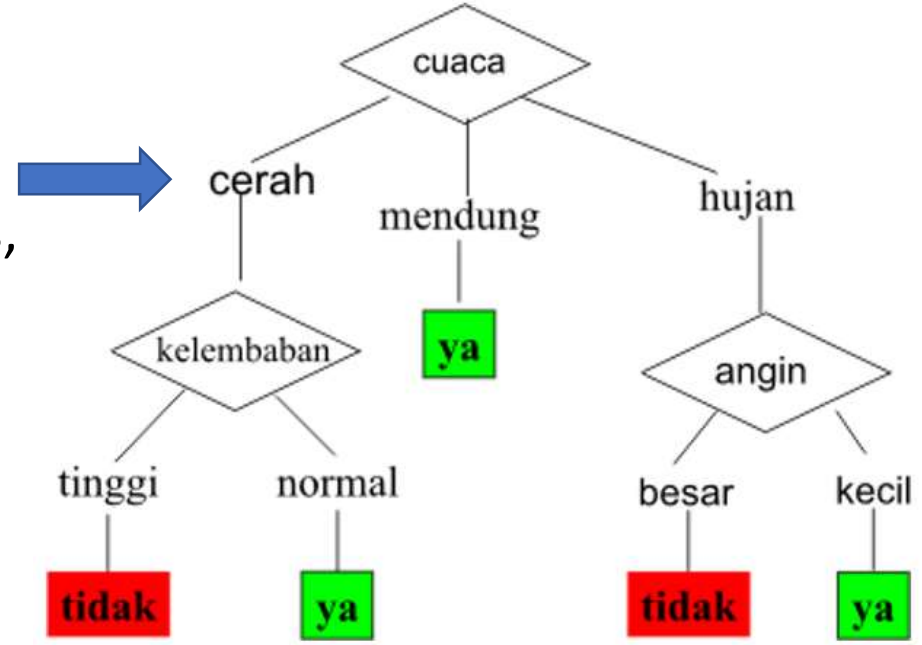
*__Note:__ jika subset atribut hanya 1 kelas, maka $nilai\ entropy = 0$

Cara Kerja Decision Tree

- Karena atribut cuaca memiliki nilai informasi gain paling besar, maka akan dijadikan sebagai simpul pemecah pertama.



- Di masing-masing anak simpul ‘cerah’, ‘mendung’ dan ‘hujan’, jika masih terdapat lebih dari satu kelas, maka ulangi langkahnya lagi untuk mencari simpul pemecah terbaik, begitu seterusnya.



Kriteria-kriteria Lain Untuk Menentukan Simpul Tree

- Selain *information gain,* kita dapat menggunakan:
  - ✓ *Gain ratio* (pilih yang paling besar). Gunakan split info untuk normalisasi

$$Gain\ ratio(A) = \frac{Gain(A)}{SplitInfo(A)}$$

dengan:

$$SplitInfo(A) = -\sum_{j=1}^{v} \frac{D_j}{D} \times \log_2\left(\frac{D_j}{D}\right)$$

  - ✓ *Indeks Gini* (pilih yang paling kecil)
    - Dalam sebuah pemecahan, hitung:
      - persentase per cabang (w_i)
      - IG_i)

$$IG_i = 1 - \sum_{k=1}^{C} p_k^2$$

Dengan p_k = probabilitas kelas $\{1,2,..C\}$

    - Hitung $\sum_{i=1}^{n}(w_i \times IG_i)$, dengan n adalah banyaknya cabang

Decision Tree – Beberapa Detail

- Menghindari overfitting
  - ✓ Salah satunya dapat dilakukan post-pruning
    - Tree yang kompleks akan rentan dengan overfitting
    - Maka dari itu, hapus simpul yang “kurang penting”
    - Ukuran “kurang penting” dapat diuji dengat set data di luar dari data training
- Atribut kontinyu
  - ✓ Secara dinamis membagi ke atribut diskret
- Beberapa varian dari metode decision tree yang diusulkan secara utuh: **ID3**, **C4.5**, **CART**
- Beberapa pengembangan dari klasifier tree-based: random forest, Xgboost (*Extreme Gradient Boosting*)

Klasifikasi Bayesian

Bayesian Classification: Why?

- A statistical classifier: performs *probabilistic prediction, i.e.,* predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers

Bayes' Theorem: Basics

- Total probability Theorem: $$P(B) = \sum_{i=1}^{M} P(B|A_i)P(A_i)$$

- Bayes' Theorem: $$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$

  - Let **X** be a data sample ("*evidence*"): class label is unknown
  - Let H be a *hypothesis* that X belongs to class C
  - Classification is to determine P(H|**X**), (i.e., *posteriori probability):* the probability that the hypothesis holds given the observed data sample **X**
  - P(H) (*prior probability*): the initial probability
    - E.g., **X** will buy computer, regardless of age, income, …
  - P(**X**): probability that sample data is observed
  - P(**X**|H) (likelihood): the probability of observing the sample **X**, given that the hypothesis holds
    - E.g., Given that **X** will buy computer, the prob. that X is 31..40, medium income

Prediction Based on Bayes' Theorem

- Given training data **X**, *posteriori probability of a hypothesis* H, P(H|**X**), follows the Bayes' theorem

$$P(H \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid H)P(H)}{P(\mathbf{X})} = P(\mathbf{X} \mid H) \times P(H) / P(\mathbf{X})$$

- Informally, this can be viewed as

  posteriori = likelihood x prior/evidence

- Predicts **X** belongs to C_i iff the probability $P(C_i|\mathbf{X})$ is the highest among all the $P(C_k|X)$ for all the *k* classes

- Practical difficulty: It requires initial knowledge of many probabilities, involving significant computational cost

Classification Is to Derive the Maximum Posteriori

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector **X** = (x_1, x_2, …, x_n)
- Suppose there are *m* classes C_1, C_2, …, C_m.
- Classification is to derive the maximum posteriori, i.e., the maximal $P(C_i|\mathbf{X})$
- This can be derived from Bayes' theorem

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- Since P(X) is constant for all classes, only

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

needs to be maximized

Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times ... \times P(x_n|C_i)$$

- This greatly reduces the computation cost: Only counts the class distribution

- If A_k is categorical, $P(x_k|C_i)$ is the # of tuples in C_i having value x_k for A_k divided by $|C_{i,D}|$ (# of tuples of C_i in D)

- If A_k is continous-valued, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

and $P(x_k|C_i)$ is

$$P(\mathbf{X}|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

Naïve Bayes Classifier: Training Dataset

Class:
C1:buys_computer = ‘yes’
C2:buys_computer = ‘no’

Data to be classified:
X = (age <=30,
Income = medium,
Student = yes
Credit_rating = Fair)

age	income	student	credit_rating	_comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayes Classifier: An Example

- $P(C_i)$: P(buys_computer = "yes") = 9/14 = 0.643
  P(buys_computer = "no") = 5/14= 0.357
- Compute $P(X|C_i)$ for each class
  P(age = "<=30" | buys_computer = "yes") = 2/9 = 0.222
  P(age = "<= 30" | buys_computer = "no") = 3/5 = 0.6
  P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444
  P(income = "medium" | buys_computer = "no") = 2/5 = 0.4
  P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667
  P(student = "yes" | buys_computer = "no") = 1/5 = 0.2
  P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667
  P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4
- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

$P(X|C_i)$: P(X|buys_computer = "yes")
= 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = "no")
= 0.6 x 0.4 x 0.2 x 0.4 = 0.019

$P(X|C_i)*P(C_i)$:
P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028
P(buys_computer = "no") = 0.007

Therefore, X belongs to class ("buys_computer = yes")

age	income	student	credit_rating	co
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Avoiding the Zero-Probability Problem

- Naïve Bayesian prediction requires each conditional prob. be **non-zero**. Otherwise, the predicted prob. will be zero

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10)
- Use **Laplacian correction** (or Laplacian estimator)
  - *Adding 1 to each case*

    Prob(income = low) = 1/1003

    Prob(income = medium) = 991/1003

    Prob(income = high) = 11/1003
  - The "corrected" prob. estimates are close to their "uncorrected" counterparts

Naïve Bayes Classifier: Comments

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.

      Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayes Classifier
- How to deal with these dependencies? Bayesian Belief Networks (Chapter 9)

Metrik Evaluasi untuk Klasifikasi

Confusion Matrix

- Diberikan data prediksi, misalkan:

n=165	Prediksi = -1	Prediksi = +1	
Aktual = -1	$TN = 50$	$FP = 10$	60
Aktual = +1	$FN = 5$	$TP = 100$	105
	55	110	

1. **TN (*True Negative*)**: output kelas negatif yang berhasil ditebak sebagai kelas negatif
2. **TP (*True Positive*)**: output kelas positif yang berhasil ditebak sebagai kelas positif
3. **FN (*False Negative*)**: output kelas positif yang salah ditebak sebagai kelas negatif
4. **FP (*False Positive*)**: output kelas negatif yang salah ditebak sebagai kelas positif

Confusion Matrix

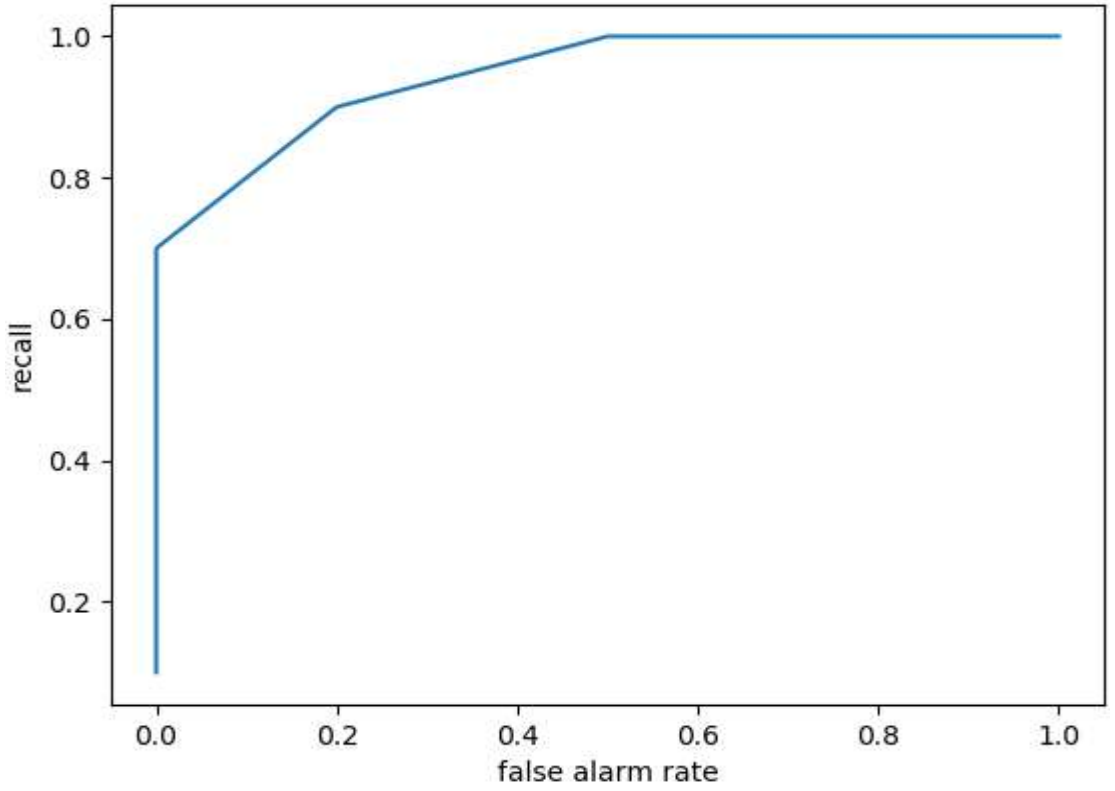
- Diberikan data prediksi, misalkan:

n=165	Prediksi = -1	Prediksi = +1	
Aktual = -1	$TN = 50$	$FP = 10$	60
Aktual = +1	$FN = 5$	$TP = 100$	105
	55	110	

1. **Akurasi**: (*TN*+*TP*)/jumlah data = (*50*+*100*)/*165*.
2. ***Miss classification rate*/error rate**: (*FP*+*FN*)/jumlah data = (*10*+*5*)/*165*.
3. ***Recall*** atau ***sensitivity*** atau ***true positive rate***: *TP*/aktual "+1"= 100/*105*.
4. **Presisi**: *TP*/prediksi "+1" = *100*/*110*.
5. ***False positive rate*** atau ***false alarm rate***: *FP*/aktual "-1" = 10/*60*.
6. ***Specificity***: *TN*/aktual "-1" = 50/*60*.

Kurva ROC (*Receiver Operating Characteristic*)

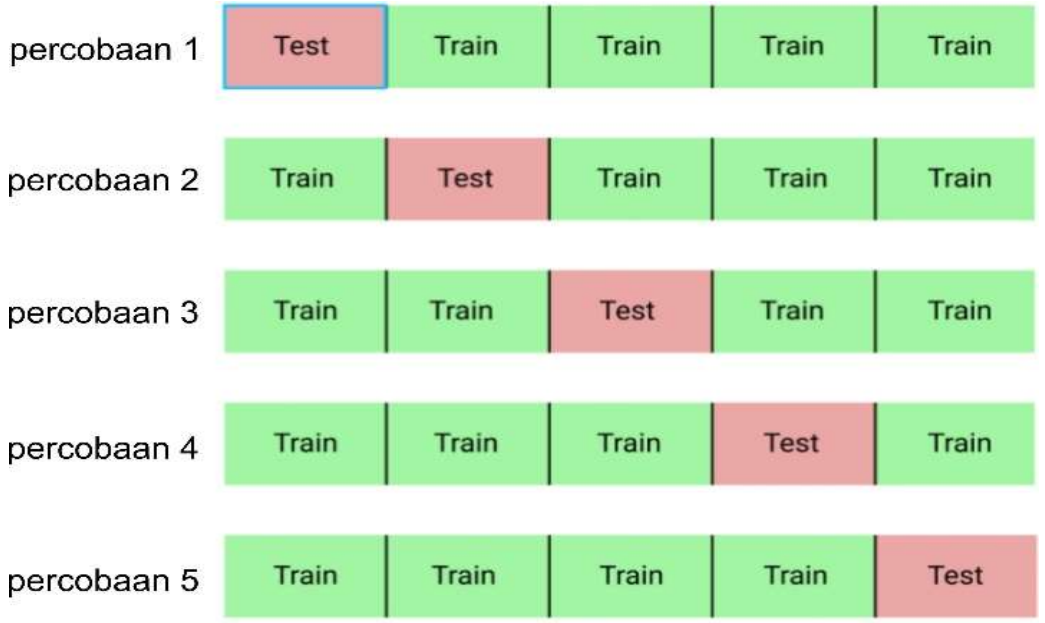
- Ketika kita menaikkan *recall*, konsekuensinya adalah menaikkan *false alarm*.
- Kurva ROC mengakum trade-off antar kedua hal tersebut, yakni dengan menghitung luas area di bawah kurva.



Cross Validation

Cross Validation

- Membagi menjadi ***k-folds***
- 1 fold menjadi data testing, selebihnya menjadi data training
- Kemudian fold data testing diganti dengan fold yang lain, begitu seterusnya seperti ilustrasi berikut, untuk 5-folds.



- Model ditraining di semua percobaan untuk selanjutnya divalidasi

Metode Ensemble

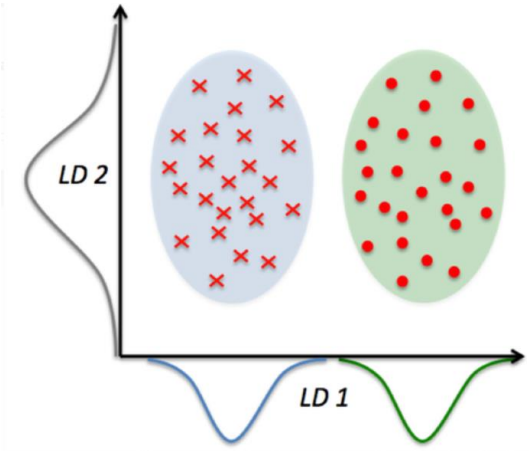
Metode Ensemble

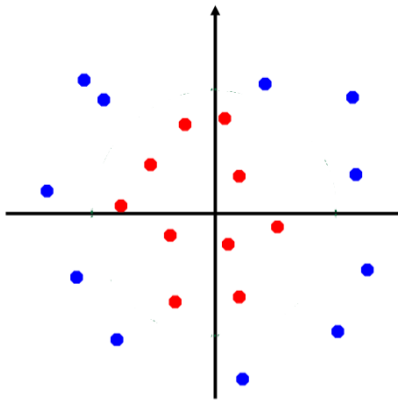
- **Pengertian**: menggabungkan beberapa model agar didapatkan performa yang lebih bagus
- Misal diberikan 5 model:
  - ✓ Lakukan prediksi data dengan 5 model tersebut
  - ✓ Lihat hasil prediksi dari kelima model tersebut
  - ✓ Hasil prediksi akhir dapat diambil dari: majority vote, dirata-rata atau dibobot.

End..

Dimentionality reduction

- Terkadang kita memiliki atribut/fitur data yang sangat banyak, sehingga:
  - ✓Komputasi menjadi berat
  - ✓Tidak bisa divisualisasikan → dimensi berapa yang maksimal masih bisa kita visualisasikan?
- **Ilustrasi**: diberikan data dengan 2 dimensi berikut, bagaimana cara terbaik untuk mereduksi menjadi 1 dimensi?





Dimentionality reduction – PCA

- Singkatan dari: Principal Component Analysis
- Tujuan: mereduksi data dengan dimensi m ke dimensi p, yang mana $p \leq m$
- Kategori pembelajaran: masuk dalam kategori *unsupervised learning*
- Ide: memproyeksikan data $\boldsymbol{X}$ ke data terproyeksi $\boldsymbol{Z}$ di mana $\boldsymbol{Z}$ adalah kemungkinan data terproyeksi dengan **nilai variansi tertinggi**.

$$\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{W}$$

Di mana:

$\boldsymbol{Z}$ = data terproyeksi, ukuran = ?
$\boldsymbol{W}$ = matriks proyeksi, ukuran = (m, p)
$\boldsymbol{X}$ = data yang ingin diproyeksikan, ukuran = (n_{data}, m)

Dimentionality reduction – PCA

$$\underset{w}{argmax}\,(varian\,Z) = \underset{w}{argmax}\,((Z - \bar{Z}\,)^2)$$

Ada yang bisa mencoba melanjutkan? Akan mendapatkan poin plus bagi yang bisa ☺

Dimentionality reduction – PCA

$$\underset{\boldsymbol{w}}{argmax}\,(varian\,\boldsymbol{Z}) = \underset{\boldsymbol{w}}{argmax}\,((\boldsymbol{Z} - \bar{\boldsymbol{Z}}\,)^2)$$

$$\underset{\boldsymbol{w}}{argmax}\,(varian\,\boldsymbol{Z}) = \underset{\boldsymbol{w}}{argmax}\,((\boldsymbol{XW} - \bar{\boldsymbol{X}}\boldsymbol{W})^2)$$

$$\underset{\boldsymbol{w}}{argmax}\,(varian\,\boldsymbol{Z}) = \underset{\boldsymbol{w}}{argmax}\left(\left((\boldsymbol{X} - \bar{\boldsymbol{X}})\boldsymbol{W}\right)^2\right)$$

$$\underset{\boldsymbol{w}}{argmax}\,(varian\,\boldsymbol{Z}) = \underset{\boldsymbol{w}}{argmax}\left(\left((\boldsymbol{X} - \bar{\boldsymbol{X}})\boldsymbol{W}\right)^{\boldsymbol{T}}(\boldsymbol{X} - \bar{\boldsymbol{X}})\boldsymbol{W}\right)$$

$$\underset{\boldsymbol{w}}{argmax}\,(varian\,\boldsymbol{Z}) = \underset{\boldsymbol{w}}{argmax}\,(\boldsymbol{W}^T(\boldsymbol{X} - \bar{\boldsymbol{X}})^T(\boldsymbol{X} - \bar{\boldsymbol{X}})\boldsymbol{W})$$

$$\underset{\boldsymbol{w}}{argmax}\,(varian\,\boldsymbol{Z}) = \underset{w}{argmax}\,(\boldsymbol{W}^T\boldsymbol{S}\,\boldsymbol{W})$$

Di mana $\boldsymbol{S} = (\boldsymbol{X} - \bar{\boldsymbol{X}})^T(\boldsymbol{X} - \bar{\boldsymbol{X}})$

Dimentionality reduction – PCA

$$\underset{\boldsymbol{w}}{argmax}\,(varian\ \boldsymbol{Z}) = \underset{w}{argmax}\,(\boldsymbol{W}^T \boldsymbol{S}\, \boldsymbol{W})$$

$$\underset{\boldsymbol{w}}{argmax}\,(varian\ \boldsymbol{Z}) = \frac{d(\boldsymbol{W}^T \boldsymbol{S}\, \boldsymbol{W})}{d\boldsymbol{W}} = 0$$

Untuk mendapatkan solusi $\boldsymbol{W}$ yang *'unique'*, tambahkan *constraint* lagrange multiplier $\boldsymbol{W}^T\boldsymbol{W} = 1$

$$\underset{\boldsymbol{w}}{argmax}\,(varian\ \boldsymbol{Z}) = \frac{d(\boldsymbol{W}^T \boldsymbol{S}\, \boldsymbol{W} - \lambda(\boldsymbol{W}^T\boldsymbol{W} - 1))}{d\boldsymbol{W}} = 0$$

$$\underset{\boldsymbol{w}}{argmax}\,(varian\ \boldsymbol{Z}) = \frac{d(\boldsymbol{W}^T \boldsymbol{S}\, \boldsymbol{W} - \lambda\boldsymbol{W}^T\boldsymbol{W} + \lambda)}{d\boldsymbol{W}} = 0$$

$$\underset{\boldsymbol{w}}{argmax}\,(varian\ \boldsymbol{Z}) = 2\boldsymbol{S}\boldsymbol{W} - \lambda 2\boldsymbol{W} = 0$$

$$2\boldsymbol{S}\boldsymbol{W} = \lambda 2\boldsymbol{W}$$

$$\boldsymbol{S}\boldsymbol{W} = \lambda\boldsymbol{W}$$

Solusi PCA adalah dengan dekomposisi Eigen

Dimentionality reduction – PCA

a. Diberikan data asli $\boldsymbol{X}$, hitung matriks kovarian $\boldsymbol{S}$.

$$\boldsymbol{S} = (\boldsymbol{X} - \overline{\boldsymbol{X}})^T(\boldsymbol{X} - \overline{\boldsymbol{X}})$$

b. Dekomposisikan $\boldsymbol{S}$ ke *eigenvector* dan *eigenvalue.*

c. Jika dikehendaki direduksi ke dimensi p, maka ambil sebanyak p *eigenvector* dari hasil langkah (b) yang memiliki nilai *eigenvalue* p-terbesar. Setiap *eigenvectornya* ini kita susun sebagai matriks kolom, yang selanjutnya akan kita akan gunakan sebagai matriks proyeksi $\boldsymbol{W}$.

d. Data yang terproyeksi dapat dihitung → $\boldsymbol{Z} = \boldsymbol{XW}$

Ingat! ***Z*** ini dimensinya sudah tereduksi dan nilai variansinya paling tinggi.

Dimentionality reduction – PCA

- Hitung $\boldsymbol{S} = (\boldsymbol{X} - \bar{\boldsymbol{X}})^T (\boldsymbol{X} - \bar{\boldsymbol{X}})$

```
mean:
 [185.72 151.12]
matriks kovarians:
 [[2287.04 1268.84]
 [1268.84 1304.64]]
```

- Dekomposisikan ke eigen

```
(eigenvalue, eigenvector) =
np.linalg.eig(matriks_kovarians)
```

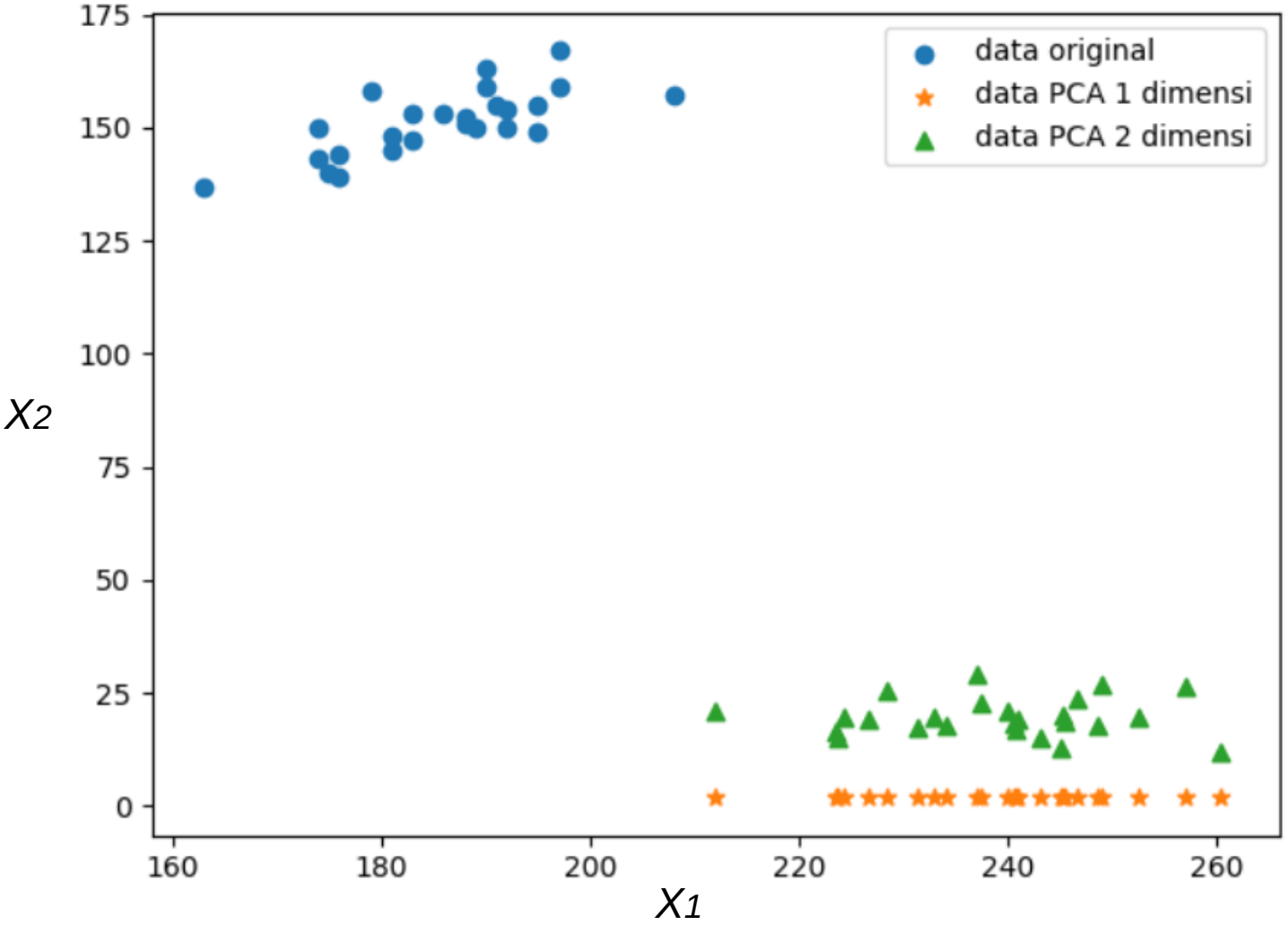
```
eigenvalue:
 [3156.44000941  435.23999059]
eigenvector:
 [[ 0.82492945 -0.5652357 ]
 [ 0.5652357   0.82492945]]
```

- Bangun matriks ***W*** dari eigenvector, dan hitung ***Z,*** contoh:

$$\boldsymbol{Z} = \boldsymbol{XW} = \begin{bmatrix} 191 & 155 \\ 195 & 149 \\ \vdots & \vdots \\ 190 & 163 \end{bmatrix} \begin{bmatrix} 0.825 \\ 0.565 \end{bmatrix} = \begin{bmatrix} 245.15 \\ 245.06 \\ \vdots \\ 248.8 \end{bmatrix}$$

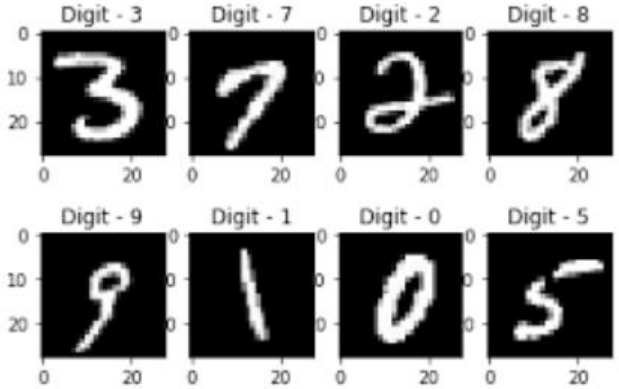
(x_1)	(x_2)
191	155
195	149
181	148
183	153
176	144
208	157
189	150
197	159
188	152
192	150
179	158
183	147
174	150
190	159
188	151
163	137
195	155
186	153
181	145
175	140
192	154
174	143
176	139
197	167
190	163

Dimentionality reduction – PCA



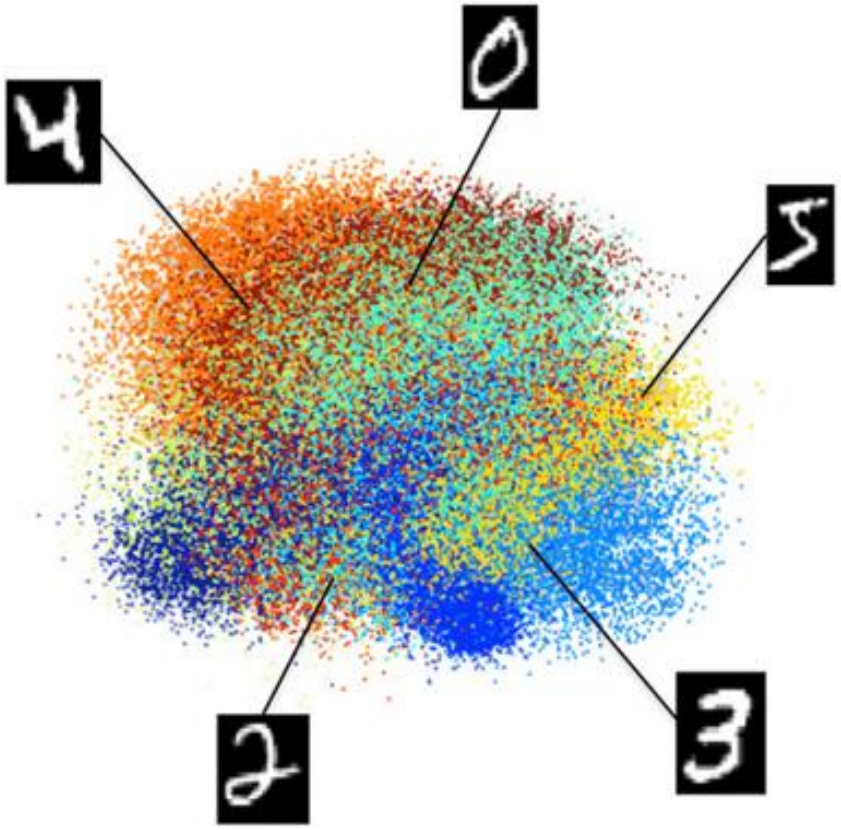
Hasil plot matriks terproyeksi ***Z*** dengan PCA dari data ***X*** (halaman sebelumnya)

Dimentionality reduction – PCA



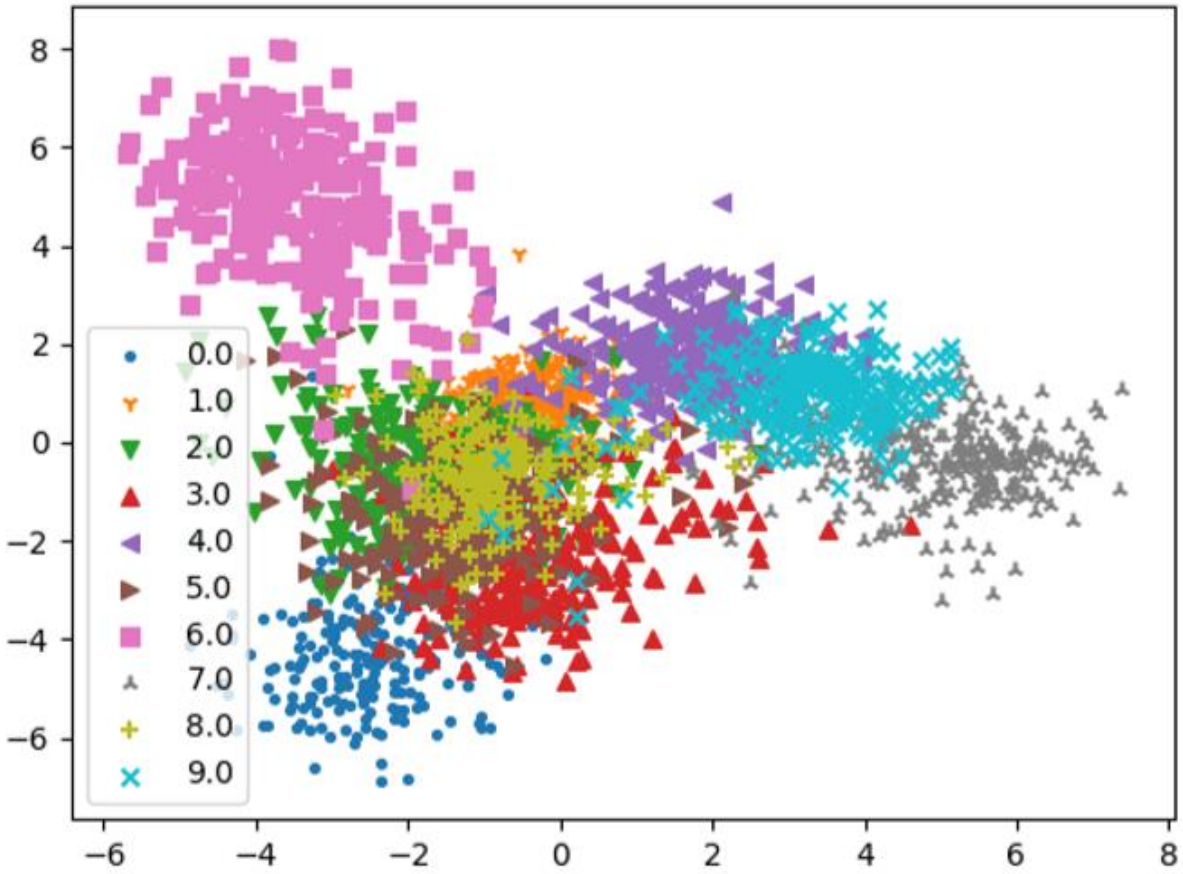
Contoh digit angka MNIST

- Angka = 0 – 9
- Ukuran = 28 x 28 = 784 fitur



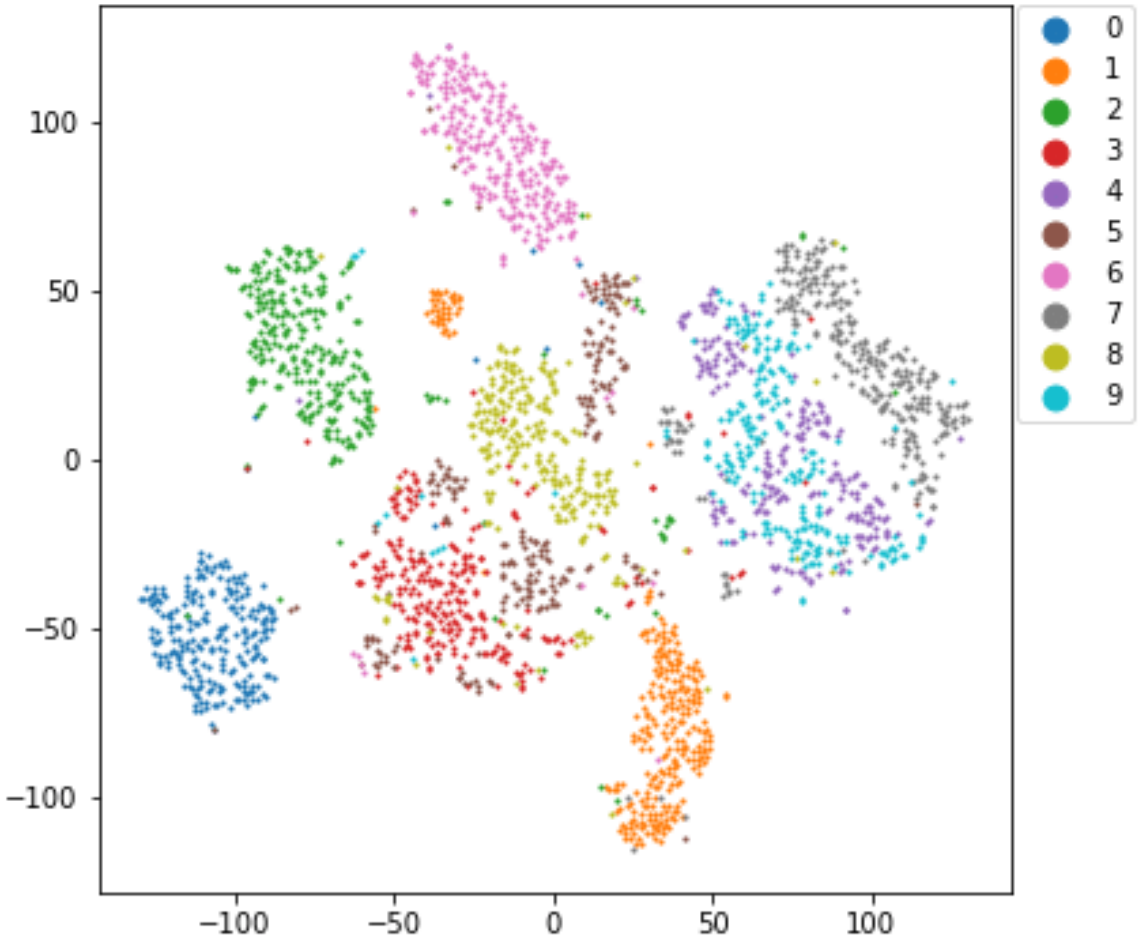
Hasil reduksi ke 2 dimensi dengan PCA

Dimentionality reduction – LDA



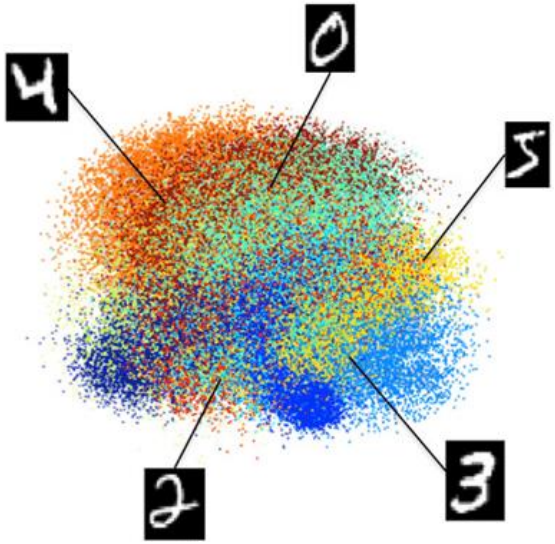
Welling, M. (2005). Fisher linear discriminant analysis. *Department of Computer Science, University of Toronto*.

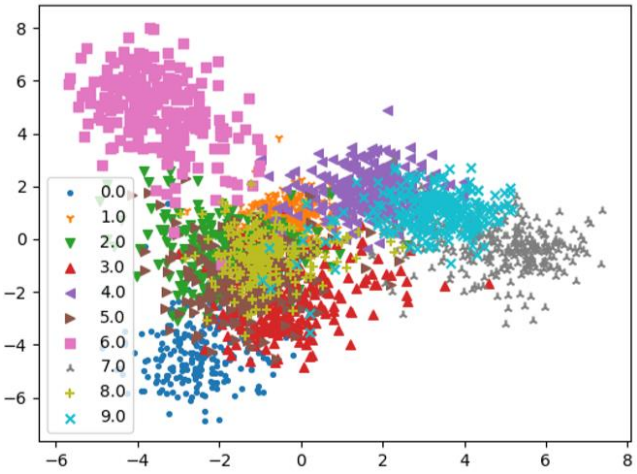
Dimentionality reduction – tSNE

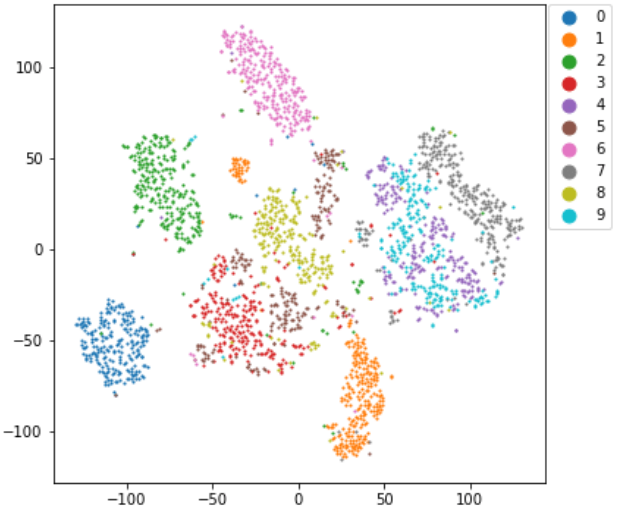


Maaten, L. V. D., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*

Dimentionality reduction – perbandingan







PCA
- Tidak butuh label
- Memaksimalkan variansi

LDA
- Butuh label
- Memaksimalkan *Fisher discriminant*

t-SNE
- Tidak butuh label
- Meminimalkan *Kullback-Leibler divergence / relative entropy*

Yang mana yang paling bagus?

End..

Recommender Systems

Disclaimer

- **The slides mostly taken from:**
  - Jannach & Freidrich (2013), Tutorial Recommendation Systems, International Joint Conference on Artificial Intelligence .
- **Other sources/references:**
  - Charu C. Aggarwal (2015), Data Mining The Textbook, Springer. Chapter 18.5
  - _____(2019), Recommendation Systems Explained, Crossing Mind

Recommendation System - Example

Facebook–"People You May Know"

Netflix–"Other Movies You May Enjoy"

LinkedIn–"Jobs You May Be Interested In"

Amazon–"Customer who bought this item also bought …"

YouTube–"Recommended Videos"

Google–"Search results adjusted"

Pinterest–"Recommended Images"

[Source: Crossing Mind, 2019]

Recommendation Problem

- **User versus Items**
  - Utility Matrix of n users and d items. Matrix D of utility values
  - The Value of: Positive Preferences only vs Positive and Negative Preferences
- **Recommendation System**
  - Content-based Recommendation: users and items are ascociated with feature-based descriptions
  - Collaborative Filtering

	GLADIATOR	GODFATHER	BEN-HUR	GOODFELLAS	SCARFACE	SPARTACUS
U_1	1			5		2
U_2		5			4	
U_3	5	3		1		
U_4			3			4
U_5				3	5	
U_6	5		4			

(a) Ratings-based utility

	GLADIATOR	GODFATHER	BEN-HUR	GOODFELLAS	SCARFACE	SPARTACUS
U_1	1			1		1
U_2		1			1	
U_3	1	1		1		
U_4			1			1
U_5				1	1	
U_6	1		1			

(b) Positive-preference utility

Figure 18.4: Examples of utility matrices.

[Source: Aggarwal, 2015]

Collaborative Filtering (CF)

- **Collaborative Filtering (CF)**
  - Pure CF approaches
  - User-based nearest-neighbor
  - The Pearson Correlation similarity measure
  - Memory-based and model-based approaches
  - Item-based nearest-neighbor
  - The cosine similarity measure
  - Data sparsity problems
  - Recent methods (SVD, Association Rule Mining, Slope One, RF-Rec, …)
  - The Google News personalization engine
  - Discussion and summary
  - Literature

[Source: Jannach & Friedrich, 2013]

Collaborative Filtering (CF)

- **The most prominent approach to generate recommendations**
  - used by large, commercial e-commerce sites
  - well-understood, various algorithms and variations exist
  - applicable in many domains (book, movies, DVDs, ..)
- **Approach**
  - use the "wisdom of the crowd" to recommend items

- **Basic assumption and idea**
  - Users give ratings to catalog items (implicitly or explicitly)
  - Customers who had similar tastes in the past, will have similar tastes in the future

[Source: Jannach & Friedrich, 2013]

Pure CF Approaches

- **Input**
  - Only a matrix of given user–item ratings
- **Output types**
  - A (numerical) prediction indicating to what degree the current user will like or dislike a certain item
  - A top-N list of recommended items

[Source: Jannach & Friedrich, 2013]

User-based nearest-neighbor collaborative filtering (1)

- **The basic technique**
  - Given an "active user" (Alice) and an item i not yet seen by Alice
    - find a set of users (peers/nearest neighbors) who liked the same items as Alice in the past **and** who have rated item i
    - use, e.g. the average of their ratings to predict, if Alice will like item i
    - do this for all items Alice has not seen and recommend the best-rated
- **Basic assumption and idea**
  - If users had similar tastes in the past they will have similar tastes in the future
  - User preferences remain stable and consistent over time

[Source: Jannach & Friedrich, 2013]

User-based nearest-neighbor collaborative filtering (2)

- **Example**
  - A database of ratings of the current user, Alice, and some other users is given:

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

  - Determine whether Alice will like or dislike *Item5*, which Alice has not yet rated or seen

[Source: Jannach & Friedrich, 2013]

User-based nearest-neighbor collaborative filtering (3)

- **Some first questions**
  - How do we measure similarity?
  - How many neighbors should we consider?
  - How do we generate a prediction from the neighbors' ratings?

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

[Source: Jannach & Friedrich, 2013]

Measuring user similarity (1)

- **A popular similarity measure in user-based CF: Pearson correlation**

  a, b : users

  $r_{a,p}$: rating of user a for item p

  P : set of items, rated both by a and b

  – Possible similarity values between -1 and 1

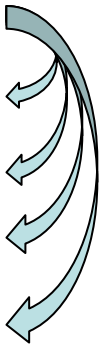
$$\boldsymbol{sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2}\sqrt{\sum_{p \in P}(r_{b,p} - \bar{r}_b)^2}}}$$

[Source: Jannach & Friedrich, 2013]

Measuring user similarity (2)

- **A popular similarity measure in user-based CF: Pearson correlation**
  - a, b : users
  - $r_{a,p}$: rating of user a for item p
  - P : set of items, rated both by a and b
  - Possible similarity values between -1 and 1

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1



sim = 0,85

sim = 0,00

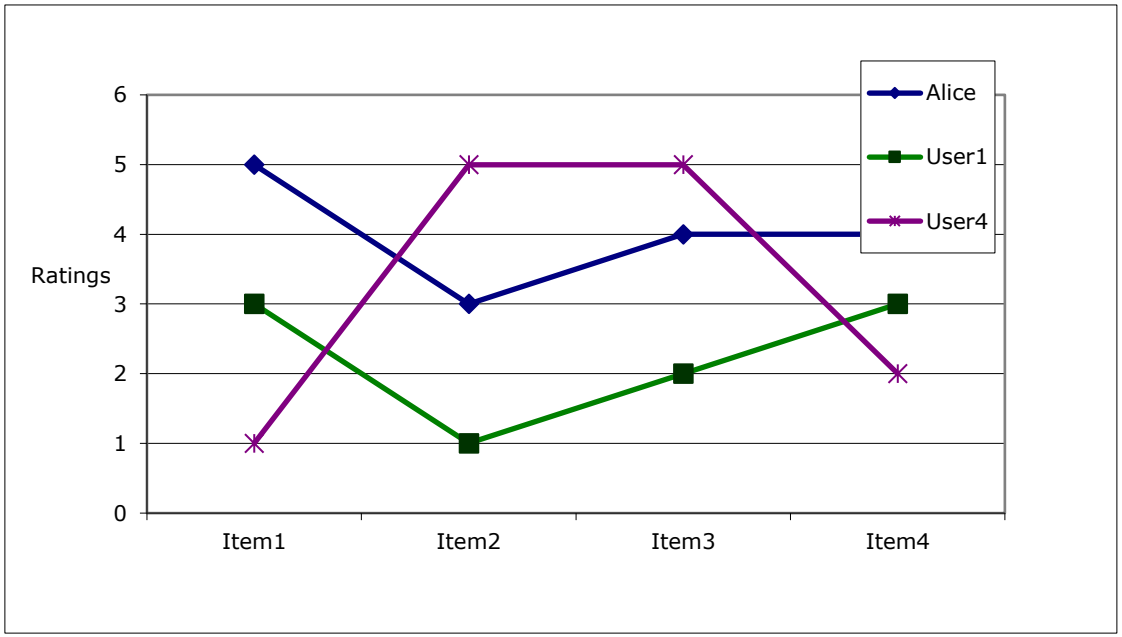
sim = 0,70

sim = -0,79

[Source: Jannach & Friedrich, 2013]

Pearson correlation

- **Takes differences in rating behavior into account**



- **Works well in usual domains, compared with alternative measures**
  - such as cosine similarity

[Source: Jannach & Friedrich, 2013]

Making predictions

- A common prediction function:

$$\boldsymbol{pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a,b)}}$$

- Calculate, whether the neighbors' ratings for the unseen item i are higher or lower than their average
- Combine the rating differences – use the similarity with a as a weight
- Add/subtract the neighbors' bias from the active user's average and use this as a prediction

[Source: Jannach & Friedrich, 2013]

Improving the metrics / prediction function

- **Not all neighbor ratings might be equally "valuable"**
  - Agreement on commonly liked items is not so informative as agreement on controversial items
  - **Possible solution**: Give more weight to items that have a higher variance
- **Value of number of co-rated items**
  - Use "significance weighting", by e.g., linearly reducing the weight when the number of co-rated items is low
- **Case amplification**
  - Intuition: Give more weight to "very similar" neighbors, i.e., where the similarity value is close to 1.
- **Neighborhood selection**
  - Use similarity threshold or fixed number of neighbors

[Source: Jannach & Friedrich, 2013]

Memory-based and model-based approaches

- **User-based CF is said to be "memory-based"**
  - the rating matrix is directly used to find neighbors / make predictions
  - does not scale for most real-world scenarios
  - large e-commerce sites have tens of millions of customers and millions of items
- **Model-based approaches**
  - based on an offline pre-processing or "model-learning" phase
  - at run-time, only the learned model is used to make predictions
  - models are updated / re-trained periodically
  - large variety of techniques used
  - model-building and updating can be computationally expensive
  - *item*-based CF is an example for model-based approaches

[Source: Jannach & Friedrich, 2013]

Item-based collaborative filtering

- **Basic idea:**
  - Use the similarity between items (and not users) to make predictions
- **Example:**
  - Look for items that are similar to Item5
  - Take Alice's ratings for these items to predict the rating for Item5

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

[Source: Jannach & Friedrich, 2013]

The cosine similarity measure

- **Produces better results in item-to-item filtering**
- **Ratings are seen as vector in n-dimensional space**
- **Similarity is calculated based on the angle between the vectors**

$$sim(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}$$

- **Adjusted cosine similarity**
  - take average user ratings into account, transform the original ratings
  - U: set of users who have rated both items a and b

$$sim(\vec{a}, \vec{b}) = \frac{\sum_{u \in U}(r_{u,a} - \overline{r_u})(r_{u,b} - \overline{r_u})}{\sqrt{\sum_{u \in U}(r_{u,a} - \overline{r_u})^2} \sqrt{\sum_{u \in U}(r_{u,b} - \overline{r_u})^2}}$$

[Source: Jannach & Friedrich, 2013]

Making predictions

- A common prediction function:

$$\boldsymbol{pred(u,p)} = \frac{\sum_{i \in \boldsymbol{ratedItem(u)}} \boldsymbol{sim(i,p)} * \boldsymbol{r_{u,i}}}{\sum_{i \in \boldsymbol{ratedItem(u)}} \boldsymbol{sim(i,p)}}$$

- Neighborhood size is typically also limited to a specific size
- Not all neighbors are taken into account for the prediction
- An analysis of the MovieLens dataset indicates that "in most real-world situations, a neighborhood of 20 to 50 neighbors seems reasonable" (Herlocker et al. 2002)

[Source: Jannach & Friedrich, 2013]

Pre-processing for item-based filtering

- **Item-based filtering does not solve the scalability problem itself**
- **Pre-processing approach by Amazon.com (in 2003)**
  - Calculate all pair-wise item similarities in advance
  - The neighborhood to be used at run-time is typically rather small, because only items are taken into account which the user has rated
  - Item similarities are supposed to be more stable than user similarities
- **Memory requirements**
  - Up to N^2 pair-wise similarities to be memorized (N = number of items) in theory
  - In practice, this is significantly lower (items with no co-ratings)
  - Further reductions possible
    - Minimum threshold for co-ratings
    - Limit the neighborhood size (might affect recommendation accuracy)

[Source: Jannach & Friedrich, 2013]

More on ratings – Explicit ratings

- Probably the most precise ratings
- Most commonly used (1 to 5, 1 to 7 Likert response scales)
- Research topics
  - Optimal granularity of scale; indication that 10-point scale is better accepted in movie dom.
  - An even more fine-grained scale was chosen in the joke recommender discussed by Goldberg et al. (2001), where a continuous scale (from −10 to +10) and a graphical input bar were used
    - No precision loss from the discretization
    - User preferences can be captured at a finer granularity
    - Users actually "like" the graphical interaction method
  - Multidimensional ratings (multiple ratings per movie such as ratings for actors and sound)
- Main problems
  - Users not always willing to rate many items
    - number of available ratings could be too small → sparse rating matrices → poor recommendation quality
  - How to stimulate users to rate more items?

[Source: Jannach & Friedrich, 2013]

More on ratings – Implicit ratings

- Typically collected by the web shop or application in which the recommender system is embedded
- When a customer buys an item, for instance, many recommender systems interpret this behavior as a positive rating
- Clicks, page views, time spent on some page, demo downloads …
- Implicit ratings can be collected constantly and do not require additional efforts from the side of the user
- Main problem
  - One cannot be sure whether the user behavior is correctly interpreted
  - For example, a user might not like all the books he or she has bought; the user also might have bought a book for someone else
- Implicit ratings can be used in addition to explicit ones; question of correctness of interpretation

[Source: Jannach & Friedrich, 2013]

Data sparsity problems

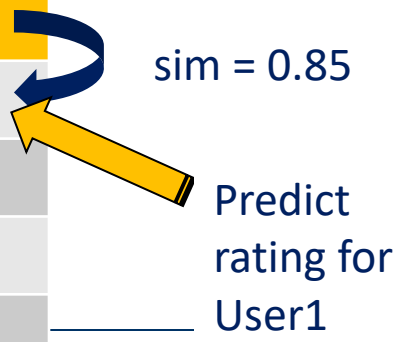
- **Cold start problem**
  - How to recommend new items? What to recommend to new users?
- **Straightforward approaches**
  - Ask/force users to rate a set of items
  - Use another method (e.g., content-based, demographic or simply non-personalized) in the initial phase
  - Default voting: assign default values to items that only one of the two users to be compared has rated (Breese et al. 1998)

[Source: Jannach & Friedrich, 2013]

Example algorithms for sparse datasets

- **Recursive CF** (Zhang and Pu 2007)
  - Assume there is a very close neighbor n of u who however has not rated the target item i yet.
  - Idea:
    - Apply CF-method recursively and predict a rating for item i for the neighbor
    - Use this predicted rating instead of the rating of a more distant direct neighbor

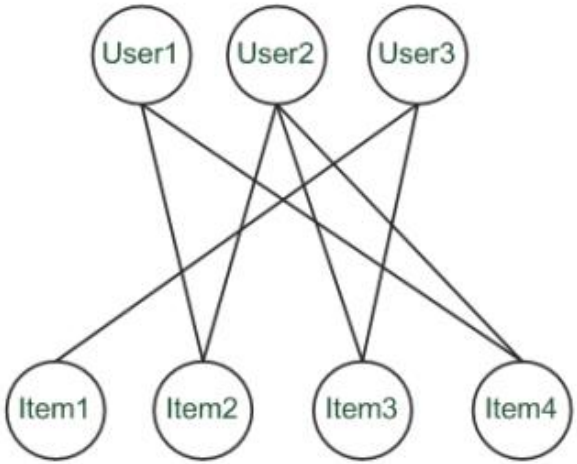
	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	?
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1



[Source: Jannach & Friedrich, 2013]

Graph-based methods (1)

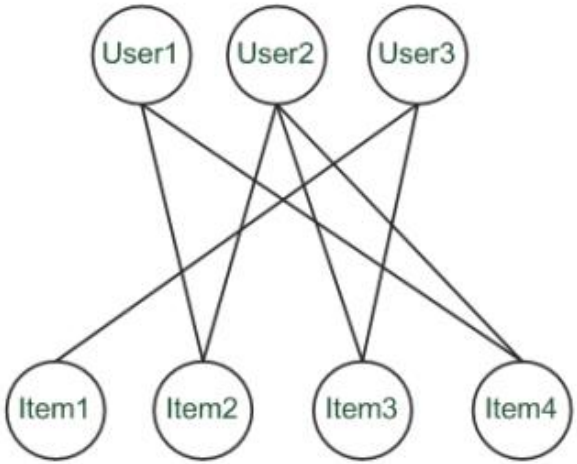
- **"Spreading activation"** (Huang et al. 2004)
  - Exploit the supposed "transitivity" of customer tastes and thereby augment the matrix with additional information
  - Assume that we are looking for a recommendation for *User1*
  - When using a standard CF approach, *User2* will be considered a peer for *User1* because they both bought *Item2* and *Item4*
  - Thus *Item3* will be recommended to *User1* because the nearest neighbor, *User2*, also bought or liked it



[Source: Jannach & Friedrich, 2013]

Graph-based methods (2)

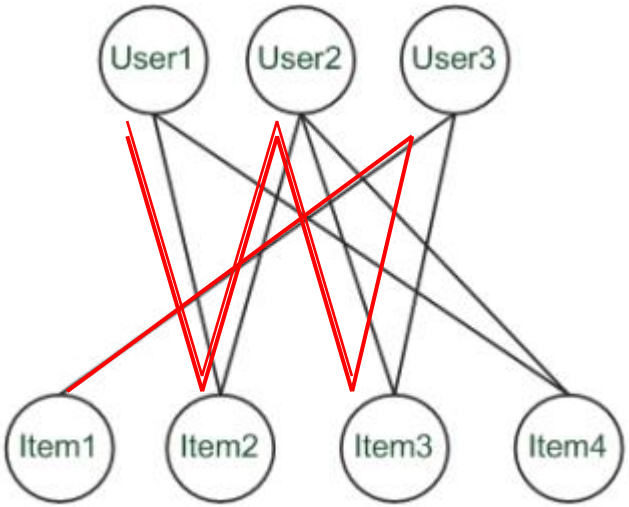
- **"Spreading activation"** (Huang et al. 2004)
  - In a standard user-based or item-based CF approach, paths of length 3 will be considered – that is, *Item3* is relevant for *User1* because there exists a three-step path (*User1–Item2–User2–Item3*) between them
  - Because the number of such paths of length 3 is small in sparse rating databases, the idea is to also consider longer paths (indirect associations) to compute recommendations
  - Using path length 5, for instance



[Source: Jannach & Friedrich, 2013]

Graph-based methods (3)

- **"Spreading activation"** (Huang et al. 2004)
  - Idea: Use paths of lengths > 3 to recommend items
  - Length 3: Recommend Item3 to User1
  - Length 5: Item1 also recommendable



[Source: Jannach & Friedrich, 2013]

More model-based approaches

- **Plethora of different techniques proposed in the last years, e.g.,**
  - Matrix factorization techniques, statistics
    - singular value decomposition, principal component analysis
  - Association rule mining
    - compare: shopping basket analysis
  - Probabilistic models
    - clustering models, Bayesian networks, probabilistic Latent Semantic Analysis
  - Various other machine learning approaches
- **Costs of pre-processing**
  - Usually not discussed
  - Incremental updates possible?

[Source: Jannach & Friedrich, 2013]

Matrix factorization

- Informally, the SVD theorem (Golub and Kahan 1965) states that a given matrix M can be decomposed into a product of three matrices as follows

$$M = U \times \Sigma \times V^T$$

  - where U and V are called *left* and *right singular vectors* and the values of the diagonal of Σ are called the *singular values*
- We can approximate the full matrix by observing only the most important features – those with the largest singular values
- In the example, we calculate U, V, and Σ (with the help of some linear algebra software) but retain only the two most important features by taking only the first two columns of U and V^T

[Source: Jannach & Friedrich, 2013]

Example for SVD-based recommendation

- **SVD:** $M_k = U_k \times \Sigma_k \times V_k^T$

U_k	Dim1	Dim2
Alice	0.47	-0.30
Bob	-0.44	0.23
Mary	0.70	-0.06
Sue	0.31	0.93

V_k^T	Terminator	Die Hard	Twins	Eat Pray Love	Pretty Woman
Dim1	-0.44	-0.57	0.06	0.38	0.57
Dim2	0.58	-0.66	0.26	0.18	-0.36

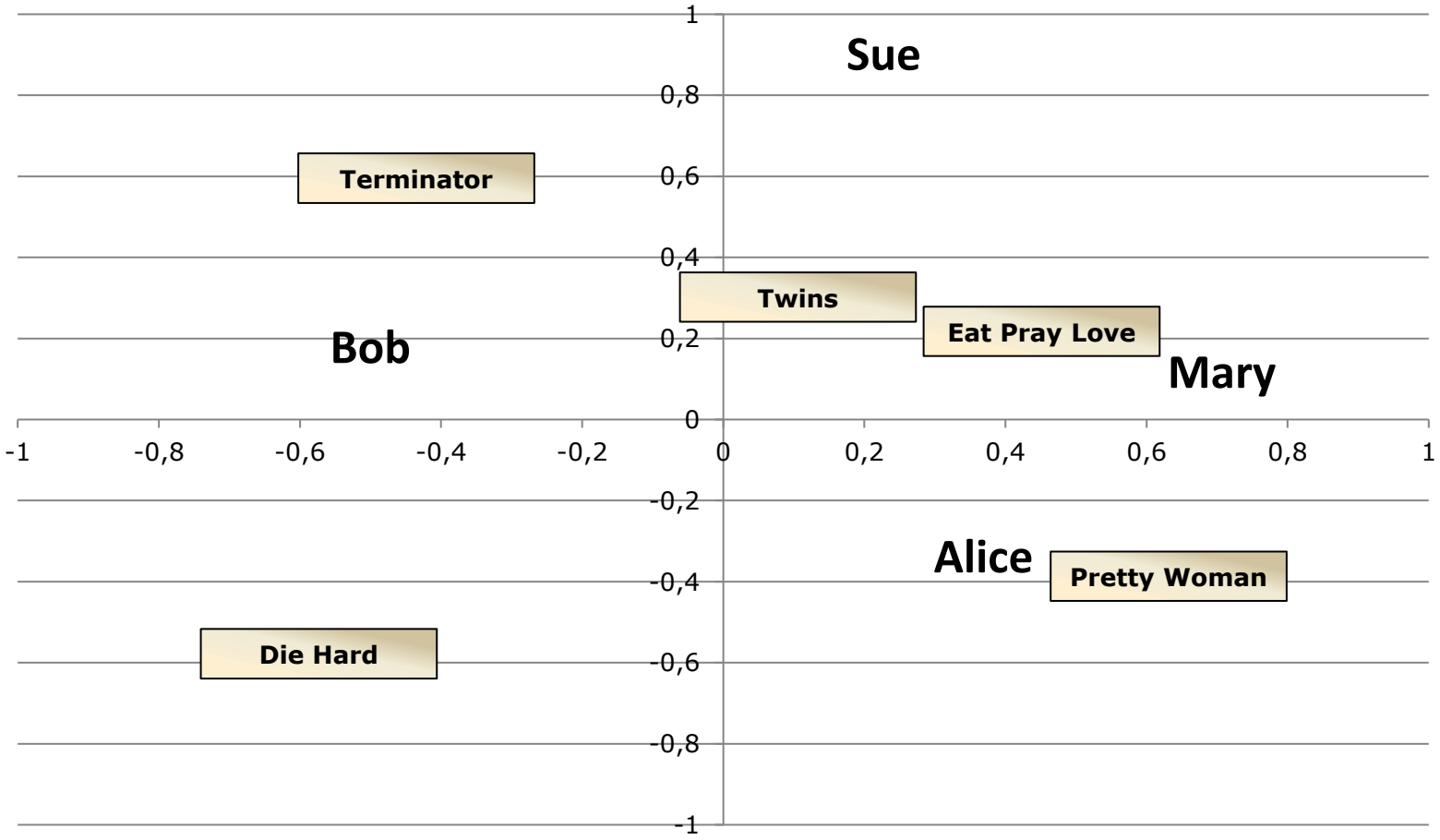
Σ_k	Dim1	Dim2
Dim1	5.63	0
Dim2	0	3.23

- **Prediction:** $\hat{r}_{ui} = \bar{r}_u + U_k(Alice) \times \Sigma_k \times V_k^T(EPL)$

= 3 + 0.84 = 3.84

[Source: Jannach & Friedrich, 2013]

The projection of U and V^T in the 2 dimensional space (U_2, V_2^T)



[Source: Jannach & Friedrich, 2013]

Discussion about dimensionality reduction (Sarwar et al. 2000a)

- **Matrix factorization**
  - Generate low-rank approximation of matrix
  - Detection of latent factors
  - Projecting items and users in the same n-dimensional space
- **Prediction quality can decrease because…**
  - the original ratings are not taken into account
- **Prediction quality can increase as a consequence of…**
  - filtering out some "noise" in the data and
  - detecting nontrivial correlations in the data
- **Depends on the right choice of the amount of data reduction**
  - number of singular values in the SVD approach
  - Parameters can be determined and fine-tuned only based on experiments in a certain domain
  - Koren et al. 2009 talk about 20 to 100 factors that are derived from the rating patterns

[Source: Jannach & Friedrich, 2013]

Association rule mining

- **Commonly used for shopping behavior analysis**
  - aims at detection of rules such as

    "If a customer purchases beer then he also buys diapers in 70% of the cases"

- **Association rule mining algorithms**
  - can detect rules of the form X → Y (e.g., beer → diapers) from a set of sales transactions D = $\{t_1, t_2, \ldots t_n\}$
  - measure of quality: support, confidence
    - used e.g. as a threshold to cut off unimportant rules
  - let $\sigma(X) = \frac{|\{x | x \subseteq ti,\ ti \in D\}|}{|D|}$
  - support = $\frac{\sigma(X \cup Y)}{|D|}$, confidence = $\frac{\sigma(X \cup Y)}{\sigma(X)}$

[Source: Jannach & Friedrich, 2013]

Recommendation based on Association Rule Mining

	Item1	Item2	Item3	Item4	Item5
Alice	1	0	0	0	?
User1	1	0	1	0	1
User2	1	0	1	0	1
User3	0	0	0	1	1
User4	0	1	1	0	0

- **Simplest approach**
  - transform 5-point ratings into binary ratings (1 = above user average)
- **Mine rules such as**
  - Item1 → Item5
    - support (2/4), confidence (2/2) (without Alice)
- **Make recommendations for Alice (basic method)**
  - Determine "relevant" rules based on Alice's transactions (the above rule will be relevant as Alice bought Item1)
  - Determine items not already bought by Alice
  - Sort the items based on the rules' confidence values
- **Different variations possible**
  - dislike statements, user associations ..

[Source: Jannach & Friedrich, 2013]

Probabilistic methods

- **Basic idea (simplistic version for illustration):**
  - given the user/item rating matrix
  - determine the probability that user Alice will like an item i
  - base the recommendation on such these probabilities
- **Calculation of rating probabilities based on Bayes Theorem**
  - How probable is rating value "1" for Item5 given Alice's previous ratings?
  - Corresponds to conditional probability P(Item5=1 | X), where
    - X = Alice's previous ratings = (Item1 =1, Item2=3, Item3= …)
  - Can be estimated based on Bayes' Theorem

$$P(Y|X) = \frac{P(X|Y) \times P(Y)}{P(X)} \qquad P(Y|X) = \frac{\prod_{i=1}^{d} P(X_i|Y) \times P(Y)}{P(X)}$$

  - Assumption: Ratings are independent (?)

[Source: Jannach & Friedrich, 2013]

Calculation of probabilities in simplistic approach

	Item1	Item2	Item3	Item4	Item5
Alice	1	3	3	2	?
User1	2	4	2	2	4
User2	1	3	3	5	1
User3	4	5	2	3	3
User4	1	1	5	2	1

X = (Item1 =1, Item2=3, Item3= …)

$$P(X|Item5 = 1)$$
$$= P(Item1 = 1|Item5 = 1) \times P(Item2 = 3|Item5 = 1)$$
$$\times P(Item3 = 3|Item5 = 1) \times P(Item4 = 2|Item5 = 1) = \frac{2}{2} \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2}$$
$$\approx 0.125$$
$$P(X|Item5 = 2)$$
$$= P(Item1 = 1|Item5 = 2) \times P(Item2 = 3|Item5 = 2)$$
$$\times P(Item3 = 3|Item5 = 2) \times P(Item4 = 2|Item5 = 2) = \frac{0}{0} \times \cdots \times \cdots \times \cdots$$
$$= 0$$

- **More to consider**
  - Zeros (smoothing required)
  - like/dislike simplification possible

[Source: Jannach & Friedrich, 2013]

Practical probabilistic approaches

- **Use a cluster-based approach** (Breese et al. 1998)
  - assume users fall into a small number of subgroups (clusters)
  - Make predictions based on estimates
    - probability of Alice falling into cluster c
    - probability of Alice liking item i given a certain cluster and her previous ratings
    - $P(C = c, v_1, \dots, v_n) = P(C = c) \prod_{i=1}^{n} P(v_i | C = c)$
  - Based on model-based clustering (mixture model)
    - Number of classes and model parameters have to be learned from data in advance (EM algorithm)
- **Others:**
  - Bayesian Networks, Probabilistic Latent Semantic Analysis, ….
- **Empirical analysis shows:**
  - Probabilistic methods lead to relatively good results (movie domain)
  - No consistent winner; small memory-footprint of network model

[Source: Jannach & Friedrich, 2013]

Slope One predictors (Lemire and Maclachlan 2005)

- **Idea of Slope One predictors is simple and is based on a *popularity differential* between items for users**
- **Example:**

	Item1	Item5
Alice	2	?
User1	1	2

- **p(Alice, Item5) = 2 + (2 - 1) = 3**
- **Basic scheme: Take the average of these differences of the co-ratings to make the prediction**
- **In general: Find a function of the form f(x) = x + b**
  - **That is why the name is "Slope One"**

[Source: Jannach & Friedrich, 2013]

RF-Rec predictors (Gedikli et al. 2011)

- **Idea: Take rating frequencies into account for computing a prediction**
- **Basic scheme:** $\hat{r}_{u,i} = \arg\max_{v \in R} f_{user}(u, v) * f_{item}(i, v)$
  - R: Set of all rating values, e.g., $R = \{1,2,3,4,5\}$ on a 5-point rating scale
  - $f_{user}(u, v)$ and $f_{item}(i, v)$ basically describe *how often* a rating v was assigned by user u and to item i resp.
- **Example:**

	Item1	Item2	Item3	Item4	Item5
Alice	1	1	?	5	4
User1	2		5	5	5
User2			1	1	
User3		5	2		2
User4	3		1	1	
User5	1	2	2		4

- **p(Alice, Item3) = 1**

[Source: Jannach & Friedrich, 2013]

2008: *Factorization meets the neighborhood: a multifaceted collaborative filtering model*, Y. Koren, ACM SIGKDD

- Stimulated by work on Netflix competition
  - Prize of $1,000,000 for accuracy improvement of 10% RMSE compared to own Cinematch system
  - Very large dataset (~100M ratings, ~480K users , ~18K movies)
  - Last ratings/user withheld (set K)
- Root mean squared error metric optimized to 0.8567
- Metrics measure error rate
  - Mean Absolute Error (*MAE*) computes the deviation between predicted ratings and actual ratings

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|p_i - r_i|$$

  - Root Mean Square Error (*RMSE*) is similar to *MAE*, but places more emphasis on larger deviation

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - r_i)^2}$$

[Source: Jannach & Friedrich, 2013]

Collaborative Filtering Issues

- **Pros:** 
  - well-understood, works well in some domains, no knowledge engineering required
- **Cons:** 
  - requires user community, sparsity problems, no integration of other knowledge sources, no explanation of results
- **What is the best CF method?**
  - In which situation and which domain? Inconsistent findings; always the same domains and data sets; differences between methods are often very small (1/100)
- **How to evaluate the prediction quality?**
  - MAE / RMSE: What does an MAE of 0.7 actually mean?
  - Serendipity (novelty and surprising effect of recommendations)
    - Not yet fully understood
- **What about multi-dimensional ratings?**

[Source: Jannach & Friedrich, 2013]

The Google News personalization engine

Google News

Search News | Search the Web

Search and browse 4,500 news sources updated continuously.

News archive search | Advanced news search | Blog search

>Top Stories
Recommended
U.S.
Business
Elections
World
Entertainment
Sci/Tech
Health
Sports
Most Popular

News Alerts
Text Version
Standard Version
Image Version
RSS | Atom
About Feeds

Top Stories Personalized News Go

Auto-generated **13 minutes ago**

Tibet's Communist Party Leader Denounces Exiled Dalai Lama

KOMO

Voice of America - **43 minutes ago**

By VOA News The head of Tibet's Communist Party has warned of a "life and death struggle" with the Dalai Lama, as China struggles to bring an end to several days of protests in the Himalayan region.

Dalai Lama threatens to resign Los Angeles Times

Comment by Jamie Metzl Executive Vice President, Asia Society

BBC News - Forbes - Reuters - Washington Post

all 5,998 news articles »

Forex - Dollar resumes weak trend on expectations Fed to cut rates ...

MSN UK News

CNNMoney.com - **2 hours ago**

HONG KONG, Mar. 19, 2008 (Thomson Financial delivered by Newstex) -- The dollar resumed its weak tone against other key currencies in afternoon Asian trade on Wednesday as investors bet the Federal Reserve will further cut interest rates to lift the ...

Commentary by John M. Berry Bloomberg

Stocks soar after Federal Reserve trims rate Houston Chronicle

Los Angeles Times - New York Times - Sacramento Bee - Financial Times

all 805 news articles »

Edit this personalized page

Fed cuts key interest rate
Los Angeles Times - all 510 news articles »

Obama on race
Los Angeles Times - all 200 news articles »

US, Russia Politely Dug In Over Missile Defense
Washington Post - all 1,096 news articles »

Sci-fi guru Sir Arthur C. Clarke dies
Vancouver Sun - all 976 news articles »

Facebook Beefs Up Privacy Options, Readies Online Chat
Washington Post - all 297 news articles »

Mills' Money Can't Buy Her Love
E! Online - all 3,490 news articles »

Boeing confident of winning back tanker deal
Reuters - all 200 news articles »

In The News

Dalai Lama
Barack Obama
Windows Vista
Halle Berry

[Source: Jannach & Friedrich, 2013]

Google News portal (1)

- **Aggregates news articles from several thousand sources**
- **Displays them to signed-in users in a personalized way**
- **Collaborative recommendation approach based on**
  - the click history of the active user and
  - the history of the larger community
- **Main challenges**
  - Vast number of articles and users
  - Generate recommendation list in real time (at most one second)
  - Constant stream of new items
  - Immediately react to user interaction
- **Significant efforts with respect to algorithms, engineering, and parallelization are required**

[Source: Jannach & Friedrich, 2013]

Google News portal (2)

- **Pure memory-based approaches are not directly applicable and for model-based approaches, the problem of continuous model updates must be solved**
- **A combination of model- and memory-based techniques is used**
- **Model-based part: Two clustering techniques are used**
  - Probabilistic Latent Semantic Indexing (PLSI) as proposed by (Hofmann 2004)
  - MinHash as a hashing method
- **Memory-based part: Analyze story *co-visits* for dealing with new users**
- **Google's MapReduce technique is used for parallelization in order to make computation scalable**

[Source: Jannach & Friedrich, 2013]

Literature (1)

- **[Adomavicius and Tuzhilin 2005]** Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, IEEE Transactions on Knowledge and Data Engineering 17 (2005), no. 6, 734–749
- **[Breese et al. 1998]** Empirical analysis of predictive algorithms for collaborative filtering, Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (Madison, WI) (Gregory F. Cooper and Seraf´in Moral, eds.), Morgan Kaufmann, 1998, pp. 43–52
- **[Gedikli et al. 2011]** RF-Rec: Fast and accurate computation of recommendations based on rating frequencies, Proceedings of the 13th IEEE Conference on Commerce and Enterprise Computing - CEC 2011, Luxembourg, 2011, forthcoming
- **[Goldberg et al. 2001]** Eigentaste: A constant time collaborative filtering algorithm, Information Retrieval **4** (2001), no. 2, 133–151
- **[Golub and Kahan 1965]** Calculating the singular values and pseudo-inverse of a matrix, Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis **2** (1965), no. 2, 205–224
- **[Herlocker et al. 2002]** An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms, Information Retrieval 5 (2002), no. 4, 287–310
- **[Herlocker et al. 2004]** Evaluating collaborative filtering recommender systems, ACM Transactions on Information Systems (TOIS) **22** (2004), no. 1, 5–53

[Source: Jannach & Friedrich, 2013]

Literature (2)

- **[Hofmann 2004]** Latent semantic models for collaborative filtering, ACM Transactions on Information Systems 22 (2004), no. 1, 89–115
- **[Huang et al. 2004]** Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering, ACM Transactions on Information Systems 22 (2004), no. 1, 116–142
- **[Koren et al. 2009]** *Matrix factorization techniques for recommender systems*, Computer **42** (2009), no. 8, 30–37
- **[Lemire and Maclachlan 2005]** Slope one predictors for online rating-based collaborative filtering, Proceedings of the 5th SIAM International Conference on Data Mining (SDM '05) (Newport Beach, CA), 2005, pp. 471–480
- **[Sarwar et al. 2000a]** Application of dimensionality reduction in recommender systems – a case study, Proceedings of the ACM WebKDD Workshop (Boston), 2000
- **[Zhang and Pu 2007]** A recursive prediction algorithm for collaborative filtering recommender systems, Proceedings of the 2007 ACM Conference on Recommender Systems (RecSys '07) (Minneapolis, MN), ACM, 2007, pp. 57–64

[Source: Jannach & Friedrich, 2013]

Regresi

Ikhtisar

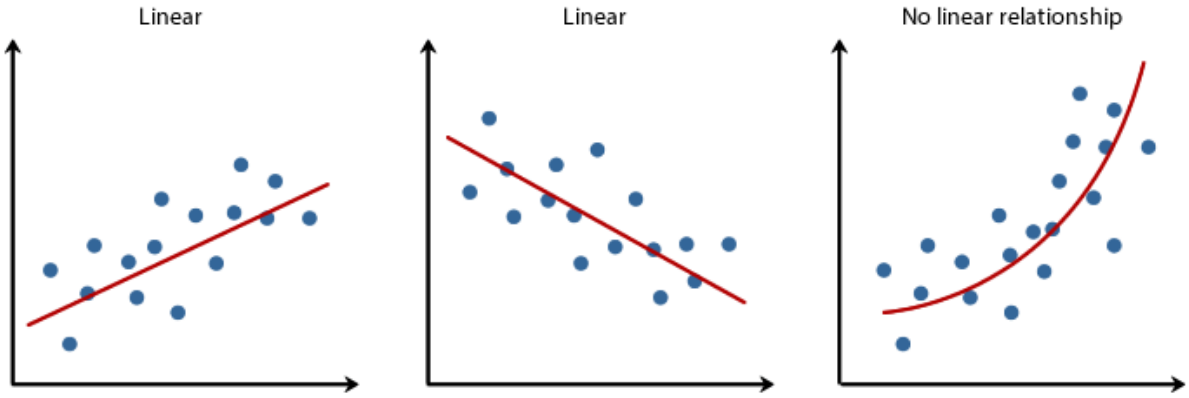
- Pengantar
- Regresi Linear
- Regresi Non-linear (Polinomial)
- Regresi dengan Regularisasi
- Regresi menggunakan ANN

Pengantar

Pengantar

- **Pengertian**: teknik pembelajaran yang fokus pada pada hubungan antara variabel independen (prediktor) dan variabel dependen (output), yang mana variabel dependennya berupa nilai **kontinyu**.

- **Contoh**:
  - ✓Memprediksi harga sebuah produk berdasarkan atribut-atribut yang diberikan
  - ✓Memprediksi *bounding box* pada deteksi objek
  - ✓Memprediksi keterlambatan sebuah pesawat



Overview regresi

- Membutuhkan label
- Outputnya **kontinyu**
- Contoh kasus:
  - ✓Memprediksi harga rumah, dengan diberikan **fitur** seperti: luas rumah, tingkat kriminalitas, akses ke transportasi umur, dsb.
  - ✓Contoh data: data '*housing*' dari UC Irvine

0.00632	18.00	2.310	0	0.5380	6.5750	65.20	4.0900	1	296.0	15.30	396.90	4.98	24.00
0.02731	0.00	7.070	0	0.4690	6.4210	78.90	4.9671	2	242.0	17.80	396.90	9.14	21.60
0.02729	0.00	7.070	0	0.4690	7.1850	61.10	4.9671	2	242.0	17.80	392.83	4.03	34.70
0.03237	0.00	2.180	0	0.4580	6.9980	45.80	6.0622	3	222.0	18.70	394.63	2.94	33.40
0.06905	0.00	2.180	0	0.4580	7.1470	54.20	6.0622	3	222.0	18.70	396.90	5.33	36.20
0.02985	0.00	2.180	0	0.4580	6.4300	58.70	6.0622	3	222.0	18.70	394.12	5.21	28.70
0.08829	12.50	7.870	0	0.5240	6.0120	66.60	5.5605	5	311.0	15.20	395.60	12.43	22.90

Fitur/atribut

Label/ground truth = harga rumah (unit = $1k)

Regresi Linear

Regresi Linear

- Regresi linear dapat kita modelkan

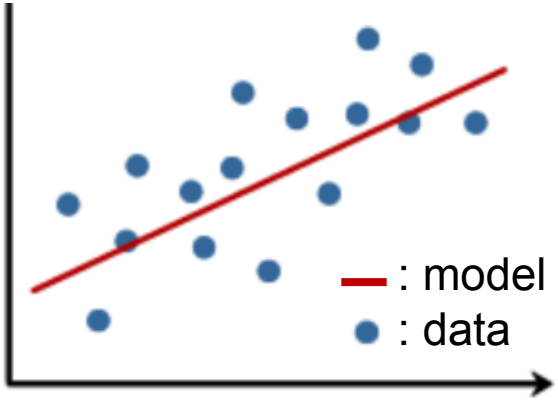
$$y(\boldsymbol{x}) = a_0 + a_1 x_1 + a_2 x_2 + \cdots + a_n x_n$$

di mana:

a_i = koefisien regresi

x_i = atribut / fitur / var. independen

y = output prediksi / var. dependen



Dalam notasi matriks, dapat kita tuliskan:

$$y(\boldsymbol{x}) = \boldsymbol{a}^T \boldsymbol{x}, \text{ dengan } \boldsymbol{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} \text{ dan } \boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

Untuk pasangan input-ouput sebanyak m data, dapat kita tuliskan:

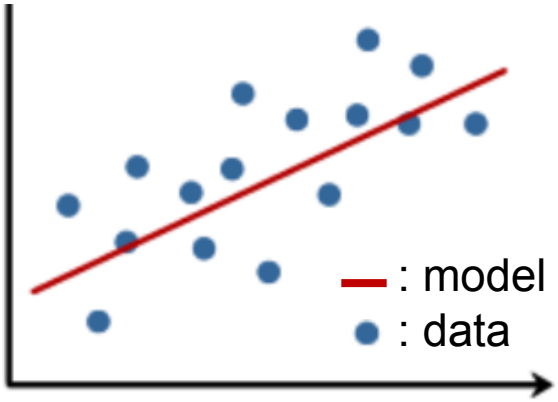
$$\begin{bmatrix} y(x_1) \\ y(x_2) \\ \vdots \\ y(x_m) \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1n} \\ 1 & x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \Leftrightarrow y(\boldsymbol{x}) = \boldsymbol{X}\boldsymbol{a}$$

Disebut sebagai matriks desain (*design matrix*)

Regresi Linear

- Untuk melatih model regresi, dapat kita lakukan dengan **meminimalkan** *loss function:*

$$f(\boldsymbol{a}) = \sum_{i=1}^{m} (y(x_i) - \bar{y}_i)^2$$



- Atau dalam notasi matriksnya:

$$f(\boldsymbol{a}) = (\boldsymbol{y}(\boldsymbol{x}) - \bar{\boldsymbol{y}})^2 = (\boldsymbol{X}\boldsymbol{a} - \bar{\boldsymbol{y}})^2,$$

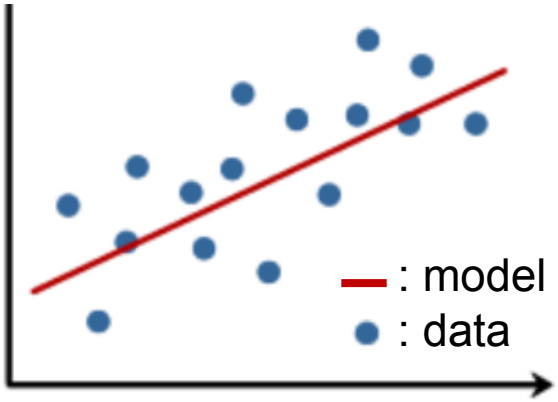
Ingat → $y(\boldsymbol{x}) = \boldsymbol{X}\boldsymbol{a} \Leftrightarrow \begin{bmatrix} y(x_1) \\ y(x_2) \\ \vdots \\ y(x_m) \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} \\ 1 & x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$

- Untuk mencari $\boldsymbol{a}$ yang meminimalkan *loss function* $f(\boldsymbol{a})$, dapat dilakukan dengan $\frac{d(f(\boldsymbol{a}))}{d\boldsymbol{a}} = 0$

Regresi Linear

- Untuk melatih model regresi, dapat kita lakukan dengan **meminimalkan** *loss function:*

$$f(\boldsymbol{a}) = \sum_{i=1}^{m} (y(x_i) - \bar{y}_i)^2$$



- Atau dalam notasi matriksnya:

$$f(\boldsymbol{a}) = (y(\boldsymbol{x}) - \bar{y}_i)^2 = (\boldsymbol{Xa} - \bar{\boldsymbol{y}})^2 = (\boldsymbol{Xa})^2 - 2(\boldsymbol{Xa})^T\bar{\boldsymbol{y}} + \bar{\boldsymbol{y}}^2$$

- Untuk mencari $\boldsymbol{a}$ yang meminimalkan *loss function* $f(\boldsymbol{a})$, dapat dilakukan dengan $\frac{d(f(\boldsymbol{a}))}{d\boldsymbol{a}} = 0$

$$\frac{d(f(\boldsymbol{a}))}{\boldsymbol{da}} = \frac{d((\boldsymbol{Xa})^2 - 2(\boldsymbol{Xa})^T\bar{\boldsymbol{y}} + \bar{\boldsymbol{y}}^2)}{\boldsymbol{a}} = 0$$

$$0 = 2\boldsymbol{X}^T\boldsymbol{Xa} - 2\boldsymbol{X}^T\bar{\boldsymbol{y}}$$

$$\boldsymbol{X}^T\boldsymbol{Xa} = \boldsymbol{X}^T\bar{\boldsymbol{y}} \Leftrightarrow \boxed{\boldsymbol{a} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\bar{\boldsymbol{y}}}$$

Dengan:

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} \\ 1 & x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix}; \bar{\boldsymbol{y}} = \begin{bmatrix} \bar{y}_1 \\ \bar{y}_2 \\ \vdots \\ \bar{y}_m \end{bmatrix}$$

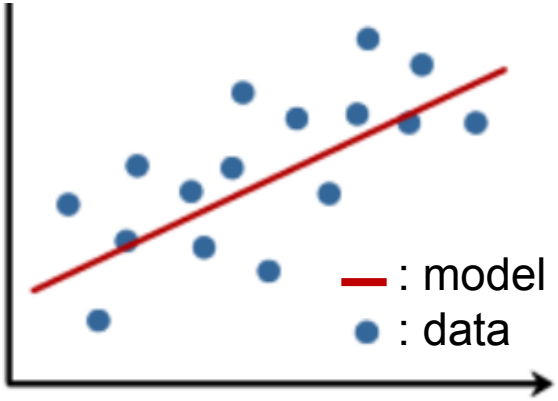
Regresi Linear

- Kita dapatkan koefisien regresi:

$$\boldsymbol{a} = \left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-\mathbf{1}}\boldsymbol{X}^{\boldsymbol{T}}\bar{\boldsymbol{y}}$$

- Maka prediksi regresinya adalah:

$$y(\boldsymbol{x}) = \boldsymbol{X}\boldsymbol{a}$$



Dengan:

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} \\ 1 & x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix}; \bar{\boldsymbol{y}} = \begin{bmatrix} \bar{y}_1 \\ \bar{y}_2 \\ \vdots \\ \bar{y}_m \end{bmatrix}$$

Regresi Linear

- Diberikan data, cari model regresinya!

Observasi	Temperatur ($x_1, satuan = C$)	Kec. katalis ($x_2, satuan = kg/jam$)	Viskositas
1	80	8	2256
2	93	9	2340
3	100	10	2426
4	82	12	2293
5	90	11	2330
6	99	8	2368
7	81	8	2368
8	96	10	2250
9	94	12	2409
10	93	11	2364
11	97	13	2440
12	95	11	2364
13	100	8	2404
14	85	12	2317
15	86	9	2309
16	87	12	2328

$$\boldsymbol{X}^T\boldsymbol{X} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 80 & 93 & \cdots & 87 \\ 8 & 9 & \cdots & 12 \end{bmatrix} \begin{bmatrix} 1 & 80 & 8 \\ 1 & 93 & 9 \\ \vdots & \vdots & \vdots \\ 1 & 87 & 12 \end{bmatrix}$$

$$\boldsymbol{X}^T\boldsymbol{X} = \begin{bmatrix} 16 & 1458 & 164 \\ 1458 & 133560 & 14946 \\ 164 & 14946 & 1726 \end{bmatrix}$$

$$\boldsymbol{X}^T\bar{\boldsymbol{y}} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 80 & 93 & \cdots & 87 \\ 8 & 9 & \cdots & 12 \end{bmatrix} \begin{bmatrix} 2256 \\ 2340 \\ \vdots \\ 2328 \end{bmatrix}$$

$$\boldsymbol{a} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\bar{\boldsymbol{y}} = \begin{bmatrix} 1566{,}08 \\ 7{,}62 \\ 8{,}58 \end{bmatrix}$$

- Jadi model regresinya:

$$y = 1566{,}08 + 7{,}62x_1 + 8{,}58x_2$$

Regresi Non-linear

Regresi Polinomial (Non-linear)

- Regresi non-linear dapat kita modelkan

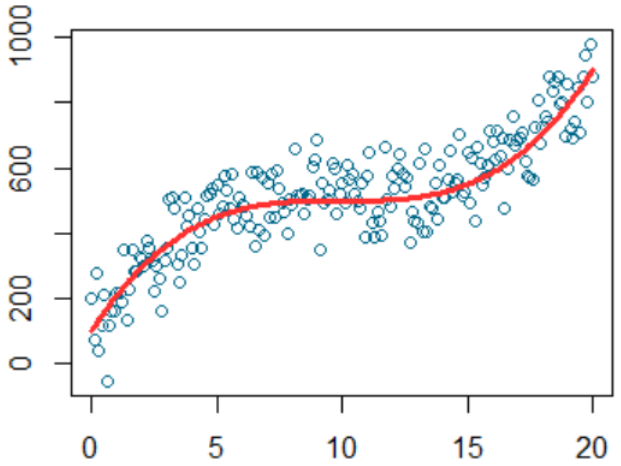
$$y(\boldsymbol{x}) = a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_n x_1^n$$

di mana:

a_i = koefisien regresi

x_i = atribut / fitur / var. independen

y = output prediksi / var. dependen



Dalam notasi matriks, dapat kita tuliskan:

$$y(\boldsymbol{x}) = \boldsymbol{a}^T \boldsymbol{x}, \text{ dengan } \boldsymbol{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} \text{ dan } \boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n^n \end{bmatrix}$$

Untuk pasangan input-ouput sebanyak m data, dapat kita tuliskan:

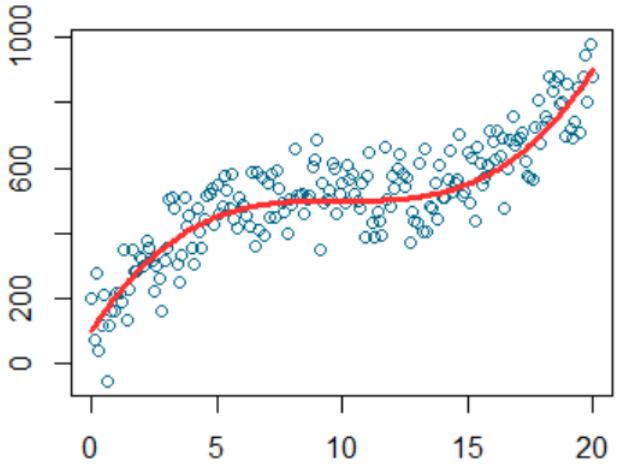
$$\begin{bmatrix} y(x_1) \\ y(x_2) \\ \vdots \\ y(x_m) \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{11}^2 & \dots & x_{11}^n \\ 1 & x_{21} & x_{21}^2 & \dots & x_{21}^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & x_{m1}^2 & \dots & x_{m1}^n \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \Leftrightarrow y(\boldsymbol{x}) = \boldsymbol{X}\boldsymbol{a}$$

Disebut sebagai matriks desain (*design matrix*)

Regresi Polinomial (Non-linear)

- Untuk melatih model regresi, dapat kita lakukan dengan **meminimalkan** *loss function:*

$$f(\boldsymbol{a}) = \sum_{i=1}^{m} (y(x_i) - \bar{y}_i)^2$$



- Atau dalam notasi matriksnya:

$$f(\boldsymbol{a}) = (y(\boldsymbol{x}) - \bar{y}_i)^2 = (\boldsymbol{Xa} - \bar{\boldsymbol{y}})^2 = (\boldsymbol{Xa})^2 - 2(\boldsymbol{Xa})^T\bar{\boldsymbol{y}} + \bar{\boldsymbol{y}}^2$$

- Untuk mencari $\boldsymbol{a}$ yang meminimalkan *loss function* $f(\boldsymbol{a})$, dapat dilakukan dengan $\frac{d(f(\boldsymbol{a}))}{\boldsymbol{a}} = 0$

$$\frac{d(f(\boldsymbol{a}))}{\boldsymbol{a}} = \frac{d((\boldsymbol{Xa})^2 - 2(\boldsymbol{Xa})^T\bar{\boldsymbol{y}} + \bar{\boldsymbol{y}}^2)}{\boldsymbol{a}} = 0$$

$$0 = 2\boldsymbol{X}^T\boldsymbol{Xa} - 2\boldsymbol{X}^T\bar{\boldsymbol{y}}$$

$$\boldsymbol{X}^T\boldsymbol{Xa} = \boldsymbol{X}^T\bar{\boldsymbol{y}} \Leftrightarrow \boxed{\boldsymbol{a} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\bar{\boldsymbol{y}}}$$

Dengan:

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_{11} & x_{11}^2 & \dots & x_{11}^n \\ 1 & x_{21} & x_{21}^2 & \dots & x_{21}^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & x_{m1}^2 & \dots & x_{m1}^n \end{bmatrix}; \bar{\boldsymbol{y}} = \begin{bmatrix} \bar{y}_1 \\ \bar{y}_2 \\ \vdots \\ \bar{y}_m \end{bmatrix}$$

Perbandingan Regresi Linear dan Non-linear

- Formula koefisien regresi linear:

$$\boldsymbol{a} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\bar{\boldsymbol{y}}$$

Dengan:

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} \\ 1 & x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix}; \bar{\boldsymbol{y}} = \begin{bmatrix} \bar{y}_1 \\ \bar{y}_2 \\ \vdots \\ \bar{y}_m \end{bmatrix}$$

- Formula koefisien regresi non-linear:

$$\boldsymbol{a} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\bar{\boldsymbol{y}}$$

Dengan:

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_{11} & x_{11}^2 & \dots & x_{11}^n \\ 1 & x_{21} & x_{21}^2 & \dots & x_{21}^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & x_{m1}^2 & \dots & x_{m1}^n \end{bmatrix}; \bar{\boldsymbol{y}} = \begin{bmatrix} \bar{y}_1 \\ \bar{y}_2 \\ \vdots \\ \bar{y}_m \end{bmatrix}$$

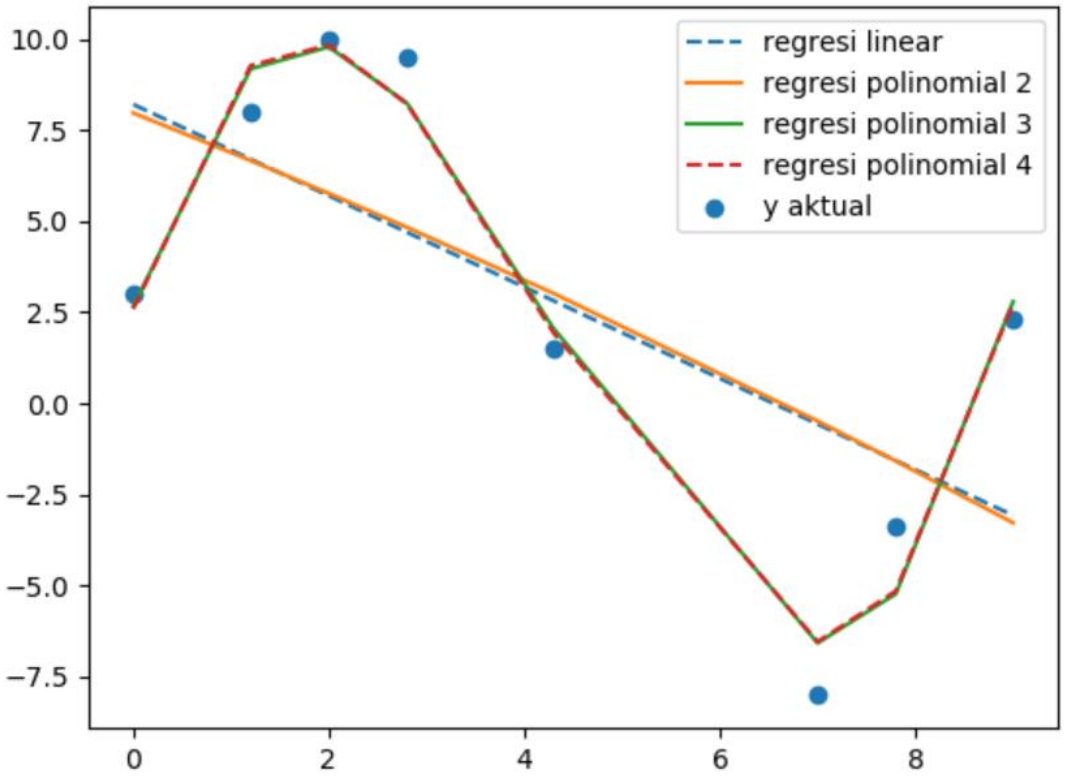
- Apa bedanya?

Latihan

Diberikan data berikut, temukan: (i) model regresi linear dan (ii) regresi polinomial orde 2!

Input data x	Output data Y
0	3
1,2	8
2	10
2,9	9,5
4,3	1,5
7	-8
7,8	-3,4
9,2	2,3

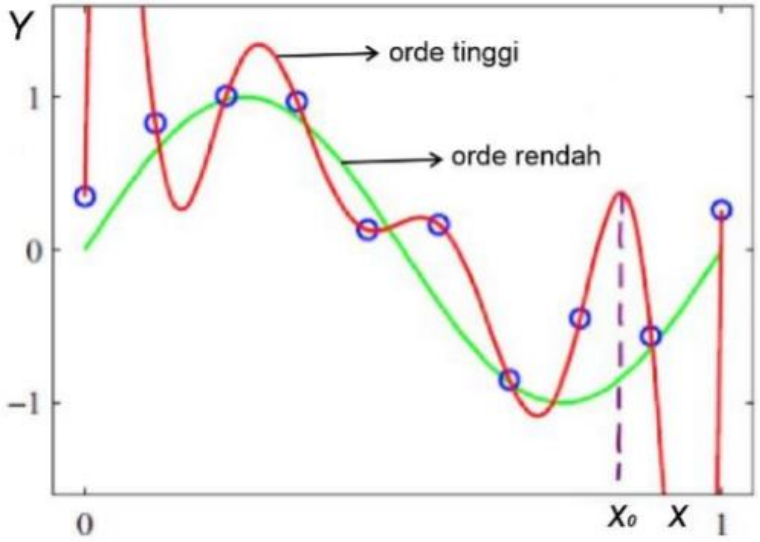
Plot model regresi



Regresi dengan Regularisasi

Regresi dengan Regularisasi

- Di regresi polinomial, semakin tinggi ordernya, model semakin dapat "*fit*" terhadap datanya
- Ingat, hal tersebut dapat menyebabkan *overfitting*.



- Untuk menghindari *overfitting* → regularisasi
  - ✓ **Pendekatan**: koefisien regresi yang kecil, lebih aman terhadap *overfitting*.

Regresi dengan Regularisasi

- Untuk menghindari *overfitting* → regularisasi
  - ✓ **Pendekatan**: koefisien regresi yang kecil, lebih aman terhadap *overfitting*.

Loss function → $f(\boldsymbol{a}) = \sum_{i=1}^{m}(y_i - \bar{y}_i)^2 + \lambda \sum_{j=1}^{n}(a_j^2)$

Untuk mendapatkan koefisien regresi yang kecil

  - ✓ Semakin tinggi λ, semakin "teregularisasi" (model semakin general)

- Formula akhir untuk mendapatan koefisien regresi $\boldsymbol{a}$:

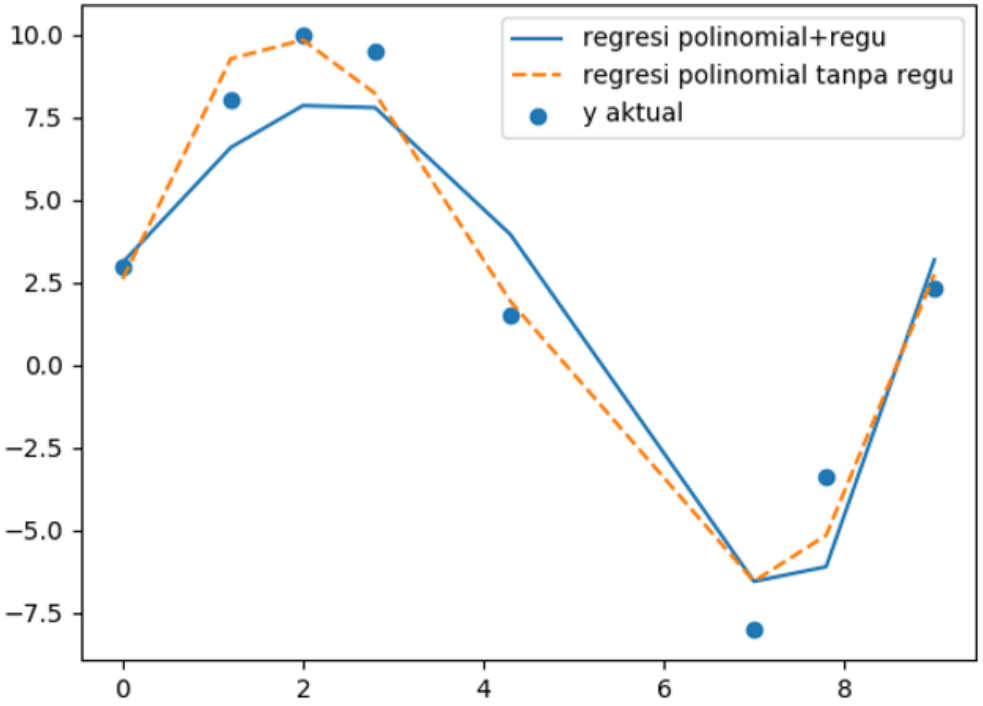
$$\boldsymbol{a} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\bar{\boldsymbol{y}}$$

Dengan $\boldsymbol{I}$ adalah matriks identitas berukuran $(n+1, n+1)$
n = orde polinomial

$$\boldsymbol{I} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

Regresi dengan Regularisasi

Input data x	Output data Y
0	3
1,2	8
2	10
2,9	9,5
4,3	1,5
7	-8
7,8	-3,4
9,2	2,3



Perbandingan Regresi:
(i) Linear, (ii) Non-linear, (iii) dengan regularisasi

- Formula koefisien regresi linear:

$$\boldsymbol{a} = \left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\boldsymbol{T}}\bar{\boldsymbol{y}}$$

Dengan:

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1n} \\ 1 & x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}; \bar{\boldsymbol{y}} = \begin{bmatrix} \bar{y}_1 \\ \bar{y}_2 \\ \vdots \\ \bar{y}_m \end{bmatrix}$$

- Formula koefisien regresi non-linear:

$$\boldsymbol{a} = \left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\boldsymbol{T}}\bar{\boldsymbol{y}}$$

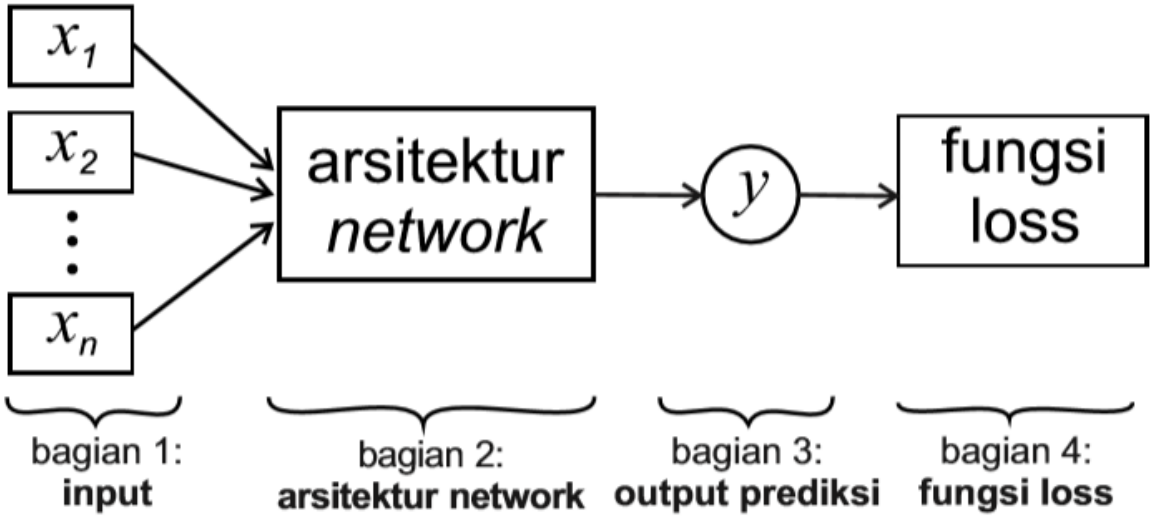
Dengan:

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_{11} & x_{11}^2 & \cdots & x_{11}^n \\ 1 & x_{21} & x_{21}^2 & \cdots & x_{21}^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & x_{m1}^2 & \cdots & x_{m1}^n \end{bmatrix}; \bar{\boldsymbol{y}} = \begin{bmatrix} \bar{y}_1 \\ \bar{y}_2 \\ \vdots \\ \bar{y}_m \end{bmatrix}$$

- Formula koefisien regresi dengan regularisasi:

$$\boldsymbol{a} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\bar{\boldsymbol{y}}$$

ANN untuk Regresi



Fungsi loss:

$$MSE = \frac{\sum_{i=1}^{m} (\overline{y} - y)^2}{m}$$

di mana :

$\overline{y}$ = label *ground truth*

y = output prediksi

m = banyaknya nilai regresi yang diprediksi

Meminimalkan MSE

Teknik-teknik Regresi Lain

- SVR (*Support Vector Regression*)
- Gaussian processes
- Bayesian regression
- Decision tree regressor

End..

Data Mining:

Concepts and Techniques

(3rd ed.)

— Chapter 10 —

Jiawei Han, Micheline Kamber, and Jian Pei
University of Illinois at Urbana-Champaign &
Simon Fraser University

Chapter 10. Cluster Analysis: Basic Concepts and Methods

- Cluster Analysis: Basic Concepts 
- Partitioning Methods
- Hierarchical Methods
- Density-Based Methods
- Grid-Based Methods
- Evaluation of Clustering
- Summary

What is Cluster Analysis?

- Cluster: A collection of data objects
  - similar (or related) to one another within the same group
  - dissimilar (or unrelated) to the objects in other groups
- Cluster analysis (or *clustering*, *data segmentation, …*)
  - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters
- Unsupervised learning: no predefined classes (i.e., *learning by observations* vs. learning by examples: supervised)
- Typical applications
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms

Clustering for Data Understanding and Applications

- Biology: taxonomy of living things: kingdom, phylum, class, order, family, genus and species
- Information retrieval: document clustering
- Land use: Identification of areas of similar land use in an earth observation database
- Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- City-planning: Identifying groups of houses according to their house type, value, and geographical location
- Earth-quake studies: Observed earth quake epicenters should be clustered along continent faults
- Climate: understanding earth climate, find patterns of atmospheric and ocean
- Economic Science: market resarch

Clustering as a Preprocessing Tool (Utility)

- Summarization:
  - Preprocessing for regression, PCA, classification, and association analysis
- Compression:
  - Image processing: vector quantization
- Finding K-nearest Neighbors
  - Localizing search to one or a small number of clusters
- Outlier detection
  - Outliers are often viewed as those “far away” from any cluster

Quality: What Is Good Clustering?

- A good clustering method will produce high quality clusters
  - high intra-class similarity: cohesive within clusters
  - low inter-class similarity: distinctive between clusters
- The quality of a clustering method depends on
  - the similarity measure used by the method
  - its implementation, and
  - Its ability to discover some or all of the hidden patterns

Measure the Quality of Clustering

- Dissimilarity/Similarity metric
  - Similarity is expressed in terms of a distance function, typically metric: $d(i, j)$
  - The definitions of distance functions are usually rather different for interval-scaled, boolean, categorical, ordinal ratio, and vector variables
  - Weights should be associated with different variables based on applications and data semantics
- Quality of clustering:
  - There is usually a separate "quality" function that measures the "goodness" of a cluster.
  - It is hard to define "similar enough" or "good enough"
    - The answer is typically highly subjective

Considerations for Cluster Analysis

- Partitioning criteria
  - Single level vs. hierarchical partitioning (often, multi-level hierarchical partitioning is desirable)
- Separation of clusters
  - Exclusive (e.g., one customer belongs to only one region) vs. non-exclusive (e.g., one document may belong to more than one class)
- Similarity measure
  - Distance-based (e.g., Euclidian, road network, vector) vs. connectivity-based (e.g., density or contiguity)
- Clustering space
  - Full space (often when low dimensional) vs. subspaces (often in high-dimensional clustering)

Requirements and Challenges

- Scalability
  - Clustering all the data instead of only on samples
- Ability to deal with different types of attributes
  - Numerical, binary, categorical, ordinal, linked, and mixture of these
- Constraint-based clustering
  - User may give inputs on constraints
  - Use domain knowledge to determine input parameters
- Interpretability and usability
- Others
  - Discovery of clusters with arbitrary shape
  - Ability to deal with noisy data
  - Incremental clustering and insensitivity to input order
  - High dimensionality

Major Clustering Approaches (I)

- Partitioning approach:
  - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
  - Typical methods: k-means, k-medoids, CLARANS
- Hierarchical approach:
  - Create a hierarchical decomposition of the set of data (or objects) using some criterion
  - Typical methods: Diana, Agnes, BIRCH, CAMELEON
- Density-based approach:
  - Based on connectivity and density functions
  - Typical methods: DBSCAN, OPTICS, DenClue
- Grid-based approach:
  - based on a multiple-level granularity structure
  - Typical methods: STING, WaveCluster, CLIQUE

Chapter 10. Cluster Analysis: Basic Concepts and Methods

- Cluster Analysis: Basic Concepts
- Partitioning Methods 
- Hierarchical Methods
- Density-Based Methods
- Grid-Based Methods
- Evaluation of Clustering
- Summary

Partitioning Algorithms: Basic Concept

- Partitioning method: Partitioning a database ***D*** of ***n*** objects into a set of ***k*** clusters, such that the sum of squared distances is minimized (where c_i is the centroid or medoid of cluster C_i)

$$E = \Sigma_{i=1}^{k} \Sigma_{p \in C_i} (p - c_i)^2$$

- Given *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
  - Heuristic methods: *k-means* and *k-medoids* algorithms
  - *k-means* (MacQueen'67, Lloyd'57/'82): Each cluster is represented by the center of the cluster
  - *k-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster

The *K-Means* Clustering Method

Algorithm: *k*-means. The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

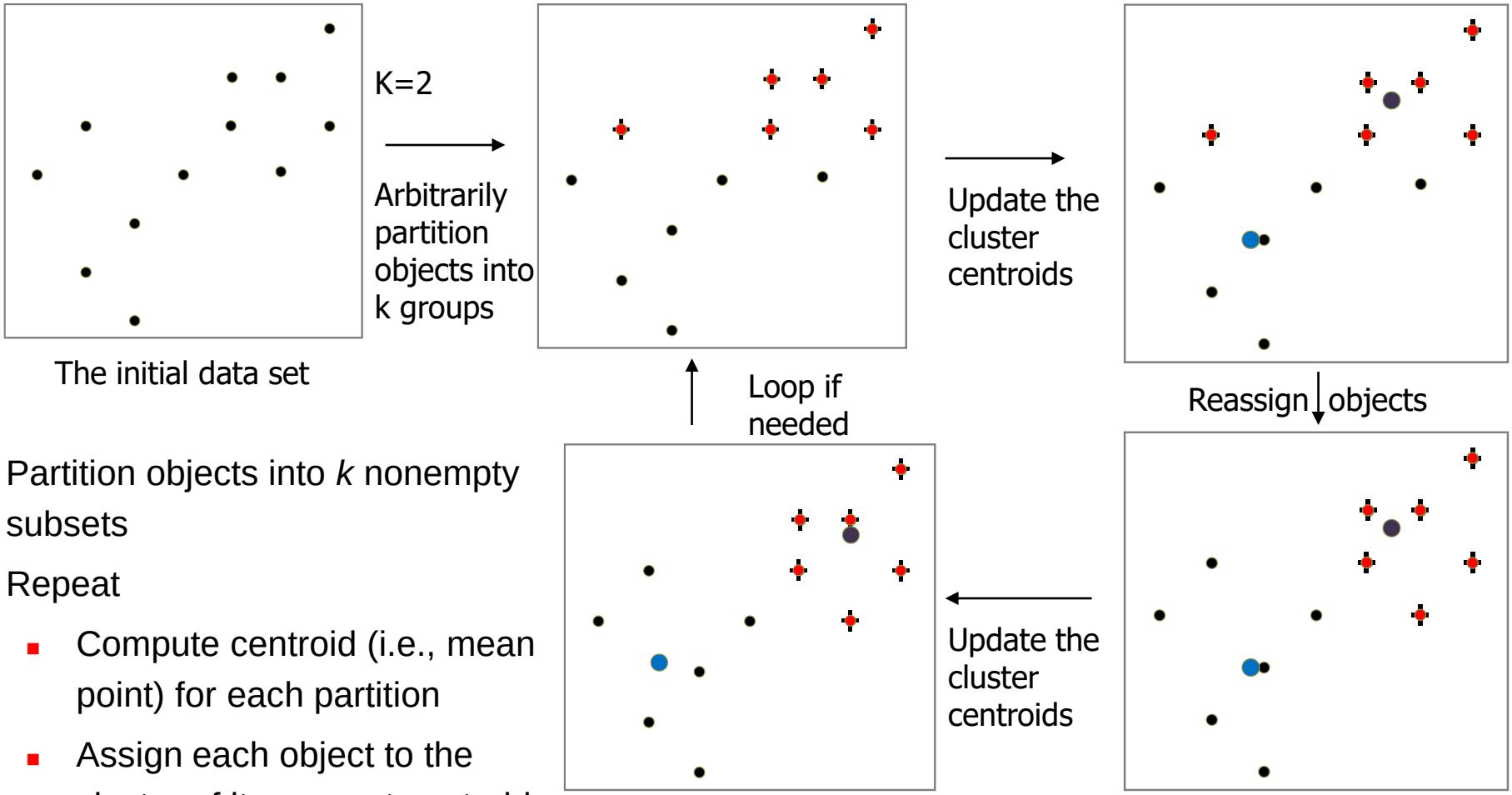
- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:

(1) arbitrarily choose *k* objects from *D* as the initial cluster centers;
(2) **repeat**
(3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
(4) update the cluster means, that is, calculate the mean value of the objects for each cluster;
(5) **until** no change;

An Example of *K-Means* Clustering



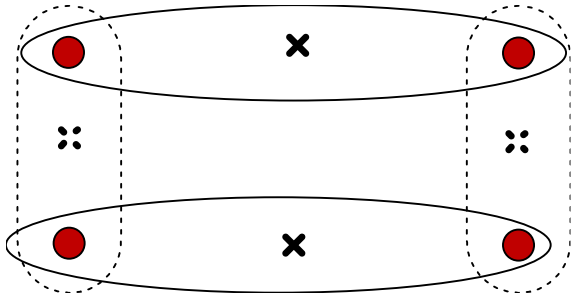
- Partition objects into *k* nonempty subsets
- Repeat
  - Compute centroid (i.e., mean point) for each partition
  - Assign each object to the cluster of its nearest centroid
- Until no change

Comments on the *K-Means* Method

- Strength: *Efficient*: $O(tkn)$, where n is # objects, k is # clusters, and t is # iterations. Normally, $k, t << n$.
  - Comparing: PAM: $O(k(n-k)^2)$, CLARA: $O(ks^2 + k(n-k))$
- Comment: Often terminates at a *local optimal*.
- Weakness
  - Applicable only to objects in a continuous n-dimensional space
    - Using the k-modes method for categorical data
    - In comparison, k-medoids can be applied to a wide range of data
  - Need to specify *k,* the *number* of clusters, in advance (there are ways to automatically determine the best k (see Hastie et al., 2009)
  - Sensitive to noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*

Variations of the *K-Means* Method

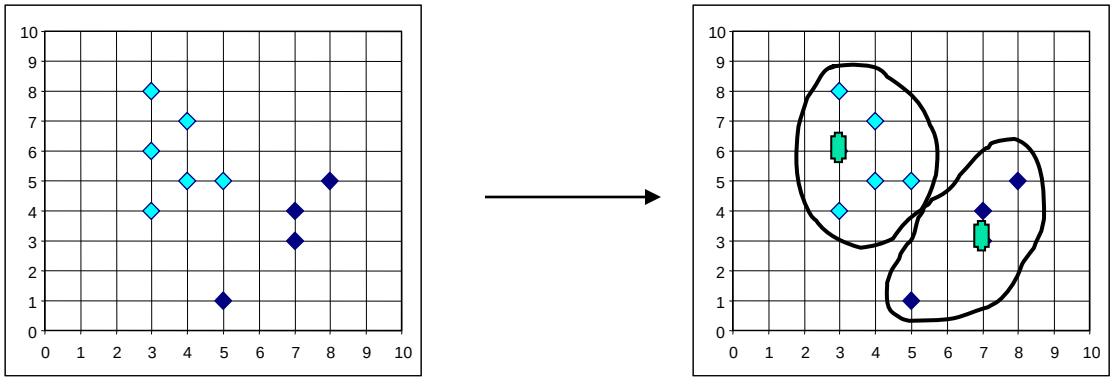
- Most of the variants of the *k-means* which differ in
  - Selection of the initial *k* means
  - Dissimilarity calculations
  - Strategies to calculate cluster means



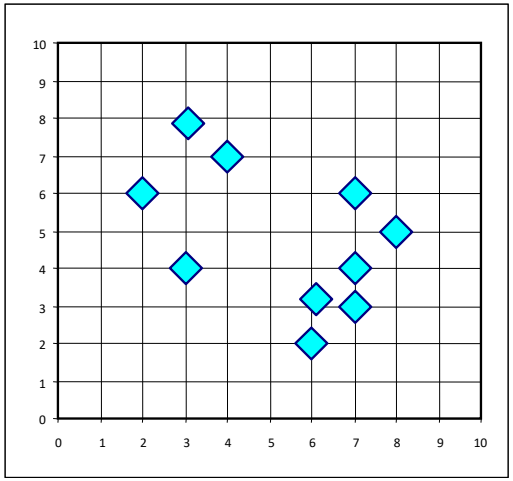
- Handling categorical data: *k-modes*
  - Replacing means of clusters with modes
  - Using new dissimilarity measures to deal with categorical objects
  - Using a frequency-based method to update modes of clusters
  - A mixture of categorical and numerical data: *k-prototype* method

What Is the Problem of the K-Means Method?

- The k-means algorithm is sensitive to outliers !
  - Since an object with an extremely large value may substantially distort the distribution of the data
- K-Medoids: Instead of taking the **mean** value of the object in a cluster as a reference point, **medoids** can be used, which is the **most centrally located** object in a cluster

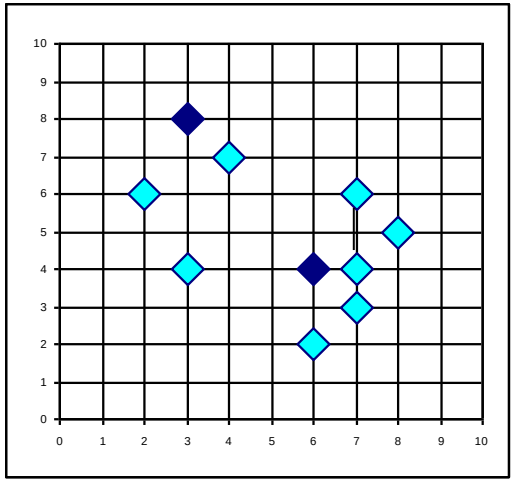


PAM: A Typical K-Medoids Algorithm



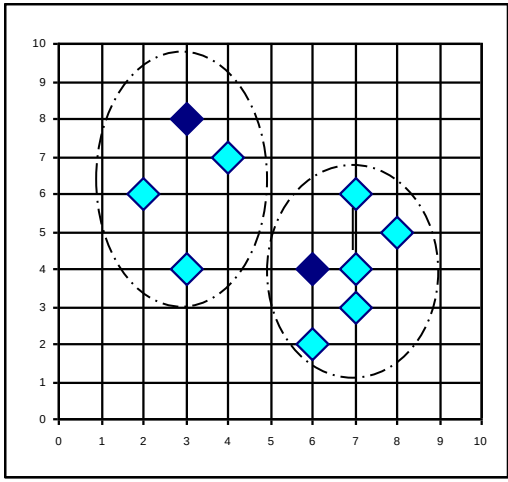
K=2

Arbitrary choose k object as initial medoids

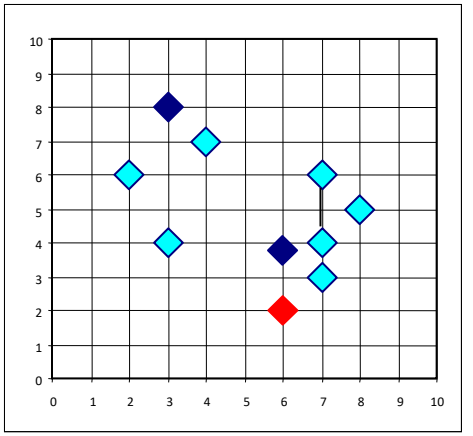


Assign each remaining object to nearest medoids

Total Cost = 20

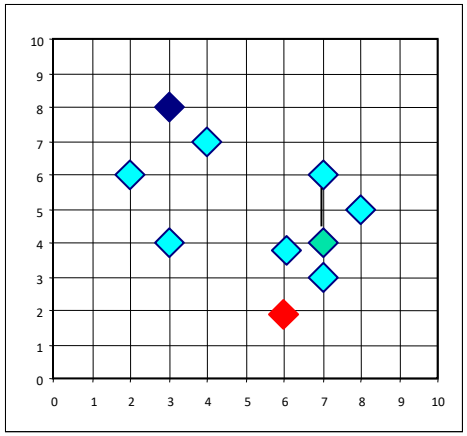


Randomly select a nonmedoid object, O_{ramdom}



Compute total cost of swapping

Total Cost = 26



Swapping O and O_{ramdom}

If quality is improved.

Do loop

Until no change

The K-Medoid Clustering Method

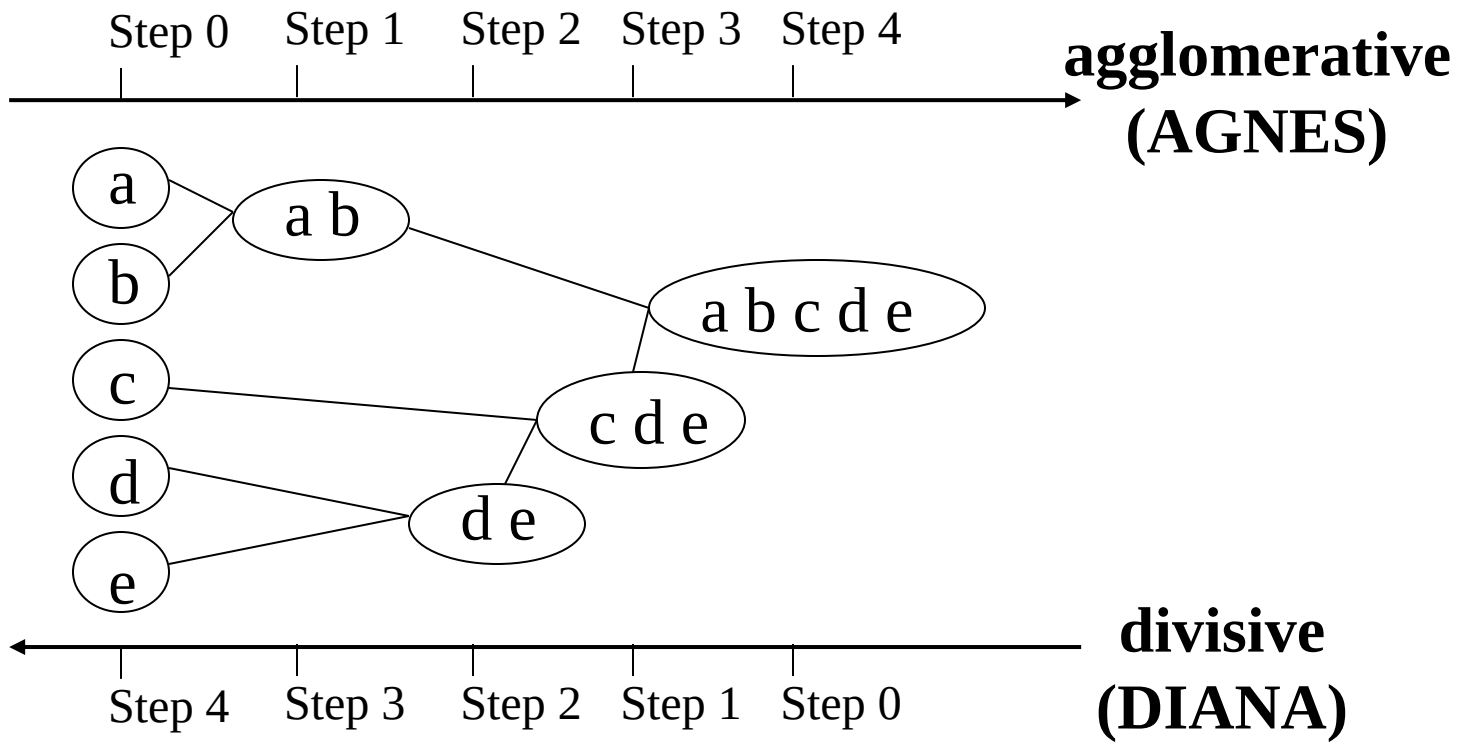
- *K-Medoids* Clustering: Find *representative* objects (medoids) in clusters
  - *PAM* (Partitioning Around Medoids, Kaufmann & Rousseeuw 1987)
    - Starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
    - *PAM* works effectively for small data sets, but does not scale well for large data sets (due to the computational complexity)
- Efficiency improvement on PAM
  - *CLARA* (Kaufmann & Rousseeuw, 1990): PAM on samples
  - *CLARANS* (Ng & Han, 1994): Randomized re-sampling

Chapter 10. Cluster Analysis: Basic Concepts and Methods

- Cluster Analysis: Basic Concepts
- Partitioning Methods
- Hierarchical Methods 
- Density-Based Methods
- Grid-Based Methods
- Evaluation of Clustering
- Summary

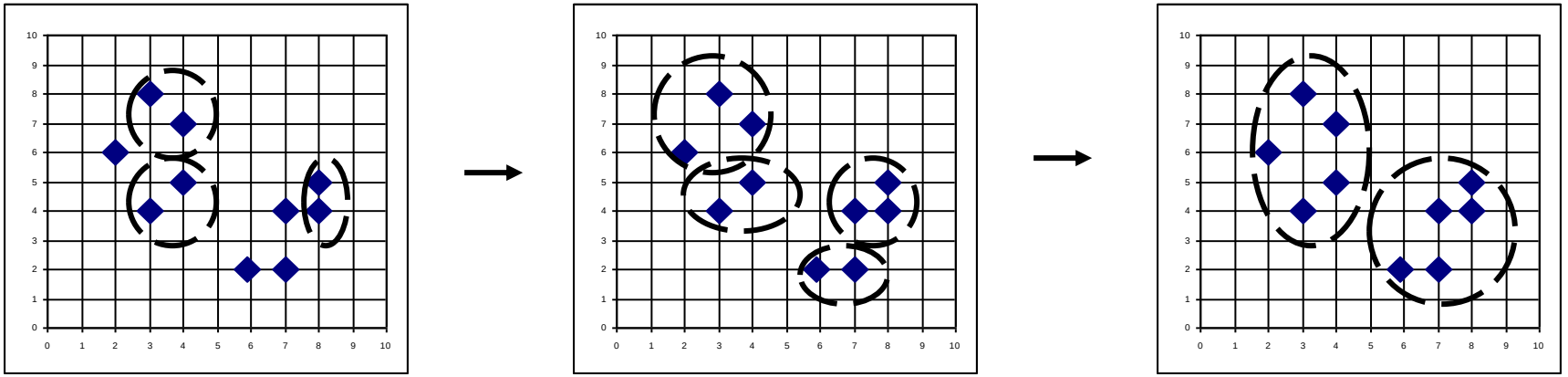
Hierarchical Clustering

- Use distance matrix as clustering criteria. This method does not require the number of clusters ***k*** as an input, but needs a termination condition



AGNES (Agglomerative Nesting)

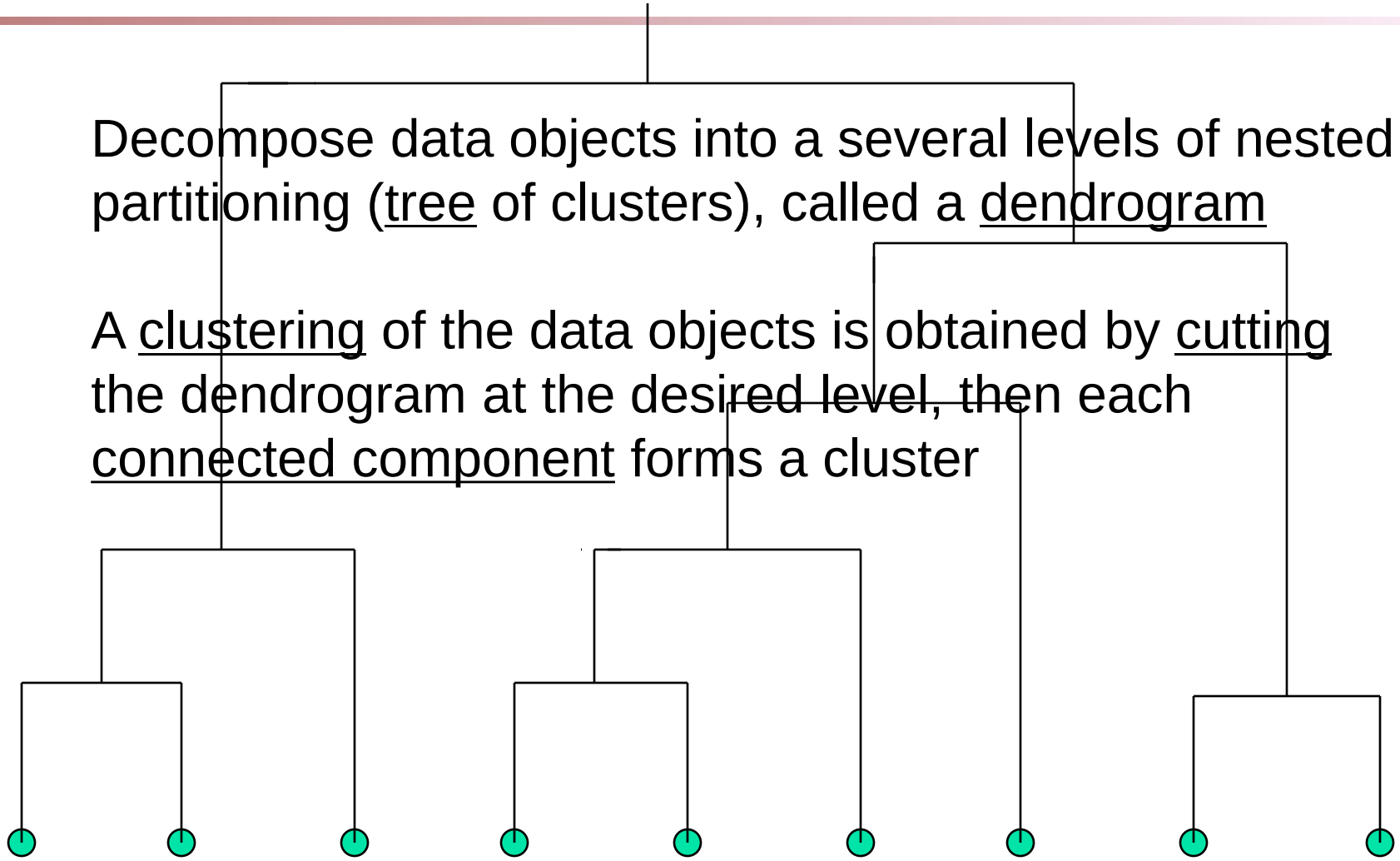
- Introduced in Kaufmann and Rousseeuw (1990)
- Implemented in statistical packages, e.g., Splus
- Use the **single-link** method and the dissimilarity matrix
- Merge nodes that have the least dissimilarity
- Go on in a non-descending fashion
- Eventually all nodes belong to the same cluster



Dendrogram: Shows How Clusters are Merged

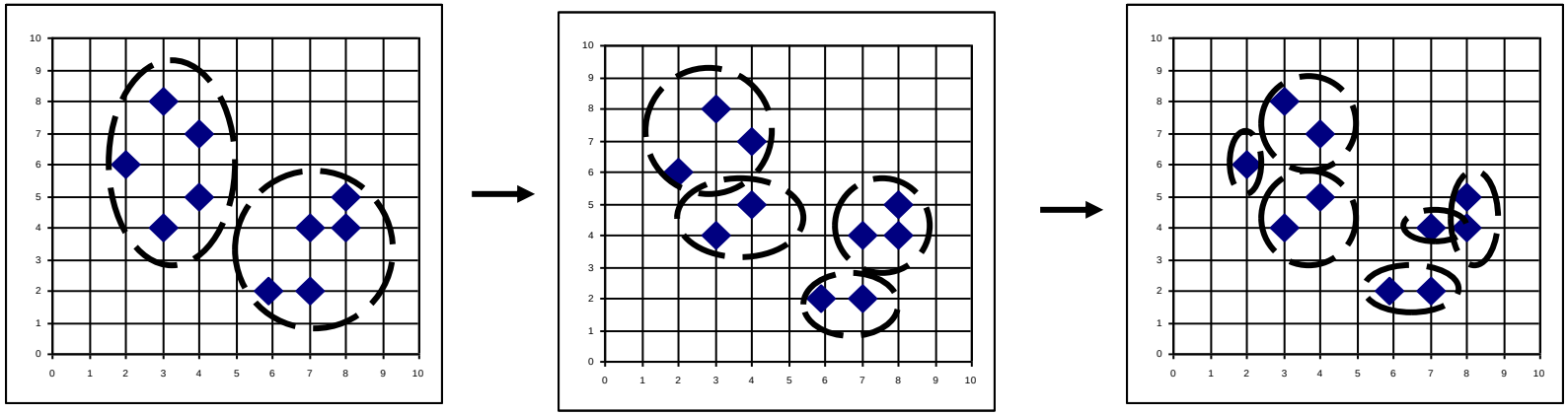
Decompose data objects into a several levels of nested partitioning (tree of clusters), called a dendrogram

A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster

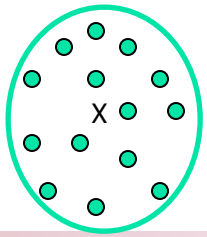


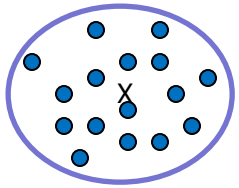
DIANA (Divisive Analysis)

- Introduced in Kaufmann and Rousseeuw (1990)
- Implemented in statistical analysis packages, e.g., Splus
- Inverse order of AGNES
- Eventually each node forms a cluster on its own



Distance between Clusters





- Single link: smallest distance between an element in one cluster and an element in the other, i.e., $dist(K_i, K_j) = min(t_{ip}, t_{jq})$
- Complete link: largest distance between an element in one cluster and an element in the other, i.e., $dist(K_i, K_j) = max(t_{ip}, t_{jq})$
- Average: avg distance between an element in one cluster and an element in the other, i.e., $dist(K_i, K_j) = avg(t_{ip}, t_{jq})$
- Centroid: distance between the centroids of two clusters, i.e., $dist(K_i, K_j) = dist(C_i, C_j)$
- Medoid: distance between the medoids of two clusters, i.e., $dist(K_i, K_j) = dist(M_i, M_j)$
  - Medoid: a chosen, centrally located object in the cluster

Centroid, Radius and Diameter of a Cluster (for numerical data sets)

- Centroid: the "middle" of a cluster

$$C_m = \frac{\sum_{i=1}^{N} (t_{ip})}{N}$$

- Radius: square root of average distance from any point of the cluster to its centroid

$$R_m = \sqrt{\frac{\sum_{i=1}^{N} (t_{ip} - c_m)^2}{N}}$$

- Diameter: square root of average mean squared distance between all pairs of points in the cluster

$$D_m = \sqrt{\frac{\sum_{i=1}^{N} \sum_{i=1}^{N} (t_{ip} - t_{iq})^2}{N(N-1)}}$$

Extensions to Hierarchical Clustering

- Major weakness of agglomerative clustering methods
  - Can never undo what was done previously
  - Do not scale well: time complexity of at least $O(n^2)$, where n is the number of total objects
- Integration of hierarchical & distance-based clustering
  - BIRCH (1996): uses CF-tree and incrementally adjusts the quality of sub-clusters
  - CHAMELEON (1999): hierarchical clustering using dynamic modeling

BIRCH (Balanced Iterative Reducing and Clustering Using Hierarchies)

- Zhang, Ramakrishnan & Livny, SIGMOD'96
- Incrementally construct a CF (Clustering Feature) tree, a hierarchical data structure for multiphase clustering
  - Phase 1: scan DB to build an initial in-memory CF tree (a multi-level compression of the data that tries to preserve the inherent clustering structure of the data)
  - Phase 2: use an arbitrary clustering algorithm to cluster the leaf nodes of the CF-tree
- *Scales linearly*: finds a good clustering with a single scan and improves the quality with a few additional scans
- *Weakness:* handles only numeric data, and sensitive to the order of the data record

Clustering Feature Vector in BIRCH

Clustering Feature (CF): ***CF = (N, LS, SS)***

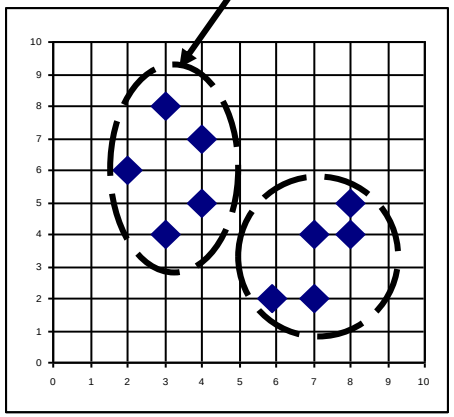
N: **Number of data points**

LS: linear sum of N points: $\sum_{i=1}^{N} X_i$

SS: square sum of N points

$$\sum_{i=1}^{N} X_i^2$$

CF = (5, (16,30),(54,190))

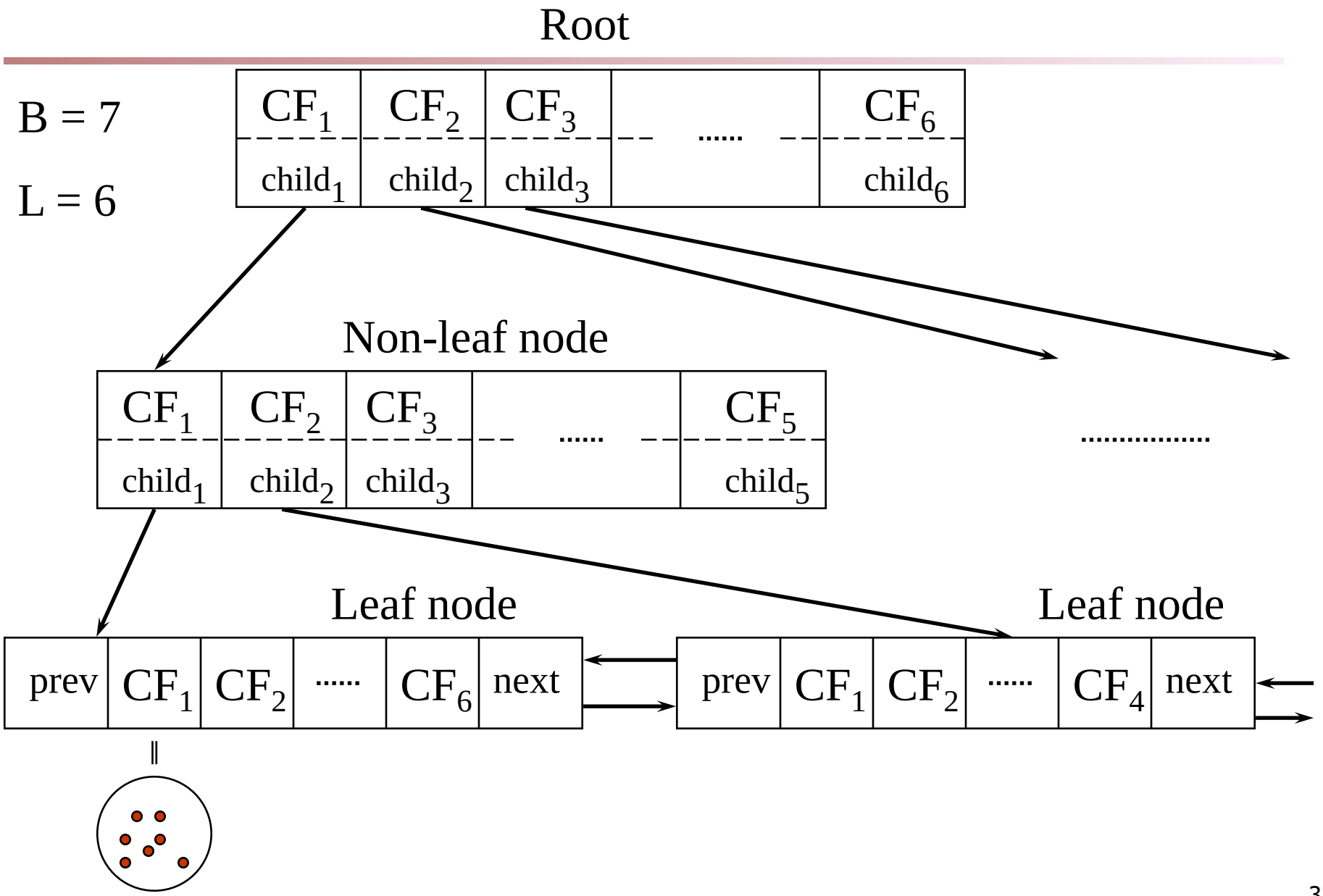


(3,4)
(2,6)
(4,5)
(4,7)
(3,8)

CF-Tree in BIRCH

- Clustering feature:
  - Summary of the statistics for a given subcluster: the 0-th, 1st, and 2nd moments of the subcluster from the statistical point of view
  - Registers crucial measurements for computing cluster and utilizes storage efficiently
- A CF tree is a height-balanced tree that stores the clustering features for a hierarchical clustering
  - A nonleaf node in a tree has descendants or “children”
  - The nonleaf nodes store sums of the CFs of their children
- A CF tree has two parameters
  - Branching factor: max # of children
  - Threshold: max diameter of sub-clusters stored at the leaf nodes

The CF Tree Structure



The Birch Algorithm

- Cluster Diameter $\sqrt{\frac{1}{n(n-1)}\sum(x_i - x_j)^2}$

- For each point in the input
  - Find closest leaf entry
  - Add point to leaf entry and update CF
  - If entry diameter > max_diameter, then split leaf, and possibly parents
- Algorithm is O(n)
- Concerns
  - Sensitive to insertion order of data points
  - Since we fix the size of leaf nodes, so clusters may not be so natural
  - Clusters tend to be spherical given the radius and diameter measures

Chapter 10. Cluster Analysis: Basic Concepts and Methods

- Cluster Analysis: Basic Concepts
- Partitioning Methods
- Hierarchical Methods
- Density-Based Methods 
- Grid-Based Methods
- Evaluation of Clustering
- Summary

Chapter 10. Cluster Analysis: Basic Concepts and Methods

- Cluster Analysis: Basic Concepts
- Partitioning Methods
- Hierarchical Methods
- Density-Based Methods 
- Grid-Based Methods
- Evaluation of Clustering
- Summary

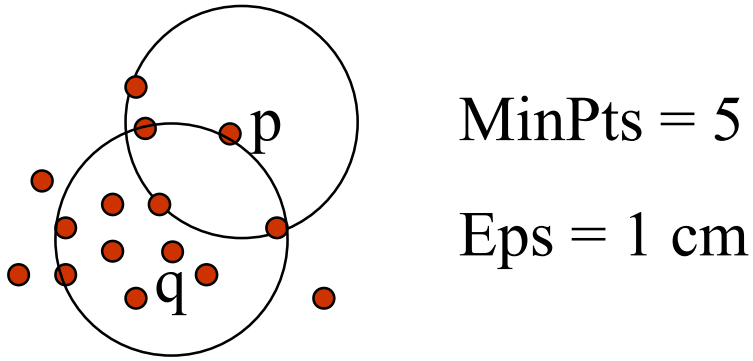
Density-Based Clustering Methods

- Clustering based on density (local cluster criterion), such as density-connected points
- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan
  - Need density parameters as termination condition
- Several interesting studies:
  - DBSCAN: Ester, et al. (KDD'96)
  - OPTICS: Ankerst, et al (SIGMOD'99).
  - DENCLUE: Hinneburg & D. Keim (KDD'98)
  - CLIQUE: Agrawal, et al. (SIGMOD'98) (more grid-based)

Density-Based Clustering: Basic Concepts

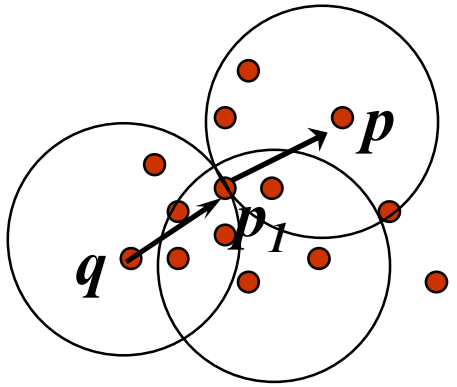
- Two parameters:
  - *Eps*: Maximum radius of the neighbourhood
  - *MinPts*: Minimum number of points in an Eps-neighbourhood of that point
- $N_{Eps}(p)$: {q belongs to D | dist(p,q) ≤ Eps}
- Directly density-reachable: A point *p* is directly density-reachable from a point *q* w.r.t. *Eps*, *MinPts* if
  - *p* belongs to $N_{Eps}(q)$
  - core point condition:

$$|N_{Eps}(q)| \geq MinPts$$

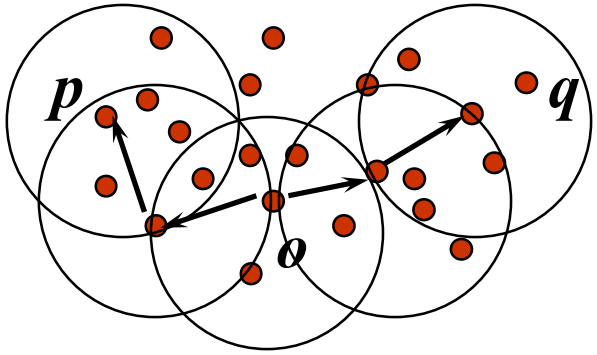


Density-Reachable and Density-Connected

- Density-reachable:
  - A point *p* is density-reachable from a point *q* w.r.t. *Eps*, *MinPts* if there is a chain of points p_1, …, p_n, p_1 = *q*, p_n = *p* such that p_{i+1} is directly density-reachable from p_i

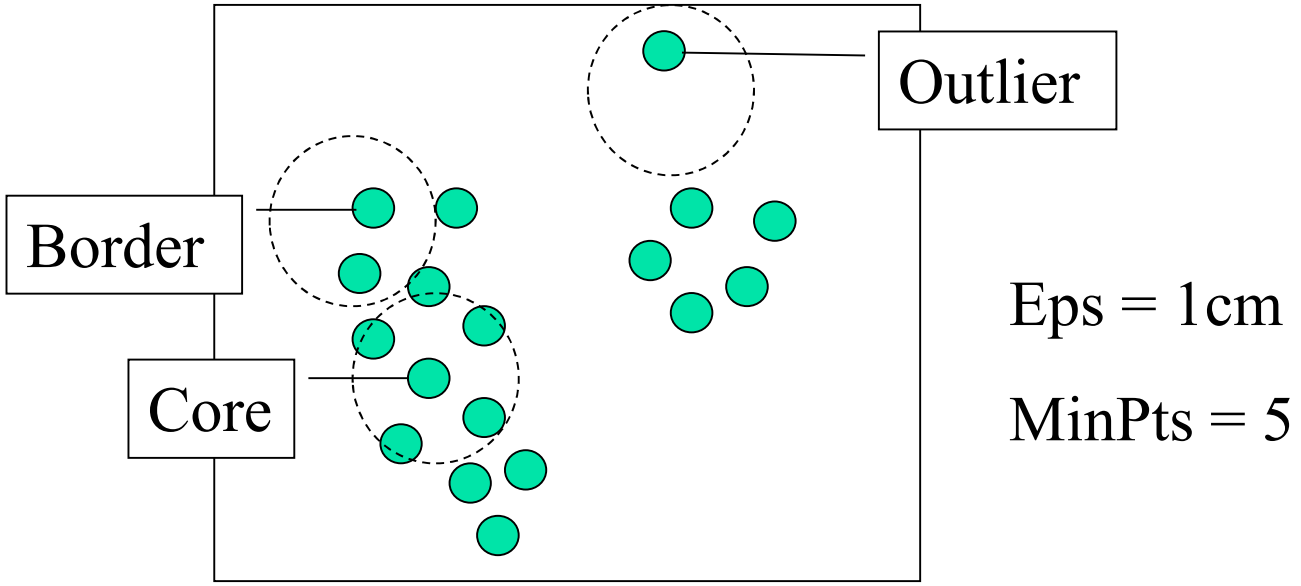


- Density-connected
  - A point *p* is density-connected to a point *q* w.r.t. *Eps*, *MinPts* if there is a point *o* such that both, *p* and *q* are density-reachable from *o* w.r.t. *Eps* and *MinPts*



DBSCAN: Density-Based Spatial Clustering of Applications with Noise

- Relies on a *density-based* notion of cluster: A *cluster* is defined as a maximal set of density-connected points
- Discovers clusters of arbitrary shape in spatial databases with noise



DBSCAN: The Algorithm

- Arbitrary select a point *p*
- Retrieve all points density-reachable from *p* w.r.t. *Eps* and *MinPts*
- If *p* is a core point, a cluster is formed
- If *p* is a border point, no points are density-reachable from *p* and DBSCAN visits the next point of the database
- Continue the process until all of the points have been processed

DBSCAN: Sensitive to Parameters

Figure 8. DBScan results for DS1 with MinPts at 4 and Eps at (a) 0.5 and (b) 0.4.

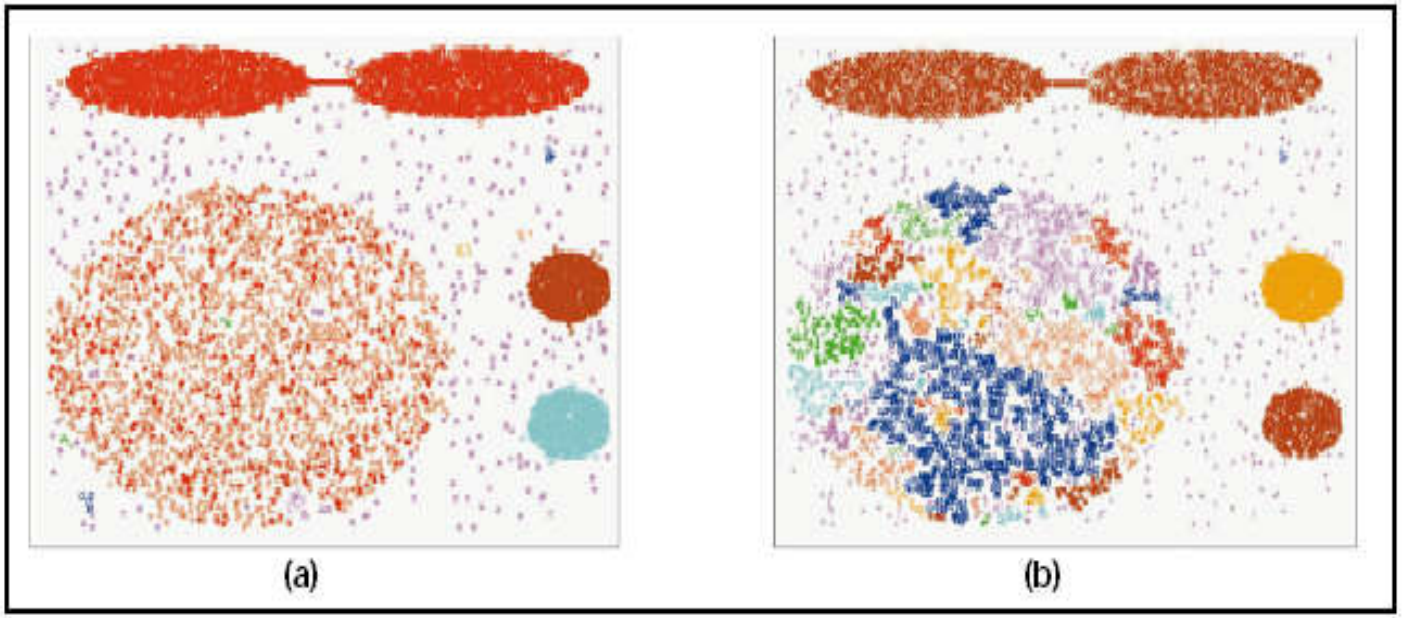
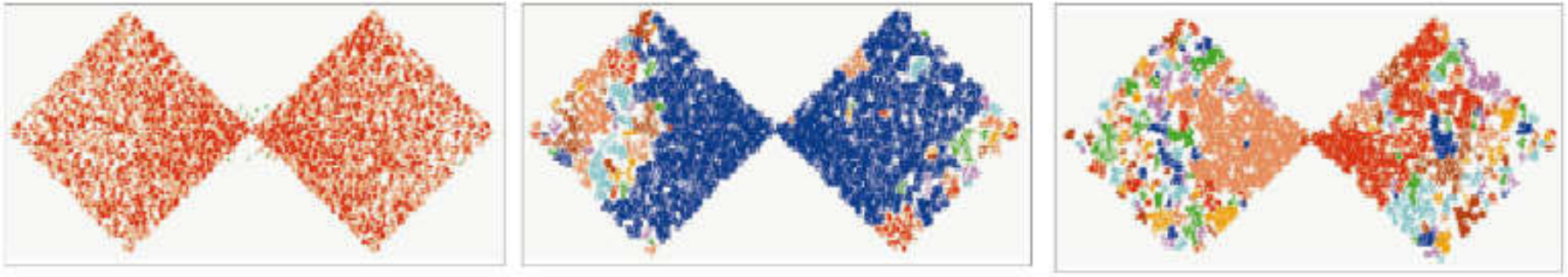


Figure 9. DBScan results for DS2 with MinPts at 4 and Eps at (a) 5.0, (b) 3.5, and (c) 3.0.



Chapter 10. Cluster Analysis: Basic Concepts and Methods

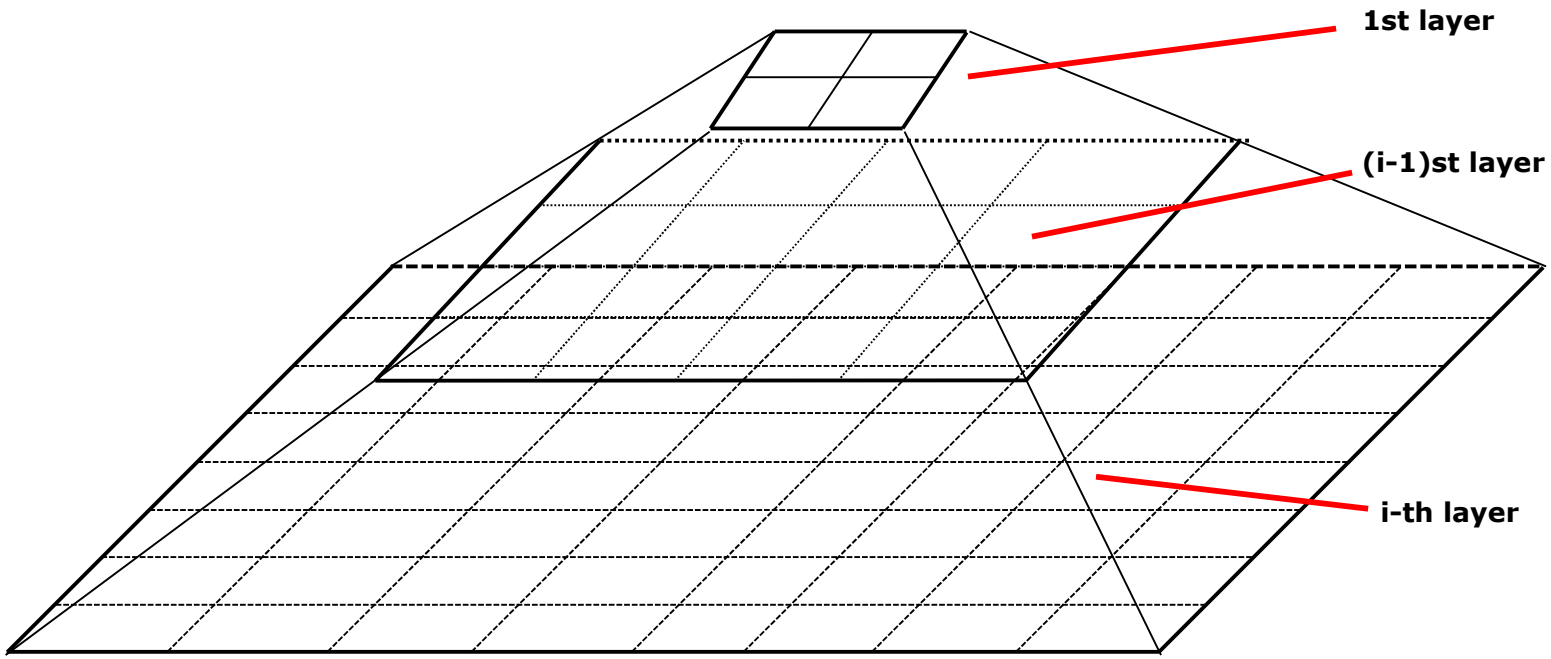
- Cluster Analysis: Basic Concepts
- Partitioning Methods
- Hierarchical Methods
- Density-Based Methods
- Grid-Based Methods 
- Evaluation of Clustering
- Summary

Grid-Based Clustering Method

- Using multi-resolution grid data structure
- Several interesting methods
  - STING (a STatistical INformation Grid approach) by Wang, Yang and Muntz (1997)
  - WaveCluster by Sheikholeslami, Chatterjee, and Zhang (VLDB'98)
    - A multi-resolution clustering approach using wavelet method
  - CLIQUE: Agrawal, et al. (SIGMOD'98)
    - Both grid-based and subspace clustering

STING: A Statistical Information Grid Approach

- Wang, Yang and Muntz (VLDB'97)
- The spatial area is divided into rectangular cells
- There are several levels of cells corresponding to different levels of resolution



The STING Clustering Method

- Each cell at a high level is partitioned into a number of smaller cells in the next lower level
- Statistical info of each cell is calculated and stored beforehand and is used to answer queries
- Parameters of higher level cells can be easily calculated from parameters of lower level cell
  - *count*, *mean*, *s*, *min*, *max*
  - type of distribution—*normal*, *uniform*, etc.
- Use a top-down approach to answer spatial data queries
- Start from a pre-selected layer—typically with a small number of cells
- For each cell in the current level compute the confidence interval

STING Algorithm and Its Analysis

- Remove the irrelevant cells from further consideration
- When finish examining the current layer, proceed to the next lower level
- Repeat this process until the bottom layer is reached
- Advantages:
  - Query-independent, easy to parallelize, incremental update
  - *O(K),* where *K* is the number of grid cells at the lowest level
- Disadvantages:
  - All the cluster boundaries are either horizontal or vertical, and no diagonal boundary is detected

Chapter 10. Cluster Analysis: Basic Concepts and Methods

- Cluster Analysis: Basic Concepts
- Partitioning Methods
- Hierarchical Methods
- Density-Based Methods
- Grid-Based Methods
- Evaluation of Clustering 
- Summary

Assessing Clustering Tendency

- Assess if non-random structure exists in the data by measuring the probability that the data is generated by a uniform data distribution
- Test spatial randomness by statistic test: Hopkins Static
  - Given a dataset D regarded as a sample of a random variable o, determine how far away o is from being uniformly distributed in the data space
  - Sample *n* points, $p_1, \ldots, p_n$, uniformly from D. For each p_i, find its nearest neighbor in D: $x_i = min\{dist(p_i, v)\}$ where *v* in D
  - Sample *n* points, $q_1, \ldots, q_n$, uniformly from D. For each q_i, find its nearest neighbor in D $-$ $\{q_i\}$: $y_i = min\{dist(q_i, v)\}$ where *v* in D and $v \neq q_i$
  - Calculate the Hopkins Statistic: $H = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i}$
  - If D is uniformly distributed, $\sum x_i$ and $\sum y_i$ will be close to each other and H is close to 0.5. If D is highly skewed, H is close to 0

Determine the Number of Clusters

- Empirical method
  - # of clusters ≈√n/2 for a dataset of n points
- Elbow method
  - Use the turning point in the curve of sum of within cluster variance w.r.t the # of clusters
- Cross validation method
  - Divide a given data set into *m* parts
  - Use *m* – 1 parts to obtain a clustering model
  - Use the remaining part to test the quality of the clustering
    - E.g., For each point in the test set, find the closest centroid, and use the sum of squared distance between all points in the test set and the closest centroids to measure how well the model fits the test set
  - For any k > 0, repeat it *m* times, compare the overall quality measure w.r.t. different *k's*, and find # of clusters that fits the data the best

Measuring Clustering Quality

- Two methods: extrinsic vs. intrinsic
- Extrinsic: supervised, i.e., the ground truth is available
  - Compare a clustering against the ground truth using certain clustering quality measure
  - Ex. BCubed precision and recall metrics
- Intrinsic: unsupervised, i.e., the ground truth is unavailable
  - Evaluate the goodness of a clustering by considering how well the clusters are separated, and how compact the clusters are
  - Ex. Silhouette coefficient

Measuring Clustering Quality: Extrinsic Methods

- Clustering quality measure: $Q(C, C_g)$, for a clustering C given the ground truth C_g.
- Q is good if it satisfies the following **4** essential criteria
  - Cluster homogeneity: the purer, the better
  - Cluster completeness: should assign objects belong to the same category in the ground truth to the same cluster
  - Rag bag: putting a heterogeneous object into a pure cluster should be penalized more than putting it into a *rag bag* (i.e., "miscellaneous" or "other" category)
  - Small cluster preservation: splitting a small category into pieces is more harmful than splitting a large category into pieces

Chapter 10. Cluster Analysis: Basic Concepts and Methods

- Cluster Analysis: Basic Concepts
- Partitioning Methods
- Hierarchical Methods
- Density-Based Methods
- Grid-Based Methods
- Evaluation of Clustering
- Summary 

Summary

- Cluster analysis groups objects based on their similarity and has wide applications
- Measure of similarity can be computed for various types of data
- Clustering algorithms can be categorized into partitioning methods, hierarchical methods, density-based methods, grid-based methods, and model-based methods
- K-means and K-medoids algorithms are popular partitioning-based clustering algorithms
- Birch and Chameleon are interesting hierarchical clustering algorithms, and there are also probabilistic hierarchical clustering algorithms
- DBSCAN, OPTICS, and DENCLU are interesting density-based algorithms
- STING and CLIQUE are grid-based methods, where CLIQUE is also a subspace clustering algorithm
- Quality of clustering results can be evaluated in various ways

CS512-Spring 2011: An Introduction

- Coverage
  - Cluster Analysis: Chapter 11
  - Outlier Detection: Chapter 12
  - Mining Sequence Data: BK2: Chapter 8
  - Mining Graphs Data: BK2: Chapter 9
  - Social and Information Network Analysis
    - BK2: Chapter 9
    - Partial coverage: Mark Newman: “Networks: An Introduction”, Oxford U., 2010
    - Scattered coverage: Easley and Kleinberg, “Networks, Crowds, and Markets: Reasoning About a Highly Connected World”, Cambridge U., 2010
    - Recent research papers
  - Mining Data Streams: BK2: Chapter 8
- Requirements
  - One research project
  - One class presentation (15 minutes)
  - Two homeworks (no programming assignment)
  - Two midterm exams (no final exam)

References (1)

- R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. SIGMOD'98
- M. R. Anderberg. Cluster Analysis for Applications. Academic Press, 1973.
- M. Ankerst, M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure, SIGMOD'99.
- Beil F., Ester M., Xu X.: "Frequent Term-Based Text Clustering", KDD'02
- M. M. Breunig, H.-P. Kriegel, R. Ng, J. Sander. LOF: Identifying Density-Based Local Outliers. SIGMOD 2000.
- M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases. KDD'96.
- M. Ester, H.-P. Kriegel, and X. Xu. Knowledge discovery in large spatial databases: Focusing techniques for efficient class identification. SSD'95.
- D. Fisher. Knowledge acquisition via incremental conceptual clustering. Machine Learning, 2:139-172, 1987.
- D. Gibson, J. Kleinberg, and P. Raghavan. Clustering categorical data: An approach based on dynamic systems. VLDB'98.
- V. Ganti, J. Gehrke, R. Ramakrishan. CACTUS Clustering Categorical Data Using Summaries. KDD'99.

References (2)

- D. Gibson, J. Kleinberg, and P. Raghavan. Clustering categorical data: An approach based on dynamic systems. In Proc. VLDB'98.
- S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. SIGMOD'98.
- S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. In *ICDE'99*, pp. 512-521, Sydney, Australia, March 1999.
- A. Hinneburg, D.I A. Keim: An Efficient Approach to Clustering in Large Multimedia Databases with Noise. KDD'98.
- A. K. Jain and R. C. Dubes. Algorithms for Clustering Data. Printice Hall, 1988.
- G. Karypis, E.-H. Han, and V. Kumar. CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling. *COMPUTER*, 32(8): 68-75, 1999.
- L. Kaufman and P. J. Rousseeuw. Finding Groups in Data: an Introduction to Cluster Analysis. John Wiley & Sons, 1990.
- E. Knorr and R. Ng. Algorithms for mining distance-based outliers in large datasets. VLDB'98.

References (3)

- G. J. McLachlan and K.E. Bkasford. Mixture Models: Inference and Applications to Clustering. John Wiley and Sons, 1988.
- R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. VLDB'94.
- L. Parsons, E. Haque and H. Liu, Subspace Clustering for High Dimensional Data: A Review, SIGKDD Explorations, 6(1), June 2004
- E. Schikuta. Grid clustering: An efficient hierarchical clustering method for very large data sets. Proc. 1996 Int. Conf. on Pattern Recognition
- G. Sheikholeslami, S. Chatterjee, and A. Zhang. WaveCluster: A multi-resolution clustering approach for very large spatial databases. VLDB'98.
- A. K. H. Tung, J. Han, L. V. S. Lakshmanan, and R. T. Ng. Constraint-Based Clustering in Large Databases, ICDT'01.
- A. K. H. Tung, J. Hou, and J. Han. Spatial Clustering in the Presence of Obstacles, ICDE'01
- H. Wang, W. Wang, J. Yang, and P.S. Yu. Clustering by pattern similarity in large data sets, SIGMOD'02
- W. Wang, Yang, R. Muntz, STING: A Statistical Information grid Approach to Spatial Data Mining, VLDB'97
- T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH : An efficient data clustering method for very large databases. SIGMOD'96
- X. Yin, J. Han, and P. S. Yu, "LinkClus: Efficient Clustering via Heterogeneous Semantic Links", VLDB'06