Unit - 5

Topics

- Mining the World Wide Web
- Page Rank Algorithm
- Text mining
- Mining Time Series Data
- Ensemble methods-Increasing the Accuracy

Web mining

- Web Mining can broadly be seen as the application of adapted data mining methods to the web.

- The Web has different facets that yield different approaches for the mining process:
  - (*i) web pages* consist of text,
  - (*ii) web pages are linked via hyperlinks, and*
  - *(iii)* user activity can be monitored via web server logs.
- These three facets lead to the distinction into the three areas of
  - web content mining,
  - web structure mining, and
  - web usage mining.

Web content mining

- For web content mining, each web page is considered as an individual document.
- Sets of web pages form a document collection
- Web page is semi structured
- content mining task is information extraction where structured information is extracted from unstructured web sites.
- The goal is to facilitate information aggregation over different web sites by using the extracted structured information.
- Typical applications are price comparison sites or news aggregators

Web Structure Mining

- For web structure mining, one considers the web as a directed graph, with the web pages being the vertices, that are connected by hyperlinks

- Application – google's page rank algorithm

Usage mining

- The records of requests of visitors of a web site are usually collected as web server logs
- The user requests indicate how the consumer perceives the visited web pages.
- Application: correlations in buying behavior, that may be used for recommendations (“People who bought *x also* bought *y.")*

Mining Methods

- All of the existing data mining techniques (eg, clustering, classification, association rules) can also be applied in web mining.

- They usually need a more extensive preprocessing, since web resources are normally not in the form of a flat table which is required for most methods.

  – Link Mining.
  – Statistical Relation Learning.
  – Social Network Analysis.

Link Mining

- Links typically represent the structure of the web.
- Analyzing these links is is often referred as Link Mining
- Link Mining is mainly divided into three major tasks:
  - the object related task like clustering based on links
  - prediction of (missing) links and
  - graph centered task like subgraph discovery.

Statistical Relation Learning (SRL)

- SRL focuses on the combination of probabilistic and logic models with the goal to develop one combined approach which is better able to describe real world phenomena.
- Traditional statistical machine learning is able to capture uncertainty but only for one relational ILP (Inductive Logic Programming)
- Relational learning approaches are able to work on multiple relation but can't handle noise.
- The combination of both tries to overcome these limitations.
- Application: Hypertext classification, topic prediction of bibliographic entries, or on any kind of social networks.

Social Network Analysis (1)

- SNA techniques analyse the network as a whole, or study properties of the individual nodes.
- Measures for the network as a whole comprise density (percentage of present edges among possible edges), diameter (length of the longest shortest path between any two nodes), clustering coefficient, etc.
- In a web mining scenario, these may be used, together with other staticistal measures, in the data understanding phase.
- A key notion for studying individual nodes is `centrality‘

Social Network Analysis (2)

- In/out degree centrality measures the number of in/outbound edges of a node, and may, be used for analysing the social status of people.

- Betweenness centrality measures the number of shortest paths a node is lying on – used in communication networks

- SNA combines the properties of individual nodes with (the growth of) the over all network, leading to a model of `preferential attachment'

Applications of Web Mining (1)

- Search
  - Search engines have become a crucial navigation component of the Web.
  - Google, uses the PageRank algorithm
  - The Hyperlink-Induced Topic Search

- Recommendations.
  - Personalised marketing has become an important business issue in the past few years.
  - Based on the easiness of collecting personal information on the web and the possibility of dynamic web page generation, recommender systems have become a major web application.
  - They make extensive use of data mining techniques

Applications of Web Mining (2)

- Topic & Trend Detection
  - The dynamic nature of the web provides another challenge: new topics may come suddenly, or trends may emerge slowly.
  - Topics and trends can be detected by comparing the evolution of web content, structure and/or usage over time.

- Communities.
  - Data mining and social network analysis methods have been deployed to discover communities, and to study their evolution over time.

Text mining (1)

- Text mining is the process of analyzing text to extract information that is useful for particular purposes.

- The phrase “text mining” is generally used to denote any system that analyzes large quantities of natural language text and detects lexical or linguistic usage patterns in an attempt to extract probably useful (although only probably correct) information.

- Compared with the kind of data stored in databases, text is unstructured, amorphous, and difficult to deal with algorithmically.

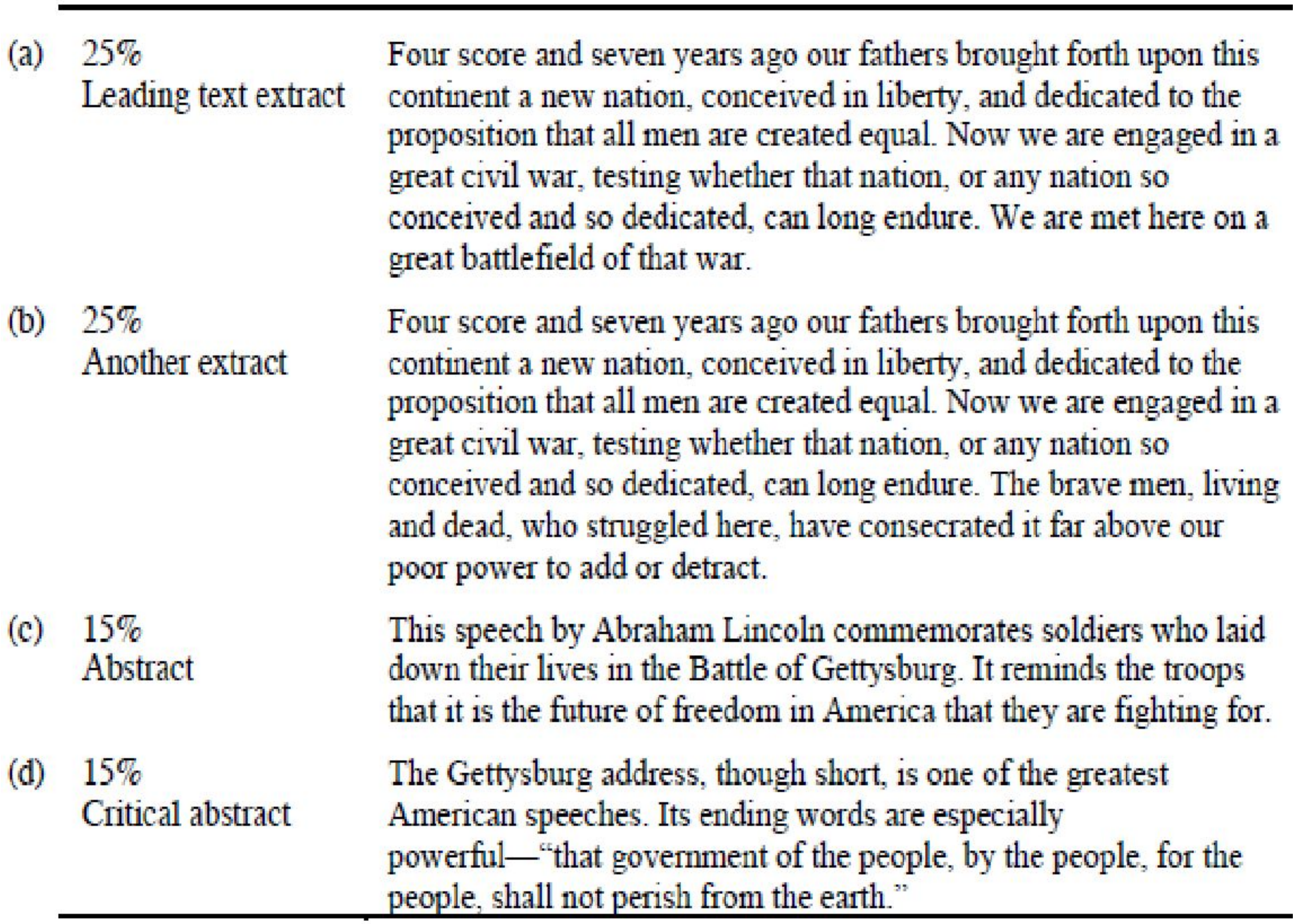
- Text mining is about looking for patterns in text.

- With text mining, the information to be extracted is clearly and explicitly stated in the text.

Text mining (2)

- It’s not hidden at all—most authors go to great pains to make sure that they express themselves clearly and unambiguously
- The problem, of course, is that the information is not couched in a manner that is amenable to automatic processing.
- Text mining strives to bring it out of the text in a form that is suitable for consumption by computers directly, with no need for a human intermediary.
- Text is just as opaque as raw data when it comes to extracting information.
- Another requirement for text mining is that the information extracted should be “potentially useful.”

Text summarization (2)



(a)	25% Leading text extract	Four score and seven years ago our fathers brought forth upon this continent a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal. Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, can long endure. We are met here on a great battlefield of that war.
(b)	25% Another extract	Four score and seven years ago our fathers brought forth upon this continent a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal. Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, can long endure. The brave men, living and dead, who struggled here, have consecrated it far above our poor power to add or detract.
(c)	15% Abstract	This speech by Abraham Lincoln commemorates soldiers who laid down their lives in the Battle of Gettysburg. It reminds the troops that it is the future of freedom in America that they are fighting for.
(d)	15% Critical abstract	The Gettysburg address, though short, is one of the greatest American speeches. Its ending words are especially powerful—"that government of the people, by the people, for the people, shall not perish from the earth."

Figure 1 Applying text summarization to the Gettysburg Address

Text summarization (1)

- A text summarizer strives to produce a condensed representation of its input, intended for human consumption.
- It may condense individual documents or groups of documents.
- As a field, summarization differs from many other forms of text mining in that there are people, namely professional abstractors, who are skilled in the art of producing summaries and carry out the task as part of their professional life.
- Studies of these people and the way they work provide valuable insights for automatic summarization.
- Useful distinctions can be made between different kinds of summaries; some are exemplified in Figure 1.

Text summarization (3)

- An *extract* consists entirely of material copied from the input—for example, one might simply take the opening sentences of a document (Figure 1a) or pick certain key sentences scattered throughout it (Figure 1b).

- In contrast, an *abstract* contains material that is not present in the input, or at least expresses it in a different way—this is what human abstractors would normally produce (Figure 1c).

- An *indicative* abstract is intended to provide a basis for selecting documents for closer study of the full text, whereas an *informative* one covers all the salient information in the source at some level of detail.

Text summarization (4)

- A further category is the *critical abstract* , which evaluates the subject matter of the source document, expressing the abstractor’s views on the quality of the author’s work (Figure 1d).

- Another distinction is between a *generic* summary, aimed at a broad readership, and a t*opic focused* one, tailored to the requirements of a particular group of users.

Document retrieval (1)

- Given a corpus of documents and a user’s information need expressed as some sort of query, document retrieval is the task of identifying and returning the most relevant documents.

- Traditional libraries provide catalogues (whether physical card catalogues or computerized information systems) that allow users to identify documents based on surrogates consisting of *metadata*—salient features of the document such as author, title, subject classification, subject headings, keywords.

- Metadata is a kind of highly structured document summary, and successful methodologies have been developed for manually extracting metadata and for identifying relevant documents based on it, methodologies that are widely taught in library school.

Document retrieval (2)

- Automatic extraction of metadata (e.g. subjects, language, author, key-phrases; see below) is a prime application of text mining techniques.

- Contemporary automatic document retrieval techniques bypass the metadata creation stage and work on the full text of the documents directly

- The basic idea is to index every individual word in the document collection.

- Effectively, documents are represented as a “bag of words”—that is, the set of words that they contain, along with a count of how often each one appears in the document.

Document retrieval (3)

- There are some practical problems: how to define a “word,” what to do with numbers; these are invariably solved by simple *ad hoc* heuristics.

- Many practical systems discard common words or “stop words”, primarily for efficiency reasons.

- A query is expressed as a set, or perhaps a Boolean combination, of words and phrases, and the index is consulted for each word in the query to determine which documents satisfy the query.

Document retrieval (4)

- Web search engines are no doubt the most widely-used of document retrieval systems.
- However, search queries are typically restricted to just a few words or phrases—usually one or two.
- In contrast, queries made by professionals to advanced document retrieval systems are often far more complex and specific.

Page Rank (1)

- **PageRank is an algorithm used by Google** Search to rank websites in their search engine results.

- PageRank is a way of measuring the importance of website pages.

- According to Google:
  - PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is.

  - The underlying assumption is that more important websites are likely to receive more links from other websites.

Page Rank (2)

- PageRank is a link analysis algorithm and it assigns a numerical weighting to each element of a hyperlinked set of documents, such as the World Wide Web, with the purpose of "measuring" its relative importance within the set.

- The numerical weight that it assigns to any given element E is referred to as the PageRank of E and denoted by PR(E) Other factors like Author Rank can contribute to the importance of an entity.

Page Rank (3)

- A PageRank results from a mathematical algorithm based on the webgraph, created by all World Wide Web pages as nodes and hyperlinks as edges, taking into consideration authority hubs such as cnn.com or usa.gov.

- The rank value indicates an importance of a particular page.

- A hyperlink to a page counts as a vote of support.

- The PageRank of a page is defined recursively and depends on the number and PageRank metric of all pages that link to it ("incoming links").

- A page that is linked to by many pages with high PageRank receives a high rank itself.

Page Rank Algorithm (1)

- The PageRank algorithm outputs a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page.

- PageRank can be calculated for collections of documents of any size.

- It is assumed in several research papers that the distribution is evenly divided among all documents in the collection at the beginning of the computational process.

- The PageRank computations require several passes, called "iterations", through the collection to adjust approximate PageRank values to more closely reflect the theoretical true value.

Page Rank Algorithm (2)

- A probability is expressed as a numeric value between 0 and 1. A 0.5 probability is commonly expressed as a "50% chance" of something happening.

- Hence, a PageRank of 0.5 means there is a 50% chance that a person clicking on a random link will be directed to the document with the 0.5 PageRank.

Page Rank Algorithm (3)

- Assume a small universe of four web pages: **A, B, C and D.**
- Links from a page to itself, or multiple outbound links from one single page to another single page, are ignored.
- PageRank is initialized to the same value for all pages.
- PageRank, assume a probability distribution between 0 and 1.
- Hence the initial value for each page in this example is 0.25.

Page Rank Algorithm (4)

- The PageRank transferred from a given page to the targets of its outbound links upon the next iteration is divided equally among all outbound links.
- If the only links in the system were from pages B, C, and D to A, each link would transfer 0.25 PageRank to A upon the next iteration, for a total of 0.75.

$$PR(A) = PR(B) + PR(C) + PR(D)$$

Page Rank Algorithm (5)

- Suppose instead that page B had a link to pages C and A, page C had a link to page A, and page D had links to all three pages.
- Thus, upon the first iteration, page B would transfer half of its existing value, or 0.125, to page A and the other half, or 0.125, to page C.
- Page C would transfer all of its existing value, 0.25, to the only page it links to, A.
- Since D had three outbound links, it would transfer one third of its existing value, or approximately 0.083, to A.
- At the completion of this iteration, page A will have a PageRank of approximately 0.458.

$$PR(A) = \frac{PR(B)}{2} + \frac{PR(C)}{1} + \frac{PR(D)}{3}$$

Page Rank Algorithm (6)

- In other words, the PageRank conferred by an outbound link is equal to the document's own PageRank score divided by the number of outbound links L( )

$$PR(A) = \frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)}$$

- In the general case, the PageRank value for any page u can be expressed as:

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)},$$

- i.e. the PageRank value for a page u is dependent on the PageRank values for each page v contained in the set Bu (the set containing all pages linking to page u), divided by the number L(v) of links from page v.

Page Rank Algorithm (7)

- **Damping factor**
- The PageRank theory holds that an imaginary surfer who is randomly clicking on links will eventually stop clicking.

- The probability, at any step, that the person will continue is a damping factor d.

- Various studies have tested different damping factors, but it is generally assumed that the damping factor will be set around 0.85.

- The damping factor is subtracted from 1 (and in some variations of the algorithm, the result is divided by the number of documents (N) in the collection) and this term is then added to the product of the damping factor and the sum of the incoming PageRank scores. That is

$$PR(A) = \frac{1-d}{N} + d\left(\frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)} + \cdots\right)$$

Mining Time Series Data(1)

- A sequence is an ordered list of events
- In time-series data, sequence data consist of long sequences of numeric data, recorded at equal time intervals (e.g., per minute, per hour, or per day).
- Time-series data can be generated by many natural and economic processes such as stock markets, and scientific, medical, or natural observations.
- Time-series databases are popular in many applications such as stock market analysis, economic and sales forecasting, budgetary analysis, utility studies, inventory studies, yield projections, workload projections, and process and quality control.
- They are also useful for studying natural phenomena (e.g., atmosphere, temperature, wind, earthquake), scientific and engineering experiments, and medical treatments.

Mining Time Series Data(2)

- **Similarity Search in Time-Series Data**
- Unlike normal database queries, which find data that match a given query exactly, a similarity search finds data sequences that differ only slightly from the given query sequence.
- Many time-series similarity queries require subsequence matching, that is, finding a set of sequences that contain subsequences that are similar to a given query sequence.
- For similarity search, it is often necessary to first perform data or dimensionality reduction and transformation of time-series data.
- Typical dimensionality reduction techniques include (1) the discrete Fourier transform (DFT), (2) discrete wavelet transforms (DWT), and (3) singular value decomposition (SVD) based on principle components analysis (PCA).

Mining Time Series Data(3)

- With such techniques, the data or signal is mapped to a signal in a transformed space.
- A small subset of the “strongest” transformed coefficients are saved as features. These features form a feature space, which is a projection of the transformed space.
- Indices can be constructed on the original or transformed time-series data to speed up a search.
- For a query-based similarity search, techniques include normalization transformation, atomic matching, window stitching, and subsequence ordering
- Numerous software packages exist for a similarity search in time-series data.

Trend Analysis in Time-Series Data (4)

- Trend analysis builds an integrated model using the following four major components or movements to characterize time-series data:

- 1. Trend or long-term movements:
  - These indicate the general direction in which a time-series graph is moving over time, for example, using weighted moving average and the least squares methods to find trend curves such as the dashed curve indicated in Figure 13.2.

- 2. Cyclic movements:
  - These are the long-term oscillations about a trend line or curve.

Mining Time Series Data(5)

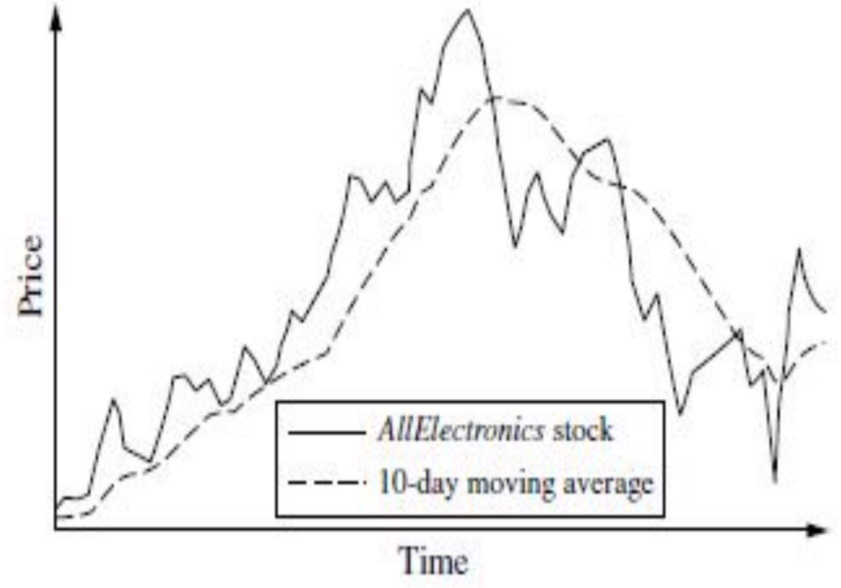


Figure 13.2 Time-series data for the stock price of *AllElectronics* over time. The *trend* is shown with a dashed curve, calculated by a moving average.

Mining Time Series Data(6)

- 3. Seasonal variations:
  - These are nearly identical patterns that a time series appears to follow during corresponding seasons of successive years such as holiday shopping seasons.
  - For effective trend analysis, the data often need to be “deseasonalized” based on a seasonal index computed by autocorrelation.

- 4. Random movements:
  - These characterize sporadic changes due to chance events such as labor disputes or announced personnel changes within companies.

- Trend analysis can also be used for time-series forecasting, that is, finding a mathematical function that will approximately generate the historic patterns in a time series, and using it to make long-term or short-term predictions of future values.

Ensemble Methods (1)

- An ensemble for classification is a composite model, made up of a combination of classifiers.
- The individual classifiers vote, and a class label prediction is returned by the ensemble based on the collection of votes.
- Ensembles tend to be more accurate than their component classifiers.
- Bagging, boosting and random forests are popular ensemble methods.
- An ensemble combines a series of *k* learned models (or *base classifiers*), *M*1, *M*2, ... , *Mk*, with the aim of creating an improved composite classification model, *M**.

Ensemble Methods (2)

- A given data set, *D*, is used to create *k* training sets, *D*1, *D*2, ..., *Dk*, where *Di* (1 <= *i*<= *k* -1) is used to generate classifier *Mi* .
- Given a new data tuple to classify, the base classifiers each vote by returning a class prediction.
- The ensemble returns a class prediction based on the votes of the base classifiers.
- An ensemble tends to be more accurate than its base classifiers
- The ensemble will misclassify ***X*** only if over half of the base classifiers are in error

Ensemble Methods (3)

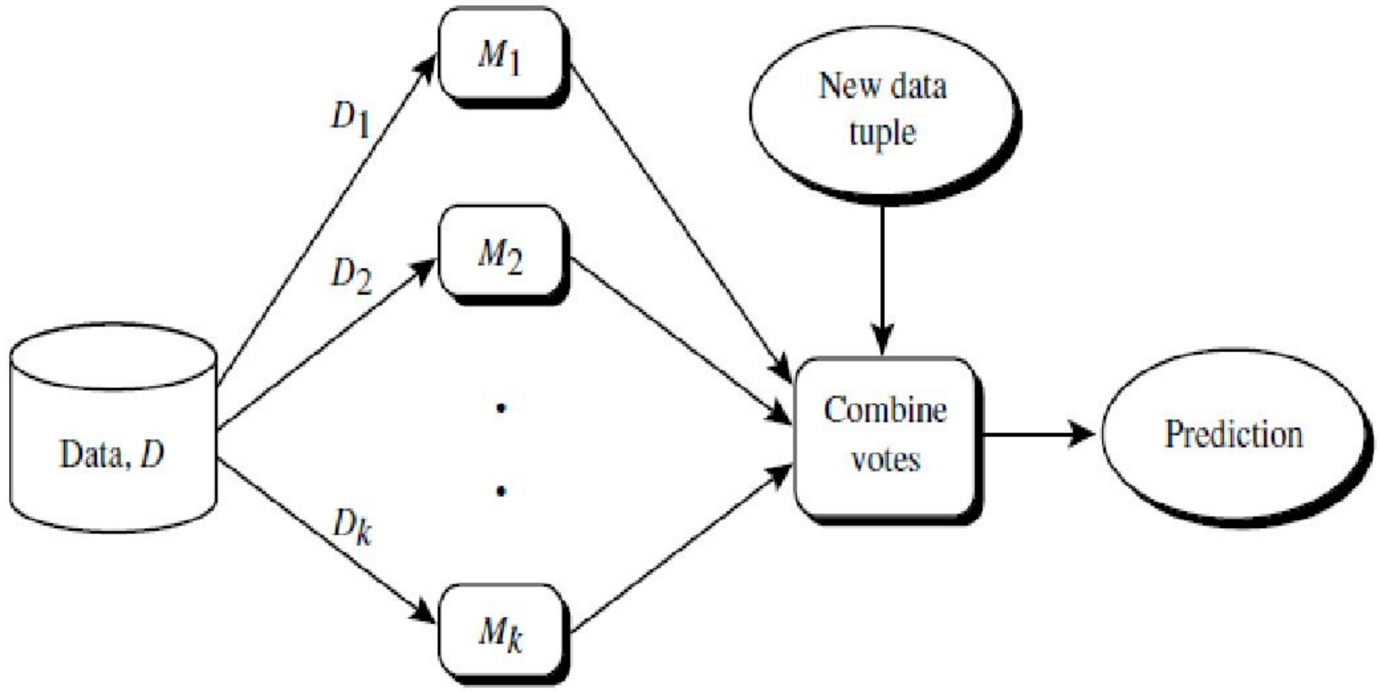


Figure 8.21 Increasing classifier accuracy: Ensemble methods generate a set of classification models, $M_1, M_2, \ldots, M_k$. Given a new data tuple to classify, each classifier "votes" for the class label of that tuple. The ensemble combines the votes to return a class prediction.

Bagging (1)

- Suppose that you are a patient and would like to have a diagnosis made based on your symptoms.
- Instead of asking one doctor, you may choose to ask several.
- If a certain diagnosis occurs more than any other, you may choose this as the final or best diagnosis.
- That is, the final diagnosis is made based on a majority vote, where each doctor gets anequal vote.
- Now replace each doctor by a classifier, and you have the basic idea behind bagging.
- Intuitively, a majority vote made by a large group of doctors may be more reliable than a majority vote made by a small group.

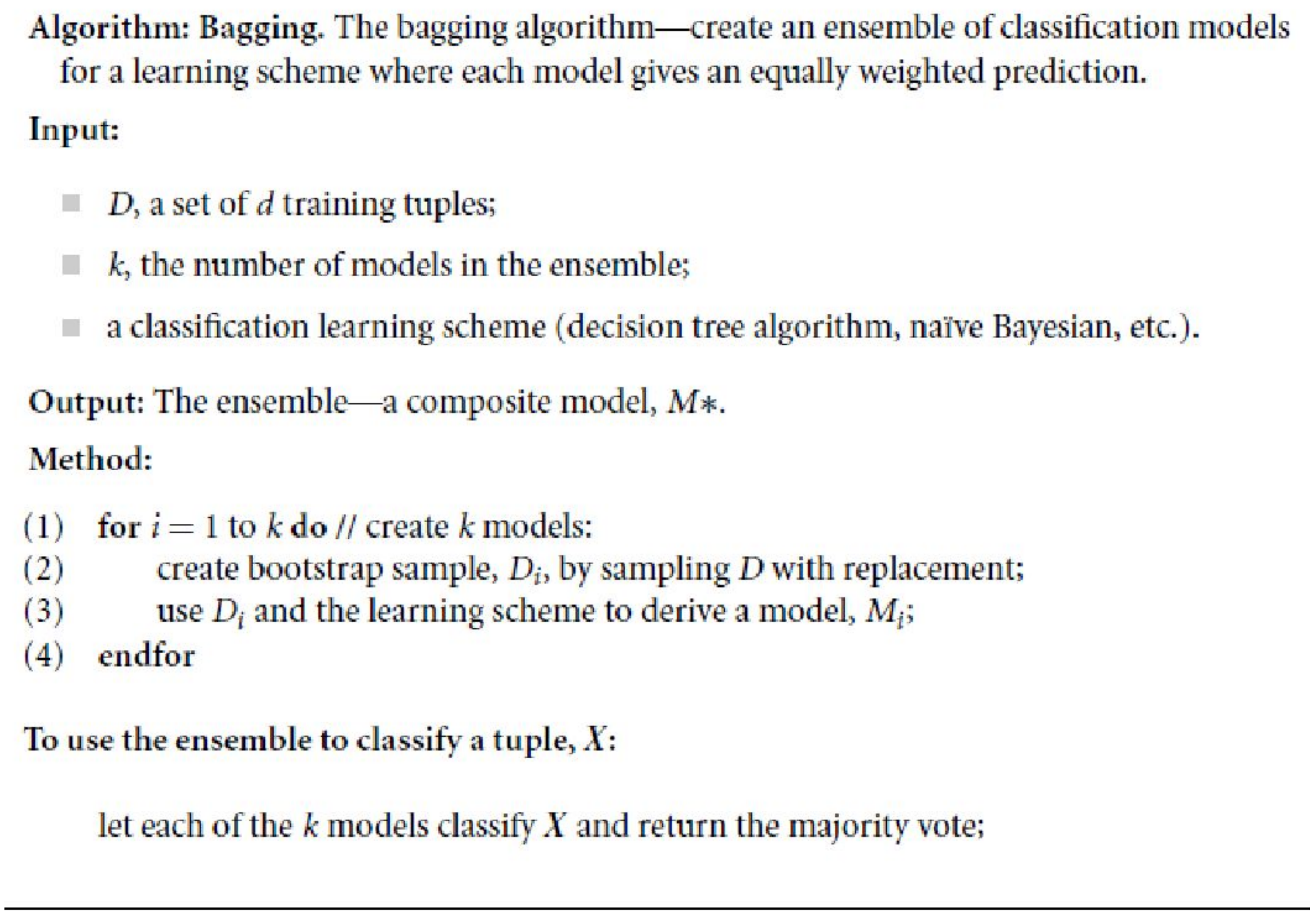
Bagging (2)

- Given a set, *D*, of *d* tuples, **bagging** works as follows.
- For iteration i (i= 1, 2,.... , k), a training set, *Di* , of *d* tuples is sampled with replacement from the original set of tuples, *D*.
- Note that the term *bagging* stands for *bootstrap aggregation*.
- Each training set is a bootstrap sample.
- Because sampling with replacement is used, some of the original tuples of *D* may not be included in *Di*, whereas others may occur more than once.
- A classifier model, *Mi* , is learned for each training set, *Di* .

Bagging (3)

- To classify an unknown tuple, ***X***, each classifier, *Mi* , returns its class prediction, which counts as one vote.
- The bagged classifier, *M**, counts the votes and assigns the class with the most votes to ***X***.
- Bagging can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple.
- The bagged classifier often has significantly greater accuracy than a single classifier derived from *D*, the original training data.
- It is more robust to the effects of noisy data and overfitting.

Bagging (4)



Algorithm: Bagging. The bagging algorithm—create an ensemble of classification models for a learning scheme where each model gives an equally weighted prediction.

Input:

- D, a set of d training tuples;
- k, the number of models in the ensemble;
- a classification learning scheme (decision tree algorithm, naïve Bayesian, etc.).

Output: The ensemble—a composite model, $M*$.

Method:

```
(1)  for $i = 1$ to $k$ do // create $k$ models:
(2)      create bootstrap sample, $D_i$, by sampling $D$ with replacement;
(3)      use $D_i$ and the learning scheme to derive a model, $M_i$;
(4)  endfor
```

To use the ensemble to classify a tuple, X:

let each of the k models classify X and return the majority vote;

Figure 8.23 Bagging.

Boosting and AdaBoost (1)

- Suppose that as a patient, you have certain symptoms.
- Instead of consulting one doctor, you choose to consult several.
- Suppose you assign weights to the value or worth of each doctor's diagnosis, based on the accuracies of previous diagnoses they have made.
- The final diagnosis is then a combination of the weighted diagnoses.
- This is the essence behind boosting.
- In **boosting**, weights are also assigned to each training tuple.

Boosting and AdaBoost (2)

- A series of *k* classifiers is iteratively learned.
- After a classifier, *Mi* , is learned, the weights are updated to allow the subsequent classifier, *Mi*+1, to “pay more attention” to the training tuples that were misclassified by *Mi* .
- The final boosted classifier, *M**, combines the votes of each individual classifier, where the weight of each classifier’s vote is a function of its accuracy.
- **AdaBoost** (short for Adaptive Boosting) is a popular boosting algorithm.

Boosting and AdaBoost (3)

- Suppose we want to boost the accuracy of a learning method.
- We are given D, a data set of d class-labeled tuples, (X1, y1), (X2, y2),... , (Xd, yd), where yi is the class label of tuple Xi.
- Initially, AdaBoost assigns each training tuple an equal weight of 1/d.
- Generating k classifiers for the ensemble requires k rounds through the rest of the algorithm.
- In round i, the tuples from D are sampled to form a training set, Di , of size d.
- Sampling with replacement is used—the same tuple may be selected more than once.

Boosting and AdaBoost (4)

- Each tuple's chance of being selected is based on its weight.
- A classifier model, *Mi* , is derived from the training tuples of *Di* .
- Its error is then calculated using *Di* as a test set.
- The weights of the training tuples are then adjusted according to how they were classified.
- If a tuple was incorrectly classified, its weight is increased.
- If a tuple was correctly classified, its weight is decreased.
- A tuple's weight reflects how difficult it is to classify— the higher the weight, the more often it has been misclassified.

Boosting and AdaBoost (5)

- These weights will be used to generate the training samples for the classifier of the next round.
- The basic idea is that when we build a classifier, we want it to focus more on the misclassified tuples of the previous round.
- In this way, we build a series of classifiers that complement each other.
- To compute the error rate of model *Mi* , we sum the weights of each of the tuples in *Di* that *Mi* misclassified.
- That is

$$error(M_i) = \sum_{j=1}^{d} w_j \times err(X_j),$$

where err(Xj) is the misclassification error of tuple Xj: If the tuple was misclassified, then err(Xj) is 1; otherwise, it is 0.

Boosting and AdaBoost (6)

- If the performance of classifier *Mi* is so poor that its error exceeds 0.5, then we abandon it.
- Instead, we try again by generating a new *Di* training set, from which we derive a new *Mi* .
- The error rate of *Mi* affects how the weights of the training tuples are updated.
- If a tuple in round *i* was correctly classified, its weight is multiplied by
  error(*Mi) / (*1-*error*(*Mi))*.
- Once the weights of all the correctly classified tuples are updated, the weights for all tuples (including the misclassified ones) are normalized so that their sum remains the same as it was before.
- To normalize a weight, we multiply it by the sum of the old weights, divided by the sum of the new weights.
- As a result, the weights of misclassified tuples are increased and the weights of correctly classified tuples are decreased.

Boosting and AdaBoost (7)

*"Once boosting is complete, how is the ensemble of classifiers used to predict the class label of a tuple, **X**?"*

- Unlike bagging, where each classifier was assigned an equal vote, boosting assigns a weight to each classifier's vote, based on how well the classifier performed.

- The lower a classifier's error rate, the more accurate it is, and therefore, the higher its weight for voting should be.

- The weight of classifier *Mi* 's vote is $$log\frac{1-error(M_i)}{error(M_i)}.$$

- For each class, *c*, we sum the weights of each classifier that assigned class *c* to ***X***.

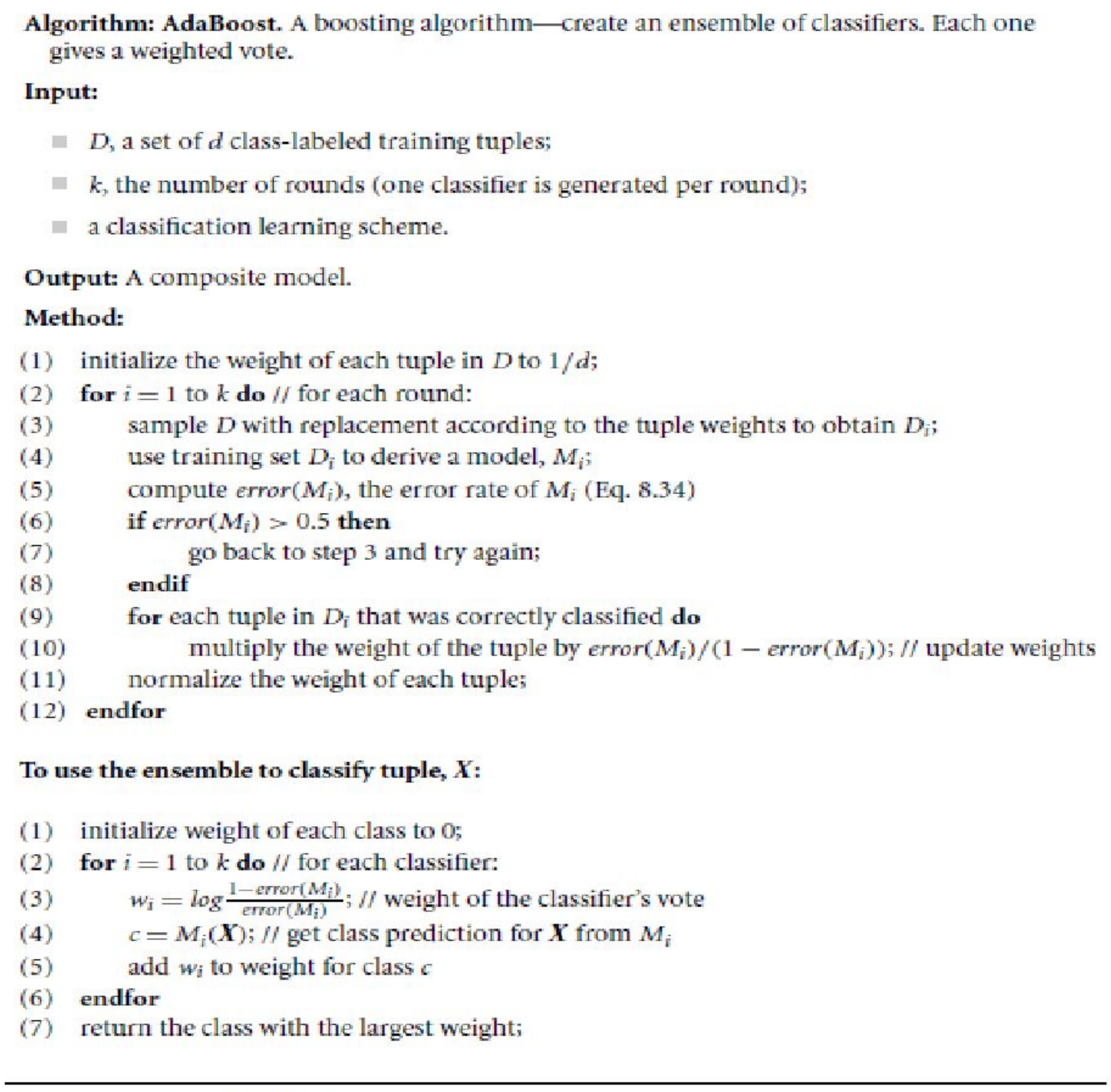
- The class with the highest sum is the "winner" and is returned as the class prediction for tuple ***X***.

Boosting and AdaBoost (8)

"How does boosting compare with bagging?"

- Because of the way boosting focuses on the misclassified tuples, it risks overfitting the resulting composite model to such data.
- Therefore, sometimes the resulting "boosted" model may be less accurate than a single model derived from the same data.
- Bagging is less susceptible to model overfitting.
- While both can significantly improve accuracy in comparison to a single model, boosting tends to achieve greater accuracy.

Boosting and AdaBoost (9)



Algorithm: AdaBoost. A boosting algorithm—create an ensemble of classifiers. Each one gives a weighted vote.

Input:

- D, a set of d class-labeled training tuples;
- k, the number of rounds (one classifier is generated per round);
- a classification learning scheme.

Output: A composite model.

Method:

```
(1)  initialize the weight of each tuple in $D$ to $1/d$;
(2)  for $i = 1$ to $k$ do // for each round:
(3)      sample $D$ with replacement according to the tuple weights to obtain $D_i$;
(4)      use training set $D_i$ to derive a model, $M_i$;
(5)      compute $error(M_i)$, the error rate of $M_i$ (Eq. 8.34)
(6)      if $error(M_i) > 0.5$ then
(7)          go back to step 3 and try again;
(8)      endif
(9)      for each tuple in $D_i$ that was correctly classified do
(10)         multiply the weight of the tuple by $error(M_i)/(1 - error(M_i))$; // update weights
(11)     normalize the weight of each tuple;
(12) endfor
```

To use the ensemble to classify tuple, X:

```
(1)  initialize weight of each class to 0;
(2)  for $i = 1$ to $k$ do // for each classifier:
(3)      $w_i = log\frac{1-error(M_i)}{error(M_i)}$; // weight of the classifier's vote
(4)      $c = M_i(X)$; // get class prediction for $X$ from $M_i$
(5)      add $w_i$ to weight for class $c$
(6)  endfor
(7)  return the class with the largest weight;
```

Figure 8.24 AdaBoost, a boosting algorithm.

Random Forests

- Imagine that each of the classifiers in the ensemble is a *decision tree* classifier so that the collection of classifiers is a “forest.”

- The individual decision trees are generated using a random selection of attributes at each node to determine the split.

- More formally, each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest

- During classification, each tree votes and the most popular class is returned.

- Random forests can be built using bagging in tandem with random attribute selection.

References

- Research paper titled “Mining the World Wide Web { Methods, Applications, and Perspectives }” by Andreas Hotho, Gerd Stumme
- PageRank From Wikipedia, the free encyclopedia
- Research paper titled “Text mining” by Ian H. Witten Computer Science, University of Waikato, Hamilton, New Zealand
- Jiawei Han and Micheline Kamber: Data Mining Concepts and Techniques, Elsevier, 3rd Edition.
- Xindong Wu and Vipin Kumar: The top ten Algorithms in Data Mining, Chapman and Hall/CRC press.
- Research paper titled “Data Mining in Genomics” by Jae K. Lee, Paul D. Williams, and Sooyoung Cheon

END OF UNIT5