Welcome

Data Mining from Large Databases

Xindong Wu
Dept of Math and Computer Science
Colorado School of Mines
Golden, Colorado 80401, USA

Email: xwu@kais.mines.edu
Home Page: http://kais.mines.edu/~xwu/
Outline

■ Why Large Databases?
■ How Large is “Very Large”?
■ Data Partitioning vs Data Sampling
■ Techniques to Deal with Large Databases
  – windowing
  – bagging and boosting
  – stacked generalization
  – batch learning and multi-layer induction
  – hierarchical meta-learning
  – aggregation of data mining rules
Why Large Databases?

■ **Existence of very large data sets.** Supermarket transactions, scientific and medical repositories, etc.

■ **Better accuracy with larger training data.** To avoid overfitting problems.

■ Meanwhile, dealing with larger data sets requires more computing resources.
How Large is “Very Large”?

According to Provost and Kolluri (1999),

- Machine Learning: 100,000 instances with a couple dozen features, or more
- Databases: 100 gigabytes, or larger (difficult to be processed simultaneously)
- KDD (from an algorithmic perspective): one million examples (100Mbyte-1Gbyte range)
Data Partitioning vs Data Sampling

■ Data partitioning
  – Break a large DB up into subsets, learn from one or more of these subsets, and possibly combine the results.

■ Random sampling
  – Randomly select a small subset of the DB for data mining.

■ Stratified sampling
  – Each class is appropriately represented in the subset.

■ Progressive sampling
  – Start with small samples and progressively increase them as long as model accuracy improves.

■ Cross validation
  – Some partitions for training and others for testing.
Windowing

- Quinlan’s **windowing** technique (1983)
  - Start with a small random sample (called a *window*), and generate a classifier for the window.
  - Test the classifier on the remaining examples, and check whether the quality (accuracy) of the classifier.
  - If the quality is not sufficient, add a set of mis-classified examples to the window and generate a new classifier.

- Windowing might not improve efficiency [Wirth & Catlett 1988].

- **Integrative windowing** (Furnkranz 1998)
  - Integrate good rules into the final theory right after their discovery to avoid re-learning them.
  - Remove examples from the window if they are covered by consistent rules.
Bagging and Boosting

Learning “ensembles of classifiers”
- Generate multiple versions of a predictor by running an existing learning algorithm many times on a set of re-sampled data.
- A data item in the original database can be used in many samples for generating different versions of the predictor.
- Bagging: a fixed size for each bag
- Boosting: incorrectly classified examples replace correctly classified ones (or get a higher weight)
- C5.0: boosting with all training data by adjusting weights of examples
Stacked Generalization

- [Wolpert 1992]
  - Uses a high-level model to combine lower-level models to achieve greater predictive accuracy.
  - Lower-level models are generated by sampling the original database
  - Cross-validation is used to generate higher-level data for the high-level model.
Multi-Layer Induction

- [Wu and Lo 1998]: Subsets (or samples) of a database are processed one by one
  - **Data Partitioning**
  - **Generalization**: A set of rules is learned.
  - **Reduction**: Behavioral examples are derived.
  - From behavioral examples, generalization can extract new rules, which are expected to correct defects and inconsistencies of previous rules.
  - Successive applications of the above generalization-reduction process allow more accurate and more complex (because of disjunctive) rules to be discovered, by sequentially handling the subsets of examples
Incremental Batch Learning
[Clearwater et al 1989]

- Rules are generated on each batch of examples
- "Almost-good-rules" are collected
- An additional filter (such as Gen-Spec method and the $n$-Best) is applied
- A RL technique (derived from the Meta-DENDRAL) is used to refine the remaining rules on the next batch of examples.
Meta-Learning [Chan 1997]

– Partition a large database into subsets, run a learning algorithm on each of the subsets, and combine the results in some principled fashion.
– Since the learning processes are independent, they can be run in parallel for speed up over a sequential system.
– Arbiters and combiners
Aggregation of Data Mining Rules

- Partition a large database into different subsets, or start with different data sources
- Discover rules in each subset or data source
- Aggregate the rules from different data sources
  - Assign weight to each data source/subset based on the number of high-belief rules it supports (a high-belief rule is supported by most data sources)
  - Aggregate rules by combining their support and confidence