Pattern Recognition - Lecture 10 Rule-Based Classifier

Dr. Dewan Md. Farid

Associate Professor, Department of Computer Science & Engineering
United International University, Bangladesh

April 22, 2018

Rule-based Classifier

Mining big data

Rule-based Classifier

Rule-based classifier is easy to deal with complex classification problems. It has various advantages:

- Highly expressive as decision tree (DT)
- Easy to interpret
- Easy to generate
- Can classify new instances rapidly
- Performance comparable to DT
- New rules can be added to existing rules without disturbing ones already in there
- Rules can be executed in any order

Adaptive Rule-based Classifier

It combines the **random subspace** and **boosting** approaches with **ensemble of decision trees** to construct a set of classification rules for multi-class classification of biological big data.

- Random subspace method (or attribute bagging) to avoid **overfitting**
- Boosting approach for classifying **noisy instances**
- Ensemble of decision trees to deal with **class-imbalance data**

It uses two popular classification techniques: **decision tree (DT)** and **k-nearest-neighbour (kNN)** classifiers.

- DTs are used for evolving classification rules from the training data.
- kNN is used for analysing the misclassified instances and removing vagueness between the contradictory rules.

Random Subspace & Boosting Method

Random subspace is an ensemble classifier. It consists of several classifiers each operating in a subspace of the original feature space, and outputs the class based on the outputs of these individual classifiers.

- It has been used for decision trees (random decision forests).
- It is an attractive choice for high dimensional data.

Boosting is designed specifically for classification.

- It converts weak classifiers to strong ones.
- It is an iterative process.
- It uses voting for classification to combine the output of individual classifiers.

Decision Tree Induction

Decision tree (DT) induction is a top down recursive divide and conquer algorithm for multi-class classification task. The goal of DT is to iteratively partition the data into smaller subsets until all the subsets belong to a single class. It is easy to interpret and explain, and also requires little prior knowledge.

- Information Gain: ID3 (Iterative Dichotomiser) algorithm
- Gain Ratio: C4.5 algorithm
- Gini Index: CART algorithm

Algorithm 1 Decision Tree Induction

Input: $D = \{x_1, \cdots, x_i, \cdots, x_N\}$
Output: A decision tree, DT.
Method:

1: $DT = \emptyset$;
2: find the root node with best splitting, $A_j \in D$;
3: DT = create the root node;
4: DT = add arc to root node for each split predicate and label;
5: **for** each arc **do**
6: D_j created by applying splitting predicate to D;
7: **if** stopping point reached for this path, **then**
8: DT' = create a leaf node and label it with c_l;
9: **else**
10: $DT' = \text{DTBuild}(D_j)$;
11: **end if**
12: DT = add DT' to arc;
13: **end for**

K-Nearest-Neighbour (kNN) Classifier

The **k-nearest-neighbour (kNN)** is a simple classifier. It uses the distance measurement techniques that widely used in pattern recognition. kNN finds k instances, $X = \{x_1, x_2, \cdots, x_k\} \in D_{training}$ that are closest to the test instance, x_{test} and assigns the most frequent class label, $c_l \rightarrow x_{test}$ among the X. When a classification is to be made for a new instance, x_{new}, its distance to each $A_j \in D_{training}$, must be determined. Only the k closest instances, $X \in D_{training}$ are considered further. The *closest* is defined in terms of a distance metric, such as Euclidean distance. The Euclidean distance between two points, $x_1 = (x_{11}, x_{12}, \cdots, x_{1n})$ and $x_2 = (x_{21}, x_{22}, \cdots, x_{2n})$, is shown in Eq. 1

$$dist(x_1, x_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2} \quad (1)$$

Algorithm 2 k-Nearest-Neighbour classifier

Input: $D = \{x_1, \cdots, x_i, \cdots, x_n\}$
Output: kNN classifier, *kNN*.
Method:

1: find $X \in D$ that identify the k nearest neighbours, *regardless* of class label, c_l.
2: out of these instances, $X = \{x_1, x_2, \cdots, x_k\}$, identify the number of instances, k_i, that belong to class c_l, $l = 1, 2, \cdots, M$. Obviously, $\sum_i k_i = k$.
3: assign x_{test} to the class c_l with the maximum number of k_i of instances.

Constructing Classification Rules

Extracting classification rules from DTs is easy and well-known process. Rules are highly expressive as DT, so the performance of rule-based classifier is comparable to DT.

- Each rule is generated for each leaf of the DT.
- Each path in DT from the root node to a leaf node corresponds with a rule.
- Tree corresponds exactly to the classification rules.

DT vs. Rules

New rules can be added to an existing rule set without disturbing ones already there, whereas to add to a tree structure may require reshaping the whole tree. Rules can be executed in any order.

Algorithm: Adaptive rule-based (ARB) classifier

- It considers a series of k iterations.
- Initially, an equal weight, $\frac{1}{N}$ is assigned to each training instance.
- The weights of training instances are adjusted according to how they are classified in every iterations.
- In each iteration, a sub-dataset D_j is created from the original training dataset D and previous sub-dataset D_{j-1} with maximum weighted instances. Only the sampling with replacement technique is used to create the sub-dataset D_1 from the original training data D in the first iteration.
- A tree DT_j is built from the sub-dataset D_j with randomly selected features in each iteration.
- Each rule is generated for each leaf node of DT_j.
- Each path in DT_j from the root to a leaf corresponds with a rule.

Algorithm 3 Adaptive rule-based classifier.

Input:
$D = \{x_1, \cdots, x_i, \cdots, x_N\}$, *training dataset;*
k, number of iterations;
DT learning scheme;
Output: *rule-set*; // A set of classification rules.
Method:

1: *rule-set* $= \emptyset$;
2: **for** i $= 1$ to N **do**
3: $x_i = \frac{1}{N}$; // initialising weights of each $x_i \in D$.
4: **end for**
5: **for** j $= 1$ to k **do**
6: **if** j==1 **then**
7: create D_j, by sampling D with replacement;
8: **else**
9: create D_j, by D_{j-1} and D with maximum weighted X;
10: **end if**
11: build a tree, $DT_j \leftarrow D_j$ by randomly selected features;
12: compute $error(DT_j)$; // the error rate of DT_j.
13: **if** $error(DT_j) \geq$ threshold-value **then**
14: go back to step 6 and try again;
15: **else**
16: *rules* $\leftarrow DT_j$; // extracting the rules from DT_j.
17: **end if**
18: **for** each $x_i \in D_j$ that was correctly classified **do**
19: multiply the weight of x_i by $(\frac{error(DT_j)}{1-error(DT_j)})$; // update weights.
20: **end for**
21: normalise the weight of each $x_i \in D_j$;
22: *rule-set* $=$ *rule-set* $\cup$ *rules*;
23: **end for**
24: return *rule-set*;
25: create sub-dataset, $D_{misclassified}$ with misclassified instances from D_j;
26: analyse $D_{misclassified}$ employing algorithm 4.

Error Rrate Calculation

The *error rate* of DT_j is calculated by the sum of weights of misclassified instances that is shown in Eq. 2. Where, $err(x_i)$ is the misclassification error of an instance x_i. If an instance, x_i is misclassified, then $err(x_i)$ is one. Otherwise, $err(x_i)$ is zero (correctly classified).

$$error(DT_j) = \sum_{i=1}^{n} w_i \times err(x_i) \tag{2}$$

If *error rate* of DT_j is less than the *threshold-value*, then rules are extracted from DT_j.

Reduced-Error Pruning

- Split the original data into two parts: (a) **a growing set**, and (b) **a pruning set**.
- Rules are generated using growing set only. So, important rules might miss because some key instances had been assigned to the pruning set.
- A rule generated from the growing set is deleted, and the effect is evaluated by trying out the truncated rule from the pruning set and seeing whether it performs well than the original rule.
- If the new truncated rule performs better then this new rule is added to the rule set.
- This process continues for each rule and for each class.
- The overall best rules are established by evaluating the rules on the pruning set.

Algorithm: Analysing Misclassified Instances

To check the classes of misclassified instances we used the kNN classifier with feature selection and weighting approach.

- We applied DT induction for feature selection and weighting approach.
- We build a tree from the misclassified instances.
- Each feature that is tested in the tree, $A_j \in D_{misclassified}$ is assigned by a weight $\frac{1}{d}$. Where d is the depth of the tree.
- We do not consider the features that are not tested in the tree for similarity measure of kNN classifier.
- We apply kNN classifier to classify each misclassified instance based on the weighted features.
- We update the class label of misclassified instances.
- We check for the contradictory rules, if there is any.

Algorithm 4 Analysing misclassified instances

Input: D, original training data;
$D_{misclassified}$, dataset with misclassified instances;
Output: A set of instances, X with right class labels.
Method:

1: build a tree, DT using $D_{misclassified}$;
2: **for** each $A_j \in D_{misclassified}$ **do**
3: **if** A_j is tested in DT **then**
4: assign weight to A_j by $\frac{1}{d}$, where d is the depth of DT;
5: **else**
6: not to consider, A_j for similarity measure;
7: **end if**
8: **end for**
9: **for** each $x_i \in D_{misclassified}$ **do**
10: find $X \in D$, with the similarity of weighted $A = \{A_1, \cdots, A_j, \cdots, A_n\}$;
11: find the most frequent class, c_l, in X;
12: assign $x_i \leftarrow c_l$;
13: **end for**

Performance Measurement

The classification accuracy:

$$accuracy = \frac{\sum_{i=1}^{|X|} assess(x_i)}{|X|}, x_i \in X \tag{3}$$

If x_i is correctly classified then $assess(x_i) = 1$, or If x_i is misclassified then $assess(x_i) = 0$.

$$precision = \frac{TP}{TP + FP} \tag{4}$$

$$recall = \frac{TP}{TP + FN} \tag{5}$$

$$F - score = \frac{2 \times precision \times recall}{precision + recall} \tag{6}$$

What is Big Data?

- Mining big data is the process of extracting knowledge to uncover large hidden information from the massive amount of complex data or databases.
- The data in big data comes in different forms including two-dimensional tables, images, documents and complex records from multiple sources.
- It must support search, retrieval and analysis.

The three V's define big data: *Volume* (the quantity of data), *Variety* (the category of data) and *Velocity* (the speed of data in and out). It might suggest throwing a few more V's into the mix: *Vision* (having a purpose/ plan), *Verification* (ensuring that the data conforms to a set of specifications) and *Validation* (checking that its purpose is fulfilled).

Big Data Vs. Business Intelligence

- Regarding data and their uses, there is a big difference between big data and business intelligence (BI).
- BI provides historical, current and predictive views of business operations, which includes reporting, online analytical processing, business performance management, competitive intelligence, benchmarking and predictive analytics.
- But in big data, the data is processed employing advanced ML and DM algorithms to extract the meaningful information/ knowledge from the data.
- In big data mining applications, very large training sets of millions of instances are common. Most often the training data will not fit in memory. The efficiency of existing ML and DM algorithms, such as DT, NB and kNN, has been well established for relatively small data sets. These algorithms become inefficient due to swapping of the training instances in and out of main and cache memories.

Mining Big Data with Rules

- Big data is so big (millions of instances) that we cannot process all the instances together at the same time.
- It is not possible to store all the data in the main memory at a time.
- We can create several smaller sample (or subsets) of data from the big data that each of which fits in main memory.
- Each subset of data is used to construct a set of rules, resulting in several sets of rules.
- Then the rules are examined and used to merge together to construct the final set of classification rules to deal with big data. As we have the advantage to add new rules with existing rules and rules are executed in any order.

Mining Big Data with Rules (con.)

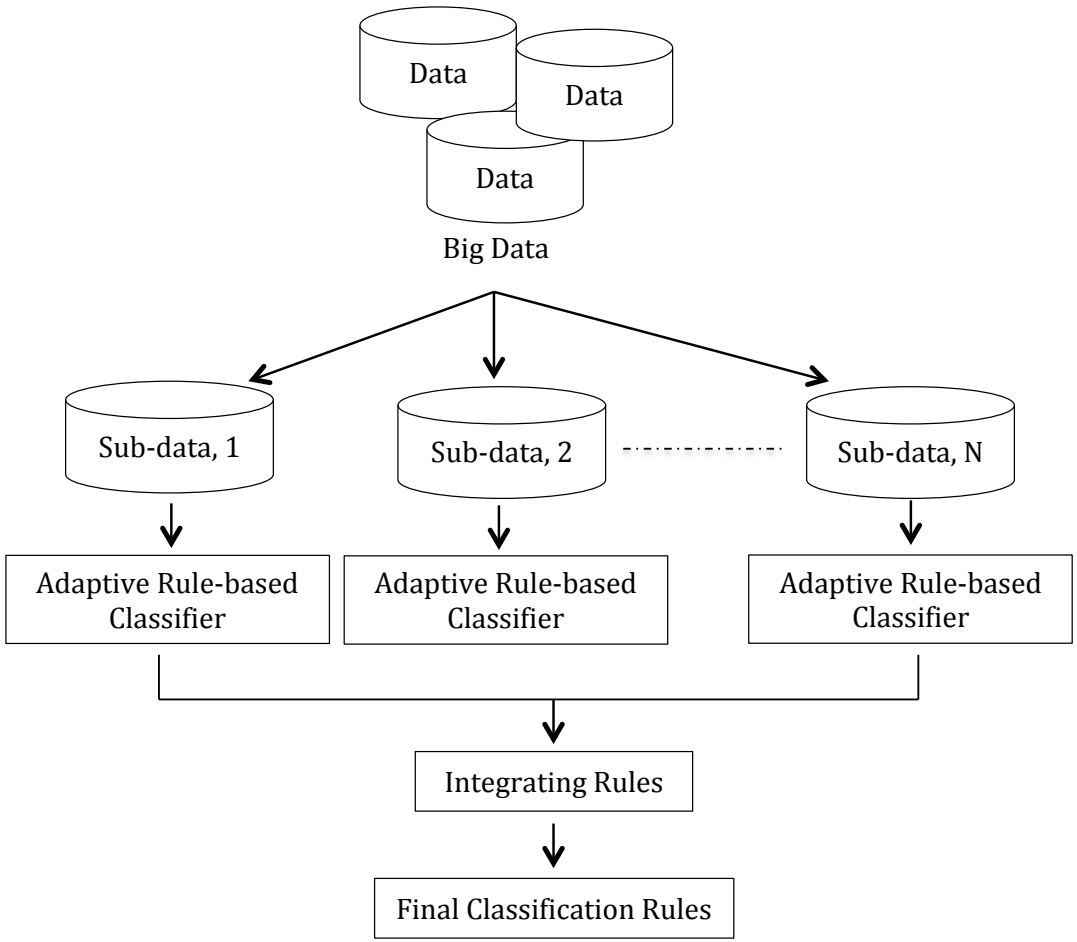

Figure: Mining big data using adaptive rule-based classifier.

*** THANK YOU ***