Journal of Biomedical Informatics 44 (2011) S17–S23

Contents lists available at ScienceDirect

Journal of Biomedical Informatics

journal homepage: www.elsevier.com/locate/yjbin

Transfer learning of classification rules for biomarker discovery and verification from molecular profiling studies

Philip Ganchev [a,*], David Malehorn [b], William L. Bigbee [b], Vanathi Gopalakrishnan [a,c]

[a] Intelligent Systems Program, University of Pittsburgh, Pittsburgh, PA, United States
[b] Department of Pathology, University of Pittsburgh, Pittsburgh, PA, United States
[c] Department of Biomedical Informatics, University of Pittsburgh, Pittsburgh, PA, United States

ARTICLE INFO

Article history:
Available online 6 May 2011

Keywords:
Biomarker discovery
Molecular profiling
Machine learning
Rule learning
Transfer learning

ABSTRACT

We present a novel framework for integrative biomarker discovery from related but separate data sets created in biomarker profiling studies. The framework takes prior knowledge in the form of interpretable, modular rules, and uses them during the learning of rules on a new data set. The framework consists of two methods of transfer of knowledge from source to target data: transfer of whole rules and transfer of rule structures. We evaluated the methods on three pairs of data sets: one genomic and two proteomic. We used standard measures of classification performance and three novel measures of amount of transfer. Preliminary evaluation shows that whole-rule transfer improves classification performance over using the target data alone, especially when there is more source data than target data. It also improves performance over using the union of the data sets.

1. Introduction

Molecular profiling data is used extensively to learn classifiers, such as rule models, and discover biomarkers for early detection, diagnosis and prognosis of diseases. Biomarkers are also critical for furthering understanding of disease mechanisms and creating treatments. The aim in biomarker discovery is to find a small set of measured variables that can be used to accurately predict a disease state. This is particularly challenging because typically we must choose among tens or hundreds of thousands of variables, representing molecules in complex mixtures, often with high measurement error. Also, data sets are typically very small, usually tens or hundreds of patients in a study. All these factors make statistical analyses more error-prone.

Fortunately, there are often multiple similar studies, each producing a data set. In order to draw on all the available data, researchers typically analyze each data set separately, then compare the biomarkers discovered [1,2]. However, this is sub-optimal because the analysis is still done on the separate small data sets. A simple way to combine the data is to use the union of the data sets. But such attempts are typically confounded by variability in sample processing and by systematic measurement error specific to each data set. For example, the same numerical measurement might mean a high abundance of some protein in one data set but low abundance in another data set.

We propose a novel framework for transfer learning, called Transfer Rule Learner (TRL), that is particularly well-suited to biomarker discovery. Transfer learning is the use of data from one learning task to help in learning another task [3]. Pan and Yang [4] offer a survey of transfer learning. Various methods for transfer learning have been applied to various domains, such as part-of-speech tagging [5] and leaf classification [6]. Transfer learning for biomarker discovery is promising because previous studies have found reproducibility of information collected in different experimental sessions when using the same protocols [1,2]. Unfortunately, many transfer learning frameworks typically produce classifiers that are difficult for human users to understand or that use many variables [3,5], which makes them less useful for biomarker discovery. Unlike other transfer learning methods, TRL transfers knowledge in the form of modular, interpretable classification rules, and uses them to seed learning of a new classifier on a new data set. Rule learning has the advantage that variable selection is embedded in the learning algorithm, and the new model uses only a few of the many measured variables to explain the data.

TRL is an extension of the classification rule learning algorithm RL [7], which been used successfully to solve biomedical problems for more than three decades [8–11] and in the past decade has been adapted and used for biomarker profiling [12–17].

We demonstrate our method on five clinical data sets, and find that more often than not, transfer learning improves performance over using one data set alone, and even more often over learning on the union of the data sets. We evaluate the methods using standard performance measures and three novel measures of transfer.

* Corresponding author.
E-mail addresses: phil.ganchev@gmail.com (P. Ganchev), vanathi@pitt.edu (V. Gopalakrishnan).

1532-0464/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jbi.2011.04.009

To our knowledge, this is the first effort to apply transfer of rules or rule structure between related biomedical data sets.

2. Materials and methods

Our transfer learning framework is based on the classification rule learner RL [7]. Models learned by RL are simple to understand and can represent non-linear relationships. RL covers data with replacement (see Section 2.1), which is beneficial when training data are scarce.

Fig. 1 shows an overview of our transfer learning approach. A source data set is used to train a set of prior rules that are then used as seeds for learning on the target data. Section 2.1 provides a brief overview of RL, which is useful in understanding transfer learning with TRL described in Sections 2.2, 2.3, 2.4.

2.1. Rule learning with RL

RL is a classification learning algorithm that outputs a rule-based classifier. RL's input is a set of training data instances, where each instance is a vector of values for the input variables, and a class value. The learned classifier comprises a set of rules of the form:

`IF < antecedent > THEN < consequent >`

where the `antecedent` consists of a conjunction of one or more variable-value pairs (conjuncts), and the `consequent` is a prediction of the class variable. For example, a rule learned from proteomic mass spectra might be:

`IF((mz.2.05=High) AND (mz.9.65=Low)) THEN Class=Control`

which is interpreted as "if the variable for m/z 2.05 kilo Daltons (kDa) has a value in the `High` range and the m/z 9.65 has a value in the `Low` range, then predict the class value Control." Values such as `Low` and `High` represent intervals of real numbers that result from discretizing the variables before training with RL. (See Section 2.2.) A rule is said to *cover* or *match* a data instance if each variable value of the instance is in the range specified in the rule antecedent. RL *covers data with replacement*, which means that multiple rules are allowed to cover the same training instance. This is unlike most other classification rule and tree learning algorithms, such as C4.5 [18] and CART [19], which cover data without replacement, so that each data instance is covered by only one rule. In small sample size data sets, covering with replacement allows RL to utilize more of the available evidence for each rule when computing the generalizability of the rule.

The classifier also includes an *evidence gathering* method for breaking ties when the antecedents of several rules are met but their consequents are different. We use the default evidence gathering method: voting weighted by the rules' certainty factor values.

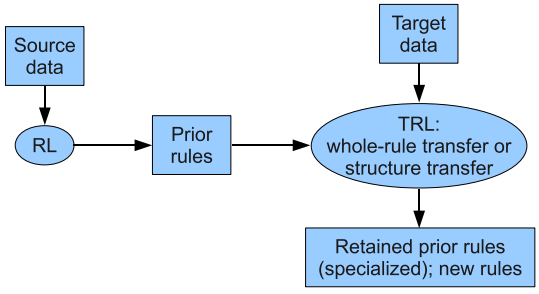

Fig. 1. The TRL framework.

RL is shown in Algorithm 1. The input is a set of data instance vectors and a set of values for the learning parameters specified by the user. The parameters define constraints on the acceptable rules in terms of a number of quantities defined with respect to a rule and a data set. The constraints are minimum coverage, minimum certainty factor value, maximum false positive rate, and inductive strengthening. *Coverage* is the fraction of training examples for which the rule antecedent is satisfied. An additional parameter is the certainty factor function. The *Certainty factor* function (CF) is a measure of the rule's accuracy; several alternative certainty factor functions are defined and implemented in RL, and the specific function to use can be specified as a parameter to the algorithm. As a CF function, we used the *true positive rate*: the number of examples the rule predicts correctly divided by the number of examples it matches. *False positive rate* is the number of examples the rule predicts incorrectly divided by the number of examples it matches. *Inductive strengthening* is a bias toward training new rules that cover uncovered training instances. Specifically, the parameter specifies the minimum number of previously uncovered examples that a proposed rule must cover. The smaller this number, the larger the overlap of instances covered by different rules. Because RL covers data with replacement, using some non-zero inductive strengthening helps to learn a more generalizable model. *Maximum conjuncts* is the maximum number of variable-value pairs allowed in the antecedent of any rule.

The algorithm proceeds as a heuristic beam search through the space of rules from general to specific [20]. Starting with all rules containing no variable-value pairs, it iteratively specializes the rules by adding conjuncts to the antecedent. It evaluates the rules and inserts promising rules onto the beam, sorted by decreasing certainty factor value. Beam search is used to limit the running time and space of the algorithm.

2.2. Transfer of rules

Algorithm 1. TRL. Differences from RL are underlined. Function `import()` takes a list of prior rules and removes from them any variables that do not appear in the data set. `satisfies()` checks if the rule satisfies the user-specified constraints. The second call to `satisfies()` checks only the minimum-coverage constraint because coverage of any specialized rule will be equal or smaller. `specialize()` creates all non-redundant specialized rules by adding to the original rule antecedent single variable-value pairs from the data set.

Input : data, a set of training instance vectors
Input : priorRules, a list of prior rules
Parameters: constraints, constraints on acceptable rules
Parameters: minCoverage, minimum-coverage constraint
beam ← `import(`*priorRules*`)`+[$\emptyset \Rightarrow$ `class`$_1$, $\emptyset \Rightarrow$ `class`$_2$...];
beam ← `sort`(beam);
model ← [];
while beam *is not empty* **do**
  beam$_{new}$ ← [];
  foreach rule ∈ beam **do**
    if `satisfies`(rule, constraints, data) **then**
      model ← model+[rule];
      beam$_{new}$ ← beam$_{new}$ + `specialize`(rule);
    else if `satisfies`(rule, minCoverage, data) **then**
      beam$_{new}$ ← beam$_{new}$ + `specialize`(rule);
    end
    beam ← `sort`(beam$_{new}$);
  end
end
Return model