On Meta-Learning Rule Learning Heuristics

Frederik Janssen and Johannes Fürnkranz

TU Darmstadt, Knowledge Engineering Group
Hochschulstraße 10, D-64289 Darmstadt, Germany

E-mail: `[janssen,juffi]@ke.informatik.tu-darmstadt.de`

Abstract

The goal of this paper is to investigate to what extent a rule learning heuristic can be learned from experience. To that end, we let a rule learner learn a large number of rules and record their performance on the test set. Subsequently, we train regression algorithms on predicting the test set performance of a rule from its training set characteristics. We investigate several variations of this basic scenario, including the question whether it is better to predict the performance of the candidate rule itself or of the resulting final rule. Our experiments on a number of independent evaluation sets show that the learned heuristics outperform standard rule learning heuristics. We also analyze their behavior in coverage space.

1 Introduction

It is well-known that learning a classification rule is essentially a search problem [10, 3], where the states are rules and the successor function is a refinement operator that returns all minimal specializations of a rule. The goal is to find a rule that maximizes the predictive performance in a domain. As this performance cannot be directly measured, an evaluation function is used to estimate the quality of a rule. Typically, the same evaluation function is also used as a *search heuristic* that allows a greedy search algorithm to focus on interesting parts of the hypothesis space. The long-term goal of our research is to understand the properties of such search heuristics. The underlying (implicit) assumption is that a rule with high quality will also produce good refinements. There has not been much work on trying to characterize the behavior of good heuristics. Notable exceptions include [9], where *weighted relative accuracy* (WRA) is proposed as a novel heuristic, and [6], in which a wide variety of rule evaluation metrics were analyzed and compared by visualizing their behavior in ROC space.

In previous work [7], we adjusted parameters of three heuristics, whose shape was predetermined. The key idea of this work is to meta-learn such a heuristic from experience, without a bias towards existing measures. To this end, we create a large meta data set which we use to learn a function that predicts the performance of a rule on an independent test set. In order to address the issue that a good rule evaluation function does not necessarily coincide with a good search heuristic for finding a rule that optimizes the evaluation function, we also analyze a setting in which the learner attempts to predict the performance of a *complete* rule from its incomplete predecessors.

2 Rule Learning Algorithm

For the purpose of this empirical study, we implemented a simple *Separate-and-conquer* (or *Covering*) rule learning algorithm [3] within the *Weka* machine learning environment [14]. Both the outer loop (the covering procedure) and the top-down refinement inside the learner are fairly standard. For details about the implementation see [4, 3].

Separate-and-conquer rule learning can be divided into two main steps: First, a rule is learned from the training data by a greedy search (the *conquer* step). Second, all examples covered by the learned rule are removed from the data set (the *separate* step). Then, the next rule is learned on the remaining examples. Both steps are repeated as long as positive examples are left in the training set. The refinement procedure, which is used inside the conquer step of the algorithm, returns all possible candidate refinements that can be obtained by adding a single condition to the body of the rule. For nominal attributes, conditions test for equal-

The original paper was published in the Proceedings of the 7th IEEE International Conference on Data Mining (ICDM-07), pages 529–534, 2007

ity with a domain value, for numerical attributes they use $>$ and $\leq$. The best among all refinements is selected.

Our implementation continues to greedily refine the current rule until no more negative examples are covered. In this case, the search stops and the best rule encountered during the refinement process is added to the theory (which is initialized as the empty theory). Note that this is not necessarily the last rule searched. We use random tie breaking for rules with equal evaluation, and filter out candidate rules that do not cover any positive examples. Rules are added to the theory as long as this increases the accuracy of the theory on the training set (this is the case when the best rule found covers more positive than negative examples).

We did not use any specific pruning technique, but solely relied on the evaluation of the rules by the used heuristic. Note, however, that this does not mean that we learn an overfitting theory that is complete and consistent on the training data (i.e., a theory that covers all positive and no negative examples), because many heuristics will prefer impure rules with a high coverage over pure rules with a lower coverage.

Multi-class problems are tackled by sorting the classes according to their frequency (least frequent first), and training binary classifiers that discriminate a class from all subsequent classes. The classifiers are then used in this order, which essentially means that a decision list is formed, in which the rules for each class appear in blocks of increasing class frequencies.

3 Rule Learning Heuristics

Numerous heuristics have been provided for inductive rule learning, a general survey can be found in [3]. Most rule learning heuristics can be seen as functions of the following four arguments:

- P and N: the number of positive/negative examples in the training set
- p and n: the number of positive/negative examples covered by the rule

Examples of heuristics of this type are the commonly used heuristics that are shown in Table 1.

As P and N are constant for a given learning problem, these heuristics effectively only differ in the way they trade off completeness (maximizing p) and consistency (minimizing n), and may thus be viewed as a function $h(p, n)$. As a consequence, each rule can be considered as a point in coverage space, a variant of ROC space that uses the absolute numbers of true positives and false positives as its axes. The preference bias of different heuristics may then be visualized by plotting the respective heuristic values of the rules on top of their positions in coverage space, resulting

Table 1. Rule learning heuristics used in this paper ($\sim$ denotes order equivalency)

precision	$= \frac{p}{p+n} \sim \frac{p-n}{p+n}$
accuracy	$= \frac{p+(N-n)}{P+N} \sim p - n$
Laplace	$= \frac{p+1}{p+n+2}$
WRA	$= \frac{p+n}{P+N}\left(\frac{p}{p+n} - \frac{P}{P+N}\right) \sim \frac{p}{P} - \frac{n}{N}$
correlation	$= \frac{p(N-n)-(P-p)n}{\sqrt{PN(p+n)(P-p+N-n)}}$

in a 3-dimensional plot $(p, n, h(p, n))$. A good way to view this graph in two dimensions is to plot the *isometrics* of the learning heuristics, i.e., to show contour lines that connect rules with identical heuristic evaluation values [6]. Another method is to plot both contour lines and the surface of the function, what we have done in our visualization (cf. Section 5.4).

The goal of our work is to automatically learn a function $h(p, n)$, which allows to predict the quality of a learned rule. However, note that most of the functions in Table 1 contain some non-linear dependencies between these values. In order to make the task for the learner easier, we will not only characterize a rule by the values p, n, P, and N, but in addition also use the following parameters as input for the meta-learning phase:

- $tpr = \frac{p}{P}$, the true positive rate of the rule
- $fpr = \frac{n}{N}$, the false positive rate of the rule
- $Prior = \frac{P}{P+N}$, the a priori distribution of positive and negative examples
- $prec = \frac{p}{p+n}$, precision, the fraction of positive examples covered by the rule

Thus, we decided to characterize a rule r by an 8-tuple $< P, N, Prior, p, n, tpr, fpr, prec >$. Some heuristics use additional components, such as the length of the rule, or the number of positive and negative examples that are covered by the rule's predecessor. We have also performed experiments that include the length of the rule (which can be found in the long version of this paper [8]), but this did not have a noticable effect on the performance. We think that the main goal of using the length of a rule is to indirectly capture the degree of generality of a rule (shorter rules cover more examples), which can be directly measured with p and n.

We will not consider statistics about a rule's predecessor, as our goal is to find a function that allows to evaluate a rule, irrespective of how it has been learned. Including them may result in different evaluations for the same rule, depending on the order in which its conditions have been added to the rule body. An example of such a heuristic is FOIL's information gain. We will also not include it in our performance measures because it actually measures the quality of refinements and not the quality of rules, which means that it cannot be used to select an optimal rule without the use of additional stopping criteria.

4 Meta-Learning Scenario

We frame the rule learning process as a search problem in the following way: Each (incomplete) rule is a state, and all possible refinements (e.g., all possible conditions that can be added to the rule) are the actions. The rule-learning agent repeatedly has to pick one of the possible refinements according to their expected utility until it has completed the learning of a rule.

In this framework, the problem of meta-learning a rule learning heuristic may be considered as a reinforcement learning problem: After learning a complete theory, the learner receives a reinforcement signal (e.g., the estimated accuracy of the learned theory), which can then be used to adjust the utility function. After a (presumably large) number of learning episodes, the utility function should converge to a heuristic that evaluates a candidate rule with the quality of the *best* rule that can be obtained by refining the candidate rule. However, for practical purposes this scenario appears to be too complex. In [2] a reinforcement learning algorithm was applied on this problem, but with disappointing results.

For this reason, we redefine the problem as a supervised learning task: Each rule is evaluated on a separate test set, in order to get an estimate of its true performance. This information is then used as the target value for rules that are characterized with the eight features discussed in Section 3. We studied both, immediate reward (where rules are trained on their own test set performance) and delayed reward (where rules are trained on the performance of their best refinement; cf. Section 5.3).

As explained above, we try to model the relation of the rule's statistics measured on the training set and its "true" performance, which is estimated on an independent test set. Therefore, we used the rule learner described above for obtaining the induced 8-tuple for each learned rule. This characterization form a training instance in the meta data set. The training signals are the performance parameters of the rule on the test set.

As we want to guide the entire rule learning process, we need to record this information not only for final rules — those that would be used in the final theory — but also for all their predecessors. Therefore all candidate rules which are created during the refinement process are included in the meta data as well. The Algorithm for creating the meta data is described in detail in [8].

It should be noted, that we ignored all rules that do not cover any instance on the test data. Our reasons for this were that on the one hand we did not have any training information for this rule , and on the other hand such rules do not do any harm (they will not have an impact on test set accuracy as they do not classify any example). In total, our meta dataset contains $87,380$ examples.

To ensure that we obtain a set of rules with varying characteristics, the following parameters were modified:

Datasets: We used 27 datasets with varying characteristics (different number of classes, attributes, instances) from the UCI Repository [12] (for a list see [8]).

5x2 Cross-validation: For each dataset, we performed 5 iterations of a 2-fold cross-validation. 2-fold cross-validation was chosen because in this case the training and test sets have equal size, so that we do not have to account for statistical variance in the precision or coverage estimates.

Classes: For each dataset and each fold, we generated one dataset for each class, treating this class as the positive one and the union of all the others as the negative class. Rules were learned for each of the resulting two-class datasets.

Heuristics: We ran the rule learner several times on the binary datasets, each time using a different search heuristic (displayed in Table 1). The first four form a representative selection of search heuristics with linear ROC space isometrics [5], while the correlation heuristic has non-linear isometrics. These heuristics represent a large variety of learning biases.

We used two different methods for learning functions on the meta data. First, we applied a simple *linear regression* based on the Akaike criterion [1] for model selection. A key advantage of this method is that we obtain a simple, easily comprehensible form of the learned heuristic function. Note that the learned model is anyhow non-linear in the basic dimensions p and n because of the non-linear terms that are used as basic features (e.g., $p/(p+n)$). Nevertheless, the type of functions that can be learned with linear

Table 2. Accuracies and theory complexities for several methods

method	MAE	Accuracy	# conditions
LinearRegression	0.22	77.43%	117.6
MLP (1 node)	0.28	77.81%	121.3
MLP (5 nodes)	0.27	77.37%	1085.8
MLP (10 nodes)	0.27	77.53%	112.7

Table 3. Accuracy and theory complexity comparison of various heuristics with training-set (p, n) and predicted ($\hat{p}, \hat{n}$) coverages (number of conditions in brackets)

args	Accuracy		Precision		WRA		Laplace		Correlation	
(p, n)	75.60%	(104.77)	76.22%	(129.17)	75.80%	(12.13)	76.89%	(118.83)	77.57%	(47.5)
$(\hat{p}, \hat{n})$	75.39%	(110.8)	76.53%	(30)	69.89%	(29.97)	76.80%	(246.8)	58.09%	(40.4)

regression is quite restricted. In order to be able to address a wider class of functions, we used *multilayer perceptrons* with back propagation algorithm and sigmoid nodes. We applied various sizes of the hidden layer (1, 5, and 10), and trained for one epoch (i.e., we went through the training data once). We have also tried to train the networks with a larger number of epochs, but the results did not improve. We used the algorithms which are implemented in *Weka* [14], initialized with the standard parameters.

A straight-forward approach to measure the fit of the learned function to the target values is to estimate its mean absolute error by a 10-fold cross validation on the meta data.

$$MAE(f, \hat{f}) = \frac{1}{m} \sum_{i=0}^{m} |\hat{f}(i) - f(i)|$$

where m denotes the number of instances, $f(i)$ the actual value, and $\hat{f}(i)$ the predicted value of instance i.

Note, however, that a low prediction error on the meta data set does not necessarily imply that the function works good as heuristic (cf. Table 2). Thus, our primary method for evaluating the learned heuristics is to use these heuristics inside the rule learner. To this end, we evaluate the rule learner on 30 UCI data sets, which have not been used during the training phase (for a list see [8]). Like the 27 data sets on which the rules for the meta data are induced, these 30 sets have varying characteristics to ensure that our method will perform well under a wide variety of conditions. On each dataset, the rule learner with the learned heuristics was evaluated with one iteration of a 10-fold cross validation. The performance over all sets was then averaged. We also evaluated the length of the learned theories in terms of number of conditions.

5 Results

5.1 Predicting Test-Set Precision

We are first interested in how accurately the out-of-sample precision of a rule can be predicted. We train a linear regression model and a neural network on the eight measurements that we use for characterizing a rule (cf. Section 3) using the test set precision as the target function. Table 2 displays results for three different neural networks, with varying numbers of nodes in the hidden layer and the linear regression. The performances of the four algorithms are quite comparable, with the possible exception of the neural network with 5 nodes in the hidden layer, which induced very large theories (over 1000 conditions on average), and also had a somewhat worse performance in predictive accuracy.

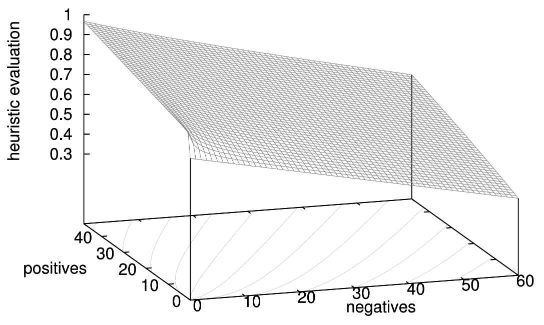
The following function was learned by the linear regression method:

$$\begin{aligned} h_{LR} \quad = \quad & 0.0001 \cdot P + 0.0001 \cdot N + 0.7485 \cdot {}^{P}\!/_{(P+N)} \\ & -0.0001 \cdot p - 0.0009 \cdot n + 0.3863 \cdot {}^{p}\!/_{(p+n)} \\ & +0.165 \cdot {}^{p}\!/_{P} + 0 \cdot {}^{n}\!/_{N} + 0.0267 \end{aligned}$$

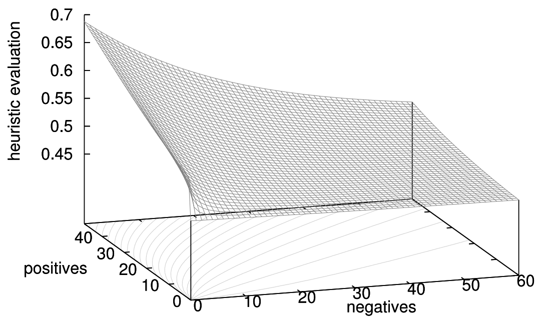
The most important feature was the *a priori* distribution of the examples in the training data followed by the precision of the rule. Interestingly, while the *tpr* has a non-negligible influence on the result, the *fpr* is practically ignored. Both the current coverage of a rule (p and n) and the total example counts of the data (P and N) have comparably low weights. This is not that surprising if one keeps in mind that the target value is in the range $[0, 1]$, while the absolute values for p and n are in a much higher range. We nevertheless had included them because we believe that in particular for rules with low coverage, the absolute numbers are more important than their relative fractions. A rule that covers only a single example will typically be bad, irrespective of the size of the original dataset.

5.2 Predicting Coverage

So far we focused on directly predicting the out-of-sample precision of a rule, assuming that this would be a good heuristic for learning a rule set. However, this choice was somewhat arbitrary. Ideally, we would like to repeat this experiment with out-of-sample values for all common rule learning heuristics. In order to cut down the number of needed experiments, we decided to directly predict the number of covered positive ($\hat{p}$) and negative ($\hat{n}$) examples. We then can combine the predictions for these values with any standard heuristic h by computing $h(\hat{p}, \hat{n})$ instead of the conventional $h(p, n)$. Note that the heuristic h only gets the



(a) Linear Regression



(b) Neural Network

Figure 1. Isometrics of the functions (final rule precision)

predicted coverages ($\hat{p}$ and $\hat{n}$) as new input, all other statistics (e.g., P,N) are still measured on the training set. This is feasible because we designed the experiments so that the training and test set are of equal size, i.e., the values predicted for $\hat{p}$ and $\hat{n}$ are predictions for the number of covered examples on an independent test set of the same size as the training set.

Table 3 compares the performance of various heuristics with measured and predicted coverage values on the 30 test sets. In general, the results are disappointing. For three of the five heuristics, no significant change could be observed, but for *WRA* and the *Correlation* heuristic, the performance degrades substantially. A rather surprising observation is the complexity of the learned theories. For instance, the heuristic *Precision* produces very simple theories when it is used with the out-of-sample predictions, and, by doing so, increases the predictive accuracy. Apparently, the use of the predicted values of $\hat{p}$ and $\hat{n}$ allows to prevent overfitting, because the predicted positive/negative coverages are never exactly 0 and therefore the overfitting problem observed with *Precision* does not occur any more. In summary, it seems that the predictions of both the linear regression and the neural network are not good enough to yield true coverage values on the test set. A closer look at the predicted values reveals that on the one hand both regression methods predict negative coverages and that on the other hand for the region of low coverages (which is the important one) too optimistic values are predicted. The acceptable performance is caused by a balancing of the two imprecise predictions (as observed with the two precision-like metrics) or rather by an induced bias which tries to omit the extreme values in the evaluations (which are responsible for overfitting).

5.3 Predicting the Value of the Final Rule

Rule learning heuristics typically evaluate the quality of the current, incomplete rule, and use this measure for greedily selecting the best candidate for further refinement. However, if we frame the learning problem as a search problem, a good heuristic should not evaluate a candidate rule with its discriminatory power, but with its potential to be refined into a good final rule. To take this into account, we applied a method which can be interpreted as an "offline" version of reinforcement learning. We simply assign each candidate rule the precision value of its final rule in one refinement process. As a consequence, all candidate rules of one refinement process have the same target value, namely the value of the rule that has eventually been selected. Because of the deletion of all final rules that do not cover any example on the test set, we decided to remove all predecessors of such rules as well. Thus, the new meta data set contains only 77,240 examples in total.

The neural network performs best with an accuracy of 78.37% followed by the linear regression which achieves 77.95% on the 30 sets used for testing. The neural network also has less conditions in average than the linear regression (53.97 vs. 95.63). Both induced heuristics outperform all of the standard heuristics (cf. Table 3). The linear regression was trained on the meta data set that only contains the 4 most important features (cf. Section 5.1). In terms of theory complexity it seems that about 50 conditions in average are necessary to obtain an accurate classifier.

5.4 Isometrics of the Heuristics

To understand the behavior of the learned heuristics, we follow the framework introduced in [6] and analyze their isometrics in ROC or coverage space. Figure 1 shows 3D-plots of the surface of the learned heuristics in a coverage space with 60x48 examples (the sizes were chosen arbitrarily). The bottom of the graph shows isometric lines that characterize this surface. Figure 1(a) displays the isometrics of the heuristic that was learned by the linear regression (the one that performed best). Figure 1(b) shows the best-performing neural network (the one that uses only one node in the hidden layer).

Apparently, both functions learn somewhat different heuristics. Although the 3D-surfaces looks fairly similar to each other (except for the stronger non-linear behavior of the neural net), the isometric lines reveal that the learned heuristics are, in fact, quite different. The isometrics of the linear regression are comparable to those of weighted relative accuracy (see [5] for an isometric plot), but with a different cost model (i.e. false negatives are more costly than false positives). The isometrics for the neural net seems to employ a trade-off similar to that of the F-measure. The shift towards the N-axis is reminiscent of the F-measure (for an illustration see [7]), which tries to correct the undesirable property of precision that all rules that cover no negative examples are evaluated equally, irrespective of the number of positive examples that they cover.

However, both heuristics have a non-linear shape of the isometrics in common, which bends the lines towards the N-axis. Effectively, this encodes a bias towards rules that cover a low number of positive examples (compared to regular precision). This seems to be a desirable property for a heuristic that is used in a covering algorithm, where incompleteness (not covering all positive examples) is less severe than inconsistency (covering some negative examples), because incompleteness can be corrected by subsequent rules, whereas inconsistency cannot.

6 Conclusion

The most important result of this work is that we have shown that a rule learning heuristic can be learned that outperforms standard heuristics in terms of predictive accuracy on a collection of databases that were not used in the meta-learning phase. Our first results were already en par with the correlation heuristic, which performed best in our experiments (cf. Table 2 and Table 3). These results were achieved with meta data that contains only a few obvious features including the out-of-sample precision as the target value. Subsequently, we tried to modify several parameters of this basic setup with mixed results. In particular, predicting the positive and negative coverage of a rule on a test set, and using these predicted coverage values inside the heuristics did not prove to be successful. Also, more complex neural network architectures did not seem to be important, linear regression and neural networks with a single node in the hidden layer performed best. On the other hand, a key result of this work is that evaluating a candidate rule by its potential of being refined into a good final rule works better than evaluating the quality of the candidate rule itself. This indicates that a clear separation of rule evaluation metrics (which characterize which rules we should look for) and search heuristics (which guide the process of finding such rules) is important. We intend to further investigate this issue.

A visualization of the learned heuristics in coverage space gave some insight into the general functionalities of the learned heuristics. In comparison to heuristics with linear isometrics, the learned heuristics have non-linear isometrics that implement a particularly strong bias towards rules with a low coverage on negative examples. This makes sense for heuristics that will be used in a covering loop, because incompleteness can be compensated by subsequent rules, whereas inconsistency cannot. Correlation, the standard heuristic that performed best in our experiments, implements a similar bias [6]. Thus, the results of this paper also contribute to our understanding of the desirable behavior of rule-learning heuristics.

Our results may also be viewed in the context of trying to correct overly optimistic training error estimates (resubstitution estimates). In particular, in some of our experiments, we tried to directly predict the out-of-sample precision of a rule. This problem has been studied theoretically in [13, 11]. In [4] meta data was also created in a quite similar way, and the authors have tried to fit various functions to the data. But the focus there is the analysis of the obtained predictions for out-of-sample precision, which is not the key issue in our experiments.

Acknowledgements: This research was supported by the *German Science Foundation (DFG)* under grant FU 580/2-1.

References

[1] H. Akaike. A new look at the statistical model selection. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[2] S. Burges. Meta-Lernen einer Evaluierungs-Funktion für einen Regel-Lerner, December 2006. Master's Thesis.

[3] J. Fürnkranz. Separate-and-Conquer Rule Learning. *Artificial Intelligence Review*, 13(1):3–54, February 1999.

[4] J. Fürnkranz. Modeling rule precision. In *LWA*, pages 147–154, 2004.

[5] J. Fürnkranz and P. A. Flach. An Analysis of Rule Evaluation Metrics. In *Proceedings 20th International Conference on Machine Learning (ICML'03)*, pages 202–209. AAAI Press, January 2003.

[6] J. Fürnkranz and P. A. Flach. ROC 'n' Rule Learning - Towards a Better Understanding of Covering Algorithms. *Machine Learning*, 58(1):39–77, January 2005.

[7] F. Janssen and J. Fürnkranz. On trading off consistency and coverage in inductive rule learning. In *LWA*, pages 306–313, 2006.

[8] F. Janssen and J. Fürnkranz. Meta-learning rule learning heuristics. Technical Report. Knowledge Engineering Group, TU Darmstadt. TUD-KE-2007-2, 2007.

[9] N. Lavrač, P. Flach, and B. Zupan. Rule evaluation measures: A unifying view. In *Proceedings of the 9th International Workshop on Inductive Logic Programming (ILP-99)*, pages 174–185, 1999.

[10] T. M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):203–226, 1982.

[11] M. Mozina, J. Demsar, J. Zabkar, and I. Bratko. Why is rule learning optimistic and how to correct it. In *Machine Learning: ECML 2006, 17th European Conference on Machine Learning*, pages 330–340, 2006.

[12] D. Newman, C. Blake, S. Hettich, and C. Merz. UCI Repository of Machine Learning databases, 1998.

[13] T. Scheffer. Finding association rules that trade support optimally against confidence. In *Principles of Data Mining and Knowledge Discovery, 5th European Conference, PKDD 2001*, pages 424–435, 2001.

[14] I. H. Witten and E. Frank. *Data Mining — Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers, 2nd edition, 2005.