Optimizing Feature Sets for Structured Data

Ulrich Rückert and Stefan Kramer

Institut für Informatik/I12, Technische Universität München, Boltzmannstr. 3,
D-85748 Garching b. München, Germany
{rueckert,kramer}@in.tum.de

Abstract. Choosing a suitable feature representation for structured data is a non-trivial task due to the vast number of potential candidates. Ideally, one would like to pick a small, but informative set of structural features, each providing complementary information about the instances. We frame the search for a suitable feature set as a combinatorial optimization problem. For this purpose, we define a scoring function that favors features that are as dissimilar as possible to all other features. The score is used in a stochastic local search (SLS) procedure to maximize the diversity of a feature set. In experiments on small molecule data, we investigate the effectiveness of a forward selection approach with two different linear classification schemes.

1 Introduction

Feature generation and selection for structured data is complicated by the vast number of possible candidate features. In graph classification, for instance, the number of ways to describe a graph is virtually unlimited, and often expert knowledge is necessary to identify relevant aspects of a graph. A popular choice is to represent a graph by features that specify whether or not a subgraph is present. If there is only limited expert knowledge available about relevant subgraph features in an application, the learning algorithm needs to generate meaningful features on its own. Unfortunately, the number of subgraphs grows exponentially with the size of the graphs, so it is clearly infeasible to use all possible subgraphs as features. Therefore, many approaches restrict the set of subgraph features to frequently occurring, frequent closed, or class-correlated subgraphs [3,2]. In all of these cases, however, there is no guarantee that the resulting feature sets have sufficient coverage over all instances of a dataset. Moreover, the resulting features may be only slight alterations of a few subgraphs. As a consequence, the number of features required to reach some level of performance may be unnecessarily high, potentially harming comprehensibility and efficiency. While we focus on graph classification in this paper, the same problems occur with other forms of structured data, for instance, in logic-based representations.

Instead of generating a bulk of features and obtaining a well-balanced coverage only incidentally, it may be worthwhile to actively construct favorable structural features in the first place. One way to minimize the number of features required

J.N. Kok et al. (Eds.): ECML 2007, LNAI 4701, pp. 716–723, 2007.

is to explicitly maximize the diversity of subgraph features. Thus, we frame the search for diverse, complementary sets of subgraph features as an optimization problem. We define a scoring function measuring the diversity of a feature set and present a stochastic local search (SLS) procedure to find subgraphs optimizing that score. Stochastic local search is motivated by the NP-hardness of the problem of finding a perfectly complementary subgraph given a set of graphs and optimal features so far. To focus the search for useful structural features even further, it is possible to extend the scoring function by balancing diversity with class correlation. The resulting feature sets are used in linear classifiers. The effectiveness of the method and its dependence on variations are tested in experiments on three small molecule datasets from cheminformatics.

2 Background and Motivation

We deal with the problem of *feature generation* for linear classifiers. In this setting, the instances are arbitrary objects and we need to find features that extract meaningful information from the objects. In particular, we are interested in features that are well suited for use by a learning algorithm to construct a predictive classifier. Information theory gives us the tools to quantify what constitutes a feature set that is informative about the target class: Let $X \in \{-1,1\}^{m\times n}$ be the training matrix containing m instances $x_1,\ldots,x_m$, where each instance $x_i = (x_i(1),\ldots,x_i(n))$ ranges over n features. X_i denotes the ith column of the training matrix, that is, the instantiation of the ith feature. Assuming a binary classification problem, we denote the class labels of the instances by $Y \in \{-1,1\}^m$. Then we are aiming at features $X_1,\ldots,X_n$ with high mutual information $I(X;Y) = I(X_1,\ldots,X_n;Y)$ between features and target class. We can write the mutual information as the difference between the entropy of X and the conditional entropy of X given Y: $I(X;Y) = H(X) - H(X|Y)$. Thus, in order to obtain highly informative features, we need to maximize $H(X) := H(X_1,\ldots,X_n)$ and to minimize $H(X|Y) := H(X_1,\ldots,X_n|Y)$. This leads to the following three criteria:

- *High correlation with the class.* Since we would like to minimize $H(X|Y)$, we are looking for features that are highly correlated with Y. This is the criterion that is most prominently applied in most traditional multi-relational learning systems. Theoretically, a single feature X_i agreeing with Y on all instances would be enough to ensure $H(X_i|Y) = 0$. In practice it is rarely possible to find such a perfect feature and often there is only a small number of features with high correlation. In such a setting, the learning algorithm also needs to consider features with comparably low correlation and the two other criteria below become relevant for optimal feature construction.
- *High feature entropy.* The joint entropy can be upper-bounded by the sum of single features: $H(X) = \sum_{i=1}^{n} H(X_i|X_{i-1},\ldots,X_1) \leq \sum_{i=1}^{n} H(X_i)$. Thus, in order to maximize $H(X)$ we need to maximize the entropy of each single feature. For Boolean features this means that each feature should divide the training instances in two parts of preferably equal size, so that it assigns -1

to roughly the same number of objects as +1. This is intuitively intriguing: a set of k features that assign +1 to only one training instance and −1 to the others can discriminate only between k different instances, whereas a set of features that divide the instances into two equal-sized parts can discriminate between up to 2^k bins of instances.

– *High inter-feature entropy.* Even if all single features have maximal entropy, it could be the case that the features are highly correlated to each other. In the most extreme case, it could be that all features assign the same labels to all instances $X_1 = \ldots = X_n$. Clearly, we need to ensure that the features complement each other and do not provide the same information all over again. In terms of information theory one can write $H(X) = \sum_{i=1}^{n} H(X_i|X_{i-1}, \ldots, X_1) \leq \sum_{i=1}^{n} H(X_i|X_{i-1})$. Thus we need to maximize $H(X_i|X_{i-1})$ for all $1 < i \leq n$. Since the features can be ordered arbitrarily, this essentially means we need to maximize $H(X_i|X_j)$ for each pair of features.

The first criterion has been dealt with to great extent in the existing literature on relational learning, the second is sometimes addressed by putting minimum frequency constraints on the features, and the third is usually not considered or included only implicitly. This is a problem in particular in the graph learning setting, where a feature indicates the occurrence or absence of a substructure in a graph. Typically, it is easy to construct substructures that appear in only a very small number of graphs, so the entropy of single features tends to be low. To avoid this, one often selects substructures that appear only with a certain minimal frequency in the graph database. Unfortunately, the resulting substructure's instantiations are often very similar to each other so that inter-feature entropy is low. For instance, mining all subgraphs that appear in at least six percent of the NCTRER dataset (see section 4) yields 83,537 frequent subgraphs. However, when comparing the instantiation X_i with X_j for all pairs of subgraphs (i, j), it turns out, that in 19% of the pairs $X_i = X_j$ and in 77% of the pairs X_i differs from X_j on less than ten instances. Hence, training matrices based on minimum frequency mining tend to be large and exhibit an unnecessarily large degree of redundancy.

On the other hand, it is easy to see that *Hadamard matrices* constitute optimal training matrices with regard to the latter two criteria, because any two columns are orthogonal (so $H(X_i, X_j) = 2$ is maximal for all $i \neq j$), and the number of ones is equal to the number of minus ones in each column (so $H(X_i) = 1$ is maximal for all i). Hadamard matrices of order 2^i can be generated using Sylvester's recursive construction:

$$H_1 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad H_i = \begin{pmatrix} H_{i-1} & H_{i-1} \\ H_{i-1} & -H_{i-1} \end{pmatrix}.$$

Hadamard matrices are also well suited for linear classifiers, because they have full rank. One can show that a matrix of rank d leads to 2^d different linear classifiers, so a large-rank matrix allows the linear learner to choose from a larger amount of different hypotheses. In the following we propose a method

that optimizes the second and third criterion for *linear classifiers* on subgraph features.

While a Hadamard matrix would be optimal for learning, it is certainly impossible to find subgraphs whose instantiations give rise to it. Instead, we are faced with the problem of finding subgraphs that approximate a Hadamard matrix as good as possible. In the following, we assume a forward selection setting and frame the problem as an iterative combinatorial optimization problem: Let $D = \{g_1, \ldots, g_m\}$ be a set of graphs (the instances), and $F = \{f_1, \ldots, f_n\}$ be a set of subgraphs (the already included features), and denote by s_i the instantiation vector of the ith subgraph f_i with regard to D, i.e., the vector whose jth component is 1 if f_i is a subgraph of g_j and -1 otherwise. We are then looking for a new subgraph f_{n+1} whose instantiation vector s_{n+1} optimizes some score quantifying the criteria explained above. For the experiments in section 4, we use what we call a *dispersion score* in the remainder of the paper:[1] $d(s_{n+1}) := \sum_{i=1}^{n} (s_i^T s_{n+1})^2$. Minimizing this score directly addresses the third criterion: if s_{n+1} and some s_i agree on many instances, the square of their dot product is large. Thus, summing up over all squared dot products essentially penalizes features that are similar to existing ones. One can show that it promotes features that discriminate between instances that have not been separated well by the existing features. It also implicitly optimizes the second criterion, because it aims at features that agree with existing features on half of the instances and disagree on the other half, thus leading on average to features which assign +1 to approximately the same number of instances as -1. Finally, the score reaches the global optimum zero precisely for the Hadamard matrix. Of course, the dispersion score does not aim at finding features that correlate well with the target. In order to also incorporate the first criterion, we extend it to not only penalize features that are similar to the existing features s_i, but also to reward features that are similar to the target class vector t: $d'(s_{n+1}) := \sum_{i=1}^{n} (s_i^T s_{n+1})^2 - n(t^T s_{n+1})^2$. The modified *class-correlated dispersion score* is designed to value dispersion to the same extent as similarity with the target. In the following section we describe our algorithmic approach to optimizing the dispersion score.

3 Stochastic Local Search for Optimal Dispersion Features

The dispersion score provides a practical way to formulate the search for features with large discriminative power as a combinatorial optimization problem. Unfortunately, due to the complexity inherent in graph operations, the problem can be extremely difficult to solve. It is clear that computing the instantiation vector for an arbitrary graph involves the repeated computation of solutions to NP-complete graph isomorphism problems. Even if one avoids these subgraph isomorphism tests, the problem can be shown to be NP-hard:

[1] In principle one could also apply mutual information instead of the dispersion score, but experiments have shown that this is not effective for the datasets in section 4.

Theorem 1. *The problem of deciding whether there exists a graph, that achieves a dispersion score of zero on a given training matrix and a given graph database is NP-hard.*

(Proof omitted)

There is no generally applicable approach to solving such a combinatorial optimization problem. However, in recent years stochastic local search (SLS) algorithms have been applied with remarkable success on similar NP-hard combinatorial problems. In particular, they are among the best algorithms available to solve hard satisfiability problems. SLS can be described as a randomized greedy walk in the space of solution candidates. More precisely, an SLS algorithm starts by generating a random solution candidate. It then iterates in a loop over two steps: in the first step, it calculates "neighboring" solution candidates according to some predefined neighborhood relation. For each neighbor, it computes a score indicating to which degree the candidate is optimal. In the second step, it randomly selects a new candidate among the neighbors with the best score. As such a pure greedy algorithm can easily be trapped in local optima, the second step is from time to time (i.e., with a predefined noise probability p) replaced by a step, where a completely random neighbor is selected as new candidate. Finally, the algorithm keeps track of the best candidate found so far and outputs this candidate as a solution after a maximum number of iterations. While modern SLS algorithms often use more sophisticated decision functions, the basic principle has been shown to be effective on a range of NP-hard problems, see e.g. [8] for an generic SLS algorithm and applications.

The SLS framework can be easily adjusted to the optimization problem presented in this paper. A solution candidate is simply a graph, and the scoring function is the dispersion score explained in the preceding section. For the neighborhood relation we generate two different kinds of neighbors: more *specific neighbors* are built by extending the current candidate graph with an edge so that the resulting subgraph occurs in at least one graph of the graph database. This avoids generating neighbors that do not occur in the database at all. More *general neighbors* are built by removing one edge from the current candidate. If the removal of the edge separates the graph into two unconnected components, we keep the larger of the two as neighbor and discard the smaller one.

It is crucial for the effectivity of an SLS algorithm that the calculation of the score function and the neighbors is as fast as possible. Unfortunately, both tasks are exceptionally expensive in our case. To compute the dispersion score and to determine more specific neighbors, one needs to identify the instantiation vector and that implies a subgraph isomorphism test with each graph in the database. To overcome this performance bottleneck, we pre-compute an index structure that stores all subgraphs up to a maximum size that occur in the database. The calculation of a candidate's instantiation vector and more specific neighbors is then just a lookup or a limited search operation in the index structure. As there is a huge number of subgraphs in a typical database, it is important to design the index structure to be space efficient, yet fast to access. We achieve this by associating each (sub-)graph with a *canonical code string*, i.e., a string that

uniquely determines the graph, and storing those canonical strings in a trie. We follow the breadth-first-search scheme [1] to compute the canonical code strings of all subgraphs with less than a user-defined number of edges. The strings are then stored in a trie structure, so that the SLS procedure can simply look up which of the database graphs contain a specific subgraph.

4 Experiments

In order to evaluate the approach, we implemented the dispersion optimizing SLS algorithm and applied it to three data sets. The NCTRER dataset [4] deals with the binding activity of small molecules at the estrogen receptor, the Yoshida dataset [10] classifies molecules according to their bio-availability, and the bloodbarr dataset [7] deals with the degree to which a molecule can cross the blood-brain barrier. For the experiments, we set the noise probability of taking a purely random step in the SLS loop to 0.2, the maximum size of subgraphs stored in the graph trie to fifteen edges and the maximal number of iterations for the SLS loop to 2000. The resulting feature sets are processed by a SVM with the C parameter set to 1 and Margin Minus Variance (MMV) optimization [9] with the p parameter set to 2. To keep the induced linear model comprehensible, we build it in an iterative fashion: we start with an empty feature set and then add the features one by one according to the optimal dispersion criterion. Whenever the number of features exceeds one hundred, we compute the linear classifier and remove the feature with the smallest weight before adding a new feature. The time to generate the trie is typically a few minutes. As it is generated once per dataset (like an index structure in a database), it does not influence the runtimes of subsequent SLS runs.

For the first experiment, we investigate to what extent SLS with dispersion and the class-correlated dispersion are able to generate training sets that are well suited for classification. To do so, we apply SLS with the two scores to construct four feature sets containing 25, 50, 150 and 300 features. We report the training set and test accuracies of SVM and MMV as estimated by tenfold cross-validation in the first four columns of table 1. The results point out some interesting insights. First of all, dispersion with class correlation is on average able to obtain a better *training accuracy* than the pure dispersion score, in particular with larger feature sets. This is expected as the pure dispersion score does not consider the class labels. However, the improvement in training accuracy does not always translate to an improvement in predictive accuracy. It does so for small feature sets up to 50 features, but for larger feature sets the difference in predictive accuracy is small even though the training accuracy is way larger for the class-correlated dispersion score. Also, MMV tends to perform better with regard to predictive accuracy than the SVM, even though its training accuracy is generally inferior to that of the SVM. Overall, MMV with the class-correlated dispersion score achieves good predictive performance for all feature set sizes.

As the SLS-based method should be particularly well-suited for obtaining small (e.g., size 25 or 50) useful feature sets, we set up an experiment comparing

Table 1. Results: percentage of correct classifications of four feature generation methods on training and test set according to tenfold cross validation

Dataset & Nr. Features		SLS (Dispersion)				SLS (Class Corr.)				MinFreq (Sorted by Size)				MinFreq (Sorted by Corr.)			
		MMV		SVM		MMV		SVM		MMV		SVM		MMV		SVM	
		Trg	Tst	Trg	Tst	Trg	Tst	Trg	Tst	Trg	Tst	Trg	Tst	Trg	Tst	Trg	Tst
yoshida	25	70.1	61.5	72.4	60.0	75.3	65.3	76.7	60.0	65.7	58.5	68.6	60.0	70.9	61.9	71.2	60.0
	50	75.6	64.5	79.2	62.3	78.0	67.2	80.5	66.4	67.7	57.0	73.4	60.0	72.2	60.4	73.1	60.0
	150	76.6	67.2	81.5	62.3	78.8	68.3	81.7	64.2	80.8	66.4	91.3	68.7	73.5	64.9	76.4	60.0
	300	77.2	66.4	81.5	63.8	85.9	66.4	96.1	65.7	85.8	66.8	96.4	68.7	76.4	66.8	83.4	63.4
NCTRER	25	84.2	82.8	83.1	77.6	82.6	81.5	82.9	76.3	80.2	76.3	80.7	59.9	79.2	78.4	79.6	59.9
	50	84.1	81.9	85.2	78.4	83.8	82.3	84.5	78.0	83.4	79.7	84.2	69.4	80.6	80.2	79.3	59.9
	150	84.2	81.0	84.5	77.2	84.3	82.3	86.1	82.8	87.1	82.3	91.3	78.0	80.8	79.7	82.2	79.3
	300	84.4	80.2	84.5	77.2	88.6	81.5	96.5	78.4	87.6	80.6	92.5	75.9	81.1	79.7	82.5	77.6
bloodbarr	25	72.7	69.6	76.8	66.5	77.1	72.5	78.2	68.2	72.9	70.4	73.6	66.5	76.2	73.7	77.3	67.5
	50	77.5	71.3	80.0	67.2	77.5	73.5	79.5	68.4	76.8	71.3	81.0	70.4	76.7	74.2	79.7	67.5
	150	77.3	69.9	81.1	69.9	78.0	74.9	80.8	68.0	83.2	75.7	90.0	75.9	78.4	72.0	85.6	70.1
	300	77.7	71.1	81.2	70.4	84.7	73.7	95.2	74.5	85.7	75.2	95.1	74.2	81.3	73.7	87.9	74.0

it to minimum-frequency and class-correlation feature generation within this range and beyond (size 150 and 300). First, we apply a subgraph mining tool to identify all subgraphs that occur in more than six percent of the dataset's graphs. Then, we sort the subgraphs by size (i.e. number of edges) or by the correlation with the target according to a χ^2 test on the 2x2 contingency table. Finally we derive four feature sets with 25, 50, 150 and 300 features from those two sorted feature sequences. Hence, the first sorting order is essentially an unsupervised propositionalization approach (similar to the one by Deshpande *et al.* [3]), while the second resembles the class-correlation based approach by Bringmann *et al.* [2]. The third and fourth column of table 1 give the training and test accuracies for MMV and the SVM. On feature sets with 150 and 300 features, the differences between dispersion-based and minimum frequency approaches are only marginal. However, in the target range of small feature sets, the SLS optimization of dispersion outperforms other approaches on two of the three datasets (Yoshida and NCTRER) in almost all pairwise comparisons.

We also compared our approach with the published accuracies (as estimated by tenfold cross validation) of two recently proposed learning systems for structured data. On the Yoshida dataset, an SVM with optimal assignment kernel [5] had 67.8% accuracy, a bit more than the dispersion-based SVM (65.7%), but less than dispersion-based MMV (68.3%). On bloodbarr, the dispersion-based approaches featured 74.9% (MMV) and 74.5% (SVM), outperforming the OA kernel SVM with 57.97% by a large margin. On the NCTRER dataset, kFOIL, an extension of FOIL incorporating an SVM in a novel evaluation function [6], had 77.6% accuracy, while the presented system yields 82.3% (MMV) and 82.8% (SVM), a significant improvement.

5 Conclusion

On structured data, a small set of strong features is vital for comprehensibility and efficiency of computation. In the paper, we framed the search for a suitable set of structural features as a combinatorial optimization problem. We proposed a scoring function fostering the diversity of feature sets and an optimization scheme based on stochastic local search (SLS) to maximize that score. The choice of SLS is motivated by the NP-hardness of finding an optimally complementary subgraph feature. In our experiments on small molecule data, we found that the optimization of dispersion pays particularly when aiming for small feature sets. Unlike many other approaches to feature selection (e.g. [11]), we can take advantage of the structure of features, intertwining structure search and feature selection. In principle, the basic approach is more general than presented here. First, it is not restricted to graphs, but could be extended to more expressive representations such as first-order logic. Second, it is not restricted to the forward selection procedure tested in section 4, since a variant of the dispersion score and SLS could be used for the optimization of fixed-size feature sets as well.

References

1. Borgelt, C.: On canonical forms for frequent graph mining. In: Proc. 3rd Int. Workshop on Mining Graphs, Trees, and Sequences, pp. 1–12 (2005)
2. Bringmann, B., Zimmermann, A., De Raedt, L., Nijssen, S.: Don't be afraid of simpler patterns. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 55–66. Springer, Heidelberg (2006)
3. Deshpande, M., Kuramochi, M., Wale, N., Karypis, G.: Frequent substructure-based approaches for classifying chemical compounds. IEEE Transactions on Knowledge and Data Engineering 17(8), 1036–1050 (2005)
4. Fang, H., Tong, W., Shi, L.M., Blair, R., Perkins, R., Branham, W., Hass, B.S., Xie, Q., Dial, S.L., Moland, C.L., Sheehan, D.M.: Structure-activity relationships for a large diverse set of natural, synthetic, and environmental estrogens. Chemical Research in Toxicology 14(3), 280–294 (2001)
5. Fröhlich, H., Wegner, J.K., Sieker, F., Zell, A.: Optimal assignment kernels for attributed molecular graphs. In: Proceedings of the 22nd ICML, pp. 225–232. ACM Press, New York (2005)
6. Landwehr, N., Passerini, A., De Raedt, L., Frasconi, P.: kFOIL: Learning simple relational kernels. In: AAAI, AAAI Press (2006)
7. Li, H., Yap, C.W., Ung, C.Y., Xue, Y., Cao, Z.W., Chen, Y.Z.: Effect of selection of molecular descriptors on the prediction of blood-brain barrier penetrating and nonpenetrating agents by statistical learning methods. Journal of Chemical Information and Modeling 45(5), 1376–1384 (2005)
8. Rückert, U., Kramer, S.: Stochastic local search in k-term DNF learning. In: Proceedings of the 20th ICML, pp. 648–655. AAAI Press (2003)
9. Rückert, U., Kramer, S.: A statistical approach to rule learning. In: Proceedings of the 23rd ICML, pp. 785–792. ACM Press, New York (2006)
10. Yoshida, F., Topliss, J.: QSAR model for drug human oral bioavailability. J. Med. Chem. 43, 2575–2585 (2000)
11. Yu, L., Liu, H.: Efficient feature selection via analysis of relevance and redundancy. J. Mach. Learn. Res. 5, 1205–1224 (2004)