CONVEX APPROXIMATION OF THE NP-HARD SEARCH PROBLEM IN FEATURE SUBSET SELECTION

Tofigh Naghibi, Sarah Hoffmann, Beat Pfister

Speech Processing Group, Computer Engineering and Networks Lab., ETH Zurich, Switzerland
{naghibi,hoffmann,pfister}@tik.ee.ethz.ch

ABSTRACT

Feature subset selection, as a special case of the general subset selection problem, attracted a lot of research attention due to the growing importance of data-mining applications. However, since finding the optimal subset is an NP-hard problem, very different heuristic search methods have been suggested to approximate it. Here we propose a new second-order cone programming based search strategy to efficiently solve the feature subset selection for large-scale problems. Experimentally, it is shown that its performance is almost always better than the greedy search methods especially when the features are strongly dependent.

Index Terms— second-order cone, search method, feature selection, convex approximation

1. INTRODUCTION

While, at least in theory, having more features should result in a more discriminative classifier, it is not the case in practice because of many reasons, foremost of which are curse of dimensionality and computational complexity. Therefore, dimensionality reduction is a must in most real-world applications. Here we mainly focus on feature subset selection as a common approach for dimensionality reduction.

Various feature selection algorithms proposed in literature may roughly be categorized into three different groups (see [1] and references therein): Wrapper methods, embedded methods and filter methods. The wrapper methods exhaustively search through the whole subset space for the optimal feature subset that maximizes the classification (generally induction algorithm) accuracy [2]. However, their main disadvantages are: First, by increasing the number of features they rapidly become prohibitive due to a so-called combinatorial explosion and second, training an induction algorithm with all the feasible feature subsets, may largely increase the over-fitting risk.

The second group of feature selection algorithms rely on internal parameters of the induction algorithm [3], [4]. Embedded approaches rank features during the training process and thus simultaneously determine both the optimal features and the parameters of the induction algorithm. However, they are highly dependent on the internal structure of the learning algorithms and cannot be seen as a general feature selection solution for all induction algorithms. Similar to the wrapper methods, they are dependent on the learning part and thus the selected subset is somehow tuned to a particular induction algorithm.

Unlike the first two groups, filter methods do not incorporate the learning part and thus show a better generalization power on the different induction algorithms. Generally, filter methods use predefined measure functions like an information-theoretic measure suggested in [5] to score the feature subsets and use a search algorithm to optimize this measure over the feature subset space.

From another standpoint, feature selection methods may be classified into three classes based on the search strategies they employ: The exhaustive search methods, random methods and greedy methods. Two main examples of the exhaustive search family are the wrapper methods and branch and bound based methods [6]. Random methods as the second class of the search strategies , usually employ an evolutionary algorithm like genetic algorithm or simulated annealing to gradually evolve an initial feature subset towards better solutions [7]. They are usually much less computationally expensive at the expense that they may not converge to the optimal solution. The third group of methods which are also called greedy hill climbing or sequential selection approaches, are very commonly used in filter methods due to their simplicity and low memory requirements [8]. Greedy algorithms iteratively evaluate a candidate subset of features, then modify the subset and evaluate if the new subset is an improvement over the old. However, since they dramatically narrow down the search space, they may not converge to the optimal solution and in some extreme cases result in a very poor performance.

Here we introduce a fourth class of the search strategies which is based on a second-order cone programming [9] (SOCP) relaxation of the subset selection problem. We show that an SOCP approximation can be employed to efficiently solve the combinatorial subset selection problem. Our experiments indicate that the proposed algorithm usually results in better performance than the greedy algorithms and especially outperforms the sequential selection methods for a) data sets with a large number of features but a very few samples. b) data sets with highly dependent features.

2. SUBSET SELECTION

We define the feature selection problem as follows:

Definition 1: Given N features, X_i for $i = 1$ to N, and a class label C (dependent variable), select a subset of $P \ll N$ features that maximizes the measure function.

From the Definition 1 it is clear that any feature selection algorithm has to address two main challenges: a) Define an appropriate measure function (or evaluation metric or score function among so many other names in literature) to evaluate the feature subsets. b) Develop an efficient search strategy that can find the optimal feature subset, in the sense of optimizing the measure function, within a reasonable amount of time.

In practice, the exact value of P can be obtained by evaluating different P-cardinality subsets with the induction algorithm. Here we use an information-theoretic measure function and develop an SOCP based search strategy to optimize this measure function.

2.1. Measure Function

Let $\mathbf{X} = [X_1, X_2, ..., X_N]$ be an N dimensional feature vector and C a dependent variable which can be a class label in case of classification or a target variable in case of regression. The mutual information function is defined as a distance from independence between $\mathbf{X}$ and C measured by the Kullback-Leibler divergence [10]. Basically, mutual information measures the amount of information shared between $\mathbf{X}$ and C by measuring their dependency level. Denoting the joint pdf of $\mathbf{X}$ and C and their marginal distributions by $Pr(\mathbf{X}, C)$, $Pr(\mathbf{X})$ and $Pr(C)$, respectively, the mutual information between the feature vector and the class label can be defined as follows:

$$I(X_1, X_2, \ldots, X_N; C) = I(\mathbf{X}; C) = \int Pr(\mathbf{X}, C) \log \frac{Pr(\mathbf{X}, C)}{Pr(\mathbf{X}) Pr(C)} \, \mathrm{d}\mathbf{X} \, \mathrm{d}C \quad (1)$$

It reaches its maximum value when the dependent variable is perfectly described by the feature set. In this case mutual information is equal to $H(C)$, the Shannon entropy of C. Despite the theoretical appeal of the mutual information measure [11], given a limited amount of data an accurate estimate of the mutual information would be impossible. Because to calculate (1), estimation of the high dimensional joint probability $Pr(\mathbf{X}, C)$ is inevitable which is, in turn, known to be an NP hard problem [12]. Therefore, several approximations of (1) have been suggested [13], [5]. Here we mainly use the Minimal Redundancy Maximal Relevance (mRMR) measure function,

$$D\{\mathbb{X}\} = \sum_{i=1}^{N} I(X_i; C) - \frac{1}{N-1} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} I(X_i; X_j) \quad (2)$$

proposed by Peng et al [5] to approximate the mutual information function. In (2), we denoted the set of X_i features with $\mathbb{X}$.

2.2. SOCP Based Search Method

A search strategy is an algorithm trying to optimize the measure function over the space of the feature subsets with 2^N members[1]. Two commonly used search algorithms in filter based feature selection methods are forward selection and backward elimination algorithms, defined as:

Definition 2: The forward selection algorithm selects a set $\mathbb{S}$ of size P iteratively as follows: 1- Initialize $\mathbb{S}_0 = \emptyset$. 2- In each iteration $i+1$, select the variable X_m maximizing $D(\mathbb{S}_i \cup X_m)$, and set $\mathbb{S}_{i+1} = \mathbb{S}_i \cup X_m$. 3- Output $\mathbb{S}_P$.

In an entirely analogous fashion, backward elimination can be described as: 1- Start with the full set of feature $\mathbb{S}_N$. 2- Iteratively remove a feature X_m maximizing the $D(\mathbb{S}_i \backslash X_m)$ and set $\mathbb{S}_{i-1} = \mathbb{S}_i \backslash X_m$ where removing X from the set $\mathbb{S}$ is denoted by $\mathbb{S} \backslash X$. 3- Output $\mathbb{S}_P$.

An experimentally comparative evaluation of several variants of these two algorithms has been conducted in [8]. Generally the main disadvantage of the forward selection method is that it only can evaluate the utility of a single feature in the limited context of the previously selected features. Contrary to forward selection, backward elimination can evaluate the contribution of a given feature in the context of all other features. Perhaps that is why it has been frequently reported to show superior performance than forward selection. However, its overemphasis on feature interactions may result in eliminating some strong independent features in favor of weak features that have some information in their interactions. Generally, in case of dependent features, where the relation between features are complicated, it is unlikely that the sequential search strategies, converge to a close to optimal solution.

To propose a new approximation method, the underlying combinatorial problem has to be studied. To this end, we may formulate the Subset Selection Problem (SSP) in Definition 1 as:

$$\begin{aligned} &\max_{\mathbf{x}} \ \mathbf{x}^T \mathbf{Q} \mathbf{x} \\ \text{(SSP)} \qquad &\sum_{i=1}^{N} x_i = P \\ &x_i \in \{0, 1\} \text{ for } i = 1, \ldots, N \end{aligned} \quad (3)$$

where $\mathbf{Q}$ is a symmetric information matrix constructed from the mutual information terms in (2) as:

$$\mathbf{Q} = \begin{pmatrix} I(X_1; C) & \cdots & -\frac{\beta}{2} I(X_1; X_N) \\ -\frac{\beta}{2} I(X_1; X_2) & \cdots & -\frac{\beta}{2} I(X_2; X_N) \\ \vdots & \ddots & \vdots \\ -\frac{\beta}{2} I(X_1; X_N) & \cdots & I(X_N; C) \end{pmatrix} \quad (4)$$

and $\mathbf{x} = [x_1, \ldots, x_N]$ is a binary vector where x_i variables are N set-membership binary variables indicating the presence of their corresponding X_i features in the feature subset. Note that for the mRMR measure function, the parameter β in (4) is equal to $\frac{1}{P-1}$.

It is straightforward to verify that for any binary vector $\mathbf{x}$, the objective function of (3) is equal to the mRMR measure function $D(\mathbb{X}_{nz})$ in (2) where $\mathbb{X}_{nz} = \{X_i | x_i = 1; i = 1, \ldots, N\}$. That is, for any given binary vector $\mathbf{x}$, (3) is just another representation of the mRMR measure. Note that in case of using other measures rather than mRMR, for instance $1 - I(\mathbf{X}; C)/H(\mathbf{X}, C)$ which is a metric version of the mutual information or correlation based measures, the matrix $\mathbf{Q}$ can be constructed by some small changes, analogously.

The (0,1)-quadratic programming problem (3) has attracted a great deal of theoretical studies because of its importance in combinatorial problems [see 14, and references therein]. One popular approach to solve this kind of problems is to use a semidefinite programming relaxation which has been proved to give a tight approximation ratio bound. However, here we follow another approach suggested in [15],[16] and [17] for solving the Max-Cut problem or its K-dense constraint version in graph theory.

Denote $\mathbf{Q}_0$ as an $N \times N$ matrix with zero diagonal entries and off-diagonal elements equal to the off-diagonal elements of $\mathbf{Q}$. Since for binary variables $x_i^2 = x_i$, the objective function of (3) can be written as $\mathbf{x}^T \mathbf{Q}_0 \mathbf{x} + \mathbf{q}^T \mathbf{x}$, where $\mathbf{q}$ is a vector of the diagonal elements of $\mathbf{Q}$. In addition, since $\mathbf{Q}_0$ is a symmetric matrix with zero diagonal entries, its largest eigenvalue is always greater than zero (because $\mathbf{Q}_0$ can be neither positive nor negative definite), that is $\lambda_{max}(\mathbf{Q}_0) \geq 0$ and $\lambda_{min}(\mathbf{Q}_0) \leq 0$. Hence, we may construct a negative-definite matrix $\mathbf{Q}_c = \mathbf{Q}_0 - \lambda_{max}(\mathbf{Q}_0)\mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Now, (3) can be rewritten as:

$$\begin{aligned} &\max_{\mathbf{x}} \ \mathbf{x}^T \mathbf{Q}_c \mathbf{x} + \mathbf{q}^T \mathbf{x} + P \lambda_{max}(\mathbf{Q}_0) \\ &\sum_{i=1}^{N} x_i = P \\ &x_i \in \{0, 1\} \text{ for } i = 1, \ldots, N \end{aligned} \quad (5)$$

By defining an auxiliary variable $\alpha = \mathbf{x}^T \mathbf{Q}_0 \mathbf{x}$ and relaxing the hard constraints $x_i \in \{0, 1\}$ to linear inequality constraints $0 \leq x_i \leq 1$,

[1] Given a P, the size of the feature subset space reduces to $\binom{N}{P}$.

the final approximation of the (SSP) can be expressed as:

$$\text{(SOCP)} \quad \begin{aligned} &\max_{\mathbf{x},\alpha} \ \alpha + \mathbf{q}^T\mathbf{x} \\ &\sum_{i=1}^{N} x_i = P \\ &x_i \in [0,1] \text{ for } i = 1,\ldots,N \\ &\alpha \leq \mathbf{x}^T\mathbf{Q}_c\mathbf{x} + P\lambda_{max}(\mathbf{Q}_0) \end{aligned} \tag{6}$$

where the last inequality comes from the fact that $\lambda_{max}(\mathbf{Q}_0)\sum_i^N x_i^2$ is smaller than $\lambda_{max}(\mathbf{Q}_0)\sum_i^N x_i$ for x_i between 0 and 1.

In this work we used Gurobi solver [18] which is sufficiently fast for our experiments. It solves a problem of size $N = 2000$ in less than 6 seconds and a large-scale SOCP problem with $N = 10000$ under an hour on a PC with an Intel Core i7 CPU.

Since the solution of (SOCP) is not necessarily a binary vector, we need more steps to attain a feasible solution. The following three steps algorithm summarizes the approximation algorithm for the (SSP). In the sequel, this algorithm is referred to as SOSS (SOCP Subset Selection) algorithm.

SOSS Algorithm:

1. SOCP: Solve (SOCP) to obtain $\mathbf{x}$.
2. Randomized rounding: Generate N random numbers u_i with uniform distribution U(0,1) and construct

$$\hat{x}_i = \begin{cases} 1 & \text{if } x_i \geq u_i \\ 0 & \text{if } x_i < u_i \end{cases} \tag{7}$$

   for all values of i between 1 to N. Select $\mathbb{X} = \{X_i | \hat{x}_i = 1\}$.
3. Size adjusting (arbitrary step): By using a greedy algorithm, add or remove one element from the set $\mathbb{X}$ at a time that has the least contribution to $D(\mathbb{X})$ to resize the $\mathbb{X}$ cardinality to P.

The randomized rounding step is a standard procedure to produce a binary solution and is widely used for designing and analyzing approximation algorithms [19].

In feature selection problem the exact satisfaction of the cardinality constraint may not be required. Since the cardinality of the solution obtained at the step two of the SOSS algorithm is usually close to P, here the size adjusting step is skipped and the solution $\mathbb{X}$ achieved in step two of the algorithm is taken as the output. Moreover, it is noteworthy to mention that the role of the randomized rounding step is more than generating a binary solution. From our experiments we noticed that due to the randomized rounding step, the algorithm produces very diverse solutions for P values close to each other in the sense that the generated feature subsets have close evaluation scores while having very few features in common. That is, unlike the sequential subset selection methods in Definition 2, where $\mathbb{S}_{i-1} \subset \mathbb{S}_i$, here $\mathbb{X}_{i-1}$ may hardly have many elements in common with $\mathbb{X}_i$. The diversity of the solutions increases the chance that the SOSS algorithm finds feature subsets that are more compatible with the subsequent classifier than the feature subset with simply the highest mRMR score.

3. EXPERIMENTS

The evaluation of a feature selection algorithm is an intrinsically difficult task since there is no direct way to evaluate the goodness of a *selection process* in general. Thus, usually a selection algorithm is scored based on the performance of its output, i.e., a feature subset, under some specific classification (regression) system, interpreted as a goal-dependent evaluation. However, this method obviously can not evaluate the generalization power of the selection process on different induction algorithms. Thus, for evaluation of the feature selection algorithms, we train the different induction algorithms with the selected features and take the average of their accuracies as the final evaluation score. This method leads to a more classifier-independent evaluation.

Data set	Arrhythmia	NCI	DBWorld	Internet Adv.
Acronym	ARR	NCI	DBW	IAD
Data Type	Continuous		Discrete	
# Features	278	9703	4702	1558
# Samples	370	60	64	3279
# Class Label	2	9	2	2

Table 1. Data sets descriptions

Some description of the five data sets we used are listed in Table 1. Except NCI data which can be found in Peng et al website [5], the rest, Arrhythmia, DBWorld emails and Internet Advertisement, are available on the UCI machine learning archive [20]. These data sets have been widely used in previous feature selection studies [5], [21]. The goodness of each feature set is evaluated with five classifiers including Support Vector Machine (SVM), Random Forest (RF), Classification and Regression Tree (CART), Neural Network (NN) and Linear Discriminant Analysis (LDA). To derive the classification accuracies, 10-fold cross-validation is performed except for the NCI and DBW data sets where leave-one-out cross-validation is used. To determine whether the difference between two algorithms is significant, the p-values of the t-Test between two algorithms are computed. Here p-value simply means the probability that one algorithm is better than the other.

Three search strategies, Forward Selection (FS), Backward Elimination (BE) and SOSS algorithm are compared with each other in our experiments. Table 2 shows the results obtained for the four data sets and five classifiers. The size of the selected subset $|\mathbb{X}|$ and the classification accuracy in percentage are given for each single experiment. The last column shows the p-values of the t-Test between the SOSS and the BE or FS algorithms (always with the one with the better performance). The larger the p-value, the higher the confidence level is. The Ave. column reports the average of the classification accuracies for each algorithm. For most of the classifiers, SOSS leads to a lower error-rate than the greedy methods.

In the case of IAD data set, SOSS and sequential methods achieve almost the same classification accuracies. This is because of the fact that the information matrix $\mathbf{Q}$ of the IAD data set has small off-diagonal elements. That is, the IAD features are more or less mutual independent. In this case, the greedy algorithms can converge to a close to optimal solution. However, a significant improvement can be seen for the NCI data where SOSS is better than BE at the confidence level of 87%. NCI is probably the most difficult (from the feature selection point of view) data set in our experiment since it has around 10000 features yet very few training samples. Since mutual information approximation for constructing $\mathbf{Q}$ is computationally expensive, we first filter out 8000 of the NCI features by simply eliminating those with small mutual information values with class label. The dominant superiority of the SOSS algorithm in this case is perhaps due to the fact that for the small size data sets like NCI, mutual information approximation is pretty

Classifiers	SVM		LDA		CART		RF		NN		Ave.	p-val
ARR Data set												
SOSS	40	81.9	38	77.9	35	79.3	38	85.5	70	74.8	79.86	0.60
FS	64	80.7	38	78.3	98	78.5	70	85.2	30	73.5	79.28	
BE	67	80.7	39	78.3	97	78.1	68	84.7	43	73.8	79.14	
NCI Data set												
SOSS	40	80.0	38	75.0	35	43.3	38	86.7	70	78.3	72.65	0.87
FS	32	78.3	11	68.3	2	45.0	12	83.3	99	70.0	69.00	
BE	26	76.6	11	68.3	2	45.0	13	85.0	31	71.7	69.33	
DBW Data set												
SOSS	127	95.3	10	93.7	8	86.0	14	92.2	108	93.8	92.18	0.64
FS	31	93.7	4	89.0	4	86.0	7	90.6	9	92.2	90.31	
BE	110	93.7	6	89.0	4	82.8	29	92.2	9	92.2	90.00	
IAD Data set												
SOSS	40	96.6	38	95.9	35	96.2	38	97.3	70	97.2	96.64	0.53
FS	109	96.2	127	95.8	127	96.7	25	97.0	52	97.2	96.58	
BE	22	96.3	163	95.9	121	96.1	109	97.2	148	97.4	96.58	

Table 2. Comparison of SOSS algorithm with greedy search methods over different data sets. For each single experiment, the first reported number is the optimal feature subset cardinality and the second is the classification accuracy achieved in that experiment.

noisy. Taking all these noisy mutual information terms together into account, as in SOSS, may result in overall noise mitigation (by assuming that mutual information values are the correct values corrupted by independent white noise). Moreover, the NCI features are highly dependent. Thus greedy search algorithms fail to find the complicated relation between the features. The ARR and DBW data sets also contain a very few samples but a large number of features. As with the NCI data set, we first prune the features with close to zero mutual information values. Interestingly, the same effect as in NCI data appears for DBW and ARR data as well (since they both have a very few samples) but at a bit lower confidence intervals.

4. CONCLUSION

In this work, we propose a new search strategy to select the optimal feature subset. The feature subset selection problem is shown to be equivalent to an instance of (0,1)-quadratic integer programming problem. By using a second-order cone programming relaxation for this combinatorial problem, we develop a convex search method which can efficiently be solved for large-scale problems. The experiments indicate that the proposed algorithm, outperforms the greedy search methods especially in case of data sets with a large number of features, yet a very few samples or when there are complicated relations between features.

5. ACKNOWLEDGMENT

This work has been supported by Swiss National Science Foundation (SNSF).

References

[1] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, 2003.

[2] R. Kohavi, *Wrappers for performance enhancement and oblivious decision graphs*, Ph.D. thesis, Stanford, CA, USA, 1996, UMI Order No. GAX96-11989.

[3] P. H. Winston, "Learning structural descriptions from examples," Tech. Rep., Cambridge, MA, USA, 1970.

[4] Tom M. Mitchell, "Generalization as search," *Artificial Intelligence*, vol. 18, no. 2, pp. 203 – 226, 1982.

[5] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, 2005.

[6] P. M. Narendra and K. Fukunaga, "A branch and bound algorithm for feature subset selection," *IEEE Transactions on Computers*, vol. C-26, no. 9, pp. 917 – 922, sept. 1977.

[7] H. Vafaie and K. De Jong, "Robust feature selection algorithms," in *Proceedings., Fifth International Conference on Tools with Artificial Intelligence, 1993. TAI '93.*, Nov 1993.

[8] D. W. Aha and R. L. Bankert, "A comparative evaluation of sequential feature selection algorithms," in *Learning from Data: Artificial Intelligence and Statistics V*. Springer-Verlag, 1996.

[9] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, New York, NY, USA, 2004.

[10] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley-Interscience, New York, NY, USA, 1991.

[11] R. Fano, *Transmission of Information: A Statistical Theory of Communications*, The MIT Press, Cambridge, MA, 1961.

[12] David Karger and Nathan Srebro, "Learning markov networks: Maximum bounded tree-width graphs," 2001.

[13] R. Battiti, "Using mutual information for selecting features in supervised neural net learning," *IEEE Transactions on Neural Networks*, vol. 5, pp. 537–550, 1994.

[14] S. Poljak, F. Rendl, and H. Wolkowicz, "A recipe for semidefinite relaxation for (0,1)-quadratic programming," *Journal of Global Optimization*, vol. 7, pp. 51–73, 1995.

[15] M. Muramatsu and T. Suzuki, "A new second-order cone programming relaxation for max-cut problems," *Journal of Operations Research of Japan*, vol. 46, 2001.

[16] B. Ghaddar, J. C. Vera, and M. F. Anjos, "Second-order cone relaxations for binary quadratic polynomial programs," *SIAM J. on Optimization*, vol. 21, no. 1, pp. 391–414, 2011.

[17] Rui Yang, "New approximation methods for solving binary quadratic programming problem," M.S. thesis, University of Illinois at Urbana-Champaign, 2010.

[18] Inc. Gurobi Optimization, "Gurobi optimizer reference manual," 2012.

[19] P. Raghavan, "Probabilistic construction of deterministic algorithms: approximating packing integer programs," *J. Comput. Syst. Sci.*, vol. 37, no. 2, Oct. 1988.

[20] A. Frank and A. Asuncion, "UCI machine learning repository," 2010.

[21] P. M. Ciarelli, E. O. T. Salles, and E. Oliveira, "An evolving system based on probabilistic neural network," in *Proceedings of BSNN*, 2010, pp. 182–187.