Rough Set Methods and Submodular Functions

Hung Son Nguyen and Wojciech Świeboda

Institute of Mathematics, The University of Warsaw,
Banacha 2, 02-097, Warsaw Poland

Abstract. In this article we discuss the connection of Rough Set methods and submodular functions. We show that discernibility measure used in reduct calculation is submodular and provides a bridge between certain methods from Rough Set theory and Submodular Function theory.

1 Introduction

In this article, we aim to highlight connections between Rough Set Theory and Submodular Function Theory. Rough Set problems (such as finding reducts, inference of decision rules, discretizing numeric attributes) are all based on approximating indiscernibility relation (in the product space $U \times U$, where U is a universe of objects). In this paper we only focus on the problem of finding a single, possibly short (decision) reduct, which is one of fundamental problems in Rough Set theory. One of natural measures of "goodness" of approximation induced by a subset of attributes in a decision system is a *discern* measure which we introduce in the first subsection. This function is submodular, hence the approximation of indiscernibility relation may be considered within the framework of Submodular Function optimization. We will discuss maximization methods that only utilize three properties of *discern* measure: submodularity, monotonicity and the ease of computation using lazy evaluations. We will also highlight the potential of applying certain Rough Set Methods to other Submodular Function optimization problems by describing an example computational problem, whose aim is to optimize (using lazy evaluations) a submodular function which takes as the argument a subset of attributes from an information (or decision) system which contains several default values. We point out the fact that optimizations of this kind can be performed for very large datasets using an SQL interface.

2 Rough Sets and Submodularity

2.1 Rough Sets

Rough Set Theory[10],[12] was introduced by Pawlak as a tool for concept approximation under uncertainty. The general idea is to provide (or derive from data) lower and upper approximations of a concept.

In this subsection we briefly review fundamental notions of Rough Set Theory: information and decision systems and decision reducts.

An *information system* is a pair $\mathbb{A} = (U, A)$, where the set U denotes the *universe of objects* and A is the set of *attributes*, i.e. mappings of the form: $a : U \to V_a$. V_a is called the *value set* of attribute a.

A decision system is an information system $\mathbb{D} = (U, A \cup \{d\})$ where d is a distinguished *decision attribute*. The remaining attributes are called *conditions* or *conditional attributes*. An example decision system is shown in Table (a).

For a subset of attributes $B \subseteq A$ we define (on $U \times U$) *B-indiscernibility relation* $IND(B)$ as follows:

$$(x, y) \in IND(B) \iff \forall a \in A \; a(x) = a(y)$$

$IND(B)$ is an equivalence relation and hence defines a partitioning of U into equivalence classes which we denote by $[x]_B$ ($x \in U$). The complement of $IND(B)$ in $U \times U$ is called *discernibility relation*, denoted $DISC(B)$. The lower and upper approximations of a concept X (using attributes from B) are defined by

$$\mathbf{L}_B(X) = \left\{x \in U : [x]_{IND(B)} \subseteq X\right\} \quad \text{and}$$
$$\mathbf{U}_B(X) = \left\{x \in U : [x]_{IND(B)} \cap X \neq \varnothing\right\}.$$

In general, reducts are minimal subsets of attributes contain necessary information about all attributes. Below we remind just two definitions.

- A *reduct* is a minimal set of attributes $R \subseteq A$ such that $IND(R) \subseteq IND(A)$.
- A *decision-relative reduct* is a minimal set of attributes $R \subseteq A$ such that $IND(R) \subseteq IND(\{dec\}) \cup IND(A)$. In other words, it is a minimal subset of attributes which suffices to discern all pairs of objects belonging to different decision classes.

We proceed with two definitions that will come in handy in reduct calculation.

A *conflict* is a pair of objects belonging to different decision classes. We define $conflicts : 2^U \to \mathbb{R}_+$ so that for $X \subseteq U$:

$$conflicts(X) = \frac{1}{2}|\{(x, y) \in X \times X : dec(x) \neq dec(y)\}|$$

We define $c : 2^A \to \mathbb{R}_+$ as follows. For $B \subseteq A$:

$$c(B) = \sum conflicts([x]_B)$$

where the summation is taken over all equivalence classes of partitioning induced by $IND(B)$. Function c is a natural extension of the definition of *conflicts* function to subsets of attributes. Subset of attributes $B \subseteq A$ is a reduct if $c(B) = c(A)$.

For a subset of attributes $B \subseteq A$ we define

$$discern(B) = c(\emptyset) - c(B)$$

Let I denote the indicator function. In the formula above:

$$c(\emptyset) = \frac{1}{2} \sum_{(x,y) \in U \times U} I\left(d(x) \neq d(y)\right)$$
$$c(B) = \frac{1}{2} \sum_{(x,y) \in U \times U} I\left(d(x) \neq d(y) \wedge (x,y) \in IND(B)\right)$$

Please notice that:

$$\begin{aligned}
discern(B) &= c(\emptyset) - c(B) \\
&= \frac{1}{2} \sum_{(x,y) \in U \times U} I\left(d(x) \neq d(y)\right) \\
&\quad - \frac{1}{2} \sum_{(x,y) \in U \times U} I\left(d(x) \neq d(y) \wedge (x,y) \in IND(B)\right) \\
&= \frac{1}{2} \sum_{(x,y) \in U \times U} I\left(d(x) \neq d(y)\right) \\
&\quad - \frac{1}{2} \sum_{(x,y) \in U \times U} I\left(d(x) \neq d(y) \wedge \forall a \in B : a(x) = a(y)\right) \\
&= \frac{1}{2} \sum_{(x,y) \in U \times U} I\left(d(x) \neq d(y) \wedge \exists a \in B : a(x) \neq a(y)\right) \\
&= \frac{1}{2} \left| DISC(\{d\}) \setminus IND(B) \right|
\end{aligned}$$

Function *discern* is closely related to decision reducts – $R \subset A$ is a decision reduct if it a minimal subset of attributes with $discern(R) = discern(A)$. A well known method for calculating short decision reducts is Maximal Discernibility heuristic (or Johnson's heuristic) [9], [2]. This method iteratively extends a set of attributes in a greedy fashion, picking in each step the attribute with the largest marginal *discern*.

2.2 Submodular Functions

Let Ω be a finite set. A set function $f : 2^{\Omega} \mapsto \mathbb{R}$ is *submodular* [1] if it satisfies any of the three following equivalent properties:

- For $T \subset S \subset \Omega$ and $x \in S \setminus T$: $f(T \cup \{x\}) - f(T) \geq f(S \cup \{x\}) - f(S)$.
- For $T, S \subset \Omega$: $f(T) + f(S) \geq f(T \cup S) + f(T \cap S)$.
- for $T \subset \Omega$ and $x, y \in \Omega \setminus T$: $f(T \cup \{x\}) + f(T \cup \{y\}) \geq f(T \cup \{x, y\}) + f(T)$.

The first of these formulations can be naturally interpreted as "diminishing returns" property.

Submodular functions naturally arise within the context of various combinatorial optimization problems and Machine Learning problems. Maximization problems involving submodular functions are usually NP-hard (see [6] and references therein), although several heuristics (with provable bounds) have been proposed in the literature.

A notable characteristic of several of these algorithms is that they may use lazy evaluation, i.e. they may update the value of the function f upon inclusion of an additional element. Another property often stressed for some algorithms is whether they are suited for optimization of arbitrary submodular functions or monotone submodular functions.

2.3 Discernibility and Submodularity

Lemma 1. *Let $\mathbb{D} = (U, A \cup \{d\})$ be a decision system. Set function $discern : 2^A \mapsto \mathbb{R}$ is a monotone increasing submodular function.*

Proof. Recall that:

$$discern(B) = \frac{1}{2} \sum_{(x,y) \in U \times U} I\left(d(x) \neq d(y) \wedge \exists a \in B : a(x) \neq a(y)\right)$$

Please notice that an unordered pair $\{x, y\}$ either does not contribute to $discern(T)$, or is counted in $discern(T)$ exactly twice.

- Notice that for $T, S \subset A$:

$$\begin{aligned} discern(T) &= \sum_{x,y \in U} I\left(d(x) \neq d(y) \wedge \exists a \in T : a(x) \neq a(y)\right) \\ &\leq \sum_{x,y \in U} I\left(d(x) \neq d(y) \wedge \exists a \in T \cup S : a(x) \neq a(y)\right) \\ &= discern(T \cup S) \end{aligned}$$

  Hence $discern$ is monotone.
- We will show that $discern(T) + discern(S) \geq discern(T \cup S) + discern(T \cap S)$. Let us first consider an unordered pair of objects $\{x, y\}$ which is counted at least once in $discern(T \cup S) + discern(T \cap S)$. It follows that this pair is discerned by an attribute $a \in T \cup S$, hence it is counted at least once in $discern(T) + discern(S)$. If a pair (x, y) is counted twice in $discern(T \cap S) + discern(T \cup S)$, then it is counted by $discern(T \cap S)$, and hence it is counted twice in $discern(T) + discern(S)$. Therefore $discern$ is submodular. ■

Please also notice that $discern(B)$ is a function of the partitioning of objects induced by $IND(B)$. In implementation of algorithms we explicitly keep the partitioning $IND(B)$ and further subdivide (shatter) it into $IND(B \cup \{a\})$ when needed (i.e. when an algorithm requests the value of $discern(B \cup \{a\})$. This property of certain submodular functions is in fact exploited by several optimization algorithms that use lazy evaluation.

In fact, in order to calculate $discern(B)$, it suffices to know the cardinalities of partitions in partitioning of d induced by $IND(B)$ (see Figure 2 and the example in the next section). In other words, it suffices to determine the appropriate contingency table (pivot table or cross tabulation).

3 Application of Rough Set methods to Submodular Function Optimization

In this section we provide an example method previously applied in Rough Set Theory that can be applied to other submodular functions.

We will focus on submodular functions f with the following two properties:

- f depends on an underlying (fixed) decision or information system and the argument of f is a subset of attributes of this decision (information) system.
- $f(B)$ is determined by the contingency table of attributes from $B \cup \{d\}$.

Examples of such functions are previously mentioned *discern*, *Entropy* of a partition, and *Gini index*[11].

In various data mining applications one faces an optimization problem in which the dataset at hand contains numerous default values. For example, in Data Mining Cup 2009, more than 90% attribute values had the same default value (zero). Another potential area is text mining, where Boolean model of Information Retrieval (and Vector Space Model) usually lead to a representation of a collection of documents which is sparse.

When mining huge data repositories, the data may be stored in a relational database and only accessed through SQL queries. A convenient data representation that can handle optimization of functions mentioned earlier is in terms of EAV (entity-attribute-value) triples, rather than tables, which often leads to data compression. Table (b) shows EAV representation system from Table (a) in which attribute values *MSc* (a_1), *High* (a_2), *Yes* (a_3) and *Neutral* (a_4) are regarded as default values (and hence omitted). We assume that values of the decision attribute are stored in a separate table.

Suppose that an optimization algorithm performs calculation of $f(B \cup \{a\})$. When the data set is represented (compressed) in EAV format, determining the partitioning of objects induced by $IND(B \cup \{a\})$ can be greatly simplified if the partitioning of objects induced by $IND(B)$ is known beforehand. It suffices to update partition identifiers of objects without missing values on attribute a. Figure 1 illustrates this step.

Partitioning of d induced by $IND(B \cup \{a\})$ suffices to determine the value of $f(B \cup \{a\})$, since the value of $f(B \cup \{a\})$ only depends on the contingency table of attributes from $B \cup \{a\}$.

Let us refer to Figure 2 for an illustration (for function *discern*). Upon determining the contingency table (which counts decision values in each partition), $c(\{a_1, a_3\})$ is the number of conflicting pairs within each partition, i.e.:

$$c(\{a_1, a_3\}) = 2 * 1 + 0 * 1 + 1 * 1 + 0 * 1 + 1 * 0 = 3$$

Similarly, there are 4 objects with decision A and 4 objects with decision R, hence $c(\emptyset) = 4 * 4$. Finally, $discern(\{a_1, a_3\}) = c(\emptyset) - c(\{a_1, a_3\}) = 13$.

In [13] we have demonstrated a SAS implementation of the greedy heuristic (for short decision reduct calculation) working on large and sparse data sets.

	Diploma	Experience	French	Reference	Decision
x_1	MBA	Medium	Yes	Excellent	Accept
x_2	MBA	Low	Yes	Neutral	Reject
x_3	MCE	Low	Yes	Good	Reject
x_4	MSc	High	Yes	Neutral	Accept
x_5	MSc	Medium	Yes	Neutral	Reject
x_6	MSc	High	Yes	Excellent	Accept
x_7	MBA	High	No	Good	Accept
x_8	MCE	Low	No	Excellent	Reject

(a)

obj.	attr.	value
x_1	a_1	MBA
x_1	a_2	Medium
x_1	a_4	Excellent
x_2	a_1	MBA
x_2	a_2	Low
x_3	a_1	MCE
x_3	a_2	Low
x_3	a_4	Good
x_5	a_2	Medium
x_6	a_4	Excellent
x_7	a_1	MBA
x_7	a_3	No
x_7	a_4	Good
x_8	a_1	MCE
x_8	a_2	Low
x_8	a_3	No
x_8	a_4	Excellent

(b)

Table 1: Example decision table (a) and EAV representation of this decision table (b).

4 Application of Submodular Functions in Rough Set Theory

In Table 2 we provide interpretation of several submodular function optimization problems in terms of decision systems. The table follows the outline given in [6], although we narrow the exposure to maximization problems and to algorithms that solve constrained problems (discern, Entropy and Gini index are all monotone). All problems mentioned in this table have approximate solvers available in an open source package SFO[6] for Matlab. Furthermore, most algorithms mentioned in the table use lazy evaluations and have provable approximation bounds. We further provide interpretations or potential applications of these problems in terms of decision systems.

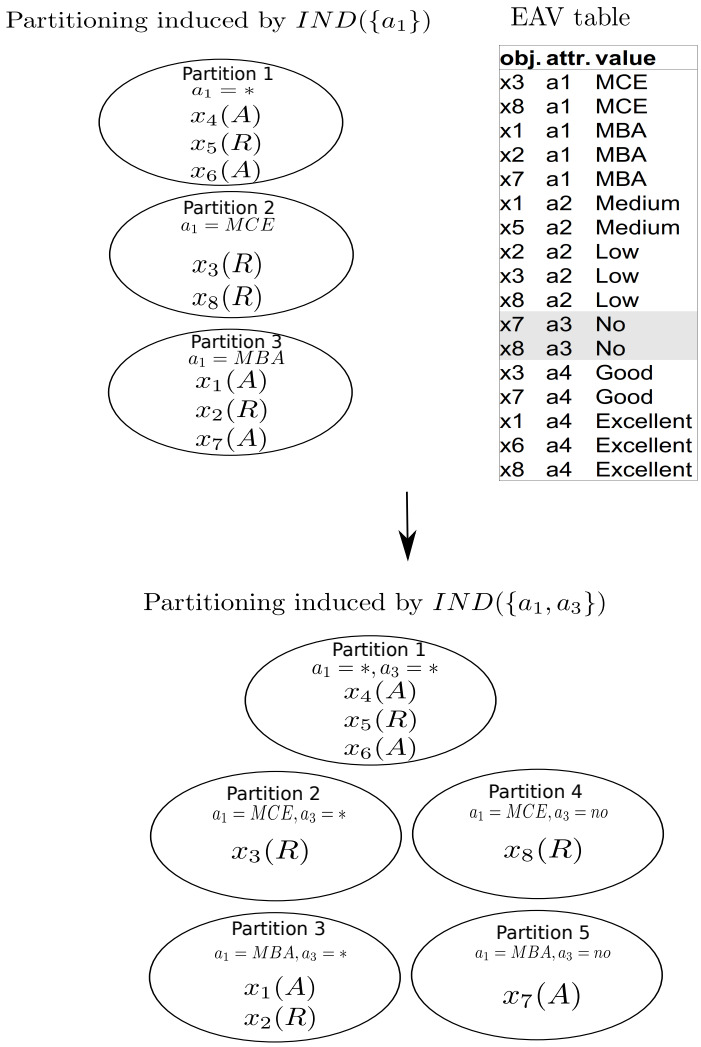
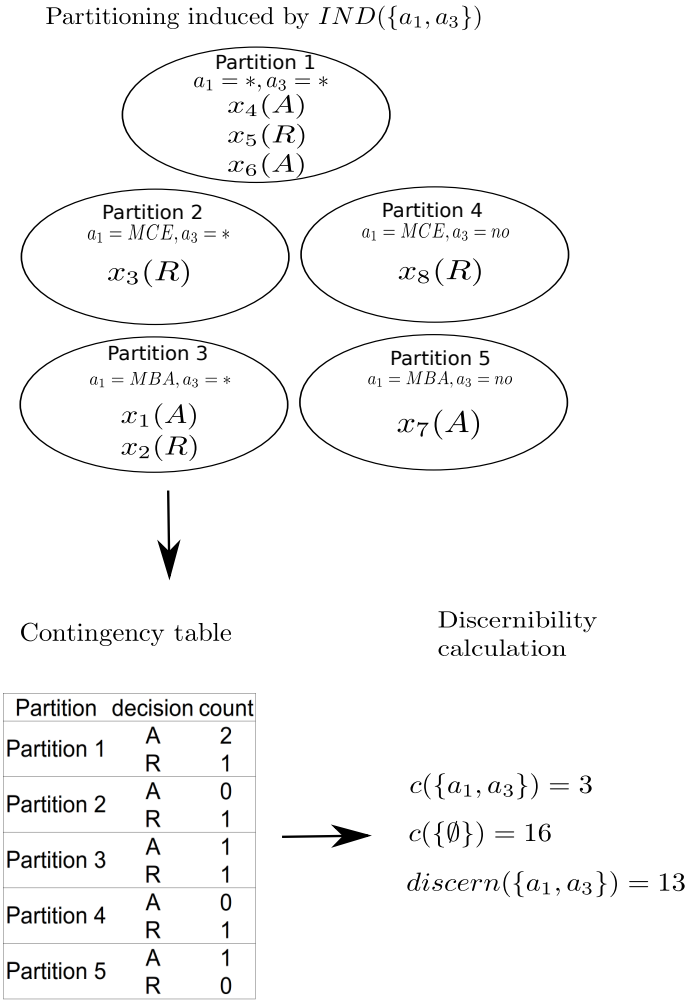


Fig. 1: Determining a subpartitioning induced by indiscernibility relation of a sparse decision system.



Partition	decision	count
Partition 1	A	2
	R	1
Partition 2	A	0
	R	1
Partition 3	A	1
	R	1
Partition 4	A	0
	R	1
Partition 5	A	1
	R	0

Fig. 2: Calculating *discern* using a contingency table.

Table 2: A subset of algorithms implemented in [6] and their interpretations or potential applications in terms of decision systems.

Greedy algorithm [8]	Constrained maximization (originally for solving the uncapacitated location problem). For a constant cost function it is similar to MD heuristic, although the stopping criterion is different – the algorithm approximately solves $A^* = argmax_A F(A)$ with $C(A) \leq B$ for a cost function C and specified budget B. Costs associated with attributes can be naturally interpreted within the context of test-cost-sensitive learning.
CELF [7]	It solves (approximately) the same optimization problem as the greedy algorithm, although it utilizes greedy and cost-agnostic greedy algorithms.
pSPIEL [3]	Similar to the formulation above, with $C(A)$ being the cost of the cheapest path connecting nodes A in a graph. An example problem that calls for a graph structure may be as follows: A decision system corresponds to a deterministic phenomena such that different attributes are measured/collected in different locations. How to choose a set of attributes sufficient to determine the decision so that the overall distance needed to traverse in order to collect values of attributes from this reduct is (not far from) minimal?
SATURATE [4]	Approximately solve $A^* = argmax_{\|A\| \leq k} min_i F_i(A)$ A potential application is determining a joint (approximate) reduct in a decision system with multiple decision variables, or an approximate reduct in a multiclass decision system such that misclassification is not overly disproportionate between classes.
eSPASS [5]	Approximately solving $max_{\|A_1 \cup \ldots \cup A_k\| \leq m} min_i F(A_i)$. This algorithm can be used to find a set of k disjoint approximate reducts or k.

5 Conclusions

In this article we presented the connection between Rough Set theory methods and methods from Submodular Function theory. The key (although very simple) observation is that discernibility is a monotone submodular function. We have briefly discussed an example method previously developed in Rough Set theory framework (i.e., lazy evaluation of discernibility when the data set contains numerous default values) and discussed its application to other submodular functions. We have also provided the interpretation or potential applications of several submodular function maximization problems in terms of Rough Set Theory and decision systems.

An example specific problem which to our knowledge has not been previously addressed is as follows: Find a set of k disjoint decision reducts of a decision sys-

tem $\mathbb{D} = (U, A \cup \{d\})$. eSPASS algorithm mentioned in [5] gives an approximate solution to this problem.

Acknowledgement:

This work is partially supported by the National Centre for Research and Development (NCBiR) under Grant No. SP/I/1/77065/10 by the Strategic scientific research and experimental development program: "Interdisciplinary System for Interactive Scientific and Scientific-Technical Information".

References

1. Fujishige, S. *Submodular Functions and Optimization*, Elsevier, 2005.
2. Johnson, D., S. Approximation algorithms for combinatorial problems In *Journal of Computer and System Sciences*, 9:256–278, 1974.
3. Krause, A., Guestrin, C., Gupta, A., Kleinberg, J. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *IPSN*, 2006.
4. Krause, A., Singh, A., Guestrin, C. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. In *JMLR*, volume 9, 2008.
5. Krause, A., Rajagopal, R., Gupta, A., Guestrin, C. Simultaneous placement and scheduling of sensors. In *Information Processing in Sensor Networks*, 2009.
6. Krause, A. SFO: A Toolbox for Submodular Function Optimization In *Journal of Machine Learning Research*, 11:1141–1144, 2010.
7. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., Glance, N. Cost-effective outbreak detection in networks. In *KDD*, 2007.
8. Nemhauser, G., Wolsey, L., Fisher, M. An analysis of the approximations for maximizing submodular set functions In *Mathematical Programming*, 14:265–294, 1978.
9. Nguyen, H. S.: Approximate Boolean Reasoning: Foundations and Applications in Data Mining. Transactions on Rough Sets: Volume 5, 2006, pages. 334–506 (2006)
10. Pawlak, Z. *Rough Sets. Theoretical Aspects of Reasoning about Data.* Springer, Formerly Kluwer Academic Publishers, Boston, Dordrecht, London, 1991.
11. Bassem Sayrafi, Dirk Van Gucht, and Marc Gyssens. Measures in databases and datamining. Tech. Report TR602, Indiana University Computer Science, 2004.
12. Komorowski, J., Pawlak, Z., Polkowski, L., Skowron, A. Rough Sets: A Tutorial In *Rough Fuzzy Hybridization: A New Trend in Decision-Making*, pages 3–98. Springer, Heidelberg, 1998.
13. Świeboda, W., Nguyen, H. S. Mining large and sparse data sets with Rough Sets. In *Proceedings of CS&P*, 2010.