A Framework for Evaluating the Smoothness of Data-Mining Results

Gaurav Misra, Behzad Golshan, and Evimaria Terzi

Boston University, Computer Science Department
{gm,behzad,evimaria}@cs.bu.edu

Abstract. The data-mining literature is rich in problems that are formalized as combinatorial-optimization problems. An indicative example is the *entity-selection* formulation that has been used to model the problem of selecting a subset of representative reviews from a review corpus [11, 22] or important nodes in a social network [10]. Existing combinatorial algorithms for solving such entity-selection problems identify a set of entities (e.g., reviews or nodes) as important. Here, we consider the following question: how do small or large changes in the input dataset change the value or the structure of the such reported solutions?
We answer this question by developing a general framework for evaluating the *smoothness* (i.e, consistency) of the data-mining results obtained for the input dataset X. We do so by comparing these results with the results obtained for datasets that are within a small or a large distance from X. The algorithms we design allow us to perform such comparisons effectively and thus, approximate the results' smoothness efficiently. Our experimental evaluation on real datasets demonstrates the efficacy and the practical utility of our framework in a wide range of applications.

1 Introduction

Given a collection of reviews about a product, which are the most valuable reviews in a collection? In a social network of users, which are the most influential nodes in the network?

Many of such *entity-selection* problems (where the term entities is used as an abstraction for reviews and node networks) have been formalized in the data-mining literature as combinatorial-optimization problems; e.g., using coverage [11, 22] or influence-maximization objectives [10]. Such formulations are characterized by a *solution space* and an *objective function*. The former defines the types of solutions the data analyst is interested in, e.g., a set of reviews or nodes. The latter provides a means for evaluating the *goodness* or *value* of every solution for a given dataset. In this context, the combinatorial methods output a single solution from the solution space – the one that optimizes (or approximates) the objective function.

While existing work focuses on designing efficient algorithms for finding (or approximating) this optimal solution, we claim that these algorithms fail to quantify how the *nature of the input dataset* affects the reported solutions. For example, if small changes in the input graph alter significantly the reported set of influential nodes, then any social study relying on existing algorithms would be highly untrustworthy. Similarly, if the set of influential nodes reported in the input dataset are identified as influential in *any* random dataset, then again the reported solution is insignificant.

In this paper, we propose a framework that allows us to quantify the relationship between the combinatorial solutions obtained by existing algorithms to the *nature of the input dataset*. More specifically, we propose the *smoothness measure*, which quantifies how the value and the structure of the reported solutions is altered by small (or large) changes in the input dataset. Related to ours is the work on statistical-significance testing of data-mining results using empirical p-values [5, 17, 16, 23]. However, the empirical p-values encode the probability that there exists a random dataset with solution with (almost) identical value to the solution obtained for the original dataset. On the other hand, the smoothness is the expected similarity between the solutions of the sampled and the original datasets. The focus on the expected similarity of the solutions, allows us to sample datasets either randomly (as in the case of p-values) or within specified neighborhoods around the original data. This flexibility allows us to quantify not only the statistical significance of the obtained solutions, but also the variation of the solution within these small neighborhoods.

Another key characteristic of our work is that it departs from the narrow characterization of solutions using only the objective function. In particular, we view the solutions as combinatorial objects that have both value and structure. After all, the objective function is only a proxy for the analyst's intuition of what constitutes a good solution. In the case of the entity-selection problem the structure of the solution is determined by the actual elements that are selected to the reported set. Therefore, when comparing solutions not only do we compare their values, but also the actual entities they contain.

Apart from the proposition of the smoothness framework, which to the best of our knowledge is new, we also present a new efficient algorithm for sampling datasets that are within a certain distance from the original dataset. This algorithm is a key component to our algorithmic solution for approximating the smoothness of the data-mining results for entity-selection problems. Our experimental results (presented in Section 4) validate the utility and the practical utility of our framework as well as our algorithms in a wide range of entity-selection problems. Some indicative ones are the selection of representative set of reviews, authors within communities as well as nodes in social networks.

Although we present the application of our framework to entity-selection problems, we point out that our framework is general and can be used to evaluate the smoothness of all data-mining results – as long as the data-mining problem has been formalized as a combinatorial optimization problem.

The rest of the paper is organized as follows: first, we describe our framework in Section 2. In Section 3 we describe our algorithms for sampling datasets and in Section 4 we present experiments that demonstrate the utility of our methods. After reviewing the related work in Section 5, we conclude the paper in Section 6.

2 Framework Overview

In this section, we provide a generic description of our framework that takes as input a dataset, an objective function, and an algorithm for optimizing it, and reports how the solution reported by the algorithm is affected by small or large changes in the input dataset.

2.1 Optimization Problems

Every combinatorial-optimization problem consists of the following components: the input dataset X, the designated solution space $\mathscr{L}$and the objective function F, which maps every solution $\mathcal{S} \in \mathscr{L}$ to a real number $F(X, \mathcal{S})$. Henceforth, we use tuple $\langle X, \mathscr{L}, F \rangle$ to represent a combinatorial optimization problem, and $\mathcal{S}^*_X$ to represent the optimal solution for input dataset X. Here, we focus on maximization problems, where $\mathcal{S}^*_X$ is the solution for which $F(X, \mathcal{S})$ is maximized:

$$\mathcal{S}^*_X = \text{argmax}_{\mathcal{S} \in \mathscr{L}} F(X, \mathcal{S}).$$

We will use $\mathcal{S}^*$ instead of $\mathcal{S}^*_X$ whenever the dataset X is clear from the context.

Note that finding $\mathcal{S}^*$ might be **NP**-hard for some problems. In these cases, different heuristics and approximation algorithms are used to obtain a suboptimal solution. We will use $\mathcal{S}^*$ to denote the solution (optimal or not) produced by the algorithm chosen to solve the problem.

Among all data-mining problems that have been formalized using this framework are the following two, which we consider in this paper: the review-selection and the node-selection problems. Consider a collection of reviews for a particular product. This product has a set of features that can be commented in a review. For instance, the features of an mp3-player are: battery, sound quality, ability to record, and etc. Each reviewer may post a review that only covers a subset of these features (say only battery and sound quality). The goal of the review-selection problem is to pick k reviews that cover the maximum number of distinct features of the product.

Similarly, in the node-selection problem, the goal is to pick a set of k important nodes from a given network. That is, to select k nodes that have the highest influence over the entire network. Of course, in order to define such nodes we need to rely on an *information propagation* model. Given such a model, information propagates from active to inactive nodes; this propagation is usually probabilistic. Using an information propagation model, the important nodes are the ones that upon activation, will lead to maximum number of (expected) active nodes in the network. There are many applications for this problem. For instance, in a given graph of social relations, the selected nodes can be used to obtain an optimal advertising strategy.

We collectively refer to the review and the node-selection problems as *entity-selection problems*. The former one has been formalized using the MaxCov formulation [11, 22], and the latter, using the MaxInfluence formulation [10]. We describe these formulations next.

MaxCov(k): Given a universe of items U and m subsets of it ($\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$), the goal of MaxCov problem is to pick k elements from $\mathcal{C}$ so that the number of distinct items that they cover from U is maximized. Formally, we would like to pick $\mathcal{S} \subseteq \mathcal{C}$ with $|\mathcal{S}| = k$ such that

$$F\text{-Cov}(\mathcal{S}) = \left| \bigcup_{C \in \mathcal{S}} C \right| \tag{1}$$

is maximized. In the case of review selection, the elements of $\mathcal{C}$ are the reviews and the universe consists of the product's features.

In an instance of MaxCov problem $\langle X, \mathscr{L}, F \rangle$, the input dataset X is the set of universe subsets $\mathcal{C}$. $\mathscr{L}$ is the solution space and contains all valid solutions like $\mathcal{S}$ where $\mathcal{S}$ is the set of k selected subsets ($\mathcal{S} \subseteq \mathcal{C}$). Finally, F is the coverage objective function given by Equation (1).

The MaxCov problem is known to be **NP**-hard. However, a greedy method of picking the subset with the largest number of uncovered elements at each step, is an $(1 - \frac{1}{e})$-approximation algorithm for the MaxCov problem. Henceforth, the output of the mentioned greedy algorithm is what we refer to as $\mathcal{S}^*$ for the MaxCov problem.

Observe that input dataset X, can be represented by a binary matrix in the following way: each column corresponds to an element in the universe U, and each row corresponds to one of the given subsets. The binary matrix has a value of 1 in position (i, j) *iff* the subset C_i contains the j-th element of U. Throughout, we consider X to be the mentioned binary matrix to keep the notation simple.

MaxInfluence(K): Given a graph $G = (V, E)$ in which all the nodes are inactive, the goal is to find the best k nodes to activate, so that the expected number of active node after propagation would be maximized. Formally, we would like to pick $\mathcal{S} \subseteq V$ with $\mathcal{S} = k$ such that the following objective function would be maximized:

$$F\text{-Influence}(\mathcal{S}) = \text{ expected number of active nodes.}$$

The computations of the expected number of active nodes depends on the propagation model. In our study, we focus on the *independent cascade* (IC) model [10].[1]

In an instance of MaxInfluence problem $\langle X, \mathscr{L}, F \rangle$, X is the input graph $G = (V, E)$. $\mathscr{L}$ is the set of all possible solutions where each solution $S \subseteq V$ and $|S| = k$. Finally, The objective function F is the F-Influence function.

Solving the MaxInfluence problem is also **NP**-hard. However, a greedy algorithm that at every step picks the node with the largest marginal increase in the objective function again achieves an $(1 - \frac{1}{e})$-approximation for the objective. Henceforth, the output of this greedy algorithm is what we refer to as $\mathcal{S}^*$ for the MaxInfluence problem.

Similar to MaxCov problem, the input dataset X can be represented using a binary matrix. Since the input dataset is a graph, X can be the adjacency matrix that describes the input graph.

[1] Our results carry over to other propagation models.

2.2 Evaluating data-mining results

Here we introduce *smoothness* and show how it can be adjusted to reveal different aspects of a given combinatorial optimization-problem.

We formally, define the smoothness of a combinatorial optimization-problem $\langle X, \mathscr{L}, F \rangle$ using the following formula:

$$\textsc{Smoothness}(X) = \sum_{X' \in \mathcal{X}} Pr(X') Sim(\mathcal{S}^*_X, \mathcal{S}^*_{X'})$$

The formula consists of three main parts: the *dataspace* $\mathcal{X}$, the *sampling probability distribution* $Pr(X')$, and the *similarity function* $Sim(\mathcal{S}^*_X, \mathcal{S}^*_{X'})$. The rest of this section describes these three components.

Dataspace: The dataspace $\mathcal{X}$ is the set of all datasets which share a *structural characteristic* with X. Intuitively, $\mathcal{X}$ is the collection of datasets which compare with X. So far, we have shown that the input dataset for both MaxCov and MaxInfluence problems is a binary matrix. As a result, we introduce two different possible choices of $\mathcal{X}$ for binary matrices: the *exchange dataspace* $\mathcal{X}_E$ and the *swap dataspace* $\mathcal{X}_W$. The former is the set of all datasets with the same number of ones as X. The latter contains the datasets that have the same row and column marginals as X. Finally, we point out that for each dataspace $\mathcal{X}$, there is a natural distance function $D(Y, Z)$ which returns the distance between datasets Y and Z in $\mathcal{X}$. Here, we use the following distance function to measure the distance between two binary matrices:

$$D(Y, Z) = |\{(i, j) | Y_{i,j} = 1 \wedge Z_{i,j} = 0\}| \tag{2}$$

Sampling probability distribution: Given a particular dataspace $\mathcal{X}$, $Pr(X')$ defines the probability of sampling dataset X'. We introduce two natural sampling schemes: *uniform* and *neighborhood* sampling. The probability distribution of sampling dataset X' in uniform and neighborhood sampling schemes are denoted using $U\text{-}Pr(X')$ and $N\text{-}Pr(X')$ respectively, and can be defined using the following formulas:

$$U\text{-}Pr(X') \propto \frac{1}{|\mathcal{X}|}$$

$$N\text{-}Pr(X') \propto e^{-\lambda D(X, X')} \tag{3}$$

Note that in the above formula, λ can be used to adjust the probability distribution. Using a high value for λ, assigns high probability to the datasets in the neighborhood of the original dataset. We use λ equal to 2 for our experiments. Finally, note that if we set λ to zero, then the probability distribution will be uniform. Also note that any distance function between datasets can be used in Equation (3). Here, we use the one defined in Equation (2).

Similarity function: The similarity function measures how close the solutions for datasets X and X' are. There are two types of similarity functions: *value-based similarity* and *structure-based similarity.* The value-based similarity, denoted as $V\text{-}Sim(\mathcal{S}^*_X, \mathcal{S}^*_{X'})$, only compares the value of the objective functions for both solutions, and can be computed using the following formula:

$$V\text{-}Sim(\mathcal{S}^*_X, \mathcal{S}^*_{X'}) = \frac{F_{\text{Max}} - |F(X, \mathcal{S}^*_X) - F(X', \mathcal{S}^*_{X'})|}{F_{\text{Max}}}.$$

In the above formula, F_{Max} is the maximum possible value of the objective function which is used to normalize the similarity measure.

The structure-based similarity, compares the combinatorial structure of the obtained solutions. The definition of this similarity measure highly depends on the combinatorial structure of the solution. Fortunately, for both MaxCov and MaxInfluence problems, the optimal solution is a set of k elements, so any similarity measure among sets can be used. In this work, we compute the structural similarity between solutions $\mathcal{S}^*_X$ and $\mathcal{S}^*_{X'}$ with cardinality k as:

$$S\text{-}Sim(\mathcal{S}^*_X, \mathcal{S}^*_{X'}) = \frac{|\mathcal{S}^*_X \cap \mathcal{S}^*_{X'}|}{k}.$$

Finally, note that the above similarity measures are normalized in order to return values between zero and one.

2.3 Discussion

Notation: Given the different choices that we have for the dataspace, the sampling probability distribution, and the similarity function, we can define eight different smoothness measures. Table 1 contains the names of all these measures along with the configuration of each part of the measure.

Table 1. The eight configurations of Smoothness

	Dataspace	Sampling	Similarity
EUV-Smoothness	Exchange	Uniform	Value
EUS-Smoothness	Exchange	Uniform	Structure
ENV-Smoothness	Exchange	Neighborhood	Value
ENS-Smoothness	Exchange	Neighborhood	Structure
WUV-Smoothness	Swap	Uniform	Value
WUS-Smoothness	Swap	Uniform	Structure
WNV-Smoothness	Swap	Neighborhood	Value
WNS-Smoothness	Swap	Neighborhood	Structure

Interpretation of smoothness: Recall that the value of Smoothness(X) represents the expected similarity of a solution sampled from $\mathcal{X}$ and the original solution $\mathcal{S}^*_X$. When smoothness is computed using uniform samples from dataspace $\mathcal{X}$, small smoothness values are an indication of the interestingness and statistical significance of the reported solution $\mathcal{S}^*_X$. In this case, small smoothness values mean that $\mathcal{S}_X^*$ cannot be obtained at random. However, when smoothness is computed from neighborhood samples, then large values of smoothness are desirable since they are indicative of the stability of $\mathcal{S}_X^*$ to small changes of the input data. Thus, in terms of smoothness, the ideal $\langle X, \mathscr{L}, F \rangle$ is one which has a small value of smoothness when sampling uniformly and a large value of smoothness when sampling in the neighborhood.

When choosing the right dataspace for smoothness evaluation of $\langle X, \mathscr{L}, F \rangle$ one must keep in mind that the exchange dataspace ($\mathcal{X}_E$) is a much larger superset of the swap dataspace ($\mathcal{X}_W$). Interestingly, depending on the nature of $\langle X, \mathscr{L}, F \rangle$ one dataspace may give more insight than the other. As an example, consider the MaxCov problem, solved by the greedy approximation algorithm. In this case, the solutions reported by the algorithm are highly dependent on the marginals of the input dataset – after all, the greedy algorithm always reports the row of the binary matrix X that has the largest number of 1s and most of the real-life datasets have heavy tailed marginal distribution. In such a case, smoothness measures computed in $\mathcal{X}_W$ would provide very limited information since samples in $\mathcal{X}_W$ maintain a constant marginal distribution and thus would tend to produce a similar solution for every sample. Therefore, the smoothness values obtained for swap dataspace would be very large. However, this large value is mainly an artifact of the algorithm used and offers limited information about the dataset itself. On the other hand, if smoothness were to be computed in $\mathcal{X}_E$ the results would be much more informative since the samples would be drawn from a much larger space allowing for more variation in the sampled datasets.

For seeing the usefulness of both the value and structural smoothness, consider the problem of identifying the right value of k for the MaxCov(k) problem. Smoothness can be used to find such k as follows: first pick a dataspace (e.g., $\mathcal{X}_E$) and compute the four smoothness measures associated with it (e.g., EUV-Smoothness(X), ENV-Smoothness(X), EUS-Smoothness(X) and ENS-Smoothness(X) for each value of k). Given that large (resp. small) values for neighborhood (resp. uniform) smoothness are desirable one can pick the k that achieves such values both in terms of the structure and the value of the solutions.

Smoothness and p-values: There is an analogy between the smoothness score Smoothness(X) and (empirical) p-values [5, 6, 9, 14, 17, 16, 18, 23], used to evaluate the statistical significance of the value of the solution obtained for $\langle X, \mathscr{L}, F \rangle$. However, p-values encode the probability that there exists a random dataset from $\mathcal{X}$ with solution of (almost) identical value to $\mathcal{S}_X^*$. On the other hand, Smoothness(X) is the expected similarity between solutions sampled from $\mathcal{X}$. Moreover, contrary to p-values, smoothness can be computed both with respect to the structure as well as the value of the solutions, and both for neighorhood and uniform samples.

However, the key advantage of smoothness – when compared to p-values – is the following: p-values simply *count* how many datasets have solutions with similar value as the original dataset. Smoothness takes into account the values of these solutions. In that way, smoothness provides a more comprehensive quantification of the structure of the solution space.

3 Sampling Datasets from the Dataspace

Computing Smoothness(X) precisely requires generating all possible datasets in $\mathcal{X}$ and running the optimization algorithm on all of them. This computation is infeasible since the number of datasets in both $\mathcal{X}_W$ and $\mathcal{X}_E$ is exponentially large in most cases. In this section, we explain how different smoothness measures can be accurately estimated by sampling from the exchange and the swap dataspaces.

3.1 Sampling from the exchange dataspace

Recall that the exchange dataspace $\mathcal{X}_E$ consists of all datasets with the same size and the same number of 1's as the input dataset X. For the rest of the discussion, we will use N_1 (resp. N_0) to denote the number 1's (reps. 0's) in X. Finally, we will use N to be the total number of entries in X: $N = N_1 + N_0$.

Uniform sampling from the exchange dataspace: The following algorithm draws a sample matrix Y uniformly at random from the exchange dataspace X_E: Randomly select N_1 entries of Y. Set them to 1, and set all the other entries to 0. The random selection of N_1 entries can be done by permuting all entries of Y and picking the first N_1 entries. As a result, the overall running time of this algorithm is $O(N)$.

Neighborhood sampling from the exchange dataspace: A naïve Markov Chain Monte Carlo (MCMC) algorithm for neighborhood sampling from X_E performs a random walk on the state space that consists of the distinct elements of X_E. A transition from $Y \in X_E$ to $Y' \in X_E$ happens via a single *exchange* operation. An exchange operation selects uniformly at random an 1-valued entry and a 0-valued entry from Y, and makes their values 0 and 1 respectively. Clearly, an exchange operation does not change the number of 1's in the matrix.

The number of possible exchanges for all matrices in $\mathcal{X}_E$ is the same and equal to $N_1 \times N_0$. This implies that the out-degree of each state in the mentioned Markov chain is $N_1 \times N_0$. Since exchange is a reversible operation, the in-degree of each state is also $N_1 \times N_0$. Based on these observations, we can conclude that the stationary distribution for this Markov chain is uniform.

By applying a *Metropolis-Hastings* technique [7, 13] on the mentioned Markov chain, we can change the uniform distribution to the desired neighborhood distribution. With Metropolis-Hastings, we do a transition from state Y to state Y' with probability $\min\{e^{\lambda(\mathrm{Dist}(X,Y)-\mathrm{Dist}(X,Y'))}, 1\}$.

Metropolis-Hastings guarantees that the stationary distribution matches our definition of neighborhood sampling. Although the above MCMC-based method obtains samples from the right distribution, it is computationally expensive; The state space of the Markov chain is $\binom{N}{N_1} \approx N^{N_1}$. Also, it is not easy to prove that the Markov chain will converge in polynomially many steps.

However, we can use the following observation to design a more efficient sampling algorithm.

Observation 1 *Consider a single transition of the naïve MCMC algorithm from state Y to state Y'. Also, assume that (i, j) and (i', j') are the positions of the selected 0-valued and 1-valued entries for the exchange operation. This means that: $Y_{i,j} = Y'_{i',j'} = 0$ and $Y_{i',j'} = Y'_{i,j} = 1$. One can observe that based on the values in the original matrix X, $D(Y', X)$ can only take one of the following three values:*

(a) If $X_{i,j} = 1, X_{i',j'} = 0$, then $D(Y', X) = D(Y, X) - 1$.
(b) If $X_{i,j} = 0, X_{i',j'} = 1$, then $D(Y', X) = D(Y, X) + 1$.
(c) If $X_{i,j} = X_{i',j'}$, then $D(Y', X) = D(Y, X)$.

The above observation hints that in this Markov chain, not every state can transition to another state. In fact, from states W_d that are within distance d from the original matrix, we can only transition to states in W_d, W_{d-1}, and W_{d+1}. The following proposition shows that the probability of such transitions can be computed analytically.

Proposition 1. *If W_d is the set of all datasets in $\mathcal{X}_E$ such that for every $Y \in W_d$ $D(Y, X) = d$, then a single exchange operation leads to dataset Y' which belongs to W_d, or W_{d-1} or W_{d+1}. The probabilities of these events are:*

$$Pr(W_d \rightarrow W_{d-1}) = \frac{d^2}{N_1 \times N_0}, \tag{4}$$

$$Pr(W_d \rightarrow W_{d+1}) = \frac{(N_1 - d)(N_0 - d)}{N_1 \times N_0} \times e^{-\lambda}, \tag{5}$$

$$Pr(W_d \rightarrow W_d) = 1 - Pr(W_d \rightarrow W_{d+1}) - Pr(W_d \rightarrow W_{d-1}).$$

Using Proposition 1, we propose a new sampling algorithm, the *Xchange-Sampler*, which is both efficient and samples from the right (neighborhood) distribution. The pseudocode of *Xchange-Sampler* is shown in Algorithm 1.

First, the algorithm forms a Markov Chain $\mathcal{M}$ with state space $\mathcal{W}$ and a transition matrix P: $\mathcal{M} = \langle \mathcal{W}, P \rangle$. Each state W_d in $\mathcal{W}$, corresponds to datasets that are within distance d form the original dataset. The probabilities of transitions from any state W_d to states W_{d-1}, W_{d+1}, and W_d are given by the equations in Proposition 1. All these transition probabilities are summarized into the transition matrix P. Using the `StationaryProb` routine, *Xchange-Sampler* first computes the stationary probability distribution π of this Markov chain. Given π, the algorithm then samples z samples from the exchange dataspace as follows: First, using the `SampleDistance` function, it samples state W_d from $\mathcal{W}$ according to the stationary distribution π. Given W_d, the `UniformSample` function samples one of the datasets within d uniformly at random. This latter

Algorithm 1 The *Xchange-Sampler* algorithm problem.

```
   **Input:** binary matrix $X$, integer $z$
   **Output:** binary matrices $Y_1, Y_2, \ldots, Y_z$
1: $\pi \leftarrow$ StationaryProb($\mathcal{W}, P$)
2: for $i = 1 \rightarrow z$ do
3:     $W_d \leftarrow$ SampleDistance($\pi$)
4:     $Y_i \leftarrow$ UniformSample($W_d$)
5:     report $Y_i$
```

step can be implemented by randomly picking d 1-valued entries and d 0-valued entries from X, and setting them to 0 and 1 respectively.

Observe, that *Xchange-Sampler* simulates the naïve MCMC approach. The difference is that in *Xchange-Sampler*, all states that are within distance d from X are merged into a single state W_d, while in MCMC they are all distinct states. Then, the Markov chain $\mathcal{M}$ is constructed in such a way, so that in the stationary probability distribution, $\pi(W_d)$ is the same as the sum of the stationary probabilities that the MCMC random walk would end in any state that is within distance d from X. Thus, *Xchange-Sampler* samples datasets by first picking their distance d (using π) and then sampling uniformly from W_d.

Observe that the number of states in the Markov chain $\mathcal{M}$ is equal to the largest distance between any two datasets in the exchange dataspace, which is at most $min\{N_1, N_0\}$. This state space is significantly smaller than the state space of the naïve MCMC method since the former is bounded by the number of entries in X, while the latter was exponential. Moreover, the naïve method performs a random walk for each sample, while *Xchange-Sampler* computes the stationary distribution π only once. Given π, *Xchange-Sampler* generates each sample in $O(N)$ time.

3.2 Sampling from the swap dataspace

Recall that all the datasets in the swap dataspace ($\mathcal{X}_W$) have the same row and column marginals as the original dataset X. Gionis et al. [5] have proposed an MCMC approach to obtain uniform samples from $\mathcal{X}_W$. In that MCMC method, the state space is the set of all possible matrices, and the transition from one state to another is done using an operation called *swap*. Similar to the way that exchange operations maintain the number of 1's in a binary matrix, the swap operations guarantees to maintain the row and column marginals of the matrix. To sample datasets from the neighborhood of the original dataset, we combine the method by Gionis et al. with an additional Metropolis-Hastings technique so that we sample from the correct neighborhood distribution.

4 Experiments

In this section, we present an evaluation of the different smoothness measures that we have defined. First, we demonstrate the usefulness of value smoothness in determining the statistical significance of data mining results. Next, we compare the value smoothness measures to the traditional method of statistical-significance testing using p-values. Lastly, we evaluate structural smoothness and gauge its ability to provide additional insights on the data-mining results.

4.1 Datasets

We use the following datasets in our experiments:

Cora: This dataset [19] is a binary matrix which represents a bipartite graph between terms and scientific papers[2]. The entry (i, j) in the matrix has a value of 1 if term i appears in paper j. The **Cora** matrix has size 1433×2708, density 1.26% and is characterized by a heavy tailed marginal distribution. We use **Cora** as input to the $\textsc{MaxCov}(k)$ problem, where our goal is to pick a set of k terms that cover the maximum number of papers (dominant words).

Bibsonomy: This dataset [3] is a binary matrix which represents a bipartite graph between authors and scientific papers tagged with the tag "programming"[3]. The entry (i, j) in the matrix has a value of 1 if author i is a co-author of the paper j. The **Bibsonomy** matrix has 4410×3862 and density of ones: 0.049%. We use **Bibsonomy** as input to the $\textsc{MaxCov}(k)$ problem, where our goal is to pick a set of k authors that cover the maximum number of publications.

Co-Authors: This dataset [12] is a collaboration network derived from the General Relativity and Quantum Cosmology category of the e-print arXiv[4]. The network contains 5242 vertices, which correspond two authors. There is an edge between two authors, if they have written a paper together. The graph contains 28968 edges and has an edge density of 0.1%. We use **Co-Authors** as input to the $\textsc{MaxInfluence}(k)$ problem where our goal is to select k authors that have the maximum influence in the network.

4.2 Value Smoothness

The experiments presented in this section, aim to demonstrate the utility of value smoothness (i.e., smoothness measures where solutions are compared using value similarity). Recall that there are four different types of value smoothness (see Table 1). In order to demonstrate the utility of each type, we conducted the following experiment: first, we computed all four measures for the **Bibsonomy**, **Cora** and **Co-Authors** datasets. Figure 1 shows the corresponding values of smoothness as a function of k.

Stability of the results: First, we observe that the neighborhood smoothness is always at least as large as uniform, in both dataspaces. This demonstrates that our optimization problems are more stable within the neighborhoods of the input datasets. For **Cora** we observe high neighborhood and low uniform

[2] Available at: `www.cs.umd.edu/projects/linqs/projects/lbc/index.html`
[3] Available at: `www.kde.cs.uni-kassel.de/bibsonomy/dumps/`
[4] Available at: `http://snap.stanford.edu/data/ca-GrQc.html`

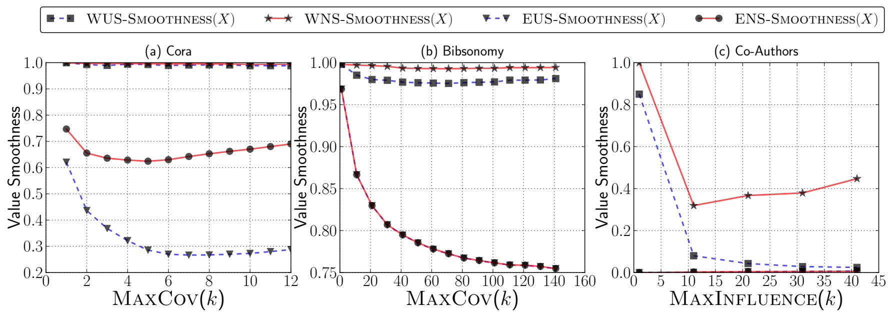

Fig. 1. Comparing the different measures of value smoothness with varying k.

smoothness making the obtained solutions stable and statistically significant. On the other hand, for **Bibsonomy** both the neighborhood and the uniform smoothness are high; this indicates that the solution obtained for this dataset is not statistically significant (i.e., could also be obtained at random). Finally, the smoothness values for the **Co-Authors** dataset is small in all cases, indicating that the solution space is unstable even in the neighborhood of the input graph.

Advantage of the exchange dataspace: The second important observation one can make from Figure 1 is that smoothness of MaxCov computed using the swap dataspace is always very close to 1, independent of whether the samples were obtained using neighborhood or uniform samplimg. This is more pronounced in **Cora** , since this datasaet also has heavy-tailed marginal distribution. As we have already discussed in Section 2.3, for this dataset the smoothness computed using $\mathcal{X}_W$ does not provide as much insight as the smoothness computed using the exchange dataspace.

Advantages of the swap dataspace: While the exchange dataspace $\mathcal{X}_E$ can prove useful for the analysis of some datasets it might prove inadequate for the analysis of others. This is particularly true when the size of $\mathcal{X}_E$ is extremely large and any reasonable number of samples is inadequate. In such cases the swap dataspace ($\mathcal{X}_W$), which is significantly smaller, gives more insightful results. For example, in 1(c) the plots for neighborhood and uniform smoothness in $\mathcal{X}_E$ completely overlap. This overlap is an artifact of the extreme sparsity of **Co-Authors**. However, the smoothness computed using $\mathcal{X}_W$ is much more informative. It demonstrates that the dataset has high smoothness in the neighborhood making it stable and at the same time has low smoothness in the uniform distribution making it statistically significant. The same effect can also be seen in **Bibsonomy** since it is even more sparse than **Co-Authors**.

4.3 Comparing value smoothness with empirical p-values

In this section, we compare smoothness to the traditional method of statistical significance testing using empirical p-values. The results of this comparison for the exchange dataspace and uniform sampling are shown in Table 2. The table shows the EUV-Smoothness values and the empirical p-values of the solutions obtained by using **Bibsonomy** as input to the MaxCov(k) problem.

Table 2. Value Smoothness and p-value for the **Bibsonomy** dataset. Sampling from exchange dataspace and 10000 samples.

	Exchange Model	
MaxCov(k)	Value Smoothness	Empirical p-value
1	0.9686	0.0000
10	0.8665	0.0000
20	0.8296	0.0000
40	0.7946	0.0000
60	0.7776	0.0000
80	0.7672	0.0000
100	0.7615	0.0000

The reported p-values suggest that the solutions obtained for for the **Bibsonomy** are statistically significant. This is because the p-values for all k are equal to zero, which means that the probability of sampling a dataset with best F-Cov value as large as **Bibsonomy** is negligible. On the other hand, the smoothness values are relatively high (larger than 0.7 in all cases). This suggests that *on expectation* the solutions on random datasets have very similar values to the solution obtained for **Bibsonomy**. Interestingly, these two findings seem to be contradictory.

The reason for this somewhat surprising result is the following: the p-values encode the probability that a random dataset has solution with value greater or (almost) equal to the original solution. In this case, there are no random datasets that satisfy this condition and therefore the p-values are equal to 0. On the other hand, the smoothness computes the average value of the solutions to the random datasets. Therefore, even if these solutions have indeed smaller values than the original one, they may still be relatively close to the original solution, yielding high smoothness values.

The above experiment is just one example where the value of smoothness is much more informative than the p-values. The reason for that is that the computation of smoothness takes into account the actual values of the solutions of *all* sampled datasets. The p-values on the other hand, simply count how many times the randomly sampled datasets have solutions with values equal to the original dataset. Therefore, the p-values ignore the values of these solutions and, inevitably, are less informative.

Note we obtained results similar to the above for other datasets as well. However, we ommit them, due to lack of space.

4.4 Structural Smoothness

Here, we conduct the same experiment as seen in Section 4.2, but we evaluate the structural smoothness instead of the value smoothness. The results we obtained are presented in Figure 2. Once again, we observe that the neighborhood is at least as smooth as the uniform distribution in all cases. In our problems, we can use structural smoothness to decide the optimal value of k. For example, by looking at the structural smoothness for **Cora** (Figure 2(a)), we can conclude that selecting value for $k = 5$ is not a good choice; this is because the smoothness computed over uniform samples for $k = 5$ is almost 1. This means that random datasets have identical solution to the input dataset. On the other hand, $k = 3$ or $k = 11$ are better choices since the the original solution has more differences with the random solutions. If we view the values of structural smoothness together with the value smoothness for the same dataset (Figure 1(a)), it becomes evident that $k = 11$ is the better choice for the value of k for **Cora**. This is because for $k = 11$, both the value and the strutural smoothness computed over random samples have relatively low values. Notice that the value smoothness alone would not have provided all the information necessary to choose the right value of k.

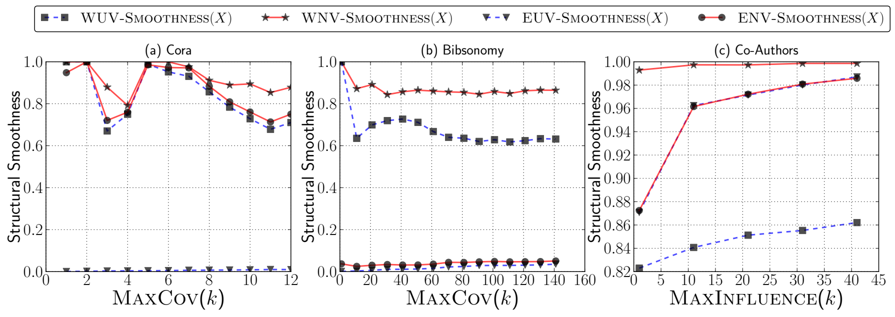

Fig. 2. Structural smoothness as a function of k.

5 Related Work

To the best of our knowledge, we are the first to introduce smoothness as a measure for evaluating how small (or large) changes in the input affect the data-mining results. However, our work has connections with existing lines of research. We discuss these connections next.

Evaluation of data-mining results: Existing work on data mining [5, 17, 16, 23] and analysis of ecological and biological datasets [6, 9, 14, 18] focuses on the evaluation of the statistical significance of the solution to $\langle X, \mathcal{S}, F\rangle$, via empirical p-values. Empirical p-values encode the probability that there exists a random dataset from $\mathcal{X}$ with solution with (almost) identical *value* to $\mathcal{S}^*_X$. On the other hand, the smoothness of a dataset is the expected similarity – in terms of value or structure – between solutions sampled from $\mathcal{X}$. However, the main difference between the smoothness framework we propose here and the empirical p-values proposed in the past is that our framework also considers non-uniform samples from $\mathcal{X}$ and explores the solutions in the "neighborhood" of X.

Moreover, our algorithms for sampling 0–1 datasets from $\mathcal{X}_E$ extend existing algorithms for uniform sampling from $\mathcal{X}_W$ [5]. However, again our algorithmic contribution goes beyond devising techniques for sampling from $\mathcal{X}_E$. For example, one of our main contributions is an efficient algorithm for sampling from the exchange dataspace $\mathcal{X}_E$.

Smoothed analysis of algorithms: Our work is also related to existing work on characterizing the stability of numerical methods [4, 8] as well as the smoothed complexity of algorithms [1, 20, 21]. The key difference between our work and existing work in these areas is that we explore the smoothness of the value as well as the structure of our results. The issue of the solutions' structure has only recently appeared in the work of Balcan et al. [2]. More specifically, Balcan et al. address the problem of finding a solution that has similar structure to the optimal solution. Although related to our study, their requirement focuses on approximating the structure of the optimal solution ignoring the rest of the solution space.

Privacy-preserving query answering: Our focus on the smoothness of the solutions to the input data instance connects our work with the work of Nissim et al. [15]. In that work, the authors focused on determining the amount of noise required for differentially-private query answering. Our focus is on studying how small changes in the data affect the data-mining results both in terms of value and in terms of structure.

6 Conclusions

In this paper, we presented a framework for evaluating the significance as well as the stability of of data-mining results. We achieved that by defining the notion of smoothness, which quantifies how these results get affected by small or large changes in the input data. In principle, our framework can be applied to all data-mining problems, which have been formalized using combinatorial-optimization formulations. For concreteness, we focused on a particular type of combinatorial-optimization problem, namely entity selection, and we demonstrated the application of our framework in this setting. From the computational point of view, our the evaluation of smoothness requires efficient sampling of datasets that are within a certain distance from the input dataset. As another contribution, we presented an efficient algorithm for obtaining such samples. Our preliminary experimental results demonstrates that the values of smoothness provide valuable insights about the structure of the solution space and the significance of the obtained data-mining results.

Acknowledgments: The authors want to thank John Byers for useful discussions. This research was supported by the NSF award #1017529 and by gifts from Microsoft, Yahoo! and Google.

References

1. D. Arthur, B. Manthey, and H. Röglin. k-means has polynomial smoothed complexity. In *FOCS*, pages 405–414, 2009.
2. M.-F. Balcan, A. Blum, and A. Gupta. Approximate clustering without the approximation. In *SODA*, pages 1068–1077, 2009.
3. Knowledge and Data Engineering Group, University of Kassel: Benchmark Folksonomy Data from Bibsonomy. Version of June 30th, 2007.
4. R. L. Burden and J. D. Faires. *Numerical Analysis.* Thomson Brooks/Cole, 2005.
5. A. Gionis, H. Mannila, T. Mielikäinen, and P. Tsaparas. Assessing data mining results via swap randomization. In *KDD*, pages 167–176, 2006.
6. N. Haiminen, H. Mannila, and E. Terzi. Comparing segmentations by applying randomization techniques. *BMC Bioinformatics*, 8, 2007.
7. W. Hastings. Monte carlo samping methods using markov chains and their applications. *Biometrika*, 57:97–109, 1970.
8. N. J. Higham. *Accuracy and Stability of Numerical Algorithms.* Society of Industrial and Applied Mathematics, 1996.
9. N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11):1746–1758, 2004.
10. D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD*, 2003.
11. T. Lappas and D. Gunopulos. Efficient confident search in large review corpora. In *ECML/PKDD (2)*, pages 195–210, 2010.
12. J. Leskovec, J. M. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *TKDD*, 1(1), 2007.
13. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.
14. R. Milo, S. Shen-Orr, S. Itzkovirz, N. Kashtan, D. C. vskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298, 2002.
15. K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, pages 75–84, 2007.
16. M. Ojala, G. C. Garriga, A. Gionis, and H. Mannila. Evaluating query result significance in databases via randomiza tions. In *SDM*, pages 906–917, 2010.
17. M. Ojala, N. Vuokko, A. Kallio, N. Haiminen, and H. Mannila. Randomization methods for assessing data analysis results on real-valued matrices. *Statistical Analysis and Data Mining*, 2(4):209–230, 2009.
18. J. Sanderson. Testing ecological patterns. *American Scientist*, 88:332–339, 2000.
19. P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.
20. D. A. Spielman and S.-H. Teng. Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. In *STOC*, pages 296–305, 2001.
21. D. A. Spielman and S.-H. Teng. Smoothed analysis: an attempt to explain the behavior of algorithms in practice. *Commun. ACM*, 52(10):76–84, 2009.
22. P. Tsaparas, A. Ntoulas, and E. Terzi. Selecting a comprehensive set of reviews. In *KDD*, pages 168–176, 2011.
23. N. Vuokko and P. Kaski. Testing the significance of patterns in data with cluster structure. In *ICDM*, pages 1097–1102, 2010.