Computer Science and Artificial Intelligence Laboratory
Technical Report

MIT-CSAIL-TR-2015-010 April 1, 2015

iBCM: Interactive Bayesian Case Model Empowering Humans via Intuitive Interaction

Been Kim, Elena Glassman, Brittney Johnson, and Julie Shah

iBCM: Interactive Bayesian Case Model Empowering Humans via Intuitive Interaction

Been Kim, Elena Glassman, Brittney Johnson and Julie Shah
Massachusetts Institute of Technology
Cambridge, MA
{beenkim@csail, elg@mit, bjohns@mit, julie_a_shah@csail}.edu

Abstract

Clustering methods optimize the partitioning of data points with respect to an internal metric, such as likelihood, in order to approximate the goodness of clustering. However, this internal metric does not necessarily translate into effective clustering from the user's perspective. This work presents the interactive Bayesian Case Model (iBCM), a model that opens a communication channel between the clustering model and the user. Users can provide direct input to iBCM in order to achieve effective clustering results, and iBCM optimizes the clustering by creating a balance between what the data indicate and what makes the most sense to the user. This model provides feedback for users and does not assume any prior knowledge of machine learning on their part. We provide quantitative evidence that users are able to obtain more satisfactory clustering results through iBCM than without an interactive model. We also demonstrate the use of this method in a real-world setting where computer language class teachers utilize iBCM to cluster students' coding assignments for grading.

1 Introduction

The ultimate goal of any clustering method is to partition data points in the most effective and useful way for the user. A number of existing methods [12] are able to group a large number of data points according to certain internal metrics. Unlike classification methods where prediction accuracy often serves as an evaluation metric, clustering metrics that directly optimize for a particular application are difficult to articulate and can be domain-specific. In clustering, there might be multiple ways to cluster data points that are equally good according to the internal metric of the given model, but some of these methods may better align with a user's knowledge or preference. For example, if a user in a retail setting wants to cluster customer data in order to build a recommendation engine and pair sales staff with appropriate customers using purchase records, clustering with respect to item categories or region of residence could yield equally good results. However, clustering with respect to region may offer a more cost-effective implementation of the engine by matching the recommendation plans with the existing team structure of the company; however, hard-coding such information does not translate to other domains, and also requires in-depth knowledge of machine learning.

The interactive Bayesian Case Model (iBCM) empowers users via direct, intuitive interaction with a clustering model, in order to incorporate their expert domain knowledge and preference while simultaneously optimizing the internal goodness of clustering metric. Moreover, the communication channel is bi-directional: iBCM communicates back to its users by providing explanations in natural language, offering insight into potential conflicts between user feedback and data point patterns and allowing for more fluent collaboration between the user and machine learning models.

The communication channel does not assume that a user has prior knowledge of machine learning. Instead, users provide examples (i.e., data points) and features of these examples in order to articulate their feedback. For example, when building a recommendation engine for films, users may think that the length of a specific movie is not an important feature for clustering, even if its use would yield good results. Instead, the user may think that genre or the number of awards each film has received are more important features. iBCM incorporates the user's feedback and propagates it to the inference engine in order to achieve better clustering. If incorporating user-suggested features does not yield good clustering, iBCM then provides feedback using the examples the user provided.

Desiderata

Ultimately, iBCM aims to enable efficient and intuitive use of clustering models, without requiring in-depth knowledge about machine learning. However, establishing intuitive interactions with a complex machine learning system while assuming no prior user knowledge introduces a number of challenges. Even if the interaction is simplified to the level of performance metrics (confusion matrix) [19], it still requires the user to understand the confusion matrix. Some domains, such as computer vision, offer a readily intuitive interaction medium [31]; however, this is not typically the case. iBCM combines Case-Based Reasoning (CBR) approaches [1, 29] with a Bayesian generative model to both generate effective clustering and provide an intuitive medium for interaction. Specifically, iBCM uses examples (data points provided by the user) for communication, enabling interaction between the model and any user with some degree of familiarity with the data. The use of examples as a medium for interaction is well-grounded in several cognitive science studies that demonstrated that exemplar-based reasoning, involving various forms of matching and prototyping, is fundamental to our most effective strategies for tactical decision-making ([26, 11, 21]). In order to assist user decision-making by leveraging large data sources, we desire machine learning algorithms that communicate in forms that are easily incorporated into the human decision-making process.

Once intuitive interaction has been established, the next challenge is to determine the correct way to incorporate user feedback in order to achieve a balance between the user's input and solutions that optimize the internal metrics of the clustering method. If there is a conflict between the suggestions made by the user and the output from the clustering model, there must be a way for the model to both provide feedback to the user and intuitively explain its internal states. Communication with the user is essential to a successful interactive system — it improves transparency, which has been identified as a major factor in establishing user trust in adaptive agents [28, 13] and improving system acceptance [17, 30].

Contributions

The key contribution of this paper is to introduce a unified framework for clustering, user interaction and explanation generation — a principled [22] interactive machine learning system. The principled approach is defined in [22] as follows: "The output representation is directly related to the parameters of the model without loss of information. The output that is presented to users is thus directly interpretable in terms of the generative process of the model." iBCM not only offers principled output as depicted in [22], but also principled input — users interact directly with a representation that maps to parameters of the model without loss of information. Principled interactive machine learning enables the model to incorporate user feedback and articulate its internal states — principled explanations generation — without a loss of information. While explanations using highly sophisticated language of statistics may offer in-depth understanding for expert users [24], iBCM provides intuitive and consistent explanations for non-expert users.

2 Related work

Types of prior art that are relevant to the described approach include interface designs intended to aid user workflow [27], the interactive visualization of clustering results [23, 16, 5] and interactive machine learning systems [2, 6, 4]. Interfaces that improve user workflow when modifying model settings are intended to reduce the burden of repeated processes to achieve the desired clustering results [27]. Instead of modifying model settings, some systems aim to improve the user's internalization of the current clustering through visualization [23, 16, 5].

One of the most commonly used methods of improving clustering is to explore multiple model parameter settings. To better support this iterative workflow, there are interfaces that allow for the tracking of model changes or comparing different models of machine learning [27]. There have also been works with the goal of designing a smart interface that can adjust to user behavior during highly dynamic tasks [5]. Although these systems can improve the efficiency of users' iterative workflows, indirect interaction with models is time-consuming. Also, this approach requires users to have in-depth knowledge of machine learning or only offers domain-specific solutions.

Another way to help users better understand clustering results is through visualization, rather than changing the clustering itself. Visualization techniques are often developed for specific domains (e.g., topic model visualization [10] or geographical data visualization [16]). Some also introduce new metrics to help users internalize and conceptualize clustering results [15]. Some of these combine their functionalities to change model parameters in the interface [16]. However, this approach assumes that the current clustering is somewhat aligned with effective clustering results from the user's perspective.

In order to overcome the limitations of indirect interaction with machine learning methods, there has been growing interest in the development of interactive machine learning systems [2], ranging from studying the theoretical aspect of interactive clustering [6] to building systems for use in specific domains [31, 23]. Some works simplified the medium of interaction to the level of performance metrics. For example, in classification tasks, [19] users specified their preferred confusion matrix for classifications. However, even this simplified method of interaction requires the user to understand the performance metrics and to know how to assign a numerical score between 0 and 1 to their input. For some domains, such as computer vision, the medium for intuitive interaction is naturally available — users can directly manipulate decision boundaries in pictures [31], for example. However, this is not typically true for high-dimensional complex data points. In more complex domains, such as clustering documents, interactive clustering systems are available [7, 23]; however, the medium of interaction assumes an expert level of user knowledge, such as n-grams for machine learning systems [7] or keyword weights for topic clusters [23]. Some works had users communicate their feedback through data points [4, 3], such as by providing positive and negative examples to train a classifier [3]. However, the difference between machine representation (e.g., the averaged features of data points) and representation of information obtained from human feedback (i.e., the examples) may lead to inconsistencies in how user feedback is incorporated into the machine.

iBCM's principled interactive machine learning addresses the challenges listed above by incorporating user feedback and articulating its internal states without the loss of information [22].

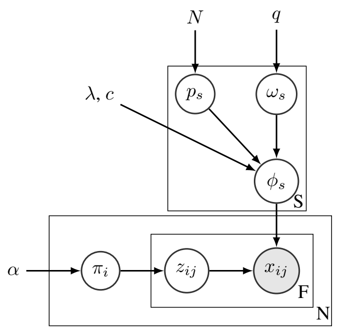
3 Interactive Bayesian Case Models (iBCM)

iBCM is an interactive version of the Bayesian Case Model (BCM), which combines case-based reasoning and a Bayesian generative model to perform accurate clustering, and has been shown to yield statistically significant improvements to objective measures of interpretability [20]. The interpretability of BCM is one of the essential ingredients for building an intuitive interactive machine learning system, especially with regard to a bi-directional communication channel between the user and the model. This section reviews BCM and presents the internal mechanisms that allow for principled interactive machine learning.

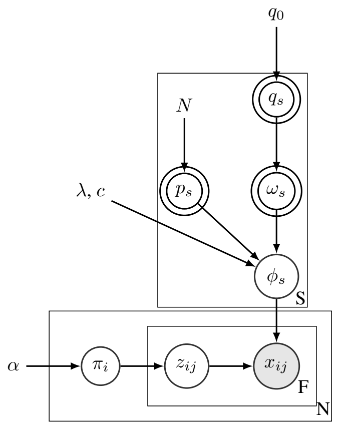
3.1 Bayesian Case Model

BCM, as introduced in [20], begins with a standard discrete mixture model [18, 9] in order to represent the underlying structure of the observations. It augments the standard mixture model with *prototypes* and *subspace feature indicators* that characterize the clusters. A graphical model of BCM is depicted in Figure 1a.

BCM begins with N observations, denoted by $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$, with each x_i represented as a random mixture over clusters. There are S clusters, where S is assumed to be known in advance. (This assumption can easily be relaxed through extension to a non-parametric mixture model, such as [8].) Vector π_i are the mixture weights over these clusters for the i^{th} observation x_i, $\pi_i \in \mathbb{R}_+^S$. Each observation has P features, and we denote the j^{th} feature of the i^{th} observation as x_{ij}. Each feature j of the observation x_i comes from one of the clusters, the index of the cluster for x_{ij} is denoted by z_{ij} and the full set of cluster assignments for observation-feature pairs is denoted by $\mathbf{z}$. Each z_{ij} takes on the value of a cluster index between



(a) Bayesian Case Model (BCM)



(b) interactive Bayesian Case Model (iBCM)

Figure 1: Graphical model depicting the BCM and iBCM. Double circle nodes represent *interacted latent variables*.

1 and S. Hyperparameters q, λ, c, and α are assumed to be fixed.

The explanatory power of BCM results from how the clusters are characterized. While a standard mixture model assumes that each cluster takes the form of a predefined parametric distribution (e.g., normal), BCM characterizes each cluster by a *prototype*, p_s, and a *subspace feature indicator*, ω_s. Intuitively, the *subspace feature indicator* selects only a few number of features that play an important role in identifying the cluster and prototype (hence, BCM clusters are *subspace clusters*). BCM intuitively defines these latent variables as follows:

Prototype, p_s: The prototype p_s for cluster s is defined as one observation in $\mathbf{x}$ that maximizes $p(p_s|\omega_s, \mathbf{z}, \mathbf{x})$, with the probability density and ω_s as defined below. The notation for element j of p_s is p_{sj}. Since p_s is a prototype, it is equal to one of the observations, such that $p_{sj} = x_{ij}$ for some i.

Subspace feature indicator ω_s: Intuitively, ω_s 'turns on' the features that are important for characterizing cluster s and selecting the prototype, p_s. Here, $\omega_s \in \{0, 1\}^P$ is an indicator variable that is 1 on the subset of features that maximizes $p(\omega_s|p_s, \mathbf{z}, \mathbf{x})$, with the probability for ω_s as defined below. Here, ω_s is a binary vector of size P, where each element is an indicator of whether or not feature j belongs to subspace s.

Kim et al. [20] show that BCM produces prediction accuracy comparable to or better than prior art for standard datasets. They also verify through human subject experiments that the prototypes and subspaces present as meaningful feedback for the characterization of important aspects of a dataset. In these experiments, the exemplar-based output of BCM resulted in statistically significant improvements to participants' performance of a task that required an understanding of clusters within a dataset, as compared to outputs produced by prior art.

3.2 Interactive Bayesian Case Model (iBCM)

The main difference between iBCM and BCM is that iBCM introduces *interacted latent variables* that represent a variable inferred through both user feedback and the data – p, ω and q (Figure 1b). Here, we introduce a new notation (double-circled nodes) for graphical models in order to represent human-*interacted latent variables*. *Interacted latent variables* are learned through user feedback and information obtained from data points. These variables differ from observed variables (shaded nodes) because they may change their value dynamically through interaction. We also distinguish them from pure latent variables (unfilled nodes) because we may want to incorporate high-confidence user feedback, even if it conflicts with inference results obtained without user interaction. Figure 1 shows the graphical models of BCM and the iBCM side-by-side. In addition to interacted latent variables, q in iBCM is a matrix, where, as in BCM, q is a scalar hyperparameter. The benefit of this augmentation is explained in Section 3.2.2.

One of the challenges of incorporating user feedback into a machine leaning system is balancing this feedback with information obtained from the data. For example, there may be a way to cluster data points that is not ideal according to the model's internal metric, but is more useful to a user. However, incorporating user feedback must be done cautiously — we must inform users when their feedback conflicts with information obtained from the data, while still maintaining a positive user experience.

This section presents two mechanisms of iBCM designed to address these challenges: confidence-based user feedback and the explanation given to users by the machine upon receiving this feedback.

3.2.1 Intuitive interaction medium

Principled interaction allows direct use of the internal representation of the model as the medium for interaction. Users interact with iBCM using a set of data points that they have provided: representative data points from each cluster (i.e., prototypes) and a set of their important features (i.e., subspaces). The only assumption this interaction makes on users is some degree of familiarity with the data points that they provide. This type of medium mimics the way that humans develop effective strategies for tactical decision-making ([26, 11, 21]).

Using data points and their features as a medium for interaction is particularly beneficial when operating within a domain that incorporates high-dimensional complex data points; for example, patent documentation, where investigating a single data point can become a daunting, time-consuming task. The iBCM interaction medium makes this simple. For example, if a user is more familiar with a particular data point or considers it to be especially important (such as a milestone patent), they can incorporate their domain knowledge by suggesting that data point as a prototype, thereby anchoring clustering with this point. The user can then suggest which features are important within that prototypical document. The ability to control prototypes and subspaces together allows for a simple yet atomic level of user control.

Communication from iBCM to users also benefits from this intuitive interaction medium. iBCM can use the same method of representation for its communication with users upon receiving feedback. (Section 3.2.3).

3.2.2 Incorporating user feedback into iBCM

The inference pipeline for incorporating user feedback involves the following three steps: 1) listening to users — updating cluster information variables ω or q according to user feedback; 2) propagating user feedback — rearranging cluster label variables z; and 3) listening to data — resampling cluster labels z and/or w.

iBCM uses a simple mechanism to appropriately incorporate user feedback: asking the user about the degree of confidence they have in their feedback. This extra information is simple enough to not burden the interaction, yet provides additional granularity for user control. Internally, the confidence-specific inference pipeline enables iBCM to appropriately adjust the impact of the feedback on the clustering results.

A. Listen to users: update interacted latent variables This section describes the different inference pipelines that iBCM applies depending on the user's confidence in their feedback. Lower-confidence feedback influences the interacted latent variable, q, which then impacts its child interacted latent variables (i.e., prototypes, p and subspaces ω) when they are resampled. Higher-confidence feedback directly influences the child interacted latent variables.

When a user provides feedback with low confidence (i.e., a user reports the he/she is 'somewhat confident' about their feedback), iBCM propagates this information to the prior level of interacted latent variable, q, as follows:

$$q_{s,f} = \begin{cases} \min(1, Hq_0) & \text{if low-confidence feedback on } \omega_{s,f} \\ q_0 & \text{otherwise,} \end{cases}$$

where H is a constant greater than 1, representing higher probability for $q_{s,f}$, and q_0 represents a constant hyperparameter (set to 0.8 in this work). Note that for BCM [20], this q variable is a scalar hyperparameter that governs the sparsity of the subspaces (e.g., the proportion of important features of a prototype). In the iBCM, q is a matrix and represents the sparsity of the subspaces of each prototype within a cluster. In the generative story, $q_{s,f}$ represents user feedback about how likely it is for feature f of cluster s to be important — and, therefore, contained within subspaces. During the inference step, variables $q_{s,f}$, once updated by user feedback, become the pseudo counts when updating the multinomial distribution, which then makes $\omega_{s,f}$ more likely to be part of subspaces.

In the case of high-confidence user feedback (i.e., a user reports that he/she is 'very confident' about their feedback), iBCM honors feedback completely by updating the ω interacted latent variable in Figure 1b as follows:

$$\omega_{s,f} = \begin{cases} \text{user specified value} & \text{if high confidence} \\ \omega_{s,f} & \text{else.} \end{cases}$$

In BCM, $\omega_{s,f}$ is 1 if feature f is an important feature for cluster s, and 0 otherwise. In iBCM, $\omega_{s,f}$ is updated to be identical to high-confidence feedback submitted by the user. Doing so updates the characteristics of cluster s, and therefore the clustering labels, z, in the inference pipeline. In combination with re-inferring the z variable (as described in Section C), this enables iBCM to learn a new clustering based on features that have been identified as important by the user.

Note that if a user is overconfident or provides careless feedback, a hard reset of the interacted latent variable ω may result in poor clustering results. In this case, iBCM provides an explanation as to why the re-inferred clustering labels, z, are inconsistent with the user's feedback (described in Section 3.2.3). This explanation is intended to offer insight for users and to improve future interaction.

B. Propagate user feedback to accelerate inference One way to propagate user feedback is to update the interacted latent variables and re-infer all other variables in the model. However, unless the inference technique guarantees a globally optimal solution, doing so only offers locally minimal solutions that may not reflect the user's feedback. In order to make sure that the new solution reflects the feedback, we require a way to quickly move the current solution to a solution space closer to the local minimum that reflects the user's feedback.

In order to move to a solution space closer to the user's feedback, iBCM propagates the feedback to the cluster labels, z. In sampling-based inference, doing so effectively moves solutions for all latent and interacted latent variables to a more favorable solution space for the user. The rationale behind this procedure is that, unlike with an inference procedure with a random start, we are not hoping for a random walk through the solution space — we know where in the solution space we want to be. Therefore, we can simulate the process of restarting Gibbs sampling multiple times with random initialization and choosing the most favorable instance with regard to the user's feedback.

To do this, iBCM updates cluster labels, z, as follows:

$$z_{i,f} = \begin{cases} s & \text{if } x_{i,f} = p_{s,f} \text{ for interacted } \omega_{s,f} \\ \text{Uniform}(1, S) & \text{otherwise,} \end{cases}$$

where $p_{s,f}$ is the value of feature f of the prototype of cluster s. For example, if user feedback indicates that feature 1 of cluster A is important (i.e., $\omega_{s,f}$ is interacted), then iBCM propagates this information by setting $z_{i,f}$ to be s for all $x_{i,f}$ that has the same value of $p_{s,f}$. All other $z_{i,f}$ are sampled uniformly over all possible cluster labels, $1, \ldots, S$.

Note that this step serves to create a balance between user feedback and patterns within the data. When re-inferring variables (as described in Section C), iBCM starts from a solution space that is closer to what the user wants, but moves toward a solution that also reflects the data patterns.

C. Listen to data: re-infer all other variables When a user provides low-confidence feedback, iBCM re-infers subspaces, ω, and clustering labels, z, by performing Gibbs sampling for only these variables. When the re-inferred interacted variable, ω, is different from what users indicated in a low-confidence case, iBCM points out where the inconsistency comes from. For example, the model might display:"You suggested that feature 1 is not important, but I think it is important. If you feel strongly about this feedback, please provide the same feedback with high confidence".

For high-confidence user feedback, iBCM only re-infers the clustering labels and completely honors the subspaces specified by users.

Note that the above approach is one of many possible ways to update interacted latent variables and re-infer other variables, and the approach can be customized. In Section 4, we show that this approach successfully incorporates user feedback.

3.2.3 Delivering feedback from the iBCM to users

The goal of providing explanations is to allow iBCM to give feedback to users when the information from patterns in the data conflicts with user-submitted feedback. Delivering these explanations opens up a communication channel between iBCM and the user, allowing for more fluent collaboration.

iBCM generates explanations using information from two sources: 1) cluster labels, z, and 2) the likelihood of prototypes and subspaces. Both types of explanation are triggered by the same condition: inconsistency between user feedback, the resulting subspaces and cluster memberships.

Explanations using cluster labels One way to generate explanations is to base them on the cluster assignments, z, where $z_{i,f}$ represents the cluster assignment of feature f of i-th data point. Note that in a mixture model, one data point may have multiple cluster assignments for each feature. This distribution of assignments is represented by π_i, a vector of length S.

Explanations are generated by analyzing data points with the same maximum element of π_i, as well as commonly shared or not shared features among those points. For example, if all assessed data points have the same value for feature B, but different values for feature A, then the explanation generated could read as follows: "101 items in group 1 have different feature A values. However, most of the items in this group have the same feature B value". This type of explanation provides common characteristics of each cluster using data points within that cluster in a "feed-forward" fashion, and represents how user input has propagated to clustering labels. However, this method does not directly offer "feedback" from iBCM — representations of internal states of the model indicating why certain data points are assigned to certain clusters.

Explanations using examples and important features In order to provide deeper insight into the internal state of iBCM, explanations can also be provided using interacted latent variables (i.e., prototypes and subspaces). Utilizing variables from the model to generate explanations offers more sophisticated information about the internal states of the model and the clustering results. These interacted variables have been shown to be effective when communicating clustering results to humans [20]. To generate explanations in this fashion, likelihoods are used as "scores" to indicate how likely each feature is to belong to the subspace. For example: "In group 1, important features are currently marked as feature A and feature B. However, feature C seems to be more important in this group". Likelihoods are defined as follows (directly from iBCM):

$$p(\omega_{sj} = b | q_{sj}, p_{sj}, \lambda, \phi, \mathbf{x}, \mathbf{z}, \alpha) \propto \begin{cases} q_{sj} \times \dfrac{\mathbf{B}(g(p_{sj}, 1, \lambda) + n_{(s,\cdot,j,\cdot)})}{\mathbf{B}(g(p_{sj}, 1, \lambda))} & \text{b = 1} \\ 1 - q_{sj} \times \dfrac{\mathbf{B}(g(p_{sj}, 0, \lambda) + n_{(s,\cdot,j,\cdot)})}{\mathbf{B}(g(p_{sj}, 0, \lambda))} & \text{b = 0,} \end{cases}$$

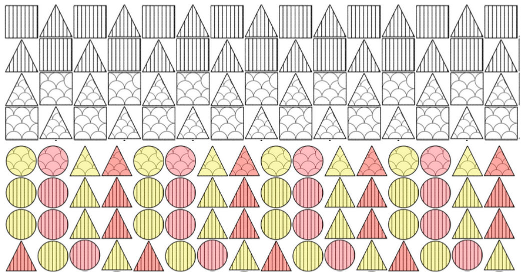


Figure 2: A subset of data points. Each data point has two features (top) or three features (bottom): shape, color and pattern.

where $g_{p_{sj},\omega_{sj},\lambda}(v) = \lambda(1 + c\mathbb{1}_{[w_{sj}=1 \text{ and } p_{sj}=\Theta_v]})$, Θ_v is a particular feature value and $\mathbf{B}$ is the Beta function. $n_{(s,\cdot,j,v)}$ is the number of times that the j^{th} feature of an observation takes feature value v and that observation is assigned to subspace cluster s (i.e., $n_{(s,\cdot,j,v)} = \sum_i \mathbb{1}(z_{ij} = s, x_{ij} = v)$). Hyperparameter λ, c and α are assumed to be fixed. These definitions follow BCM [20].

Note that providing this type of explanation is only possible with principled interactive machine learning, where internal clustering, user interaction and explanations are all done in the same representation without loss of information [22]. This enables iBCM to articulate its internal states to users in the same intuitive and consistent way.

4 Evaluation

We performed a human experiment, along with a proof-of-concept study, to validate our approach. The experiment was designed to measure quantifiable improvements in how well final clustering performed with iBCM matches the intent of the subject compared with clustering with BCM. The proof-of-concept study was a demonstration of iBCM in a real-world setting (computer science education).

4.1 Human subject experiment

The goal of this experiment is to obtain objective measures of iBCM's performance. We designed an experiment such that the quality of clustering obtained through interaction can be evaluated while controlling for data characteristics that can affect clustering results. In order to do this, we constructed datasets that could be clustered in multiple ways, all of which were equal in terms of likelihoods, with the optimal ways to cluster according to internal metrics also known a priori.

There were eight conditions in the experiment (shown in Table 1), each of which had three variables: the number of features, number of clusters and balance. The number of features reflects the complexity of the domain. The data points in the first four conditions had two features (shape

Question	1	2	3	4	5	6	7	8
# of features	2	2	2	2	3	3	3	3
is balanced	T	F	T	F	T	F	T	F
# of clusters	2	2	3	3	4	4	5	5

Table 1: Experiment design. Participants were randomly assigned to one of four groups, with four participants per group. "Is balanced" was identified as true (T) when there were an equal number of data points with each feature, and false (F) when there were more data points with a particular feature value than others (e.g., more red squares then yellow squares).

and pattern, shown on the top of Figure 2), while points in the remaining four conditions each had three features (shape, pattern and color, depicted on the bottom of Figure 2). Each feature had two possible values; for example, the color could be either red or yellow. The number of clusters reflects the complexity of interaction between iBCM and the user. When the number of clusters is even, clustering can be performed such that the number of data points falling into each cluster is exactly the same. If the number of clusters is odd, some clusters must contain fewer data points than others. The phrase "is balanced" was defined as the presence of an equal number of data points with the same feature values. For example, if there are an equal number of red circles, yellow circles, red triangles and yellow triangles, then "is balanced" is true. This factor was intended to test how subjects' interaction with iBCM is influenced by the complexity of a dataset. Note that a balanced dataset can have a different number of data points that fall into an individual cluster due to the number of clusters.

The experiment was designed using a Latin square to assign the conditions to the 24 participants in a balanced manner. Each subject answered nine questions, with the first considered a practice question that was not included in the analysis.

The subjects performed the following steps for each question: First, subjects were asked how they wanted to cluster the data; for example, according to shape or color. Randomly ordered data points were shown to the subjects at this point. This step collected the ground truth for subjects' intent, which would later be compared with the clustering results following interaction with iBCM. We then showed the clustering results of BCM, which essentially selects one of the optimal ways to cluster the data points, as the dataset was designed to have multiple, equally likely optimal solutions. At the third step, subjects rated how well the clustering they had been shown matched their preferred clustering on a five-point Likert scale, with 1 indicating they strongly disagreed and 5 that they strongly agreed that the two clustering attempts matched. Next, subjects interacted with iBCM to provide their feedback and customize the

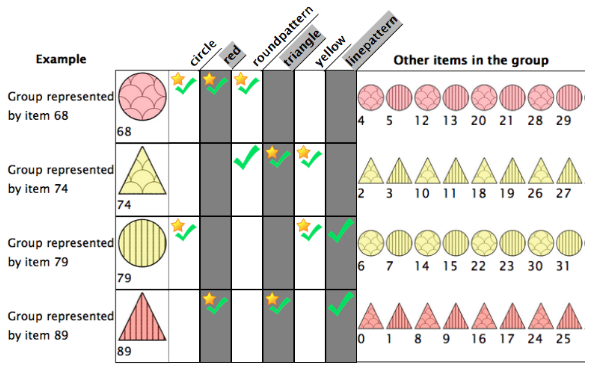


Figure 3: Graphical user interface for interaction. Each row represents a cluster. Prototypes of each cluster are shown on the left. The numbers below them are data ID. Subspaces are marked as stars. Each data point has three features: shape, color and pattern.

clustering. Finally, when subjects indicated that they were done, they were asked to rate how well the final clustering matched with their preferred clustering on the same five-point Likert scale. All participants took mandatory breaks after every other question in order to manage fatigue. The goal of iBCM is to achieve stronger subject agreement that their clustering results matched their preference following interaction with the model. Note that according to the model's internal metric, the quality of clustering at step 2 is likely to be equal to that of the final clustering.

Figure 3 shows the interface used for this experiment. In this interface, each row represents a cluster, where the example that best represents the cluster (i.e., prototype) is shown on the left of each row. The name of each data point is simply a number depicted below each point. Features (denoted with a check mark) and important features (denoted with a check mark and star) are depicted to the right of each prototype. For example, on the second row, the data number 74 has the 'round pattern', 'triangle' and 'yellow' features, but the important features for characterizing this cluster are 'yellow' and 'triangle.'

Subjects were able to provide feedback using this interface in two ways: by clicking a check mark to designate a feature as important (when clicked, a check mark changed to a check mark with a star) or unimportant, and by clicking one of the items marked as "other items in the group" in order to promote it to become a prototype of a cluster. When an item was clicked, subjects were prompted with options to designate it as a prototype of any of the clusters.

When subjects provided either type of feedback, the interface asked them to indicate their degree of confidence in that feedback (i.e., 'somewhat confident' or 'very confident'). All feedback was cached in iBCM, which incorporated and recomputed clustering data points only when subjects clicked the 'rerun' button. Subjects were allowed to provide as much feedback as they wanted before hitting 'rerun.' They were able to click the 'rerun' button a maximum of 50 times; however, all subjects were satisfied with their clustering results before reaching this number of interactions.

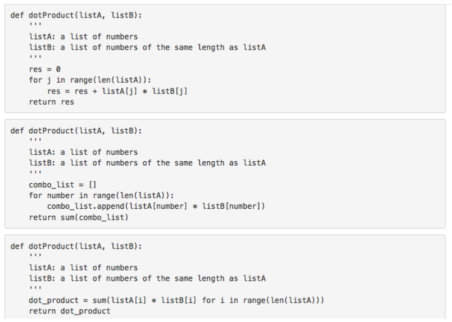
Question 1	$z = 6.10, p < .001$
Question 2	$z = 6.09, p < .001$
Question 3	$z = 6.13, p < .001$
Question 4	$z = 6.08, p < .001$
Question 5	$z = 6.09, p < .001$
Question 6	$z = 6.08, p < .001$
Question 7	$z = 6.11, p < .001$
Question 8	$z = 6.11, p < .001$

Table 2: Human subject experiment results

For all conditions, regardless of the number of features, number of clusters or whether the dataset was identified as balanced or not, iBCM achieved significantly better agreement with the users' intentions, as shown in Table 2. The complexity of the domain, the interaction with iBCM or of the dataset did not influence the effectiveness of the model. We used the two-sided Wilcoxon signed rank test for paired observations to assess the statistical significance of the observed differences in agreement before and after interaction with iBCM. Unlike a t-test, the Wilcoxon signed rank test does not assume normal data distribution. This experiment validates that the subjects achieved significantly more satisfying results using iBCM compared with BCM, even though the results from BCM provided equally good solutions according to the model's internal metric. In the next section, we present a real-world implementation of a system utilizing iBCM for further validation.

4.2 A demonstration of the iBCM system for online education

Here, we present a proof-of-concept implementation of iBCM for a real-world application: introductory programming education. Our institute, for example, offers several courses that teach basic programming to hundreds of students simultaneously. It can take hours for teachers to grade exams for courses of this size, and many teachers generate grading rubrics for each problem based on a small random sample of student solutions. Rather than depend on such a sample, we implemented a system for introductory Python programming teachers that uses iBCM as the main computational engine for grouping. The system utilizes iBCM to help teachers group their students' assignments for grading and provide helpful comments for the students. We invited teachers of introductory Python computer language classes (i.e., domain experts) to interact with the system to cluster students' assignment submissions, to show that using iBCM allows the users to easily explore variations across hundreds of student solutions before designing a grading rubric or composing feedback for students.

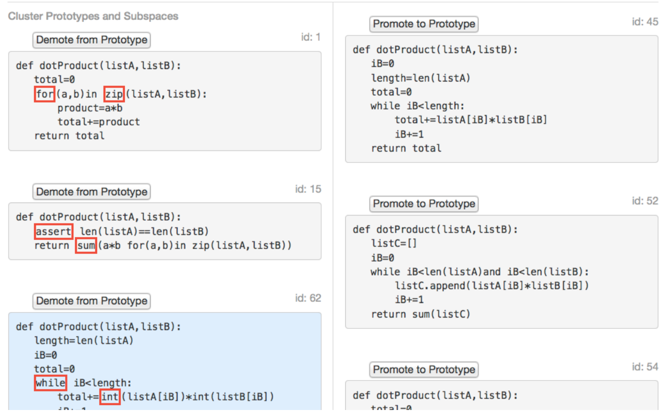


```
def dotProduct(listA, listB):
    '''
    listA: a list of numbers
    listB: a list of numbers of the same length as listA
    '''
    res = 0
    for j in range(len(listA)):
        res = res + listA[j] * listB[j]
    return res
```

```
def dotProduct(listA, listB):
    '''
    listA: a list of numbers
    listB: a list of numbers of the same length as listA
    '''
    combo_list = []
    for number in range(len(listA)):
        combo_list.append(listA[number] * listB[number])
    return sum(combo_list)
```

```
def dotProduct(listA, listB):
    '''
    listA: a list of numbers
    listB: a list of numbers of the same length as listA
    '''
    dot_product = sum(listA[i] * listB[i] for i in range(len(listA)))
    return dot_product
```

(a) The raw data



```
def dotProduct(listA,listB):
   total=0
   for(a,b)in zip(listA,listB):
       product=a*b
       total+=product
   return total
```

```
def dotProduct(listA,listB):
   iB=0
   length=len(listA)
   total=0
   while iB<length:
       total+=listA[iB]*listB[iB]
       iB+=1
   return total
```

```
def dotProduct(listA,listB):
   assert len(listA)==len(listB)
   return sum(a*b for(a,b)in zip(listA,listB))
```

```
def dotProduct(listA,listB):
   listC=[]
   iB=0
   while iB<len(listA)and iB<len(listB):
       listC.append(listA[iB]*listB[iB])
       iB+=1
   return sum(listC)
```

```
def dotProduct(listA,listB):
   length=len(listA)
   iB=0
   total=0
   while iB<length:
       total+=int(listA[iB])*int(listB[iB])
```

```
def dotProduct(listA,listB):
```

(b) iBCM with features extracted from using OverCode

Figure 4: Interface combining iBCM and OverCode to help teachers grade and provide useful feedback for students

Education is one of the many domains for which iBCM could be useful — domains where clustering results must reflect both patterns within data and the knowledge of domain experts. These experts — in this case, teachers — have accumulated years of knowledge that the clustering algorithm can leverage to make the clustering results more effective. For example, each teacher may have a different style of grading and providing feedback — some may focus on the key concepts presented in the class, while others may focus on students' coding practices, such as the use of good variable names and comments, as much as understanding the key concepts. Systems that can simplify the grouping of assignments are particularly useful for massive open online courses (MOOCs). Reviewing thousands of students' assignments is a very time-consuming task, especially when also trying to provide constructive feedback on an individual level. iBCM allows teachers to incorporate their domain knowledge in order to achieve the grouping of assignments that is most effective for their task.

However, building a complete system for use in engineering education introduces an additional challenge into the interactive machine learning system: processing raw data. If keywords within the raw code data are simply used as features, they may lose important functional similarities and differences. For example, a small feature difference in keywords may not change the functionality of a statement, as would be the case with the use of different variable names. However, another small difference in features, such as using `for` versus `while` in Python, may result in potentially large differences in functionality depending on the context. Processing raw data requires an understanding of the structure of the code, as well as language-specific knowledge.

We use a method called OverCode [14] to extract relevant features for iBCM. OverCode uses both static and dynamic analysis to combine similar solutions that perform the same computations, but may use different variable names or statement order. OverCode was developed to help MOOC instructors explore variations among student solutions and provide appropriate feedback to a large number of students. It has been shown to allow teachers to more quickly develop a high-level view of students' understanding and misconceptions, as well as provide feedback that is relevant for more students [14]. Note that the output codes from OverCode are both human-readable and executable (snippets are shown in Figure 4b). OverCode renames variables to reflect their behavior. It find common variables that behave the same way in many solutions run on the same test case, and then renames those common variables to most popular name. Inputs to iBCM are binary vectors indicating the existence of the keywords, including renamed variables and language-specific keywords, such as `listA`, `assert`, `while`.

Figure 4b depicts the interface that subjects used to interact with iBCM. On the left, it shows prototypical examples (one of the students' submissions) for each cluster. The parts surrounded by red rectangles indicate subspaces (important features). When a prototype is selected — highlighted in blue, as in the last row in Figure4 — assignments that fall into that cluster are depicted on the right.

We conducted a pilot study to gather more qualitative feedback about iBCM. In this study, subjects were given three conditions: raw code submissions (static interface, as shown in Figure 4a), submissions processed using OverCode (static interface) and OverCode utilizing iBCM (interactive interface, shown in Figure 4b). The different conditions served as reference points for the subjects to give more qualitative feedback about iBCM. Each subject received four different domains (problem sets), with the first domain considered just for practice and not included in analysis. Three introductory Python classes teachers were invited to participate and functioned as domain experts. The order in which conditions were shown to a participant was randomly selected. The teachers were asked to create a grading rubric and provide helpful comments for the students.

The strength of iBCM particularly stood out in providing a better exploration tool for the teachers to discover submissions with different approaches to solve assignments. In a post-experiment questionnaire, subjects reported that they were able to provide feedback for a larger portion of students' submissions using iBCM than with the other two conditions. One participant noted that, "Prototype 1 is a good example. Now I found someone who uses [some other feature]...". Subjects also said that using iBCM provided a good starting point and that the ability to interact with the model allowed them to modify a cluster as they chose. "This is cool," one commenter said. "A lot of these people are consistent with the prototype. I think this is actually a pretty good prototype, so I'll keep it." Another noted that, "Given three prototypes to start off with gave me a starting point, instead of endless stream. And if I didn't like one of the prototypes I can go search for another one". The usefulness of examples rather than keywords as a communication tool was also noted by some subjects. One participant said that the keyword given in the context of the code enabled her to process what students were trying to do, which in turn helped her to select a keyword from that code. Another commented, "I enjoyed being able to rapidly pick keywords and update clusters to see how common they were". One participant who observed iBCM before the other two conditions said that "In other conditions [raw data and OverCode], I did find myself sometimes wishing I could click on a keyword to see how common it was, after being able to do that in condition 1 [iBCM]".

Over the course of our experiment, we also encountered challenges that should be addressed when making a system for use in this complex domain. First, there is a need to communicate the probabilistic nature of the clustering method; even a single outlier within the cluster caused confusion for subjects. This could be addressed by establishing the appropriate expectations about the benefits of the randomness of the tool for exploration — in other words, introducing the model as a tool for 'discovery' rather than 'organization'. Subjects also asked for richer features: Instead of keywords, they wanted to interact with higher-level features, such as the structure of the code. Extracting such features is an area of active research [25], and doing so would increase the productivity of the tool.

5 Conclusion

In this work, we introduced an interactive clustering model that can provide effective results for the user. iBCM communicates with users to incorporate their feedback into the clustering model and share its internal states. We showed that users were statistically significantly more in agreement that the final clustering matches with their preferred way to cluster data points when using iBCM during human experiments. We also demonstrated iBCM's potential real-world use by implementing a system to help computer science teachers to grade and provide feedback to their students.

References

[1] A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 1994.

[2] S. Amershi, M. Cakmak, B. Knox, and T. Kulesza. Power to the people: The role of humans in interactive machine learning. *AI Magazine*.

[3] S. Amershi, J. Fogarty, A. Kapoor, and D.S. Tan. Effective end-user interaction with machine learning. In *AAAI*, 2011.

[4] S. Amershi, J. Fogarty, and D. Weld. Regroup: Interactive machine learning for on-demand group creation in social networks. In *SIGCHI*. ACM, 2012.

[5] S. Amershi, B. Lee, A. Kapoor, R. Mahajan, and B. Christian. CueT: human-guided fast and accurate network alarm triage. In *SIGCHI*. ACM, 2011.

[6] M.F. Balcan and V. Blum. Clustering with interactive feedback. In *Algorithmic Learning Theory*. Springer, 2008.

[7] R. Bekkerman, H. Raghavan, J. Allan, and K. Eguchi. Interactive clustering of text collections according to a user-specified criterion.

[8] D.M. Blei, T.L. Griffiths, and M.I. Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *JACM*, 2010.

[9] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent Dirichlet allocation. *JMLR*, 2003.

[10] A.J. Chaney and D.M. Blei. Visualizing topic models. In *ICWSM*.

[11] M.S. Cohen, J.T. Freeman, and S. Wolf. Metarecognition in time-stressed decision making: Recognizing, critiquing, and correcting. *Human Factors*, 1996.

[12] C. Fraley and A.E. Raftery. How many clusters? which clustering method? answers via model-based cluster analysis. *The Computer Journal*, 1998.

[13] A. Glass, D.L. McGuinness, and M. Wolverton. Toward establishing trust in adaptive agents. In *IUI*. ACM, 2008.

[14] E.L. Glassman, J. Scott, R. Singh, and R.C. Miller. OverCode: visualizing variation in student solutions to programming problems at scale.

[15] J. Grimmer and G. King. General purpose computer-assisted clustering and conceptualization. *PNAS*, 2011.

[16] D. Guo, D.J. Peuquet, and M. Gahegan. ICEAGE: Interactive clustering and exploration of large and high-dimensional geodata. *GeoInformatica*, 2003.

[17] J.L. Herlocker, J.A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *CSCW*, 2000.

[18] T. Hofmann. Probabilistic latent semantic indexing. In *ACM SIGIR*, 1999.

[19] A. Kapoor, B. Lee, D. Tan, and E. Horvitz. Interactive optimization for steering machine classification. In *Human Factors in Computing Systems*. ACM, 2010.

[20] B. Kim, C. Rudin, and J.A. Shah. The Bayesian Case Model: A generative approach for case-based reasoning and prototype classification. In *NIPS*, 2014.

[21] G.A. Klein. Do decision biases explain too much. *HFES*, 1989.

[22] P. Krafft, J. Moore, B. Desmarais, and H.M. Wallach. Topic-partitioned multinetwork embeddings. In *NIPS*, 2012.

[23] H. Lee, J. Kihm, J. Choo, J. Stasko, and H. Park. iVisClustering: An interactive visual document clustering via topic modeling. In *Computer Graphics Forum*. Wiley Online Library, 2012.

[24] J.R. Lloyd, D. Duvenaud, R. Grosse, J.B. Tenenbaum, and Z. Ghahramani. Automatic construction and natural-language description of nonparametric regression models. In *AAAI*, 2014.

[25] G.C. Murphy and D. Notkin. Lightweight lexical source model extraction. *TOSEM*, 1996.

[26] A. Newell and H.A. Simon. *Human problem solving*. Prentice-Hall Englewood Cliffs, 1972.

[27] K. Patel, N. Bancroft, S.M. Drucker, J. Fogarty, A.J. Ko, and J. Landay. Gestalt: integrated support for implementation and analysis in machine learning. In *UIST*. ACM, 2010.

[28] P. Pu and L. Chen. Trust building with explanation interfaces. In *IUI*. ACM Press, 2006.

[29] S. Slade. Case-based reasoning: A research paradigm. *AI magazine*, 1991.

[30] N. Tintarev and J. Masthoff. Designing and evaluating explanations for recommender systems. In *Recommender Systems Handbook*. Springer, 2011.

[31] M. Ware, E. Frank, G. Holmes, M. Hall, and I.H. Witten. Interactive machine learning: letting users build classifiers. *International Journal of Human-Computer Studies*, 2001.