Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

PIVE: Per-Iteration Visualization Environment for Real-Time Interactions with Dimension Reduction and Clustering*

Hannah Kim,[1] **Jaegul Choo,**[2] **Changhyun Lee,**[3] **Hanseung Lee,**[3] **Chandan K. Reddy,**[4] **Haesun Park,**[1]
[1]Georgia Institute of Technology, Atlanta, GA, USA; hannahkim@gatech.edu, hpark@cc.gatech.edu
[2]Korea University, Seoul, South Korea; jchoo@korea.ac.kr
[3]Google Inc., Mountain View, CA, USA; khakice@gmail.com, hanseung@google.com
[4]Virginia Tech, Arlington, VA, USA; reddy@cs.vt.edu

Abstract

One of the key advantages of visual analytics is its capability to leverage both humans's visual perception and the power of computing. A big obstacle in integrating machine learning with visual analytics is its high computing cost. To tackle this problem, this paper presents PIVE (Per-Iteration Visualization Environment) that supports real-time interactive visualization with machine learning. By immediately visualizing the intermediate results from algorithm iterations, PIVE enables users to quickly grasp insights and interact with the intermediate output, which then affects subsequent algorithm iterations. In addition, we propose a widely-applicable interaction methodology that allows efficient incorporation of user feedback into virtually any iterative computational method without introducing additional computational cost. We demonstrate the application of PIVE for various dimension reduction algorithms such as multidimensional scaling and t-SNE and clustering and topic modeling algorithms such as k-means and latent Dirichlet allocation.

1 Introduction

The innate ability of humans to quickly acquire insights through visualization has been a key factor in the growth in visual analytics (Keim 2002; Thomas and Cook 2005). To leverage humans' visual perception in data analytics, an increasing amount of effort has been made to utilize various computational methods in visual analytics (Buja, Cook, and Swayne 1996; Seo and Shneiderman 2002). However, the significant amount of computing time required to run these methods has been a critical hurdle against the effective integration of machine learning in visual analytics. Even worse, as machine learning becomes more advanced and capable, they often require more computations, making it virtually impossible to perform real-time interactive visualizations with them. Therefore, even the state-of-the-art in visual analytics adopts only a few standard techniques and does not properly leverage the advantages of advanced machine learning methods.

However, several important aspects have been largely overlooked in previous studies. Specifically, this paper focuses on the following aspects: (1) humans' perceptual precision and (2) the iterative behavior of machine learning. First, we notice that *visual perception does not require highly precise outputs from machine learning methods*. For example, when perceiving the value of π, most people think of its approximate value, e.g., 3.14, and knowing it more accurately, e.g., 3.1415926, does not make much difference in practice. Second, modern machine learning methods usually obtain the solution via iterative processes. Their important characteristic is that *a major improvement of the solution typically occurs in early iterations* while only minor changes occur in the later iterations. It indicates that the approximate, low-precision outputs can be obtained much earlier before the full iterations finish. Motivated by these two crucial observations, we postulate that, in visual analytics, there is no need for users to wait utill the algorithms are completely finished and get the final precise result.

In response, we propose a novel approach called PIVE (**P**er-**I**teration **V**isualization **E**nvironment), which visualizes the intermediate results from algorithm iterations as soon as they become available, achieving an efficient real-time interactive visualization with machine learning. Unlike many previous approaches that treat a machine learning method as a black box, PIVE breaks it down to an iteration level and tightly integrates them with visual analytics so that a user can check and interact with the visualization of machine learning outputs. To avoid any delays in this process, PIVE parallelizes computation and visualization via multi-threading.

With PIVE, a user can efficiently perform multiple interactions with machine learning in real time since it drastically reduces the turn-around time of a single interaction from full iterations to a few. The main idea of our interaction methodology is to allow a user to interact with the intermediate output, which then affects subsequent algorithm iterations. Since such a methodology does not require any major algorithmic modifications nor computational overhead, a user can efficiently perform multiple interactions with machine learning in real time.

2 Related Work

Efficient Interactive Visualization Numerous studies focused on the efficient interactive visualization of large-scale data. A straightforward approach is to use sampled data (Fisher et al. 2012; Ellis and Dix 2006). Another type of

*This work extends a poster at IEEE VIS '14 (Choo et al. 2014), which won the VAST best poster award.

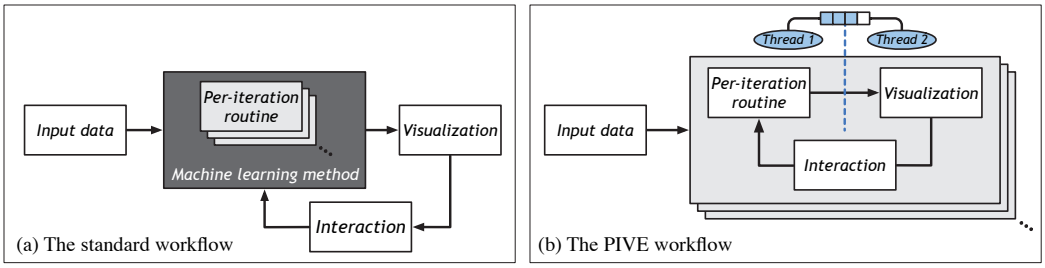


Figure 1: Comparison of the standard and PIVE workflows. In (a), a machine learning method is treated as a black box, which gives the output only after its iterations finish. In contrast, PIVE (b) splits a machine learning method into iterations, visualizing and interacting with intermediate results during iterations.

popular approaches relied upon multi-threading techniques to separate data processing and computation from visualization and rendering (Ma 2009; Yu et al. 2010; Tu et al. 2006; Piringer et al. 2009). However, none of these approaches have exploited the nature of the iterative processes in most machine learning methods, which makes a clear distinction of PIVE.

User Interaction with Machine Learning There have been significant efforts to provide a general framework to visualize machine learning results (Johnston 2001; Thearling et al. 2001) and to improve the interactivity with them (Mulder, van Wijk, and van Liere 1999). Several studies added interaction capabilities to dimension reduction (Williams and Munzner 2004; Brown et al. 2012; Kim et al. 2016; Kwon et al. 2017), clustering (Seo and Shneiderman 2002; Lee et al. 2012; Schreck et al. 2009), classification (Bosch et al. 2013; van den Elzen and van Wijk 2011), and topic modeling (Choo et al. 2013a; Kuang, Choo, and Park 2015). However, most of them do not efficiently support their interactions at an iteration level. In this sense, PIVE, which leverages the iteration-wise behavior of machine learning, potentially bears a great impact in achieving this goal.

Progressive Visual Analytics Similar to our work, the concept of progressive visual analytics, which generates meaningful partial results of an algorithm and interacts with it, was recently introduced (Stolper, Perer, and Gotz 2014; Mülbacher et al. 2014). PIVE realizes the idea to tightly integrate machine learning with visual analytics at an iteration level by customizing various well-known machine learning methods in established visual analytics systems.

3 Per-Iteration Visualization Environment

In this work, we focus on iterative methods, which refine approximate solutions over iterations. As summarized in Algorithm 1, given a set of data items X and parameter vector α, at t-th iteration, iterative methods refine previous solution Y^{t-1} into Y^t. The iterations continue until a stopping criterion is satisfied, say, at $t = T$. We note that the intermediate output Y^t has the same form throughout the iterations as the final output Y^T, and PIVE directly utilizes an intermediate output Y^t for real-time interactive visualization.

Algorithm 1 Iterative methods

Input: $X = \{x_1, \cdots, x_n\}$ and parameter α
Output: $Y = \{y_1, \cdots, y_n\}$
$t \leftarrow 0$
Initialize $Y^t = \{y_1^t, \cdots, y_n^t\}$
repeat
  $t \leftarrow t + 1$
  /* Per-iteration routine */
  for $i \leftarrow 1, \cdots, n$ **do**
    $y_i^t \leftarrow f(\{X, Y^0, \cdots, Y^{t-1}\}, \alpha)$
  $Y^t \leftarrow \{y_1^t, \cdots, y_n^t\}$
until a stopping criterion is satisfied
$T \leftarrow t$ /* Final iteration index */
$Y \leftarrow Y^T$ /* Final output */

3.1 Overall Workflow

In Fig. 1(a), a machine learning method is run, and it gives outputs to a visualization module only after finishing its iterations. Subsequently, each user interaction goes through another entire set of iterations. On the contrary, PIVE immediately delivers intermediate results from algorithm iterations to the visualization module, as shown in Fig. 1(b). Accordingly, users can immediately initiate their interactions with machine learning, which then affects its subsequent iterations. In this manner, user interactions are performed at an iteration level, which is the key to support real-time user interactions.

3.2 Interaction Methodology

The basic types of interactions with machine learning includes changing its parameters and selecting/removing data subsets of interest. PIVE allows users to immediately check the effect of parameter changes in real time.

In addition, we consider more sophisticated interactions that allow users to manipulate the intermediate output by replacing part of them with user-specified values. Depending on whether a user wants to fix these new values over the subsequent iterations, we categorize our interaction methodology into *soft* and *hard replacements*.

As summarized in Algorithm 2, the main difference between the soft and hard replacements is that the latter skips the updating step for user-selected data items and uses the replaced outputs throughout the iterations. Unlike the filtering

Algorithm 2 Soft and hard replacement interactions

1: Input: an iteration index t at which a user interaction was performed, interacted data item indices $I = \{i_1, \cdots, i_l\}$, their new values $\{\tilde{y}_{i_1}, \cdots, \tilde{y}_{i_l}\}$, and a new parameter $\tilde{\alpha}$
2: $\alpha \leftarrow \tilde{\alpha}$
3: **for** $i \leftarrow i_1, \cdots, i_l$ **do**
4: $\quad y_i^t \leftarrow \tilde{y}_i$
5: **if** Hard replacement **then**
6: $\quad$ Replace the per-iteration routine in Algorithm 1 as follows:
7: $\quad$ /* Per-iteration routine*/
8: $\quad$ **for** $i \leftarrow 1, \cdots, n$ **do**
9: $\quad\quad$ **if** $i \notin I$ **then**
10: $\quad\quad\quad y_i^t \leftarrow f(\{X, Y^0, \cdots, Y^{t-1}\}, \alpha)$
11: $\quad\quad$ **else**
12: $\quad\quad\quad y_i^t \leftarrow y_i^{t-1}$
13: Continue from the iteration index $t+1$ until a stopping criterion is satisfied in Algorithm 1

interaction, these replaced and fixed outputs stillaffect the updated outputs for the rest of the data items. In this sense, the hard replacement interaction converts the original machine learning method into a constrained or semi-supervised method. In addition, this interaction has an advantage of saving computational time in the subsequent iterations by skipping the updating steps for the fixed data items. This can be useful when a user wants to focus the computational resource on the remaining data items.

On the other hand, a soft replacement interaction replaces the outputs of selected data items at a particular iteration, but the later iterations continue updating them just as they update the rest of the data items. In this respect, soft replacement interactions can be viewed as user-driven re-initialization of the algorithm. This can be useful in finding a better local optimum for non-convex problems or finding a good initialization in a user's own manner.

3.3 Further Considerations

Stability and Convergence[1] PIVE poses a challenge as the difficulty in deciding when to start interactions with machine learning. First, a visual stability issue exists. Because PIVE continuously updates the visualization, if intermediate outputs change significantly and frequently, the corresponding visualizations become inconsistent, thus preventing a user from analyzing and interacting with them. The second issue is whether intermediate outputs from machine learning are close enough to the final solution for users to start analyzing and interacting with them. To help determine whether the intermediate result is sufficiently stable and close to the final solution, we provide users with separate charts showing the corresponding measures as well as visual encoding within existing visualizations.

[1]We define 'stability' and 'convergence' as visual stability and visual convergence throughout this paper.

Computational Overhead Since PIVE has to repetitively process intermediate outputs, additional computations are incurred. We use a multi-threading approach to handle them. We separate the entire process into two concurrent processes/threads (blue ellipses in Fig. 1(b)). The computational thread deals with the computations within algorithm iterations while the visualization thread works on post-processing and rendering. These two threads communicate via a message queue (Fig. 1(b)). Since modern commodity computers are usually equipped with a multi-core CPU, these two threads can be executed in parallel without much performance loss compared to the standard approach.

4 Applications to Machine Learning

In this section, we present the applications of PIVE to several dimension reduction and clustering methods. For demonstration, we altered existing visual analytics systems.

For the two dimension reduction methods, we have improved FodavaTestbed visual analytics system (Choo et al. 2013b),[2] which supports various dimension reduction methods in high-dimensional data analysis. For k-means clustering, we have customized a well-known visual analytics system for document analysis, Jigsaw (Stasko, Görg, and Liu 2008).[3] Finally, for latent Dirichlet allocation, we have modified an interactive document clustering system called iVisClustering (Lee et al. 2012).

4.1 Dimension Reduction

Given n data items, $X = \{x_1, \cdots, x_n\} \in \mathbb{R}^{m \times n}$, dimension reduction generates their 2D coordinates, $Y = \{y_1, \cdots, y_n\} \in \mathbb{R}^{2 \times n}$ that will be used in a scatter plot.

Multidimensional Scaling (MDS) MDS (Cox and Cox 2000) attempts to preserve the distances/relationships of data items in a lower-dimensional space. MDS solves

$$\min_{y_1, \ldots, y_n} \sum_{1 \le i \le n} \sum_{1 \le j \le n} \left(d_{ij}^x - d_{ij}^y\right)^2, \tag{1}$$

where d_{ij}^x and d_{ij}^y are the given pairwise distances between the i-th and j-th data items in the original m-dimensional and the reduced 2-dimensional spaces, respectively.

t-Distributed Stochastic Neighbor Embedding (t-SNE) t-SNE (van der Maaten and Hinton 2008) tries to minimize the difference between pairwise probability distribution P^x over X and P^yover Y by solving

$$\min_{y_1, \ldots, y_n} KL(P^x \| P^y) = \min_{y_1, \ldots, y_n} \sum_{1 \le i \le n} \sum_{1 \le j \le n} p_{ij}^x \log \frac{p_{ij}^x}{p_{ij}^y}, \tag{2}$$

where $KL(P^x \| P^y)$ is the Kullback-Leibler (KL) divergence between P^x and P^y.

[2]http://fodava.gatech.edu/fodava-testbed-software

[3]http://www.cc.gatech.edu/gvu/ii/jigsaw/

User Interaction Capabilities Typically, the dimension reduction outputs are visualized in a scatter plot. Other than basic interactions such as changing parameters and selecting/filtering data items, a natural user interaction is to move data points on a scatter plot. We achieve this "*point-moving*" interaction by utilizing the soft and hard replacement interactions described in Algorithm 2. That is, once a user selects and moves l points to new positions in a scatter plot, their current intermediate output $\left\{y_{i_1}^t, \cdots, y_{i_l}^t\right\}$ gets updated as their new positions$\{\tilde{y}_{i_1}, \cdots, \tilde{y}_{i_l}\}$ (line 4 in Algorithm 2). As discussed in Section 3.2, the soft replacement interaction can be thought of as restarting the dimension reduction method with new initial points. On the other hand, the hard replacement interaction skips the updating step for the user-selected data items, while their fixed coordinates still affect the rest of the data items in later iterations. For instance, those data items with close relationships to the fixed data items may be pulled towards them while those with remote relationships may be pushed away from them. These interactions can reveal interesting knowledge about high-dimensional data relationships without additional computations.

Stability and Convergence To show the stability of intermediate outputs, we propose a quantitative measure at iteration t as an average number of the k nearest neighbor changes from the previous iteration $t-1$, i.e.,

$$S_{DR}^1(t) = \frac{1}{nk} \sum_{1 \le i \le n} \left| N_k\left(y_i^t\right) - N_k\left(y_i^{t-1}\right) \right|, \quad (3)$$

where $N_k\left(y_i^t\right)$ is the set of the k nearest neighbor data items of y_i^t at the iteration t. We also compute an average number of the original k nearest neighbors preserved in a low-dimensional space as

$$S_{DR}^2(t) = \frac{1}{nk} \sum_{1 \le i \le n} \left| N_k\left(y_i^t\right) \cap N_k\left(x_i\right) \right|, \quad (4)$$

where $N_k\left(x_i\right)$ is the set of the original k nearest neighbor data items of x_i. A lower value of Eq. (3) indicates a more stable visualization, and a higher value of Eq. (4) indicates a better preservation of given neighborhood relationships.

Second, we visually encode the actual changes of data items during iterations in a scatter plot by drawing the polyline showing the trajectory of each data point over the past few iterations. We also draw a transparent circle whose radius is equal to the total length of the polyline, i.e., the total amount of coordinate changes of the data item, at the same position of the data item. This visual encoding tells us which data points are more stable/unstable than the others. See Figs. 3 and 6 for an example.

4.2 Clustering

Given n data items, $X = \{x_1, \cdots, x_n\} \in \mathbb{R}^{m \times n}$, and the number of clusters c, a clustering method generates their cluster indices, $Y = \{y_1, \cdots, y_n\} \in \mathbb{R}^{1 \times n}$, where $y_i \in \{1, \cdots, c\}$.

k-means k-means repeats (1) minimizing the sum of squared distances between data items and their corresponding cluster centroids and (2) updating cluster assignments.

Latent Dirichlet Allocation (LDA) LDA (Blei, Ng, and Jordan 2003) computes two outputs: the distribution of each topic over words and the distribution of each document over topics. From a clustering viewpoint, the former corresponds to a cluster representative vector μ_j for topic cluster j, and the latter corresponds to a soft-clustering coefficient, which is used to determine y_i by taking the topic index with the maximum value. LDA updates these two sets of outputs alternately, similar to k-means iterations.

Nonnegative Matrix Factorization (NMF) NMF (Lee and Seung 1999) has been successfully utilized in document clustering and topic modeling (Kuang and Park 2013). NMF approximates a nonnegative matrix X as the product of two low-rank nonnegative matrices W and H, which can be interpreted as cluster representatives and membership coefficients, respectively, in the clustering context. One can compute y_i as the largest element index in the i-th column of H. NMF iteratively updates W and H.

User Interaction Capabilities A straightforward interaction is to change cluster assignments of user-selected items. By utilizing the cluster-level interactions, we support cluster splitting and merging in both soft and hard replacement interactions. When merging clusters, the data items in the two clusters to be merged are assigned the same cluster indices. Accordingly, we dynamically reduce c by one. When splitting a cluster, we randomly select a subset of data items in the cluster and assign their new cluster indices as $c+1$, and increase c by one. After these steps, the subsequent iterations are performed.

Stability and Convergence For convergence measure, we use the relative number of cluster membership changes at a given iteration t with respect to the previous iteration, i.e.,

$$S_{CL}(t) = \frac{1}{n} \sum_{1 \le i \le n} I\left(y_i^t \neq y_i^{t-1}\right). \quad (5)$$

By monitoring this measure over iterations, a user can check the stability of the clustering result.

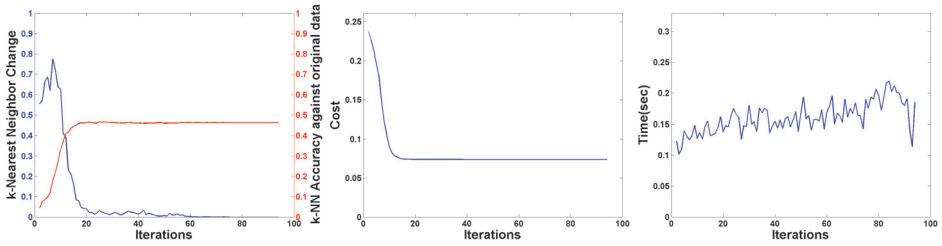
To directly visualize the cluster membership changes, we draw the line connecting the previous cluster label to the current position of a particular data item in the visualization. A large number of lines indicates that the clustering result is going through significant changes. See Figs. 7 and 8 for an example. Additionally, in document clustering, we color-code each keyword in a cluster summary depending on whether the keyword has an increasing (red-colored) or decreasing (blue-colored) importance in the corresponding cluster. See Fig. 10 for an example. In this manner, a user can have a clear understanding of the topic changes over iterations.

5 Experiments

In this section, we present the analyses on the iteration-wise behaviors of machine learning methods as well as various user interaction scenarios in PIVE.

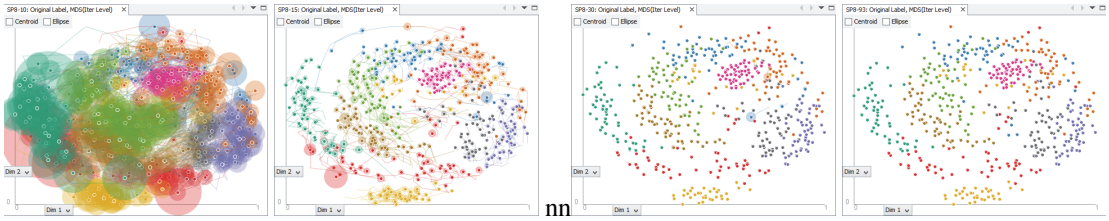
5.1 Iteration-wise Behavior and Visualization

Fig. 2 shows the iteration-wise behavior of MDS along with its computing times. Both our stability/convergence mea-



(a) Convergence and accuracy (b) MDS criterion values in Eq. (1) (c) Computing time

Figure 2: Iteration-wise behavior of MDS for 500 handwritten digit data represented in 16 dimensions.



(a) Iteration 10 (b) Iteration 15 (c) Iteration 30 (d) Iteration 93 (final)

Figure 3: MDS scatter plots for 500 handwritten digit data represented in 16 dimensions.

sures and the MDS criterion value become stable within 20 iterations out of 93 in total, indicating that a small number of iterations suffices visual analytics applications (Figs. 2(a)(b)). This trend can also be found via our visual stability/convergence encoding schemes (Section 4.1). That is, as indicated by large circles and long polylines in Fig. 3(a), the major visual changes occur in early iterations. On the other hand, the result at around the 30th iteration (Fig. 3(c)) is virtually the same as the final one at the 93rd iteration (Fig. 3(d)). Throughout the entire iterations, each iteration takes roughly the same computing time as seen in Fig. 2(c). Therefore, instead of performing a large number of MDS iterations, PIVE quickly provides a user with a sufficiently good visualization. This becomes critical in large-scale data where each iteration requires a lot of time. A similar argument applies to t-SNE. From our measures shown in Figs. 4(a)(b), a stable result can be found as early as at the 130th iteration out of 1,000 in total.

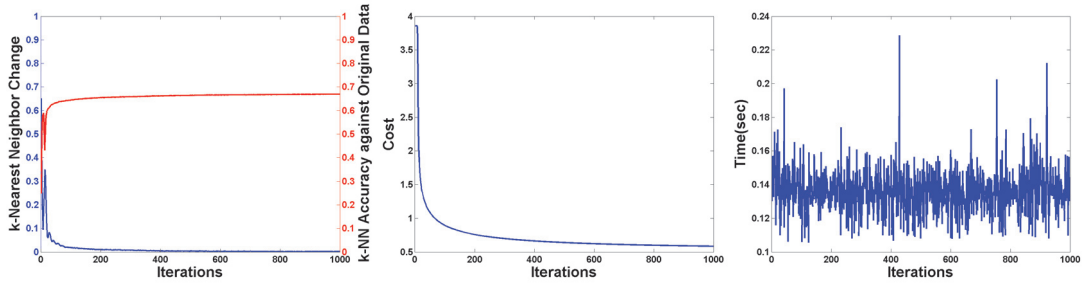
In clustering, Figs. 5(a)(b) presents the iteration-wise behavior of k-means. As seen in Fig. 5(a), significant changes from early iterations diminish quickly as iterations proceed. Nonetheless, the computing time per iteration remains almost the same (Fig. 5(b)). Finally, LDA shows a different behavior from the above-discussed methods in Figs. 5(c)(d). Although the cluster membership changes between iterations generally decrease and the intermediate solutions get close to the final solutions (Fig. 5(c)), the cluster memberships change significantly even after a large number of iterations, e.g., 1,200 iterations. In iVisClustering, we confirm that the top keywords of each topic remain relatively unchanged after several hundreds of iterations, but the randomness of the sampling-based algorithm may prevent LDA from generating consistent outputs for stable visualizations.

5.2 User Interaction Scenarios[4]

We now show interaction scenarios discussed in Section 4.

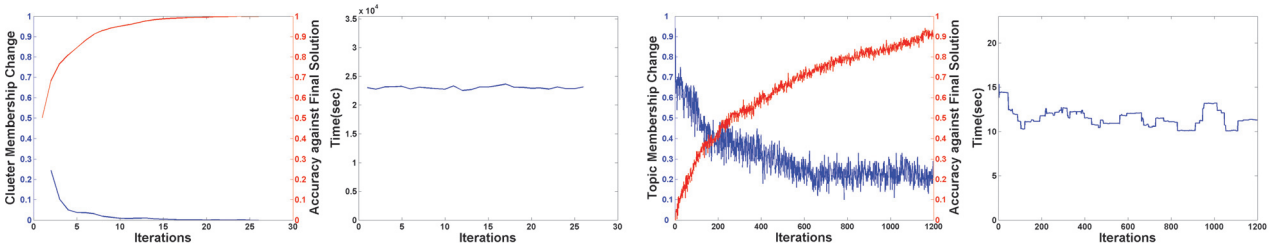
Moving Data Points in t-SNE Fig. 6 shows a sequence of multiple point-moving interactions in t-SNE (Section 4.1) for spoken letter data with 26 classes corresponding to individual alphabet letters. After significant visualization changes, e.g., Fig. 6(a), a sufficiently stable visualization (Fig. 6(b)) still contains many overlapping clusters. Therefore, we move the points representing the letter 'c' (red arrow in Fig. 6(b)) away from the overlapping clusters. As a result, the letter cluster 'z' (blue arrow in Fig. 6(c)) is separated out accordingly, which gives an insight that the letters 'c' and 'z' are pronounced similarly. Second, we move some data points in the letter cluster 'w' (red arrow in Fig. 6(d)), but the neighboring letters 'm' and 'n' (blue arrow in Fig. 6(e)) are not pulled towards the moved points. It indicates that the letter 'w' does not actually sound similar to 'm' and 'n' although the initial visualization did not show this. Next, we move the letter 'q' (red arrow in Fig. 6(e)) out of the cluttered region. In response, the letter 'u' (blue arrow in Fig. 6(f)) is also separated from the overlapping clusters and pulled towards the moved cluster 'q'. This makes sense because these two letters 'q' and 'u' sound similar but quite different from the other previously overlapping letters, 'b', 'd', 'e', 'g', 'p', 't', and 'v', all of which are pronounced with the '–ee' sound at the end. Finally, we increase separations between the two letters 'l' and 'o', which sound quite different, by moving parts of them away from each other (red arrow in Fig. 6(g)). Now, their separation becomes clearer in the visualization (blue arrow in Fig. 6(h)).

[4]A demo video is at http://tiny.cc/aaai17pive.



(a) Convergence and accuracy (b) t-SNE criterion values in Eq. (2) (c) Computing time

Figure 4: Iteration-wise behavior of t-SNE for 1,558 spoken letter data represented in 618 dimensions.



(a) Convergence and accuracy (b) Computing time (c) Convergence and accuracy (d) Computing time

Figure 5: Iteration-wise behavior of (a-b) k-means for CHI papers from 1999-2010 and (c-d) LDA for 20 newsgroups data.

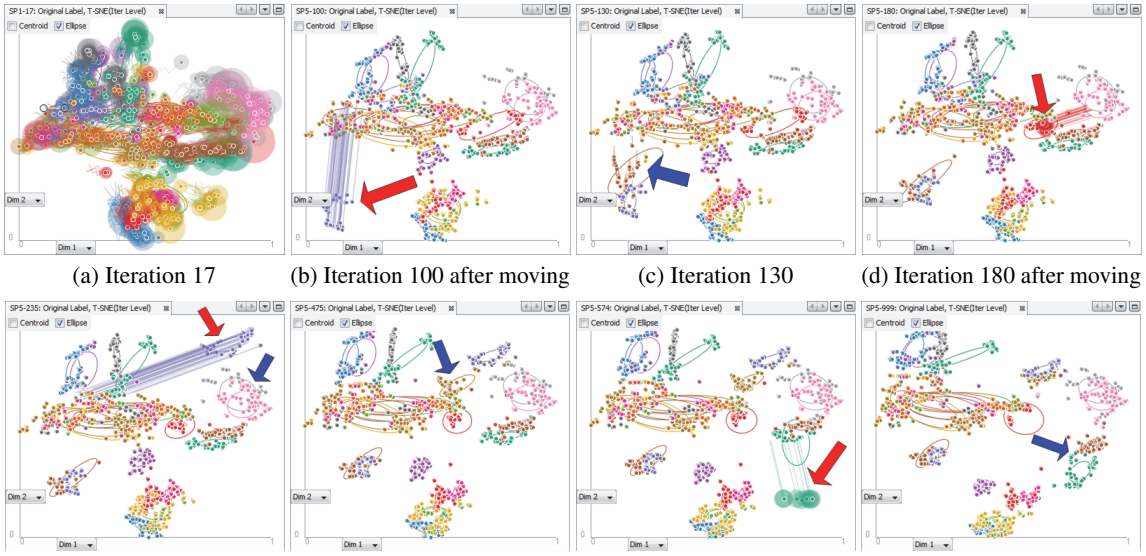
Freezing and Splitting/Merging Clusters in k-means Using k-means that we customized in Jigsaw (Stasko, Görg, and Liu 2008), we first demonstrate an interaction of fixing/freezing cluster assignments for selected data items via hard replacement (Section 4.2). Fig. 7 shows this interaction on the CHI conference papers published between 1999 and 2010. At early iterations, the cluster membership changes are significant (Fig. 7(a)), but the clustering results become much stable after a few iterations, e.g., the sixth iteration out of 26 in total (Fig. 7(b)). At this point, we choose three stable clusters that have clear meaning (green rectangles in Fig. 7(b)) and fix the cluster indices of data items in these clusters, which amount to 32% of the total data items. The final solution due to this interaction (Fig. 7(c)) does not differ much from that without this interaction (Fig. 7(d)). Further analysis shows that less than 10% of the final cluster memberships differ between the two cases, as seen from the increasing blue line reaching 90% accuracy with respect to the final solution without the interaction in Fig. 9(a). The computing time taken for the subsequent iterations drops significantly as shown by the blue line in Fig. 9(b). Next, we merge multiple small, semantically related clusters and split large, unclear clusters, as shown in Fig. 8. In the sixth iteration (Fig. 8(a)), we merge two similar clusters (green rectangles) and split an unclear cluster (purple rectangles). The subsequent iterations (Figs. 8(b)(c)) form a properly merged cluster 'task, performance, models', and a new cluster 'mobile, phone, device' is unveiled from the cluster split, which would not have been found without this interaction (Fig. 7(d)).

Filtering Noisy Documents to Improve Topics in LDA An available interaction with LDA in iVisClustering is to filter those documents with no strong relationships to any particular topics. After filtering, the remaining documents are used to re-run LDA to generate a clearer set of topics. With PIVE, given several mixed topics (black rectangles in Fig. 10(b)), we performed this interaction at around the 300th iteration out of 1,000 in total. Such an interaction successfully improves topic quality at around the 700th iteration (Fig. 10(c)), which, without PIVE, would have taken two full sets of 1,000 iterations of LDA.

6 Discussions

Broad Applicability Our soft and hard replacement interaction methods have fundamental differences from most of the existing interaction methods, e.g., previous point-moving interactions in dimension reduction (Endert et al. 2011; Brown et al. 2012). In particular, our methods do not require any major algorithmic modifications unlike other existing methods. In this respect, ours have a great potential to convert almost any iterative machine learning methods to its interactive version, which would greatly increase their utility in visual analytics. Furthermore, our replacement-based methodology directly interprets a user interaction in the same form as an algorithm output, e.g., low-dimensional coordinates in dimension reduction and cluster assignments in clustering. Thus, our methods do not involve any ambiguous inference of such user interactions to a model parameter, e.g., new Bayesian prior parameter values (Endert et al. 2011) and new linear combination coefficients in a weighted Euclidean distance model (Brown et al. 2012).

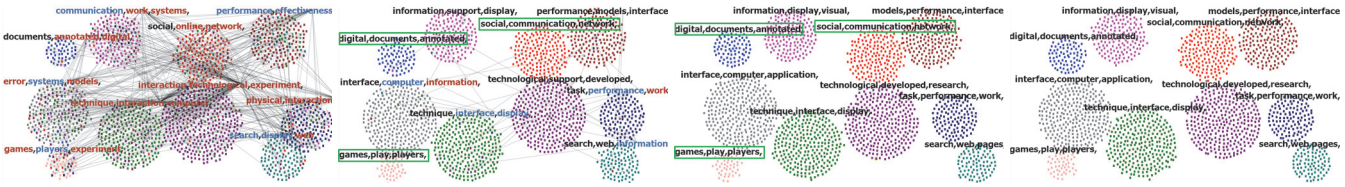
Furthermore, PIVE can significantly benefit various machine learning and data mining tasks including classification,



(a) Iteration 17 (b) Iteration 100 after moving (c) Iteration 130 (d) Iteration 180 after moving

(e) Iteration 235 after moving (f) Iteration 475 (g) Iteration 574 after moving (h) Iteration 999 (final)

Figure 6: Point-moving interactions with t-SNE for 1,558 spoken letter data represented. in 618 dimensions



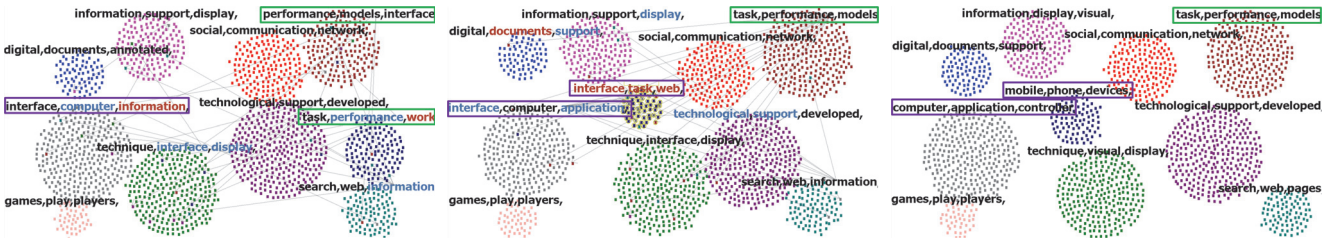
(a) Iteration 2 (b) Iteration 6 (c) Iteration 25 (final) after freezing clusters (d) Iteration 26 (final) without interactions

Figure 7: Cluster freezing interactions with k-means in Jigsaw. (b) At iteration 6, the green-colored clusters are fixed. The final result with/out this interaction is (c)/(d), respectively. The CHI papers published from 1999-2010 were used.

regression, anomaly detection, and association rule mining. In classification and regression, the training process performed in support vector machines, decision trees, and deep neural network can significantly benefit from PIVE via interactively changing parameters, removing noisy features, and correcting misclassified data items for better prediction performance. Similarly, in anomaly detection and association rule mining, PIVE can also help users steer the algorithm in real time to obtain the results in their own manner.

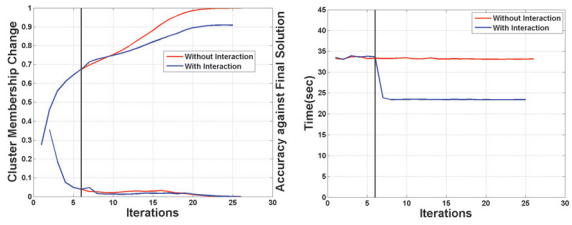
Degree of Influence The simplicity of PIVE bears a potential limitation that the interaction effect reaches only locally. In the above-mentioned existing work, even if a user interaction was performed in a small portion of data, newly adjusted model parameters affect the entire data. On the contrary, this rarely happens when using our methods since our user interactions influence only closely-related data instances and clusters. As a result, a user may have to frequently perform multiple interactions until a satisfactory result is obtained. Even so, PIVE mitigates this drawback by allowing a user to fluidly perform multiple interactions and to easily steer the algorithm output through multiple local changes. This important aspect makes our PIVE-based interactions truly compelling in many scenarios.

Optimal Frequency of Visualization Update Currently, the update frequency of visualizing iteration-wise results is mainly dependent on the speed of algorithm iterations. However, such a frequency may be too fast to keep track of or too slow to spend one's time on. In addition, if the total computing time of an algorithm is short, users may prefer waiting for the entire algorithm iterations to visualizing every intermediate result per iteration. Therefore, it would be important to consider the optimal frequency of visualization update. To handle this issue, one can update the visualization after multiple iterations are performed if each iteration is too fast. If each iteration takes much time, one can further split one iteration into individual data level to provide a faster visualization update. For instance, when visualizing k-means results, each iteration updates the entire set of data items in terms of their cluster indices, which may take long time to finish. In this case, PIVE may update the visualization at the level of an individual (or multiple) data item(s), and accordingly, interactions can be effectively performed during



(a) Iteration 6 (b) Iteration 7 (c) Iteration 26 (final) after split/merge

Figure 8: Split and merging interactions with k-means. From (a) to (b), the green clusters are merged and the purple cluster is split at the sixth iteration. The final result is in (c). The CHI papers published from 1999-2010 were used.



(a) Convergence and accuracy (b) Computing time

Figure 9: Iteration-wise behaviors of k-means with and without the cluster freezing interaction in Fig. 7(b). The black vertical lines represent the interaction time.

the visualization update with an optimal frequency.

7 Conclusions

We present PIVE (**P**er-**I**teration **V**isualization **E**nvironment), a novel framework that supports real-time interactions with machine learning. PIVE visualizes intermediate results during algorithm iterations and allows a user to perform interactions via soft and hard replacement in real time. We also discussed various issues of PIVE and their solutions in terms of stability and convergence as well as computational overheads.

We plan to apply this idea to expand the visual analytic capabilities using machine learning in various manner (Kim et al. 2017).

Acknowledgments

The work of these authors was supported in part by the NSF Grants CCF-0808863, IIS-1707498, IIS-1619028, and IIS-1646881, the DARPA XDATA program Grant FA8750-12-2-0309, and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2016R1C1B2015924). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of funding agencies.

References

Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research (JMLR)* 3:993–1022.

Bosch, H.; Thom, D.; Heimerl, F.; Puttmann, E.; Koch, S.; Kruger, R.; Worner, M.; and Ertl, T. 2013. Scatterblogs2: Real-time monitoring of microblog messages through user-guided filtering. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 19(12):2022–2031.

Brown, E.; Liu, J.; Brodley, C.; and Chang, R. 2012. Dis-function: Learning distance functions interactively. In *Proc. the IEEE Conference on Visual Analytics Science and Technology (VAST)*, 83–92.

Buja, A.; Cook, D.; and Swayne, D. 1996. Interactive high-dimensional data visualization. *Journal of Computational and Graphical Statistics* 5(1):78–99.

Choo, J.; Lee, C.; Reddy, C. K.; and Park, H. 2013a. UTOPIAN: User-driven topic modeling based on interactive nonnegative matrix factorization. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 19(12):1992–2001.

Choo, J.; Lee, H.; Liu, Z.; Stasko, J.; and Park, H. 2013b. An interactive visual testbed system for dimension reduction and clustering of large-scale high-dimensional data. In *Proc. SPIE 8654, Visualization and Data Analysis (VDA)*, 1–15.

Choo, J.; Lee, C.; Kim, H.; Lee, H.; Reddy, C. K.; Drake, B. L.; and Park, H. 2014. PIVE: Per-iteration visualization environment for supporting real-time interactions with computational methods. In *Proc. the IEEE Conference on Visual Analytics Science and Technology (VAST Poster)*, 241–242.

Cox, T. F., and Cox, M. A. A. 2000. *Multidimensional Scaling*. Chapman & Hall/CRC.

Ellis, G., and Dix, A. 2006. Enabling automatic clutter reduction in parallel coordinate plots. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 12(5):717–724.

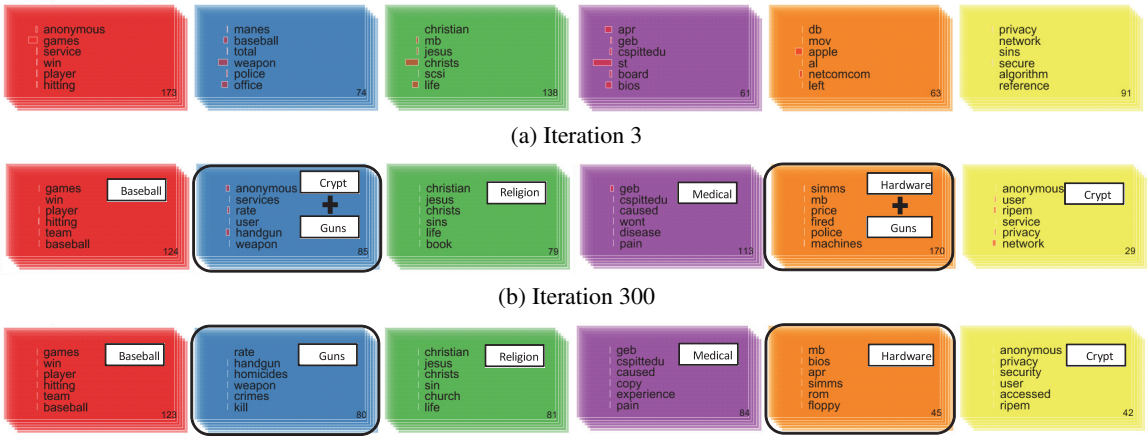
Endert, A.; Han, C.; Maiti, D.; House, L.; Leman, S.; and North, C. 2011. Observation-level interaction with statistical models for visual analytics. In *Proc. the IEEE Conference on Visual Analytics Science and Technology (VAST)*, 121–130.

Fisher, D.; Popov, I.; Drucker, S.; and schraefel, m. 2012. Trust me, i'm partially right: incremental visualization lets analysts explore large datasets faster. In *Proc. the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 1673–1682.

Johnston, W. 2001. Model visualization. *Information Visualization in Data Mining and Knowledge Discovery* 223–228.

Keim, D. 2002. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 8(1):1–8.

Kim, H.; Choo, J.; Park, H.; and Endert, A. 2016. InterAxis: Steering scatterplot axes via observation-level interaction. *IEEE*



(a) Iteration 3

(b) Iteration 300

(c) Iteration 700

Figure 10: Filtering interactions with LDA. Documents with unclear cluster memberships are filtered out at the 300th iteration (out of 1000), and the topics become clearer in the later iterations. 20 newsgroups data was used.

Transactions on Visualization and Computer Graphics (TVCG) 22(1):131–140.

Kim, M.; Kang, K.; Park, D.; Choo, J.; and Elmqvist, N. 2017. Topiclens: Efficient multi-level visual topic exploration of large-scale document collections. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 23(1):151–160.

Kuang, D., and Park, H. 2013. Fast rank-2 nonnegative matrix factorization for hierarchical document clustering. In *Proc. the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 739–747.

Kuang, D.; Choo, J.; and Park, H. 2015. Nonnegative matrix factorization for interactive topic modeling and document clustering. *Partitional Clustering Algorithms* 215–243.

Kwon, B. C.; Kim, H.; Wall, E.; Choo, J.; Park, H.; and Endert, A. 2017. AxiSketcher: Interactive nonlinear axis mapping of visualizations through user drawings. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 23(1):221–230.

Lee, D. D., and Seung, H. S. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401:788–791.

Lee, H.; Kihm, J.; Choo, J.; Stasko, J.; and Park, H. 2012. iVisClustering: An interactive visual document clustering via topic modeling. *Computer Graphics Forum (CGF)* 31(3pt3):1155–1164.

Ma, K.-L. 2009. In situ visualization at extreme scale: Challenges and opportunities. *IEEE Computer Graphics and Applications (CG&A)* 29(6):14–19.

Mülbacher, T.; Piringer, H.; Gratzl, S.; Sedlmair, M.; and Streit, M. 2014. Opening the black box: Strategies for increased user involvement in existing algorithm implementations. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 20(12):1643–1652.

Mulder, J. D.; van Wijk, J. J.; and van Liere, R. 1999. A survey of computational steering environments. *Future Generation Computer Systems* 15(1):119–129.

Piringer, H.; Tominski, C.; Muigg, P.; and Berger, W. 2009. A multi-threading architecture to support interactive visual exploration. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 15(6):1113–1120.

Schreck, T.; Bernard, J.; Von Landesberger, T.; and Kohlhammer, J. 2009. Visual cluster analysis of trajectory data with interactive kohonen maps. *Information Visualization* 8(1):14–29.

Seo, J., and Shneiderman, B. 2002. Interactively exploring hierarchical clustering results. *Computer* 35(7):80–86.

Stasko, J.; Görg, C.; and Liu, Z. 2008. Jigsaw: supporting investigative analysis through interactive visualization. *Information Visualization* 7(2):118–132.

Stolper, C. D.; Perer, A.; and Gotz, D. 2014. Progressive visual analytics: User-driven visual exploration of in-progress analytics. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 20(12):1653–1662.

Thearling, K.; Becker, B.; DeCoste, D.; Mawby, B.; Pilote, M.; and Sommerfield, D. 2001. Visualizing data mining models. *Information Visualization in Data Mining and Knowledge Discovery* 24:205–222.

Thomas, J. J., and Cook, K. A. 2005. *Illuminating the path: The research and development agenda for visual analytics*. IEEE Computer Society Press.

Tu, T.; Yu, H.; Ramirez-Guzman, L.; Bielak, J.; Ghattas, O.; Ma, K.-L.; and O'Hallaron, D. R. 2006. From mesh generation to scientific visualization: an end-to-end approach to parallel supercomputing. In *Proc. the ACM/IEEE conference on Supercomputing*, 12–12.

van den Elzen, S., and van Wijk, J. 2011. Baobabview: Interactive construction and analysis of decision trees. In *Proc. the IEEE Conference on Visual Analytics Science and Technology (VAST)*, 151–160.

van der Maaten, L., and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research (JMLR)* 9:2579–2605.

Williams, M., and Munzner, T. 2004. Steerable, progressive multidimensional scaling. In *Proc. the IEEE Symposium on Information Visualization (InfoVis)*, 57–64.

Yu, H.; Wang, C.; Grout, R.; Chen, J.; and Ma, K.-L. 2010. In situ visualization for large-scale combustion simulations. *IEEE Computer Graphics and Applications (CG&A)* 30(3):45–57.