EigenSense: Saving User Effort with Active Metric Learning

Eli T Brown
Tufts University
Department of Computer Science
161 College Ave.
Medford, MA, USA
ebrown@cs.tufts.edu

Remco Chang
Tufts University
Department of Computer Science
161 College Ave.
Medford, MA, USA
remco@cs.tufts.edu

ABSTRACT

Research in interactive machine learning has shown the effectiveness of live human interaction with machine learning algorithms in many applications. Metric learning is a common type of algorithm employed in this context, using feedback from users to learn a distance metric over the data that encapsulares their own understanding. Less progress has been made on helping users decide which data to examine for potential feedback. Systems may make suggestions for grouping items, or may propose constraints to the user, generally by focusing on fixing areas of uncertainty in the model. For this work in progress, we propose an active learning approach, aimed at an interactive machine learning context, that tries to minimize user effort by directly estimating the amount of change that potential inputs will have on the model and querying users appropriately.

With EigenSense, we use eigenvector sensitivity in the pairwise distance matrix induced by a distance metric over the data to estimate how much a given user input might affect the metric. We evaluate the techinique by comparing the output points it proposes for user consideration against what an oracle would like to choose as inputs.

Categories and Subject Descriptors

I.5.5 [**Pattern Recognition**]: Implementation—*Interactive Systems*; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Human Factors

Keywords

Active Learning, Metric Learning, Interactive Machine Learning

1. INTRODUCTION

The field of interactive machine learning has demonstrated the effectiveness of using human interaction to improve machine learning results, and simultaneously using machine learning algorithms to improve user experiences. Example systems help people group or model their data without having to understand machine learning techniques [14].

One concept in the machine learning apparatus underlying many examples is metric learning. Understanding the similarity between objects is a powerful way to model them for grouping or labelling. In metric learning, a distance function over the data is learned from side information about the data, often in the form of constraints for what objects are similar. For an interactive context, the algorithm must update incrementally by making improvements iteratively with increasing user feedback.

User feedback is expensive, since human throughput at reviewing data is far lower than a computer's throughput at analysis. In order to maximize utility of user efforts, active learning researchers develop techniques to query users for feedback in ways that will help the machine learner. A common approach is to query users about points chosen so that the user feedback will resolve uncertainty in the model.

Coming from an interactive learning perspective, we target our active learning method directly toward predicting how strong any given user input will be. Using our method, a user can judiciously spend the effort of developing feedback on data that will affect the underlying model as much as possible.

In this work in progress, we introduce the EigenScore, a measure that leverages "eigenvector sensitivity" to predict how much a potential user input will change an underlying metric learning model. We then propose EigenSense, which uses EigenScores to guide a user toward making the most productive feedback while minimizing her effort (in terms of data points examined). Finally we provide two types of evidence of the efficacy of this algorithm. First we compare the EigenScores to ground truth measurements of what it estimates, amounts of change introduced by certain metric learning constraints. Second, we show with simulations that the few points selected for user review by EigenSense often include the best possible choices as evaluated by an oracle.

2. MOTIVATION: EIGENVECTOR SENSITIVITY TO FIND CRITICAL POINTS

The eigenvectors of a matrix have been used to represent its underlying structure for applications in many domains including connectivity graph analysis [33], face recognition [43], and clustering [47]. The eigenvectors of symmetric matrices A for which entry A_{ij} represents some measure of distance between objects i and j is of particular relevance. For example, the PageRank [33] algorithm uses an $n \times n$ pairwise matrix to represent the transition probabilities between pairs of the n webpages. Here entry A_{ij} corresponds to the probability of landing on node (page) j during a length one

KDD 2014 New York, New York, USA

random walk, having started at node i. Raising that matrix to the k power gives a matrix of the probabilities of landing on node j having started a *length* k random walk at node i. Increasing powers of A^k will show the assymptotic behavior of flow through the graph. Conveniently, following from the orthoganality and definition of eigenvectors, the dominant eigenvector approximates this quantity: For a real, symmetric matrix A, suppose we have the eigenvalues $\lambda_1, \ldots, \lambda_n$ sorted in decreasing order, and their corresponding eigenvectors $\mathbf{v_1}, \ldots, \mathbf{v_n}$. Because the eigenvectors are orthogonal, any vector $\mathbf{x} \in \mathbb{R}^n$ can be written as a linear combination of the eigenvectors, with coefficients α_i. We can observe the asymptotic behavior:

$$\begin{aligned} x &= \alpha_1 \mathbf{v_1} + \alpha_2 \mathbf{v_2} + \ldots + \alpha_n \mathbf{v_n} \\ Ax &= \alpha_1 A\mathbf{v_1} + \alpha_2 A\mathbf{v_2} + \ldots + \alpha_n A\mathbf{v_n} \\ A^k x &= \alpha_1 \lambda_1^k \mathbf{v_1} + \alpha_2 \lambda_2^k \mathbf{v_2} + \ldots + \alpha_n \lambda_n^k \mathbf{v_n} \\ &= \alpha_1 \lambda_1^k \left(\mathbf{v_1} + \frac{\alpha_2}{\alpha_1} \left(\frac{\lambda_2}{\lambda_1} \right)^k \mathbf{v_2} + \ldots + \frac{\alpha_n}{\alpha_1} \left(\frac{\lambda_n}{\lambda_1} \right)^k \mathbf{v_n} \right) \end{aligned}$$

Note that because the dominant eigenvalue, $\lambda_1 \geq \lambda_i, i = 2 \ldots n$ in the final sum, the $\mathbf{v_1}$ term dominates.

When studying population dynamics, biologists take advantage of this fact of with matrix L, called a "Leslie" matrix, where each element L_{ij} represents an organism surviving from age i to age j [1]. To see the equilibrium point of a population, biologists then study the dominant eigenvector of this matrix [22].

Extending this technique, to see how the population can be affected by environmental factors, biologists adjust survival rates at different times in the lifecycle by editing the matrix and reconsider the new dominant eigenvalue. This sensitivity of the eigenvalue to change in particular matrix entries is the eigenvalue sensitivity [22].

Motivated by biologists' successes with eigenvalue sensitivity and considering the similar behavior of the dominant eigenvector of $n \times n$ pairwise transition matrices, we adopt these ideas to the context of active metric learning. We will use the dominant eigenvector of a pairwise distance matrix as a standin for its overall structure, and calculate the sensitivy of that eigenvector with respect to changes in the matrix. Using this model, we will estimate how much individual user inputs, i.e. constraining points close together corresponding to lowering an entry in the distance matrix, will affect the structure of the distance matrix and the underlying distance metric.

3. RELATED WORK

This work builds on previous efforts in both machine learning and human-computer interaction. We begin with an overview of related work in interactive machine learning and then discuss metric and active machine learning.

3.1 Interactive Machine Learning

Interactive machine learning researchers strive to use human interaction with machine learning to improve machine learning results and improve user experiences, leveraging computers' raw analytical power and humans' reasoning skills to acheive results greater than either alone [39]. Several methods coupling machine learning techniques with visualization to cluster or classify data have been proposed [6, 10, 17]. Systems have been built for grading [11], network alarm triage [5], building social network groups [4], ranking search results in user context [2], managing overeating [16], and searching for images [3].

Another vein of this research, from visual analytics, focuses on data analysis tasks, and on effectively leveraging user interaction to refine an underlying model, generally by adjustments of layouts or clusterings [14, 20, 23, 13].

All of this work integrates human reasoning with machine learning without asking the human to understand the machine learning. However, these systems do not offer strong active learning support to help the user give the most efficient feedback. The EigenSense approach aims to provide this support by guiding the user toward the most important data to review.

3.2 Metric Learning

Many of the examples above of interactive machine learning use metric learning algorithms at their core. This powerful approach has been the subject of much research since 2003 [8, 12, 19, 25, 34, 38, 42, 45, 46, 48, 51, 52] and has proven applicable in many real-world application domains including information retrieval, face verification and image recognition [18, 26, 29].

Most of these methods assume that the machine learner is given additional information beyond the data itself, most often as pairwise constraints between data points, i.e. that they should or should not be close together. Using this side information, existing methods seek to learn a distance metric such that the distance between similar examples should be relatively smaller than that between dissimilar examples.

It is generally assumed that a domain expert can easily provide pairs of similar data points and pairs of dissimilar data points, but that assumption implies a perfectly accurate user who is motivated and available to review all of the data. This work in progress begins to address this gap by introducing a technique for paring down what points an expert would actually have to review in an interactive machine learning context by trying to guide a user toward constraints that will be most impactful to the distance metric.

3.3 Active Learning

Active learning is a form of semi-supervised machine learning in which the learner iteratively queries the user for additional information while building a model for classification or clustering. The key idea behind active learning is that an algorithm can achieve greater accuracy or performance with fewer training labels if it is allowed to choose the most helpful labels [36].

A common approach is to select the data points that are most uncertain to classify. Different measures of the uncertainty are based on the disagreement in the class labels predicted by an ensemble of classification models [1, 32, 37], by distance to the decision boundary [15, 35, 41], by the uncertainty of an unlabeled example's projection using the Fisher information matrix [31, 53], or with Bayesian analysis that takes into account the model distribution [24, 30, 40, 54].

Active learning and metric learning come together in several recent works, where authors determine what the user should see based on uncertainty of labels and coverage of the dataset [21], or the median points in groups with the same label [44]. In Yang, et al., the authors select pairs of points for feedback based on the uncertainty of deciding their closeness [50].

Querying the user about pairs of points is the common approach in a sub-category of active learning algorithms called active clustering, where the end goal is a clustering instead of labels, and so the user iteratively provides constraints. Points are chosen by uncertainty [28], or by most informative example pairs [9]. One work by Xu, et al., is especially related to ours. The authors learn a

[1]The matrix represents different age groups' rates of survival and reproduction by setting the first row to the fertility rate of each age group, and the lower off-diagonal entries to organism survival probabilities from one age group to the next.

two-class spectral clustering with active learning by examining the eigenvectors of the pairwise distance matrix to find points on the boundary of being put in either cluster [49].

Generally, active learning methods are based on querying the user for one unit of feedback at a time. In our approach the user plays an active roll in deciding what feedback to provide: no suggestions are given without an initial seed point of interest from the user, and then, several suggestions are provided for the user to peruse.

4. APPLICATION CONTEXT

The EigenSense method is best understood within a real interactive learning context. In prior work, Brown, et al., created Dis-Function [14], a system that shows an analyst high-dimensional real-valued data in a 2D projection, and learns a distance function iteratively though user feedback. Feedback is provided by dragging together points that should be closer together. However, the user is given no information about what points would be helpful to the metric learning backend.

For illustration, we have integrated EigenSense into Dis-Function. When a user clicks a point of interest, EigenSense responds by showing several points that may be of interest relative to the first. Figure 1 presents a screenshot of this modified Dis-Function, specifically the data projection portion. The data have been arranged using multidimensional scaling (MDS), and colored based on a spectral clustering. The user has clicked the point indicated by a red X. In response, EigenSense adds colored squares showing which points may be of interest relative to that X. Darker colors show a higher eigenvector sensitivity score, or EigenScore (see Section 5).

These predictions of which points could provide the strongest update to the model are intended to guide the user towards giving the machine learner fruitful feedback, and taking best advantage of precious expert user time.

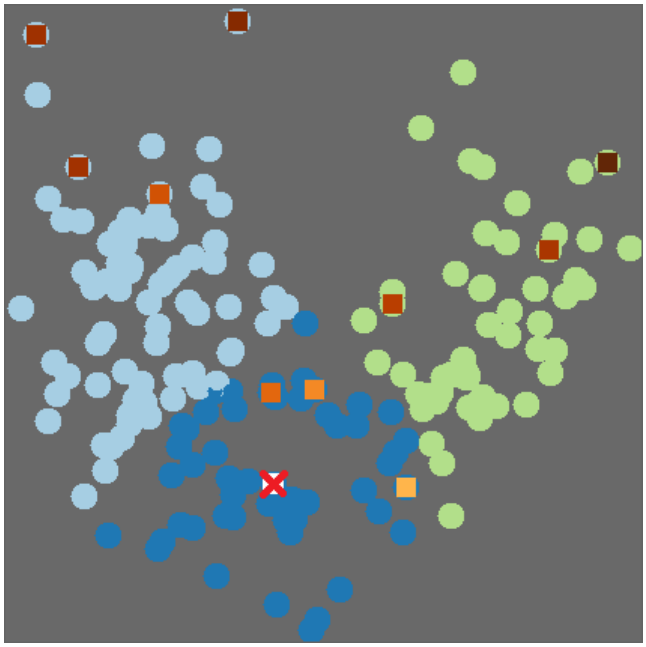


Figure 1: EigenSense demonstrated on an interactive scatterplot of projected data. All data points are laid out with MDS and colored by a spectral clustering. The point with a red X is the one the user clicked, asking what other data should be considered in relation to that point. The colored squares show the EigenSense response, with darker colors indicating higher EigenScores (see Section 5.1). Only the top five percent of scores from each cluster are highlighted, helping the user target the most fruitful data to examine.

5. THE EIGENSENSE METHOD

In our interactive machine learning context, we have presented a user with data and need useful side information to improve our learned model. More specifically, in the context provided by Section 4, we assume a user examines a visualization of data and notices points of interest, perhaps outliers, cluster exemplars, or points aligned with personal expertise. We aim to answer, given one point of interest selected by the user, which are other points that the user should examine. We chose this interaction paradigm for two reasons: first, the user guides the process as opposed to simply being used for point comparisons, and second, the input starting point sharply reduces the computational complexity (see Section 5.2). In these points suggested for user examination, our ideal is to uncover the point that would make the strongest update to the model for the user's effort, leveraging their expertise and efforts efficiently.

In this section, we introduce a technique using eigenvector sensitivity on a pairwise distance matrix to provide these predictions of strong model updates. First we associate a score (called the EigenScore) with any pair of data points. The EigenScore of a pair is designed to predict the strength of user feedback about that pair to updating the model. We then present the EigenSense algorithm, which uses EigenScores to recommend top candidate points to the user.

5.1 Calculating EigenScores

The EigenScore between two points represents our prediction for how strongly a change in distance between them would affect the underlying structure of the pairwise distance matrix. Specifically, it is a measure of the sensitivity of the dominant eigenvector of that matrix to changes in elements corresonding to pairs of data points.

Given a distance function and the data set with N points, we calculate the pairwise distance matrix

$$D \in \mathbb{R}^{N\times N} \text{ where } D_{ij} = distance(x_i, x_j)$$

for all pairs x_i and x_j. Note that no specific type of distance function is required. To model how that matrix changes with specific x_i and x_j assumed to be perfectly close togther, i.e. because the user specified so with feedback, we construct a new distance matrix D' which is identical to D, except that we set $D_{ij} = D_{ji} = 0$. These entries now reflect that x_i and x_j should be close to one another.

We next compute the dominant eigenvector for D, called v_1, and for D', which we indicate with v'_1. We compute the cosine similarity between v_1 and v'_1. Note that we desire a dissimilarity metric, showing how much v'_1 is different from v_1, so we define EigenScore(x_i, x_j) = 1 - CosineSimilarity(v_1, v'_1). Algorithm 1 summarizes this process.

Note that computing the function $eig(D)$ to return the dominant eigenvector is computationally expensive if implemented using factorization methods [27]. Techniques such as SVD [27] and the Cholesky decomposition [27] return all eigenvectors of the matrix D. However, because computing the EigenScore requires only the dominant eigenvector (and because we are restricted to a real-valued symmetric matrix), we can dramatically improve performance by using the Lanczos method [27], which returns only the dominant eigenvector and which we denote $eigs(D, 1)$ as in MATLAB.

Algorithm 1: EigenScore

Input: Data points x_i, x_j, distance matrix D
Output: $ES_{ij} \in [0,1]$

Calculate $v_1 = eigs(D,1)$ [dominant eigenvector]
Let $D' = D$
Set $D'_{ij} = D'_{ji} = 0$
Calculate $v'_1 = eigs(D',1)$
Set $\hat{v}_1 = normalize(v_1)$, $\hat{v}'_1 = normalize(v'_1)$
$ES_{ij} = 1 - \hat{v}_1 \cdot \hat{v}'_1$
return ES_{ij}

5.2 Using EigenScores to make EigenSense

The EigenScore algorithm maps a pair of points to a scalar value, which implicitly provides a ranking over pairs of points by their potential impact on the distance metric. This section addresses how to use this ranking with the goal of reducing user effort.

Calculating EigenScores over all pairs of points is prohibitively expensive, but recall that in our usage context, the user has selected one point of interest and we must suggest options for the second point of a pair for the user to constrain. This requires only $(N-1)$ evaluations of EigenScore. We further limit the number of suggestions the user sees to some proportion $k \in (0,1]$ of the total data to save the user from examining every point. Rather than returning a fully ranked list of the top $k*(N-1)$ of the $(N-1)$ total points, we want to choose a *diverse* set of points for consideration. Our rationale is that we expect high EigenScores to correspond to pairs of points where user constraints would cause big updates to the model, but these may not be good updates. For example, outliers in the dataset will often contribute to high EigenScores, but should not necessarily be constrained as close to other data.

To create the desired set of suggestions, we first cluster the data (using the current learned distance function), then sort the points in each cluster c by their EigenScore and return the top $k*|c|$ points within each cluster. This algorithm is detailed in Algorithm 2.

The performance of our implementation is critical to demonstrating the feasibility of this technique for interactive systems. Our prototype system provides EigenSense recommendations on demand as response to interaction with a visualization. The current implementation connects to MATLAB from C# via a COM interface to take advantage of the Lanczos algorithm for quickly calculating the dominant eigenvector. Still, as an example of performance capability, on a laptop with an Intel i5 480M processor, for a dataset of about 200 points with about 20 dimensions, an EigenSense response takes about one second.

Algorithm 2: EigenSense

Input: Initial point x_i, distance matrix D, set of clusters C, threshold k
Output: S, a set of model-critical points

foreach *cluster* $c \in C$ **do**
  foreach *point* $x_j \in c$ **do**
    Compute $ES_{ij} = EigenScore(x_i, x_j, D)$
  Let S_c be the set of $k \times |c|$ points with the highest ES_{ij}
Let $S = \bigcup S_c$
return S

6. EXPERIMENTS AND RESULTS

We validate the accuracy and effectiveness of our proposed method through two experiments on test datasets from the UCI Machine Learning Repository[7]. First, we compare EigenScores against actual values of the quantity they estimate and see that they could be an effective low-cost estimator of model change. Second, we evaluate the accuracy of EigenSense by considering the quality of the sets of points it offers to users compared against the ground truth best points. We show that guided by EigenSense, a user could pick high-quality inputs while reviewing small amounts of data.

6.1 Experiment 1: Compare To Ground Truth

In this experiment we evaluate the EigenScores by comparing them directly to the value they are attempting to estimate. Recall that in our interactive metric learning context, EigenScores are an estimate of how much the distance matrix, as a standin for the distance metric itself, would be changed by constraining a given pair of points. The ground truth is prohibitively expensive to calculate for an interactive system, but can be prepared offline.

For three datasets, starting from scratch with no constraints, we used our prototype system (with interactive metric learning based on Brown et al. [14]) to calculate for each possible pair of points the actual change in distance function resulting from constraining the pair. The graphs in Figure 2 show the comparison of these values to the EigenScores. We use weighted Euclidean distance functions, thus the initial distance function is parameterized by the vector $\Theta_{init} = (1/M, ..., 1/M)$ of length M. In the graphs, the horizontal axis is the change in the distance metric from applying the constraint and the vertical axis is the EigenScore:

$$1 - \text{CosineSimilarity}(\Theta_{init}, \Theta_{post\_constraint})$$

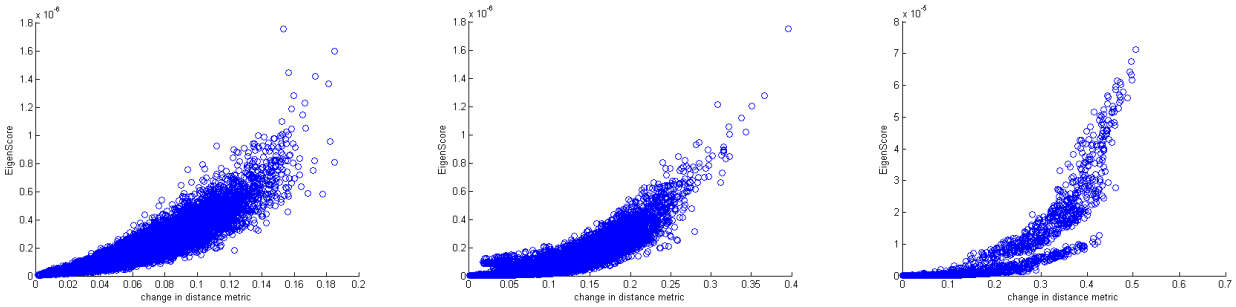
Although the correlations between EigenScores and actual distance metric change are not obvious linear relationships, it is apparent from visual inspection that the quantities are related. This first pass evaluation shows the promise of EigenScores as an estimate of distance metric change, which implies that it could be an inexpensive way to predict model change for interactive machine learning.

6.2 Experiment 2: Evaluate Suggestion Quality

The goal of this experiment is to determine the quality of EigenSense recommendations by comparing them to the choices an oracle would make. Given an oracle that can rank all user feedback options in terms of which yield the best distance functions, we look to see how the EigenSense recommendations rank in that list.

We simulate choices of a point of interest x_i by the user, and then compute both the oracle's ranking of all possible constraint pairs with x_i, and the set of EigenSense options that would be presented to the user. Specifically, the oracle takes advantage of the labels for these test datasets and calculates, for all pairs of constraints that include x_i, the accuracy (with k-NN) of the distance metric resulting from an update with the given constraint. That is, given one point x_i, the oracle applies the system's metric learning algorithm with each constraint pair $(x_i, x_j) \forall x_j$, and evaluates each resulting distance function at classifying the data with k-NN. The accuracy scores of these evaluations provide a ranking over the constraint pairs. We compare the EigenSense options against the oracle ranking by finding the EigenSense recommendation with minimum oracle rank.

Figure 3 shows the results of our experiment. Each graph line corresponds to a different dataset, and each plotted point represents an average over ten simulations, each of which simulated ten user



(a) Wine (b) Ion (c) Parkinsons

Figure 2: Experiment 1 - Comparison between EigenScores and actual change to distance metric for three example datasets. Each point in each graph is a pair of datapoints. The horizontal axis is the amount the underlying distance function would change if that pair of points were constrained together. The vertical axis is the EigenScore for that pair of points.

inputs. Simulated users picked a first point randomly then some (not necessarily optimal) EigenSense recommendation for the second. In total, each plotted point represents 100 uses of EigenSense. The horizontal axis is the k parameter of EigenSense (see Algorithm 2 and Section 5.2), which determines how many points will be shown to the user. Because the vertical axis shows the best oracle ranking of the EigenSense points, lower scores are better. It is no surprise that with larger values of k, where the user is being shown more points, the opportunity for the best-ranked points to be included is higher. Using a low value of k means showing the user few points and saving effort, whereas using a high value means showing more points but having a better chance to show the absolute best ones. The results of this experiment suggest that, depending on the dataset, a user could give strong feedback to a metric learner while only reviewing less than ten percent of the data, or in some cases, substantially less.

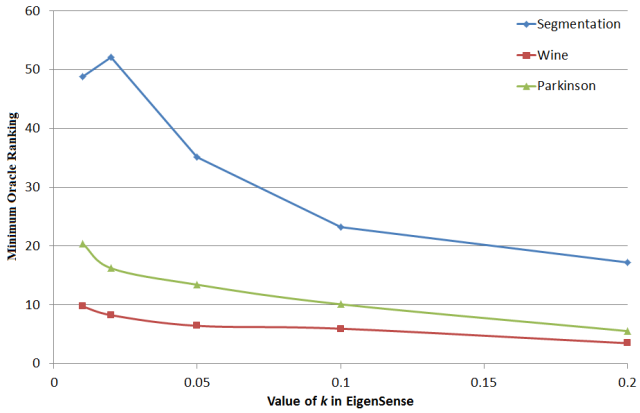


Figure 3: Experiment 2 - the horizontal axis shows values of the k parameter to EigenSense, i.e. how much data is shown to the user. The vertical axis shows the minimum (best) rank of the EigenSense recommendations in the oracle's ordering of all possible point pairs. Note that, as expected, as more data is shown to the user (k increases), there is more chance of the best possible options being revealed (rank decreases). Even with a small amount of data revealed, the EigenSense suggestions provide strong options.

7. FUTURE WORK

Although we have collected the presented evidence of EigenSense's effectiveness, there are opportunities for improving the algorithm itself. For example, there are several variations on how to generate pairwise distance or similarity matrices. Further, the the performance of the implementation could be improved by using a library implementation of the Lanczos method for calculating the dominant eigenvector, instead of using MATLAB via COM calls.

The performance improvement is critical for the main thrust of future work, which is to complete the evaluation of the technique by testing it with human subjects. In particular, participants in a user study will use the tool to cluster some images with known classes. We can then evaluate their comfort with the tool, confidence in the recommendations, and progressive accuracy of the distance metrics learned from their inputs to see if they do better with or without EigenSense.

8. CONCLUSION

This paper contributes to the study of interactive metric learning by applying active learning to reduce the workload of the human actor. We introduced the concept of EigenScores based on eigenvector sensitivity of distance matrices, and then applied these to create the EigenSense algorithm, which identifies and recommends points for user consideration given an initial exploratory direction. We presented evidence of the effectiveness of the algorithm by demonstrating its correlation with ground truth values of the quantity it estimates, and then by showing the frequency with which EigenSense presents the best possible option to users. Our results indicate that EigenSense could help save human workload by vastly reducing number of data points to be considered while maintaining near-optimal metric learning results.

9. REFERENCES

[1] N. Abe and H. Mamitsuka. Query learning strategies using boosting and bagging. In *ICML*, pages 1–9, 1998.

[2] R. Agrawal, R. Rantzau, and E. Terzi. Context-sensitive ranking. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 383–394. ACM, 2006.

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

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

Human Factors in Computing Systems, pages 21–30. ACM, 2012.

[5] S. Amershi, B. Lee, A. Kapoor, R. Mahajan, and B. Christian. Human-guided machine learning for fast and accurate network alarm triage.

[6] G. Andrienko, N. Andrienko, S. Rinzivillo, M. Nanni, D. Pedreschi, and F. Giannotti. Interactive visual clustering of large collections of trajectories. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 3–10. IEEE, 2009.

[7] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[8] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, pages 937–965, 2005.

[9] S. Basu, A. Banerjee, and R. Mooney. Active semi-supervision for pairwise constrained clustering. In *ICDM*, pages 333–344, 2004.

[10] S. Basu, S. M. Drucker, and H. Lu. Assisting Users with Clustering Tasks by. pages 394–400.

[11] S. Basu, C. Jacobs, and L. Vanderwende. Powergrading: a clustering approach to amplify human effort for short answer grading.

[12] M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the 21st International Conference on Machine Learning*, pages 81–88, 2004.

[13] J. Broekens, T. Cocx, and W. Kosters. Object-centered interactive multi-dimensional scaling: Ask the expert. In *Proceedings of the 18th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC)*, pages 59–66, 2006.

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

[15] C. Campbell, N. Cristianini, and A. Smola. Query learning with large margin classifiers. In *ICML*, pages 111–118, 2000.

[16] E. Carroll, M. Czerwinski, A. Roseway, A. Kapoor, P. Johns, K. Rowan, and M. Schraefel. Food and mood: Just-in-time support for emotional eating. In *Affective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on*, pages 252–257, Sept 2013.

[17] J. Choo, H. Lee, J. Kihm, and H. Park. iVisClassifier: An interactive visual analytics system for classification based on supervised dimension reduction. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 27–34. IEEE, 2010.

[18] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 539–546, 2005.

[19] J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon. Information-theoretic metric learning. In *Proceedings of International Conference on Machine Learning*, pages 209–216, 2007.

[20] M. Desjardins, J. MacGlashan, and J. Ferraioli. Interactive visual clustering. In *Proceedings of the 12th International Conference on Intelligent User Interfaces*, pages 361–364. ACM, 2007.

[21] S. Ebert, M. Fritz, and B. Schiele. Active metric learning for object recognition. In A. Pinz, T. Pock, H. Bischof, and F. Leberl, editors, *Pattern Recognition*, volume 7476 of *Lecture Notes in Computer Science*, pages 327–336. Springer Berlin Heidelberg, 2012.

[22] S. P. Ellner and J. Guckenheimer. *Dynamic models in biology*. Princeton University Press, 2011.

[23] A. Endert, P. Fiaux, and C. North. Semantic interaction for visual text analytics. In *Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems (CHI)*, pages 473–482. ACM, 2012.

[24] Y. Freund, H. Seung, Sebastian, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning Journal*, pages 133–168, 1997.

[25] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems*, pages 513–520, 2004.

[26] M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? Metric learning approaches for face identification. In *Proceedings of the International Conference on Computer Vision*, pages 498–505, 2009.

[27] M. Heath. *Scientific Computing*. The McGraw-Hill Companies, Incorporated, 2001.

[28] T. Hofmann and J. Buhmann. Active data clustering. In *NIPS*, pages 528–534, 1997.

[29] S. Hoi, W. Liu, M. Lyu, and W. Ma. Learning distance metrics with contextual constraints for image retrieval. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2072–2078, 2006.

[30] R. Jin and L. Si. A bayesian approach toward active learning for collaborative filtering. In *Uncertainty in Artificial Intelligence*, pages 278–285, 2004.

[31] D. MacKay. Information-based objective functions for active data selection. *Neural Computation*, pages 590–604, 1992.

[32] P. Melville and R. J. Mooney. Diverse ensembles for active learning. In *ICML*, pages 74–83, 2004.

[33] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[34] R. Rosales and G. Fung. Learning sparse metrics via linear programming. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 367–373, 2006.

[35] N. Roy and A. Mccallum. Toward optimal active learning through sampling estimation of error reduction. In *ICML*, pages 441–448, 2001.

[36] B. Settles. Active learning literature survey. *University of Wisconsin, Madison*, 2010.

[37] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *workshop on Computational learning theory*, pages 287–294, 1992.

[38] C. Shen, J. Kim, L. Wang, and A. Hengel. Positive semidefinite metric learning with boosting. In *Advances in Neural Information Processing Systems 22*, pages 1651–1659, 2009.

[39] S. Stumpf, V. Rajaram, L. Li, W.-k. Wong, M. Burnett, T. Dietterich, E. Sullivan, and J. Herlocker. Interacting Meaningfully with Machine Learning Systems :. 67:639–662, 2009.

[40] S. Tong and D. Koller. Active learning for parameter estimation in bayesian networks. In *NIPS*, pages 647–653,

2001.

[41] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *Journal Of Machine Learning Research*, pages 999–1006, 2001.

[42] L. Torresani and K. C. Lee. Large margin component analysis. In *Advances in Neural Information Processing Systems*, pages 1385–1392, 2007.

[43] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 586–591. IEEE, 1991.

[44] F. Wang, J. Sun, T. Li, and N. Anerousis. Two heads better than one: Metric+active learning and its applications for it service classification. In *Data Mining, 2009. ICDM '09. Ninth IEEE International Conference on*, pages 1022–1027, Dec 2009.

[45] K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems*, pages 10:207–244, 2006.

[46] K. Weinberger and L. Saul. Fast solvers and efficient implementations for distance metric learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1160–1167, 2008.

[47] Y. Weiss. Segmentation using eigenvectors: a unifying view. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 975–982. IEEE, 1999.

[48] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, pages 505–512, 2002.

[49] Q. Xu, K. Wagstaff, et al. Active constrained clustering by examining spectral eigenvectors. In *Discovery Science*, pages 294–307, 2005.

[50] L. Yang and R. Jin. Bayesian active distance metric learning. *UAI*, 2007.

[51] Y. Ying, K. Huang, and C. Campbell. Sparse metric learning via smooth optimization. In *Advances in Neural Information Processing Systems 22*, pages 2214–2222, 2009.

[52] Y. Ying and P. Li. Distance metric learning with eigenvalue optimization. In *Journal of Machine Learning Research*, pages 1–26, 2012.

[53] T. Zhang and F. Oles. A probability analysis on the value of unlabeled data for classification problems. *ICML*, pages 1191–1198, 2000.

[54] Y. Zhang, W. Xu, and J. Callan. Exploration and exploitation in adaptive filtering based on bayesian active learning. In *ICML*, pages 896–903, 2003.