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 encapsulates 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 impact on the model of potential inputs, and querying users accordingly.

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 technique 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 labeling. In metric learning, a distance function over the data is learned from side information about the data, often in the form of constraints for which 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.

Emphasizing the interactive learning perspective, our approach keeps the user in control, providing suggestions only when the user initiates a direction of inquiry. We target our active learning method toward predicting the impact of any given user input. 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 first introduce 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 his or her effort (in terms of data points examined). Finally we provide two types of evidence of the efficacy of this algorithm. First, we compare EigenScores to the ground-truth of what they estimate: the amount that particular constraints would change the metric learning model. 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 random walk, having started at node i. Raising that matrix to the power k 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 asymptotic behavior of flow through the graph. Conveniently, following from the definition and orthogonality of eigenvectors, the dominant eigenvector approximates this quantity. For a real, symmetric matrix A, suppose we have the eigenval-
ues $\lambda_1, \ldots, \lambda_n$ sorted in decreasing order, and their corresponding eigenvectors $v_1, \ldots, v_n$. Because the eigenvectors are orthogonal, any vector $x \in \mathbb{R}^n$ can be written as a linear combination of the eigenvectors, with coefficients $\alpha$. We can observe the asymptotic behavior:

$$x = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$$
$$Ax = \alpha_1 A v_1 + \alpha_2 A v_2 + \ldots + \alpha_n A v_n$$
$$A^k x = \alpha_1 \lambda_1^k v_1 + \alpha_2 \lambda_2^k v_2 + \ldots + \alpha_n \lambda_n^k v_n$$

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

When studying population dynamics, biologists take advantage of this fact with a matrix $L$, called a “Leslie” matrix, where each element $L_{ij}$ represents an organism’s survival prospects to age $i$ from age $j$. To see the equilibrium point of a population, biologists 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 in Leslie matrices, we consider the behavior of the dominant eigenvector of our related $n \times n$ pairwise distance matrices, and we adapt the concept of sensitivity to the context of active metric learning. We will use the dominant eigenvector of a pairwise distance matrix as a stand-in for its overall structure, and calculate the sensitivity of that eigenvector with respect to changes in entries of that matrix.

Since each entry corresponds to a pair of data points, we will use this approach to estimate how individual user inputs, i.e. user constraints that certain pairs of points have small distances between them, 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 achieve 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 with 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 certain pairs should or should not be close together. With that information, metric learning techniques learn a distance function optimized to produce relatively small distances between points that belong close together, and large distances between those that belong far apart [52].

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 its model. 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]. Yang and Jin select pairs of points for feedback based on the uncertainty of deciding their closeness [50].

In a sub-category of active learning algorithms called active clustering, the end goal is a clustering instead of classification, and the common approach is to gather constraints by iteratively querying the user about pairs of points. 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 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 role in deciding what feedback to provide: no sug-
gestions are given without an initial seed point of interest from the user, and then, several suggestions are provided for the user to pursue.

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 through user feedback. Feedback is provided by dragging together points that should be closer together. The method is effective with appropriate user feedback, but 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 marked by a red X. In response, EigenSense adds colored squares showing which points may be of interest relative to that X. Darker colors indicate a higher eigenvector sensitivity score, or EigenScore (see Section 5.1).

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

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, having an initial point sharply reduces the computational complexity (see Section 5.2). In deciding which points to suggest for user examination, our ideal is to uncover the point that would make the strongest update to the model with the user’s feedback, leveraging expertise efficiently to minimize effort.

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 a model update corresponding to user feedback about that pair. 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 its elements, which correspond to pairs of data points.

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

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

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 together, 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

\[
\text{EigenScore}(x_i, x_j) = 1 - \text{CosineSimilarity}(v_1, v'_1)
\]

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 \(eigs(D, 1)\) in MATLAB.

5.2 Using EigenScores to make EigenSense

Figure 1: EigenSense demonstrated on an interactive scatter-plot of projected data – all data points are laid out with multidimensional scaling (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.
Algorithm 1: EigenScore

Input: Data points \( x_i, x_j \), distance matrix \( D \)
Output: \( ES_{ij} \in [0, 1] \)
1. Calculate \( v_1 = eigs(D, 1) \) [dominant eigenvector]
2. Let \( D' = D \)
3. Set \( D'_j = D'_j = 0 \)
4. Calculate \( v'_j = eigs(D', 1) \)
5. Set \( ES_{ij} = 1 - \text{CosineSimilarity}(v_1, v'_j) \)
6. return \( ES_{ij} \)

The EigenScore algorithm maps a pair of points to a scalar value representing potential impact on the distance metric, and thus implicitly provides a ranking over pairs of points. 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 for an interactive context. However, recall that in our usage context, the user has selected one point of interest and we must suggest options for a second point to pair with the first for a potential user constraint. Evaluating possibilities for just the choices of a second point 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 \( k \ast (N - 1) \) 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 may contribute to high EigenScores, but should not necessarily be used in constraints.

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 \( k \ast |c| \) points within each cluster. This process 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
1. foreach cluster \( c \in C \) do
2.   foreach point \( x_j \in c \) do
3.     Compute \( ES_{ij} = \text{EigenScore}(x_i, x_j, D) \)
4.   Let \( S_c \) be the set of \( k \times |c| \) points with the highest \( ES_{ij} \)
5.   Let \( S = \bigcup S_c \)
6. 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 metric, as a stand-in 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 \( \Theta_{\text{init}} = (1/M, \ldots, 1/M) \) 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:

\[ 1 - \text{CosineSimilarity}(\Theta_{\text{init}}, \Theta_{\text{post_constraint}}) \]

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 our test datasets to calculate, 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 inputs. Simulated users picked a first point randomly then some
Figure 2: Experiment 1 – In this comparison between EigenScores and the quantity they estimate, each point in each graph represents a pair of data points from the appropriate dataset. The horizontal axis shows the actual amount the underlying distance function changes when a given pair of points is constrained together. The vertical axis shows the EigenScore for that pair of points.

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 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 the number of data points to be considered while maintaining near-optimal metric learning results.

9. REFERENCES


