Visualizations of Machine Learning Behavior with Dimensionality Reduction Techniques

Keywords: visualization, machine learning, dimensionality reduction

Bo Gao BO.GAO@CS.KULEUVEN.BE
K.U.Leuven - Department of Computer Science, Celestijnenlaan 200A, 3001 Leuven, Belgium

Joaquin Vanschoren JOAQUIN@LIACS.NL
Leiden University - LIACS, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands

Abstract

There are many different approaches to machine learning, and each approach has its own characteristics and behavior. In order to investigate the aspects of these approaches, large amounts of machine learning experiments with high dimensionality(data characteristics, algorithm characteristics, parameters settings, evaluation metrics, etc.) are generated and collected within databases, such as the Experiment Database. To enable the user to gain insight into this mass of meta-data about machine learning algorithms efficiently and effectively, different Dimensionality Reduction techniques are investigated. Based on this, a visualization tool is built to help users analyze the behavior of learning algorithms. The experiment results of these techniques on different meta-datasets are discussed in this paper.

1. Introduction

The Experiment Database (ExpDB) is a public database designed to collect and organize large numbers of past experiments donated by many researchers (Vanschoren et al., 2011). Each experiment may contain different algorithms, different parameter settings and different datasets, etc. We call the data that describe the details of the experiments meta-data. Unfortunately, the high dimensionality of the meta-dataset becomes a prohibiting factor for us to gain insights in the algorithms. Indeed, many trends may occur in a (curved) low-dimensional subspace (or manifold) of the data. In other words, the data points may lie close to a manifold of much lower dimensionality than that of the original data space (Bishop, 2006). As such, we look into different Dimensionality Reduction (DR) techniques which aim to map the data from a high dimensional space to a low dimensional one (typically 2D or 3D), so that an overview picture of the data is presented to the researcher. As the saying goes: a picture is worth a thousand words. Large amounts of data can be imprinted on a single image that is far more comprehensible. Furthermore, new knowledge can be perceived and discovered through visualizations. Therefore the researcher should be able to use the visualizations to analyse machine learning behavior efficiently.

Hence, we can summarize our research goal as twofold: First, we aim to investigate the relationships between datasets and machine learning algorithms. For instance, which parameter setting of which algorithm can achieve high performance (e.g. predictive accuracy) on what kind of datasets, or which type of learning algorithms (e.g. rule-based or decision trees or kernel methods) are more robust on certain datasets. We also hope that through visualizations, new relationships can be discovered. Second, we want to evaluate different state-of-the-art DR techniques in the context of machine learning meta-datasets.

In the remainder of this paper, we discuss the key implemented DR techniques in Section 2, and show the results of the evaluations and visualizations of the DR techniques on machine learning meta-dataset(s) in Section 3. Finally, in Section 4, we conclude.

Appearing in *Proceedings of the 20th Machine Learning conference of Belgium and The Netherlands.*

2. Dimensionality Reduction

In this section, we discuss a few representative DR techniques, including the traditional linear as well as several state-of-the-art non-linear ones. Afterwards, we give a brief overview of all implemented DR techniques. Because of the scope of this paper, we try to keep the mathematical equations to a minimum throughout the text, and emphasize on the concepts and intuitions behind these techniques.

A note on notations: given a dataset, we use N to denote the number of instances (rows), with i being the index for each instance ($i = 1, ..., N$). In the original high dimensional space, D denotes dimensionality, or the number of attributes (columns), and X_i denotes the i_{th} instance (row) of the dataset. In the low dimensional space, M denotes the number of attributes after mapping ($M < D$), and Y_i denotes the i_{th} instance of the mapped dataset.

2.1. Linear DR Techniques

2.1.1. PCA

Principal Component Analysis (PCA) can be formulated as the orthogonal projection of the original data onto the (linear) *principal subspace* such that the original variance in the data is maximally preserved. The algorithm performs eigenvalue decomposition on the $D \times D$ covariance matrix of the original dataset (Bishop, 2006). The obtained M eigenvectors with the largest eigenvalues are called the *principal components*(PC's) of the original dataset. The low dimensional data is the projection of the original data onto the PC's. PCA has several limitations. First, it assumes the linearity of data, whereas in real world, the intrinsic structure of the data can be so "curvy" that a "rigid" subspace (comprised of a set of orthogonal PC's) becomes an inadequate approximation. Second, it assumes a single Gaussian distribution of the data. Third, it focuses on preserving the largest variances, which makes the approximation vulnerable towards noise. Many variants of PCA have been proposed, such as Principal Curves (Hastie & Stuetzle, 1989), Probabilistic PCA (Tipping & Bishop, 1999), Kernel PCA (Schölkopf et al., 1998), etc. From these variants, only Kernel PCA is currently implemented in the visualization tool.

2.1.2. CLASSICAL SCALING

Classical Scaling (Cox & Cox, 1994) (CS) is a type of MDS (MultiDimensional Scaling). The aim of MDS is to preserve the pairwise distances as much as possible in the mapping. The low dimensional data can be derived via eigenvalue decomposition on the distance matrix. It is shown that CS is intrinsically the same as PCA (Van der Maaten et al., 2009). Like PCA, CS also assumes the data to be linear. It also ignores the neighborhood information of a data point and large distances are preserved better than small ones causing the detailed information among "close" data points is overlooked. Other (non-linear) types of MDS are proposed to address these weaknesses of CS, such as Isomap and Sammon Maps, which will be discussed in Section 2.2.3 and 2.2.6 respectively.

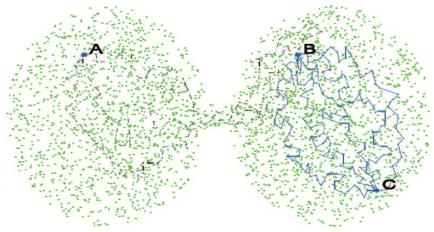


Figure 1. An illustration of Markov random walks in Diffusion Maps

2.2. Non-Linear DR Techniques

In order to find the intrinsic, often non-linear manifold embedded in the original data space, besides constructing a mixture of linear models (such as Probabilistic PCA), we have an alternative: to consider a single non-linear model. Many techniques have been invented or adapted in this way, yielding distinctive merits. We implemented the following representative state-of-the-art non-linear DR techniques in the visualization tool.

2.2.1. DIFFUSION MAPS

Diffusion Maps build a probabilistic model with pairwise *transition probabilities* to reflect the degree of closeness between each pair of data points (e.g. points A and B). The higher this probability is, the easier it becomes to "walk" from A to B (Lafon & Lee, 2006). As shown in Figure 1, the given set of data points (green dots) form two clusters, data point A is in the left cluster, B and C are in the right cluster. Since there are more paths between B and C than between B and A, it will be easier for B to "walk" to C than to A. Based on the transition probabilities, *diffusion distances* (Lafon & Lee, 2006) are defined between pairs of data points. Diffusion Maps aim to find a low dimensional representation in which the Euclidean distances between pairs of data points match the diffusion distances as well as possible. This is achieved by eigenvalue decomposition on the transition probability matrix $\mathbf{P^{(t)}}$ (Lafon & Lee, 2006).

2.2.2. KERNEL PCA

Kernel PCA(KPCA) is a non-linear extension of PCA, which transforms the original data into a higher-

dimensional feature space using a kernel function (Schölkopf et al., 1998) and performs regular PCA in the feature space to obtain "curved" principal components. The low dimensional representation can be found through the eigenvalue decomposition on the kernel matrix of the original data. The main limitation of KPCA is the selection of an appropriate kernel and the parameter configuration of that kernel. Different methods have been proposed to solve the kernel-selection problem, e.g. hold-out testing (Golub et al., 1979), semi-definite programming (Graepel, 2002), etc. Still, these methods are computationally expensive (Van der Maaten et al., 2009).

2.2.3. ISOMAP

Isometric Feature Mapping (Isomap) (Balasubramanian & Schwartz, 2002) is similar to Classical Scaling, but instead of a Euclidean distance matrix, Isomap uses a geodesic distance matrix. The geodesic distance between two data points is the accumulative distance of the shortest path between the two points found by Dijkstra's Algorithm (Dijkstra, 1959) on a graph. However, the geodesic distance is vulnerable towards "short-circuiting": where the data points that are far away from each other are taken as neighbors.

The neighborhood graph is constructed as follows. First, a Euclidean distance matrix is constructed as in Classical Scaling. Second, a neighborhood graph is constructed based on the Euclidean distances, in which only data points considered to be neighbors are connected, and each connection is assigned a weight. There are two approaches to find a data point A's neighbors: (1) If the Euclidean distance between A and a point B is smaller than a predefined threshold ε, then B is A's neighbor, and a connection is assigned to A and B (this is called ε-Isomap); (2) Rank the Euclidean distances between A and all the other data points, select the K nearest points as A's neighbors ($K < m-1$, this is called k-Isomap). We will call this k-Isomap in further discussion.

2.2.4. LLE

Locally Linear Embedding (LLE) (Roweis & Saul, 2000) also constructs a (k-nearest neighbor) graph representation of the manifold of the original data. But in contrast to Isomap, which uses geodesic distances to characterize the global geometry of the manifold, LLE focuses on preserving the local geometry. We can imagine that a data point and its k neighbors form a local plane: the data point would become the plane's topological center, and the plane is unique to the data point. A set of reconstruction weights is defined so that each data point X_i can be represented as a linear combination of its k neighbors (Eq.1).

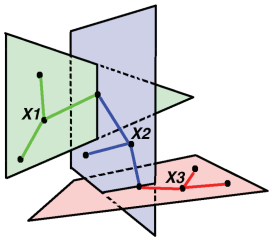


Figure 2. Example sub-planes in LLE, the data points X_1, X_2 and X_3 are the topological centers for each plane

$$X_i \approx w_{i,1}X_i^{(1)} + w_{i,2}X_i^{(2)} + ... + w_{i,k}X_i^{(k)} \quad (1)$$

with

$$\sum_{j=1}^{K} w_{i,j} = 1. \quad (2)$$

We call $w_{i,j}$ the reconstruction weights. An illustration is shown in Figure 2, in which three sub-planes in the original data are identified by the three data points X_1, X_2, X_3 and their respective neighbors.
The reconstruction weights are then derived by solving a linear system (Saul & Roweis, 2000) to minimize the cost function:

$$\phi_1(W) = \sum_{i=1}^{N} \left| X_i - \sum_{j=1}^{k} w_{i,j} X_i^{(j)} \right|^2 \quad (3)$$

Because the reconstruction weights are invariant to translation, rotation and rescaling, we have:

$$\phi_2(Y) = \sum_{i=1}^{N} \left| Y_i - \sum_{j=1}^{k} w_{i,j} Y_i^{(j)} \right|^2 \quad (4)$$

The low dimensional representation is derived via the eigenvalue decomposition on the sparse matrix (Roweis & Saul, 2000): $(I-W)^T(I-W)$ to minimize the cost function (Eq.4), I is an $N \times N$ identity matrix.

2.2.5. LAPLACIAN EIGENMAPS

Laplacian Eigenmaps are similar to LLE in the sense that they both try to preserve the local geometrical properties of the manifold of data and they both use the sparse weight matrix based on a neighborhood graph. After the neighborhood graph is constructed, which is the same as that in Isomap and LLE, a weight matrix is directly computed based on a kernel function (E.g. a Gaussian kernel in Eq.5):

$$w_{i,j} = \begin{cases} e^{-\frac{|X_i - X_j|^2}{2\sigma^2}} & if\ neighbor(X_i, X_j) \\ 0 & else \end{cases} \quad (5)$$

A cost function (an sum of the weighted distances between a data point and its k nearest neighbors) is then minimized:

$$\phi(Y) = \sum_{i=1}^{N}\sum_{j=1}^{N} \left(|Y_i - Y_j|^2 w_{i,j} \right) \quad (6)$$

The Gaussian kernel function emphasizes the small distances between data points more than the large distances. In other words, the closer neighbors contribute more to the cost function than the farther ones. The cost function can be minimized through a eigenvalue decomposition on the matrix $D^{-1}L$, with D being the diagonal matrix $D_{i,i} = \sum_{j=1}^{N} w_{i,j}$, W being the sparse weight matrix and $L = D - W$ (Belkin & Niyogi, 2001). The neighborhood graph-based approaches have several limitations: the construction of the neighborhood graph is susceptible to overfitting, and the local linearity assumption is susceptible to discontinuities in the manifold of data.

2.2.6. SAMMON MAPS

Sammon Maps is another type of MDS. In contrast to the DR techniques discussed previously, Sammon Maps (Sammon, 1969) do not perform eigenvalue decomposition on a (transformed) proximity matrix to minimize a cost function and find the low dimensional coordinates of the original data. Instead, it tries to find the low dimensional mapping through an iterative process.

$$\phi(Y) = \frac{1}{\sum_{i<j} d(X_i, X_j)} \sum_{i<j} \frac{(d(X_i, X_j) - d(Y_i, Y_j))^2}{d(X_i, X_j)} \tag{7}$$

In each iteration, the error (Eq.8) is computed:

$$e_{i,j} = \gamma \frac{(d(X_i, X_j) - d(Y_i, Y_j))}{d(Y_i, Y_j)} (Y_i - Y_j) \tag{8}$$

The low dimensional coordinates are updated iteratively:

$$Y_i^{new} = Y_i + e_{i,j} \tag{9}$$

$$Y_j^{new} = Y_j - e_{i,j} \tag{10}$$

with $\gamma > 0, i, j \in [1, N]$, γ is the learning rate.

Being a non-spectral technique, Sammon Maps are less susceptible to high dimensionality. The Sammon cost function is shown in Eq.7, which is similar to that of Classical Scaling. However, one possible limitation of Sammon Maps is weight $1/d(X_i, X_j)$ can lead to overfitting.

2.3. An Overview of the DR techniques

As illustrated in Figure 3: from the eight DR techniques we have discussed, PCA and Classical Scaling are linear, the rest are non-linear. Among the non-linear ones, Kernel PCA, Diffusion Maps and Laplacian Eigenmaps (LE) utilize kernel functions. LE, LLE and Isomap are all based on neighborhood graphs, in which LE and LLE use a sparse proximity matrix, while Isomap uses a full one. We can also distinguish Sammon Maps from the others by its non-spectrality.

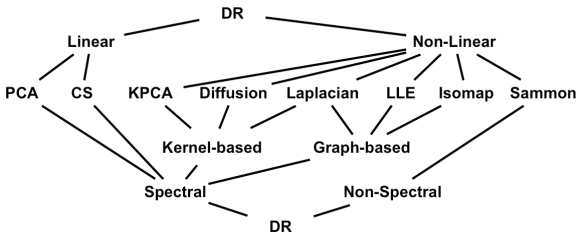


Figure 3. Categorization of DR techniques.

Table 1. The attribute description of ***bagging***.

Name	Description
1.nr_iterations	number of bagging iterations
2.baseLearner	base learner used
3.dataset	name of the dataset
4.classCount	number of classes
5.nr_attributes	number of the attributes
6.nr_sym_attributes	number of symbolic attributes
7.nr_num_attributes	number of numeric attributes
8.lm_naiveBayes	performance of naive bayes landmarker
9.lm_1nn	performance of 1-nearest neighbor landmarker
10.lm_1rule	performance of 1-rule landmarker
11.nr_examples	number of data points
12.nr_missingValues	number of missing values
13.class_entropy	entropy of the class attribute
14.default_accuracy	default accuracy of the dataset
15.evaluation	predictive accuracy of bagging

3. Experiments

In this section, we apply the eight DR techniques with different parameter settings to two machine learning meta-datasets: ***bagging*** and ***algo-performance***, and discuss the results.

3.1. Datasets

bagging contains 1437 runs of the standard Bagging algorithm, with different parameter settings and base-learners, on different classification UCI[1] datasets. The attributes of the dataset are summarized in Table 1. Note that landmarkers are simplified algorithms used to characterize a dataset (Pfahringer et al., 2000). Attributes 2 and 3 are categorical, and are used as color labels in the visualizations, the rest are numerical, which serve as the input to DR techniques.

algo-performance contains the predictive accuracies of 293 algorithm-parameter combinations on 83 different UCI datasets (D=83). E.g. combination SVM-C-1.0-Polynomial-E-3 is a support vector machine with complexity constant 1.0 and a polynomial kernel with power 3. Each row is a different algorithm-parameter combination (all the algorithms are from Weka[2]) and each column is the performance on a specific dataset.

[1] http://archive.ics.uci.edu/ml/
[2] http://www.cs.waikato.ac.nz/ml/weka/

3.2. Evaluations

3.2.1. TRUSTWORTHINESS AND CONTINUITY

When we reduce the dimensionality of the data, the topological properties of the data in the original space will not be completely preserved, which leads to distortions. There are two types of distortions: (1) Data points that are originally far away from each other are mapped close to each other in the low dimensional space. (2) Data points that are originally close to each other are mapped far away instead. In order to measure the quality of a dimensionality reduction, we use Trustworthiness and Continuity (Venna & Kaski, 2006) to characterize the first and the second distortion respectively. Trustworthiness is defined as follows:

$$Trustworthiness\,(k) = 1 - A\,(k) \sum_{i=1}^{N} \sum_{j \in U_k} (r_{i,j} - k) \quad (11)$$

with

$$A\,(k) = \begin{cases} \frac{2}{N \cdot K \cdot (2N - 3k - 1)} & if\ k < \frac{N}{2} \\ \frac{2}{N \cdot (N-k) \cdot (N-k-1)} & if\ k \geq \frac{N}{2} \end{cases} \quad (12)$$

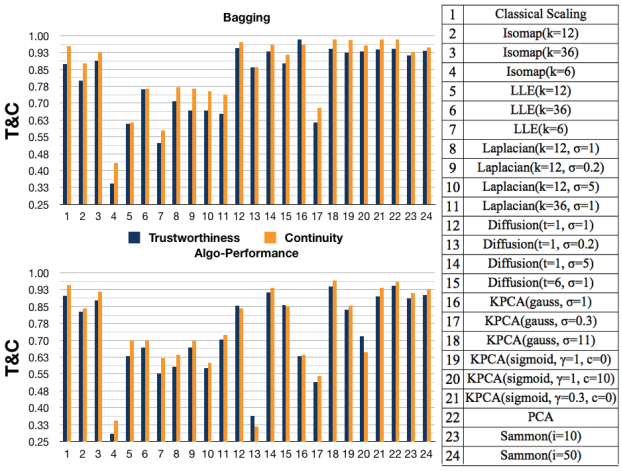
In which N is the number of data points, k is the predefined number of neighboring points, and $r_{i,j}$ is the rank of the data point j according to the data point i in the original space: the closer j is to i, the lower $r_{i,j}$ will be. The ranks are natural numbers. U_k is the set of data points that are within the k-nearest neighbors of data point i in the low dimensional space but do not appear in the original space. $A(k)$ is a scaling factor that scales the second term, i.e. the error term. The more data points with high ranks (the data points that are far away from each other) are wrongly mapped close to each other in the low dimensional space, the larger the error term will be. Thus, the trustworthiness ranges from 0 to 1, 1 means completely trustworthy, 0 means completely untrustworthy. Similarly, Continuity is defined as following:

$$\text{Continuity}\,(k) = 1 - A\,(k) \sum_{i=1}^{N} \sum_{j \in V_k} (\hat{r}_{i,j} - k) \quad (13)$$

$\hat{r}_{i,j}$ is the rank of data point j according to data point i in the low dimensional space: the closer j is to i, the lower $\hat{r}_{i,j}$ will be. V_k is the set of data points that are within the k-nearest neighbors of data point i in the original space but do not appear in the low dimensional space. When data points are neighbors in the original space but not in the visualization, this will increase the error. Continuity also ranges from 0 to 1, a larger number means a better continuity.

3.2.2. RESULTS

For each dataset, the Trustworthiness(T) and Continuity(C) with 6 different k-values[3] are measured for each DR technique. The T and C scores are then computed for each dataset. The resulting graphs are shown in Figure 4: the table on the right denotes the indices of the DR techniques with their particular parameter settings. We use Gaussian kernels in Diffusion Maps, Kernel PCA and Laplacian Eigenmaps, and also use Sigmoid kernels in Kernel PCA for comparison. As we can see in Fig.4, for ***bagging***, DR No.12 has relatively high Continuity, though not the best, and it has high Trustworthiness comparing with other DR with high Continuity. Also note that DR No.16, 21 and 22 are also very good visualization candidates on ***bagging***. For ***algo-performance***, DR No.22 achieves the highest Trustworthiness and Continuity. As such, we will discuss the visualizations of DR12: Diffusion Maps ($t = 1, \sigma = 1$) and DR22: PCA, i.e. 22 on the two datasets respectively. The visualizations of KPCA(gauss, $\sigma = 1$) on ***bagging*** and Sammon Maps($i = 50$) on ***algo-performance*** are also shown in section 3.3.3 to give the reader an impression of other DR techniques. Due to the space limitations, we will not discuss the latter two visualizations in detail.



1	Classical Scaling
2	Isomap(k=12)
3	Isomap(k=36)
4	Isomap(k=6)
5	LLE(k=12)
6	LLE(k=36)
7	LLE(k=6)
8	Laplacian(k=12, σ=1)
9	Laplacian(k=12, σ=0.2)
10	Laplacian(k=12, σ=5)
11	Laplacian(k=36, σ=1)
12	Diffusion(t=1, σ=1)
13	Diffusion(t=1, σ=0.2)
14	Diffusion(t=1, σ=5)
15	Diffusion(t=6, σ=1)
16	KPCA(gauss, σ=1)
17	KPCA(gauss, σ=0.3)
18	KPCA(gauss, σ=11)
19	KPCA(sigmoid, γ=1, c=0)
20	KPCA(sigmoid, γ=1, c=10)
21	KPCA(sigmoid, γ=0.3, c=0)
22	PCA
23	Sammon(i=10)
24	Sammon(i=50)

Figure 4. The experiment results of different DR techniques on the machine learning meta-datasets.

3.3. Visualizations

3.3.1. ON THE *BAGGING* DATASET

First, we apply the diffusion maps technique on the *bagging* dataset. The result is shown in Figure 5. In each sub-figure, a different dimension is used to color the data points. In Figure 5.2, we color the three base learners IBk, J48 and Random Tree with red, green and blue color respectively. For categorical values, each category gets a unique color, and for numerical values, points are colored ranging from red (high) to blue (low).

It is interesting to see which attributes correlate with

[3] k is selected so that three are smaller than $N/2$ and the other three are larger than $N/2$, N being the number of instances in the dataset.

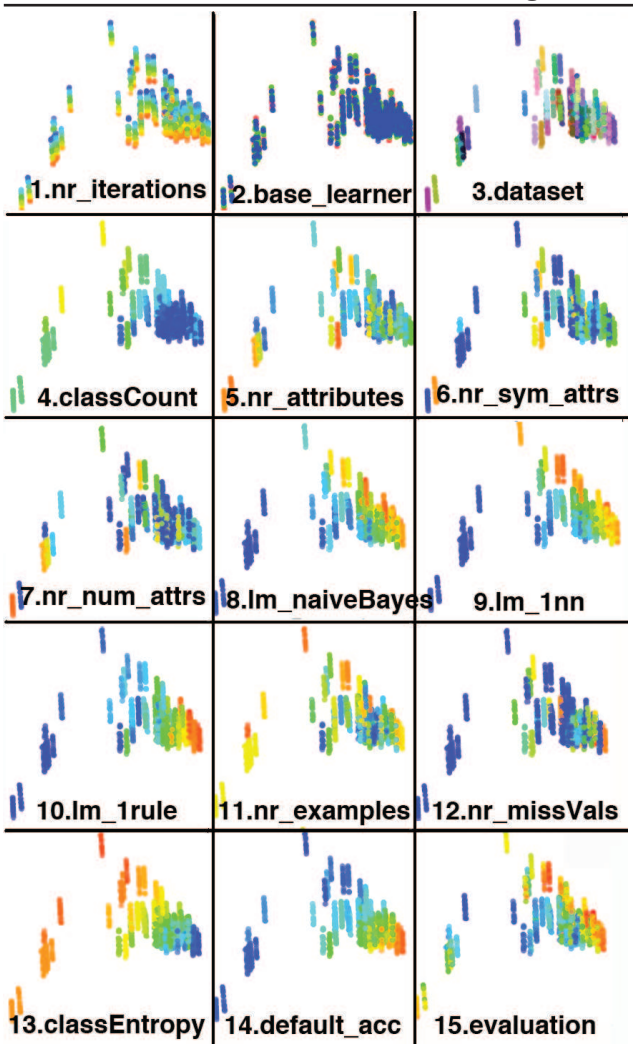


Figure 5. Diffusion Maps ($\sigma = 1, t = 1$) on the *bagging* dataset, colored based on the 15 attributes. For categorical values, same category has the same color which is distinguishable from other categories. For numerical values, color range "red..orange..yellow..green..blue" indicate a series of values from being high to being low.

the dimensions generated by diffusion mapping. The Y-axis correlates with the number of bagging iterations (Figure 5.1), while the X-axis correlates with the landmarking results (Figure 5.8-10). The latter roughly indicate how easily the data can be modeled with simple classifiers. Since the data can be most accurately mapped to these two dimensions, they have a large impact on the distribution of the data.

When looking at the actual clusters that formed, it is clear that the data is clustered by dataset. This is not surprising, since the data contains a lot of attributes for dataset characteristics. Figure 6 shows these clusters in more detail, with labels for several datasets. Still, we also see more general clusters, which seem to correlate with the class count (Figure 5.4) and class entropy (Figure 5.13). For instance, there is one dense cluster on the right for datasets with few classes.

Another interesting discovery is that the landmarkers seem to correlate well with the final evaluation of the experiment, but only on the generally 'easier' datasets on the right. On the leftmost datasets, landmarkers perform badly (the blue dots in Figure 5.8-10), while the evaluation of the complete learning algorithms is generally good. Especially the fact that these are datasets with a low default accuracy, high class entropy and higher class count leads us to believe that the "simple" models generated by landmarkers are just too simple to capture the structure in the data, even if complete versions of the algorithm can do this. Also note that the different landmarkers disagree on the top-most datasets (Figure 5.8-10), which tend to have many classes. Clearly, some landmarkers are more robust to many classes than others.

When looking at Figure 5.2, we see that diffusion mapping does not separate the different base learners. This is not surprising as there are no numerical attributes that describe the base-learning algorithm used. Measurable numerical properties of the learning algorithms should be added to the data so that they may be included in the diffusion mapping.

The evaluations on the same datasets tend to have similar scores: we can see them forming small clusters in the plot as shown in Figure 6. Still, some clusters (datasets) show a vertical gradient, indicating the effect of the number of iterations. The effect is not as pronounced as we expected, but this is probably due to the fact the results for different base-learners overlap with each other. Also, on some datasets the Bagging algorithms can achieve particularly high or low scores, e.g. the dataset Sick always has high evaluations with the Bagging algorithms, whereas the dataset Abalone always has low evaluations.

Finally, all this shows that dimensionality reduction techniques are extremely useful to study the performance of algorithms under many different variables: different parameter settings, different datasets, and

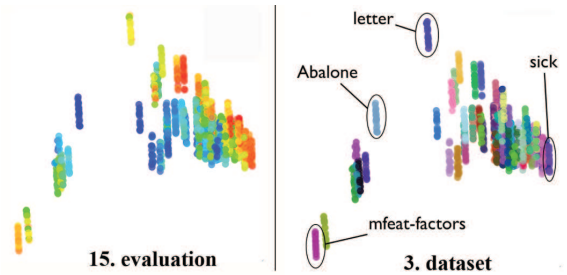


Figure 6. Diffusion Maps ($\sigma = 1, t = 1$) on the *bagging* dataset, a detailed illustration.

different properties of the algorithms and datasets. The high number of attributes can successfully be reduced to a representation which retains most of the information and immediately allows us to look for interesting patterns on sight.

3.3.2. ON THE *ALGO-PERFORMANCE* DATASET

Second, we apply the PCA technique on the *algo-performance* dataset.

The result is shown in Figure 7. In Figure 7.1, we can find some outliers such as "'Raced-Incremental-Logit-Boost-default" and "Bagging-I-1..25-ZeroR", which are far away from the "big cluster". This indicates that these algorithm-parameter combinations perform very differently from their peers on the given datasets. Also, when combining with Figure 7.2 and 7.3, we discover that the outliers perform poorly on the *letter* and *anneal* datasets, since the dots representing them are colored blue. On the other hand, the combinations in the"big cluster" generally have a high performance on the two datasets. One step further, when coloring the algorithm-paramenter combinations based on their general categorization (the combinations belonging to SMO (Support Vector Machines) are blue, the ones belonging to Bagging are green and the others are red), we can see that SMO methods mostly occupy the top of the visualization, and looking back to Figure 7.2 and 7.3, we discover that SMO methods perform very well on the *letter* dataset but very poorly on the *anneal* dataset.

Finally, let's look at the small clusters formed in the outliers. With the help of the interactive function in the visualization tool, we see that the Bagging algorithms with the base learners HyperPipe and ConjunctiveRule form their own small clusters in the outliers, whose performance is poor on the given datasets. We also discover that the Bagging algorithms with the base learners Multi-layer Perceptron (MLP) and J48 have much better performance. Many more patterns can be discovered in the visualization.

3.3.3. OTHER DR VISUALIZATIONS ON *BAGGING* AND *ALGO-PERFORMANCE*

From Figure 8 we see a similar distribution of points as Figure 5. The points are vertically aligned according to the number of iterations of bagging algorithms, and horizontally aligned according to the general 'difficulty' of the dataset.

From Figure 9 we see a similar distribution of points as Figure 7. The major cluster as well as small peripheral clusters can be identified.

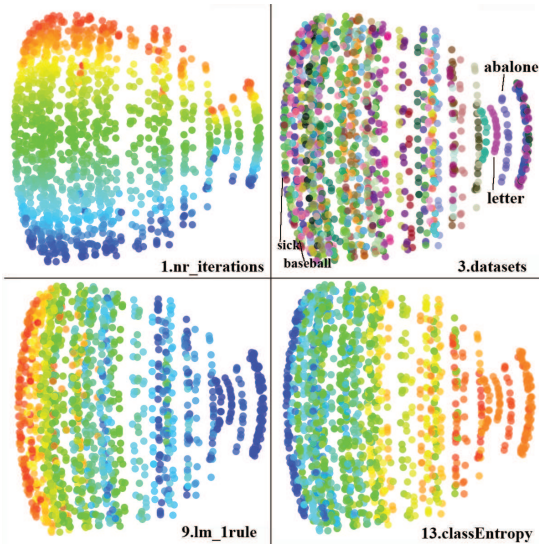


Figure 8. KPCA (gauss, $\sigma = 1$) on the *bagging* dataset, colored based on attributes 1, 3, 9 and 13.

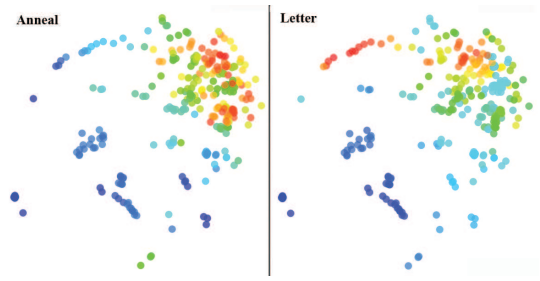


Figure 9. Sammon Maps ($i = 1$) on the *algo-performance* dataset, colored based on attributes 'anneal' and 'letter'.

4. Conclusions

In this paper, we investigated different dimensionality reduction (DR) techniques for visualizing high dimensional meta-data on machine learning algorithms, retrieved from the Experiment Database (ExpDB). First, we provided an overview of interesting DR techniques. Next, we evaluated these DR techniques on trustworthiness and continuity to select the adequate techniques for visualizing two different sets of meta-data.

We observed that the same DR techniques with different parameter settings can lead to significantly different results, such as Isomap with k = 6 and k = 36. Kernel and/or probabilistic techniques (such as Kernel PCA and Diffusion Maps) achieve more satisfying results than graph-based techniques (such as LLE and Isomap) on the two meta-datasets. Especially sparse graph-based techniques, such as LLE, do not perform well. Indeed, in order to achieve high performance with kernel-based techniques, the type of the kernel and the corresponding parameters have to

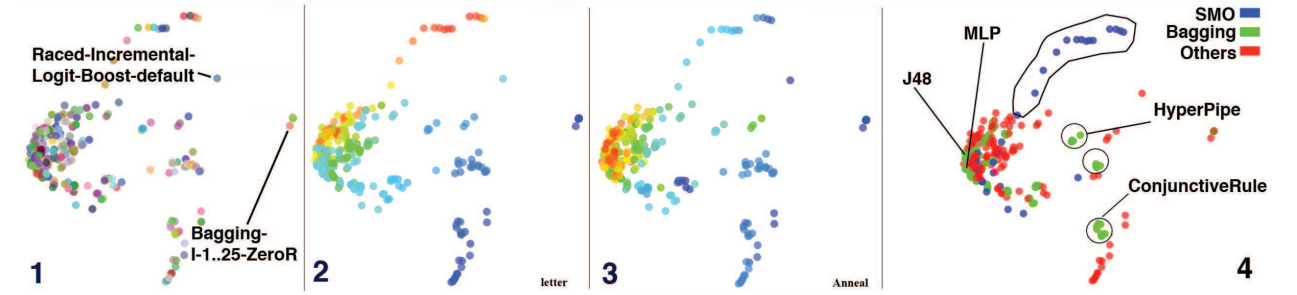


Figure 7. PCA on the *bagging* dataset. The first three sub-figures are colored based on (1) the categorical attribute "dataset", the performance of all the algorithm-parameter combinations on (2) the *letter* dataset and (3) the *anneal* dataset. The forth sub-figure is colored based on a general categorization of the algorithm-parameter combinations.

be adjusted several times manually, which is rather inconvenient. Also, we find that linear DR techniques have stable (and sometimes even better) performance compared to their non-linear peers. Non-spectral techniques (namely Sammon Maps) generally give good results. Furthermore, similar to kernel-based techniques, the other nonlinear DR techniques require parameter optimization.

Determining which technique is preferable depends not only on parameter configuration, but also the given datasets. Selecting or synthesizing appropriate techniques based on a given dataset is an interesting direction to look into in future work. For example, one could build a model that selects an appropriate DR technique with appropriate parameter(s) for a given dataset based on Trustworthiness and Continuity. Furthermore, other DR techniques, especially deep-structured ones are worth investigation. Next, we could use DR techniques to compare how algorithm performance on the UCI 'benchmark datasets' compare to performance results on real-world datasets. We can also store the results of the experiments of DR on machine learning meta-data (let's call them DR meta-data to distinguish with ML meta-data) into a database (e.g. ExpDB) so that we can investigate their performance in detail.

Finally, the produced DR visualization tools is of course also applicable for any other high dimensional data.

References

Balasubramanian, M., & Schwartz, E. L. (2002). The Isomap Algorithm and Topological Stability. *Science*, *295*, 7.

Belkin, M., & Niyogi, P. (2001). Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in Neural Information Processing Systems 14* (pp. 585–591). MIT Press.

Bishop, C. M. (2006). *Pattern recognition and machine learning (information science and statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Cox, T., & Cox, M. (1994). *Multidimensional scaling*. London: Chapman & Hall.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*, 269–271. 10.1007/BF01386390.

Golub, G., Heath, H., & Wahba, G. (1979). Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, *21*, 215–224.

Hastie, T., & Stuetzle, W. (1989). Principal curves. *Journal of the American Statistical Association*, *84*, 502–516.

Lafon, S., & Lee, A. (2006). Diffusion maps and coarse-graining: a unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *28*, 1393 –1403.

Pfahringer, B., Bensusan, H., & Giraud-Carrier, C. (2000). Meta-learning by landmarking various learning algorithms. *In Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 743–750). Morgan Kaufmann.

Roweis, S., & Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*, 2323–2326.

Sammon, J. W. (1969). A nonlinear mapping for data structure analysis. *IEEE Trans. Comput.*, *18*, 401–409.

Saul, L. K., & Roweis, S. T. (2000). *An introduction to locally linear embedding* (Technical Report).

Schölkopf, B., Smola, A., & Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.*, *10*, 1299–1319.

Tipping, M. E., & Bishop, C. M. (1999). Mixtures of probabilistic principal component analysers. *Neural Computation*, *11*, 443–482.

Van der Maaten, L., Postma, E., & Van den Herik, J. (2009). Dimensionality reduction: A comparative review. *TiCC-TR 2009-005*.

Vanschoren, J., Blockeel, H., Pfahringer, B., & Holmes, G. (2011). Experiment databases: A new way to share, organize and learn from experiments. *Machine Learning*.

Venna, J., & Kaski, S. (2006). Local multidimensional scaling. *Neural Networks*, *19*, 889–899.