Unsupervised Outlier Detection and Semi-Supervised Learning

Adam Vinueza
Department of Computer Science
University of Colorado
Boulder, Colorado 80302
vinueza@colorado.edu

Gregory Z. Grudic
Department of Computer Science
University of Colorado
Boulder, CO 80302
grudic@cs.colorado.edu

Abstract

A familiar problem in machine learning is to determine which data points are outliers when the underlying distribution is unknown. In this paper, we adapt a simple algorithm from Zhou *et al*[3], designed for semi-supervised learning, and show that it not only can automatically detect outliers by using local and global consistency of data points, but also automatically select optimal learning parameters, as well as predict class outliers for points introduced after training.

1 Introduction

Given a set of data points drawn from some probability distribution, an *outlier* with respect to that distribution is an unlikely point. A familiar problem in machine learning is to determine which data points are outliers when the underlying distribution is unknown. In this paper, we examine a simple algorithm from Zhou *et al*[3], designed for semi-supervised learning, and show that it also can automatically detect outliers by using local and global consistency of data points. This algorithm has several notable features: (1) it performs at least as well as the best known outlier-detection algorithms; (2) it allows for the *automatic* selection of optimal learning parameters; (3) it predicts class outliers; and (4) it predicts class outliers for *points introduced after training*.

2 Algorithm

2.1 The Original Semi-Supervised Learning Algorithm

Let X be a set $\{x_1, \ldots, x_n\} \subset \mathbb{R}^m$ of data points, and let L be a set $\{1, \ldots, c\}$ of labels. Let the first $l < n$ points in X be labeled, the rest unlabeled. We wish to predict the labels of the points $\{x_{l+1}, \ldots, x_n\}$. To this end, consider a nonnegative $n \times c$ matrix F. By assigning a point x_i to class j just in case $\arg\max_{j \leq c} = F_{ij}$, we obtain a classification on X. The original algorithm of Zhou *et al* [3] makes use of X and knowledge of the labels of $x_1, \ldots, x_l$, with the aim of finding an F that gives us a good prediction of the labels of the unlabeled points in X.

The algorithm works as follows. Construct an affinity matrix W such that $W_{ij} = \exp(-\|x_i - x_j\|^2/2\sigma^2)$ if $i \neq j$ and $W_{ii} = 0$. Now, let D be the diagonal matrix such

that D_{ii} is the sum of the i-th row of W: then $D^{-1/2}WD^{-1/2}$ normalizes the rows of W. Finally, let Y be the $n \times c$ matrix such that $Y_{ij} = 1$ if x_i has label j, and $Y_{ij} = 0$ otherwise.

We may now define the cost function:

$$Q(F) = \frac{1}{2}\left(\sum_{i,j=1}^{n} W_{ij}\left\|\frac{1}{\sqrt{D}_{ii}}F_i - \frac{1}{\sqrt{D}_{jj}}F_j\right\|^2 + \mu\sum_{i=1}^{n}\|F_i - Y_i\|^2\right), \tag{1}$$

where F_i is the i-th row of F and $\mu > 0$. Then for variable candidate matrices F, our classifying function is:

$$F^* = \arg\min_F Q(F). \tag{2}$$

The first term term in $Q(F)$ maintains local consistency by constraining classification of nearby points to not change too much (recall that W_{ij} encodes nearness of x_i and x_j). The second term constrains classification to not stray too far from the initial assignment of labels. The algorithm tries to maintain global consistency by balancing between the two terms, as $Q(F)$ is punished by points differing in class from nearby initially labeled ones: the initially labeled points serve as reference anchors for classification. And we respect the fact that these constraints may compete by incorporating the weighting parameter μ.

It can be shown by differentiating $Q(F)$ with respect to F and doing some algebraic manipulation that

$$F^* = \beta\left(I - \alpha S\right)^{-1} Y, \tag{3}$$

where $\alpha = 1/(1+\mu)$ and $\beta = \mu/(1+\mu)$: F^*, then, is the matrix that allows for a good classification on X. Several experiments in [3] show that this algorithm yields good classifications on data sets.

In [4], this algorithm is applied to the task of determining a weighted relevance metric that respects both local and global consistency. In particular, it is applied to show that it yields the same ranking list as Google's PageRank algorithm. Notably, queries and pages are represented as vectors, and a query plays the role of an initially labeled point. What makes the analogy between queries and initially labeled points notable is that, in $Q(F)$, labelings constrain the final assignment of labels to points. Looking at labelings in this way suggests a natural use of the algorithm to detect outliers: if labeled points are regarded as paradigmatic instances of a class, and we construct an empirical cdf of the values in $F*$ that lead to classifications, those points that lie below a given threshold can be considered to be outliers. This is what we see in Figure 1:

Thus, points distant from labeled points are more prone to being identified as outliers. A significant disadvantage of this approach that it makes outlierness relative to our choice of points to label. If we are to use this algorithm to identify outliers objectively, we need to somehow separate the main work of the algorithm from the initial assignment of labels. In the next section, we see how this is done.

2.2 Clustering with Local and Global Consistency

From equation (3), it is evident that the solution to the semi-supervised learning problem only depends on the labels after the the matrix $(I - \alpha S)$ has been inverted. This matrix only contains the training data inputs, $\{x_1, ..., x_n\}$, and it is this property that we will exploit to derive our clustering algorithm. We define a matrix U as:

$$U = \beta\left(I - \alpha S\right)^{-1} = \left[u_1^T, ..., u_n^T\right] \tag{4}$$

and note that U defines a graph or diffusion kernel (as described in [1], [2]). In addition, the columns of U, denoted by u_i^T, define distances between training points on these graphs, which can be interpreted as distances along a manifold [4]. The ordering of these distances

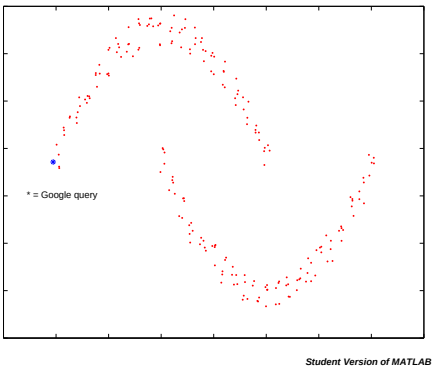


Figure 1: Sample Google Query Graph

along each manifold is maintained independent of scaling. From U, we create a new matrix V, by scaling the columns of U to have unit length. We define this V matrix as:

$$V = \left[u_1^T \left\|u_1^T\right\|^{-1}, ..., u_n^T \left\|u_1^T\right\|^{-1}\right] = \left[v_1^T, ..., v_n^T\right] \tag{5}$$

Note that, by definition, $||v_i|| = 1$.

Based on this column normalized matrix V, we define the proposed version of semi-supervised clustering:

$$F_V^* = VY = \left[f_{V1}^T, ..., f_{Vn}^T\right]$$

Finally, we further normalize the columns of F_V^* to give:

$$G^* = \left[f_{V1}^T \left\|f_{V1}^T\right\|^{-1}, ..., f_{Vn}^T \left\|f_{Vn}^T\right\|^{-1}\right]$$

A class label is assigned to point x_i as:

$$y_i = \arg\max_{j \leq c} G_{ij}^*$$

where G_{ij}^* is the (i, j) element of G^*.

2.3 Semi-Supervised Model Selection via Optimization

We define a distance (along a manifold specified by U) between points x_i and x_j to be:

$$d_M\left(x_i, x_j\right) = 1 - v_i v_j^T \tag{6}$$

The intuition behind this distance measure is that two points on a manifold are identically if the order of distances between all other points in the training set is identically, and the relative distances are identical. If this is the case for points x_i and x_j, then $d_M(x_i, x_j) = 0$. Conversely, the if point x_i have completely different distances along U to other points in the training data than point x_j, then $d_M(x_i, x_j)$ will approach 1. This leads to our definition of a distance matrix:

$$D_M = 1 - \begin{pmatrix} v_1 v_1^T & \dots & v_1 v_n^T \\ \vdots & \ddots & \vdots \\ v_n v_1^T & \cdots & v_n v_n^T \end{pmatrix} = \begin{pmatrix} d_M(x_1, x_1) & \dots & d_M(x_1, x_n) \\ \vdots & \ddots & \vdots \\ d_M(x_n, x_1) & \cdots & d_M(x_n, x_n) \end{pmatrix} \quad (7)$$

Next, let $\mathbf{p}_j$ be the set of points that belong to class j. Using matrix D_m we can define the mean distance between points in class j as:

$$\overline{D}_M^{jj} = E\left[D_M(\mathbf{p}_j, \mathbf{p}_j)\right]$$

where $D_M(\mathbf{p}_j, \mathbf{p}_j)$ denotes all entries of D_M corresponding to columns and rows of points $\mathbf{p}_j$ and $E[]$ is the average value of these. Similarly, The mean distance between points in class j and points in class k is give by:

$$\overline{D}_M^{jk} = E\left[D_M(\mathbf{p}_j, \mathbf{p}_k)\right]$$

Given that our goal is to find semi-supervised models that maximize the distances between points in different classes, while minimizing the distances between points in the same class, we can now state the optimization problem we are solving. Specifically,

$$\Omega = \max_{\alpha,\sigma} \left[E\left[\overline{D}_M^{jk}\right]_{\left\{\substack{k=1,\dots,c \\ j=1,\dots,c \\ i\neq j}\right\}} - E\left[\overline{D}_M^{jj}\right]_{\{j=1,\dots,c\}} \right] \quad (8)$$

2.4 Outlier Detection

We define a cluster independent outlier point to be one that is, on average, furthest away to all other points. This can be directly calculated from equation (7) by taking the average of the columns of D_M as follows and defining a outlier cluster independent vector O_d as follows:

$$O_d = \frac{1}{n}\left[\sum D_{M1}^T, ..., \sum D_{Mn}^T\right] = [O_{d1}, ..., O_{dn}]$$

where the element O_{di} is the average distance (in manifold space) between point x_i and all the other points and $D_M = \left[D_{M1}^T, ..., D_{Mn}^T\right]$. Thus by ordering the vales of O_{di} in increasing order, we order the points from furthest to closest, and the points appearing first in the list constitute the outliers.

Similarly, we can find outliers within a cluster j by looking at the $D_M^{jj} = D_M(\mathbf{p}_j, \mathbf{p}_j)$ matrix defined above. Specifically, we obtain an outlier O_d^j vector for cluster j as follows: $O_d^j = \frac{1}{n}\left[\sum D_{M1}^{jjT}, ..., \sum D_{Mn}^{jjT}\right] = \left[O_{d1}^j, ..., O_{dn}^j\right]$ where O_{di}^j is the mean distance of x_j to all other points in its cluster. Thus the point which has maximum O_{di}^j is the one which is *most inside* the cluster, while the point that has minimum O_{di}^j is *most outside* of the cluster.

2.5 Classifying New Points

In order to cluster a new point without adding it to S and re-inverting the matrix $(I - \alpha S)$, we once more use the property that two points are similar if they have similar distances to

all other points. However, this time we measure similarity using the S matrix as follows. Given a point x_k, we calculate $W_{kj} = exp(-\|x_k - x_j\|^2/(2\sigma^2))$, for $j = 1, ...n$ and obtain a vector W_k. We then calculate the $D_k = \sum_{j=1}^{n} W_k(j)$ and compute the vector in the S matrix that is associated with x_k), as $S_k = D_k^{-1/2} W D^{-1/2}$. Finally we normalize S_k to have length 1 and call it S_k^1 and similarly normalize the rows of S to also have length 1, denoting this matrix by S^1. We then obtain a set of coefficients $\Theta = (\theta_1, ..., \theta_n)^T = S^1(S_k^1)^T$. This vector has the property that if $x_k = x_i$, then $\theta_i == 1$, but if x_k is very far away from x_i then θ_i will approach zero. Therefore, θ_i measures the closeness of x_k to x_i in S matrix space (with $\theta_i = 1$ being really close and $\theta_i = 0$ really far). We use this property to assign x_k to a cluster by creating an $F_k = [v_1\Theta^T, ..., v_n\Theta^T]$ and assigning $y_c = \arg\max_{j \leq c} F_k$.

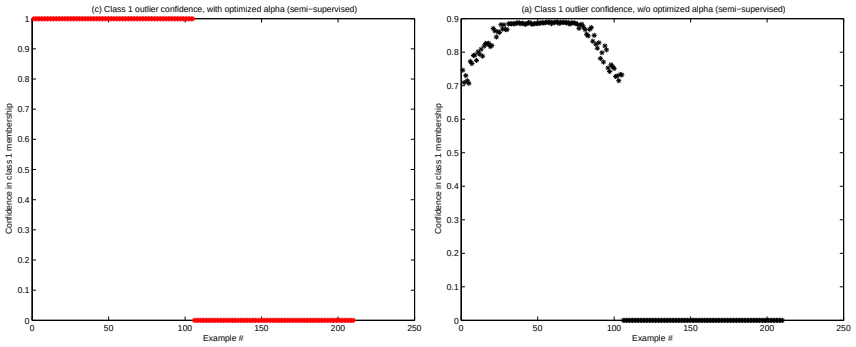
3 Experiments

We experimented on the Two Moons data set, the USPS digits data set, the 20 newsgroups data set, and the 6-class synthetic data set. For each of the data sets, we ran two sets of experiments: those that optimized both α and σ, and those with α preset to 0.99.

3.1 Two Moons Data

Figure 2 depicts the toy data set consisting of points generated into two clusters of intertwining crescent moons. We wish to classify points in each moon in the same way, and classify points in different moons differently. Also, if we look at the shape of the clusters, it is evident that the further a point is from the center of the thickest part of each cluster, the better candidate for outlierness it is.

The graphs in Figure 3 illustrate how our algorithm performs on this data set. Points 1 to 100 are in the top moon; points 101 to 200 are in the bottom. Moreover, points in a class are ordered by their value along the x-axis. When α is optimized, it is impossible to see in 3a any difference in relative outlierness between points in a given class; relative outlierness can be seen more clearly in 3b . The shape of the graph in that area mirrors the shapes of the two moons, which is what we should expect.



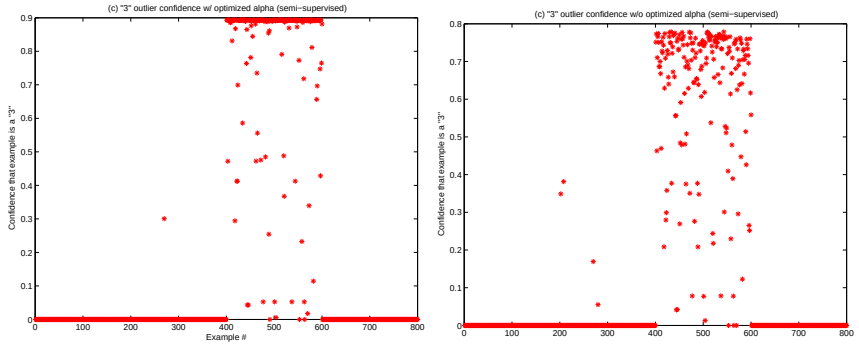
a) Optimized b) Unoptimized

Figure 2: Class 1 Outlier Confidence

3.2 USPS Digits Data

In this set of experiments, we attempted to classify representations of handwritten digits, using the USPS handwritten 16×16 digits dataset. Our four classes comprised the digits

1, 2, 3, and 4. We used 800 examples, 200 per class. We ran the experiments with only one labeled point from each class. The original algorithm run on this data set had an error rate of 9.38 percent with unoptimized α and an error rate of 6 percent with optimized α; by contrast, our algorithm, without optimizing for α, had an error rates of 3 percent without and 1.25 percent with optimized α. This is an encouraging sign that our algorithm can reliably select a good model with very few labeled points.



a) Optimized b) Unoptimized

Figure 3: USPS Outliers for Digit 3

Figure 3 shows the algorithm's confidence in class outlierness, given for each class, when optimized or unoptimized, and for both clustering and semi-supervised versions. As with the Two Moons data, the higher the y-value, the more confident the algorithm is that a data point belongs to a given class. Note that optimizing α and labeling a small number of examples yield significantly better estimates of outlierness.

3.3 20 Newsgroups Data

In this set of experiments, we used the 20 newsgroups data set (version 20-news-18828). The topics were chosen from *rec*, which contains *autos*, *motorcycles*, *baseball*, and *hockey*. The articles were processed by the Rainbow software package, using options to skip headers, remove tokens on SMART's stoplist system, ignoring words in 5 or fewer documents, and stem words before counting. This yielded 3970 document vectors in a 8014-dimensional space, which were normalized into TFIDF representation. We calculated distance between points by using 1 minus the cosine of the angle between them. Test errors were averaged over 10 trials, and we used varying numbers of total labeled points, as indicated in Table 1.

As it was impossible to make a reasonable guess as to what it meant for a document vector to be an outlier, say, with respect to the class *hockey*, we did not estimate outlierness for this data set.

3.4 Time Series Data

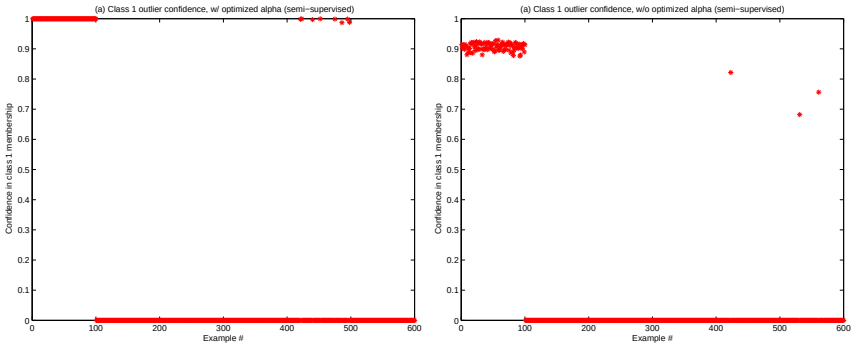
This set of experiments was performed on the Synthetic Control Time Series data set from the UCI database. Let us note at the outset that this data set is notoriously difficult for machine learning algorithms in general, so the high error rates should not be discouraging. We used 100 examples from each of the 6 classes. As with the USPS data set, we labeled only one example from each class.

The original algorithm, when run on this data set, had error rates of 31.2 and 35.5 with unoptimized and optimized α, respectively. (We optimized with respect to our algorithm only.) Our algorithm, by contrast, had error rates of 14.5 and 10.2 percent with unoptimized

Pts Labeled	α = 0.99	Optimal α	Orig. Algorithm
4	33%	30%	46%
10	30%	29%	40%
15	28%	25%	37%
20	24%	20%	34%
25	22%	19%	31%
30	20%	18%	28%
40	20%	18%	26%
50	21%	18%	21%

Table 1: Accuracy of Text Classification on 20 Newsgroups Data Set

and optimized α, respectively. Thus, normalization and optimization seems to dramatically improve results. Figure 4 illustrates the algorithm's confidence that a particular point belongs to a Class 1. Note the similarity between the variability here and the variability in the hard-to-classify USPS digits.



a) Optimized b) Unoptimized

Figure 4: Time Series Outliers for Digit 3

4 Conclusion

Zhou *et al*'s algorithm from [4] represents an exciting starting point for research into semi-supervised outlier detection. By normalizing and seeking to minimize class differences between nearby points while maximizing class differences between distant points, the algorithm discussed here improves on the very encouraging results from the original algorithm. In further research, we will work on refining optimization procedures and further theoretical analysis.

References

[1] DeCoste, D. & Schölkopf, B. (2002) Training Invariant Support Vector Machines. *Machine Learning* **46**, pp. 161-90.

[2] Kandola, J., Shawe-Taylor, J., & Cristianini, N. (2002) Learning Semantic Similarity. *NIPS*.

[3] Zhou, D., Bousquet, O., Lal, T.N., Weston, J. & Schölkopf, B. (2004) Learning with Local and Global Consistency. In S. Thrun, L. Saul and B.Schölkopf (eds.), *Advances in Neural Information Processing Systems* **16** (in press). Cambridge, MA: MIT Press.

[4] Zhou, D., Weston, J., Gretton, A., Bousquet, O., & Schölkopf, B. (2004) Ranking on Data Manifolds. In S. Thrun, L. Saul and B.Schölkopf (eds.), *Advances in Neural Information Processing Systems* **16** (in press). Cambridge, MA: MIT Press.