

# Automatic Locally Adaptive Smoothing for Tree-Based Set Estimation

Gabriel Chandler<sup>1</sup> and Leif Johnson<sup>2</sup>

<sup>1</sup>Department of Mathematics, Connecticut College, New London, CT 06320-4196 USA

<sup>2</sup>School of Statistics, University of Minnesota, Minneapolis, MN 55455

**ABSTRACT:** Tree-based methods similar to CART have recently been utilized for problems where the main goal is to estimate some set of interest. It is often the case that the boundary of the true set is smooth in some sense, however tree-based estimates will not be smooth, as they will be a union of ‘boxes’. We propose a general methodology for smoothing such sets that allows for varying levels of smoothness on the boundary automatically. The method is similar to the idea underlying support vector machines, that is applying a computationally simple technique to data after a non-linear mapping to produce smooth estimates in the original space. As an example, we consider the problem of level set estimation for regression functions and the dyadic tree based method of Willett and Nowak (2007).

**Additional Key Words and Phrases:** smoothing, tree-based estimation, locally adaptive, level set

## 1. Introduction

Tree based methods have been a popular tool in many areas of statistics for some time. Their popularity for problems of classification and regression dates back at least to the work of Breiman et al (1984). These methods provide nonlinear and nonparametric approaches that are computationally feasible and simple to interpret. For instance, in the classification problem, the decision regions (the set for which a particular decision will be made) are given as unions of ‘boxes’ in our decision space (rectangular regions corresponding to nodes of the tree). As a result, given a new observation, it is very simple to check which decision should be made. The utility of tree based techniques, as seen below, has been extended to areas of statistics where the goal of the analysis can be better thought of as set estimation. As a result, the property of sets being unions of boxes may no longer be preferable, as we might expect our true sets to be smooth in some regard.

This problem is not unique to tree-based estimates, as other ‘partition’ based methods suffer the same problems. For instance, we mention the work of Arias-Castro, Donoho and Huo (2006) in filament (a 1-dimensional manifold embedded into a random field) estimation. Though their work dealt with testing for the existence of such a structure in point process data, extending their method to estimation would result in a sequence of adjacent parallelograms, which conflicts with our belief that the true curve is smooth, and thus smoothing should be required.

A particular example we study in detail presently is the estimation of the level set of a regression function. The level set is defined as the set for which the function exceeds a certain threshold, denoted by  $\gamma$ . Formally, we suppose we have a response variable  $y \in \mathfrak{R}$  related to an independent variable  $\mathbf{x} \in [0, 1]^d$  via  $y = f(\mathbf{x}) + \epsilon$  with  $\epsilon \sim (0, \sigma^2)$ . The level set is then  $S_\gamma := \{x : f(x) \geq \gamma\}$ . Willet and Nowak (2007) proposed a method for estimation based on dyadic trees, which we describe in detail below. The resulting estimator is again a collection of boxes, though often it is reasonable to assume that the true level set has a smooth boundary in some sense.

We propose a method for smoothing estimates of sets that allows for varying levels of smoothness along the boundary of the set automatically. The idea is similar to that employed by support vector machines (SVMs), the use of a non-linear map to produce smooth estimates in the original data space. Based on a rough estimate of the original boundary via a tree-based algorithm or similar, we map data near the boundary into a new space in such a way that boxes in this new space correspond to smooth curves in the original space that should better align with the boundary of the true set. Whereas using an SVM requires both a selection of kernel and associated tuning parameters (for instance a Gaussian kernel with covariance matrix  $\Sigma$ ), the particular map used here is suggested completely by the data. The methodology also has the potential to overcome shortcomings in implementing some of these tree based methods, either the reliance on proper selection of tuning parameters or issue related to computational complexity of the algorithm, as discussed below.

Section 2 provides motivation for the methodology through a discussion of level set estimation for regression functions and the method of Willet and Nowak (2007). Section 3 discusses the methodology which we call tree-warping. Simulation results are found in section 4.

## 2. Regression Level Set Estimation and Dyadic Trees

### 2.1 Regression Level Sets

Regression level set estimation shows up in many applications. For instance, Willet and Nowak (2007) consider a noisy satellite image and estimate flood planes at various water levels. In this case, the response

variable is altitude, quantified by the grey scale value of each pixel. Thus, the flood plane at a water level  $\gamma$  is the complement of the regression level set. Polonik and Wang (2005) relates level set estimation to optimal operation conditions, noting that setting  $\gamma = \sup f(x)$  yields the location of the modes of the function. Several applications of regression level sets in image processing applications are contained in the introduction of Willet and Nowak (2007).

Our goal is to estimate the  $S_\gamma$  based on observing data  $(\mathbf{x}_i, y_i)$ ,  $i = 1, \dots, n$ . One might consider the simple approach of estimating the entire regression function given the data (via a kernel smoother, etc.) then take as our estimate the set on which the estimated function exceeds the threshold, i.e.  $\hat{S}_\gamma := \{x : \hat{f}(x) \geq \gamma\}$ . However, estimation of the entire regression function is a more general problem than level set estimation, which is likely to lead to sub-optimal result, as density estimation is usually concerned with a global optimization criterion, with no guarantees regarding local behavior. As a result, several authors have recently proposed methods that estimate the level set directly, without first estimating the entire regression function. For instance, Polonik and Wang (2005) use the excess mass functional to estimate the level set, Scott and Davenport (2007) use support vector machines, and Willet and Nowak (2007) propose a method based on dyadic decision trees. It is this tree based approach that is used as motivation for the methodology discussed below.

## 2.2 Dyadic trees

Dyadic trees, unlike CART, have splits that are predetermined and not dependent on the data. The leaves of the tree correspond to regions that are found by recursively splitting larger regions in half along one dimension. The maximum number of splits in any direction is pre-determined by the user and is denoted as  $k_{max}$ . The largest possible tree for  $k_{max} = 3$  with  $d = 2$  (in which no branches have been pruned), is shown in figure 1. Each region is thus a square with side length  $2^{-k_{max}} = 1/8$ .

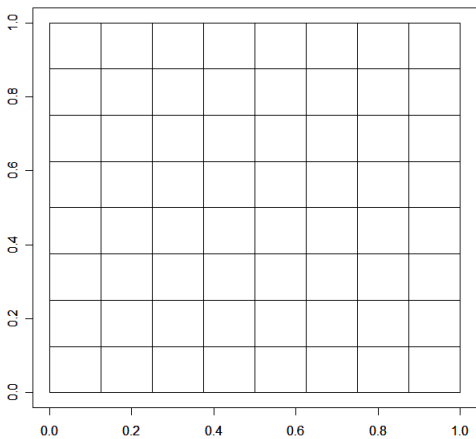


Figure 1: The largest unlabeled tree for  $k_{max} = 3$

Unlike CART-type methods, where the location of each split is based on a greedy algorithm, the smaller class of trees resulting from dyadic trees allows explicit optimization within the class. This is due to the smaller class of possible estimators. In addition, by choosing an optimization criterion that is additive over subtrees, a pruning strategy that minimizes this criterion is feasible and in fact simple to implement. This is

accomplished by the algorithm given by Blanchard et al (2004). The optimization criterion given by Willett and Nowak (2007) is given as a linear combination of the empirical risk

$$\hat{R}_n(T) = \frac{1}{n} \sum_{i=1}^n \hat{e}_S(X_i, Y_i)$$

where

$$\hat{e}_S(X_i, Y_i) = \frac{\gamma - Y_i}{2A} \left[ \mathbf{1}_{\{\mathbf{x}_i \in \mathbf{S}\}} - \mathbf{1}_{\{\mathbf{x}_i \in \bar{\mathbf{S}}\}} \right] \quad (1)$$

and a penalty term for tree-complexity,  $\Phi_n(T)$ , which depends not only on the size of the boxes, but on the symmetry of the tree as well. The value of  $A$  in (1) is a bound on the data needed for the asymptotics for their method.

Specifically,

$$\Phi_n(T) := \sum_{L \in \pi(T)} \sqrt{\frac{8(\log(2/\delta) + [L] \log 2) \hat{p}_L(\delta)}{n}},$$

where  $[L]$  denotes the number of bits required to encode the position of the leaf  $L$  in the tree  $\pi(T)$ ,  $\delta$  relates to the probability the true risk and empirical risk are close and  $\hat{p}_L(\delta)$  is related to the probability content of  $L$ .

In particular, we seek to minimize

$$\hat{R}(T) + \rho \hat{\Phi}_n(T) \quad (2)$$

where  $0 < \rho < 1$  plays the role of a smoothing parameter. The larger the value of  $\rho$  the less complex the resulting tree, as small boxes will be more likely to be trimmed in favor of a larger box attained by combining two smaller boxes.

There is a need to properly balance the penalty term against the empirical risk. A selection of  $\rho = 0$  and a sufficiently large value of  $k_{max}$  returns a tree in which the level set consists of the collection of boxes containing single points who's corresponding dependent variable exceeds  $\gamma$ . In the presence of noise, this estimate will severely overfit the data. A selection of  $\rho = 1$  returns a level set estimate which only consists of very big boxes and thus most likely not a very accurate estimate of the underlying set.

### 2.3 The need for small boxes

In order for the tree-based estimate to resemble the true level set (in terms of symmetric difference for instance), it is likely that small boxes will need to be kept in the estimate. This will especially be the case if the actual boundary of the level set is somewhat smooth. Consider the level set shown in figure 2, where it is clear that using  $k_{max} = 3$  cannot result in a very precise estimate. Thus, larger values of  $k_{max}$  would need to be considered for precise estimates.

As a result, we realize that the precision of our estimate is both a function of  $k_{max}$  as well as  $\rho$ . Regardless of the size of  $k_{max}$ , if  $\rho$  is set too large for the particular problem, the resulting estimate will likely contain mostly large boxes. Notice for the level set of figure 2, even for a good choice of  $\rho$ , for any finite  $k_{max}$ , the

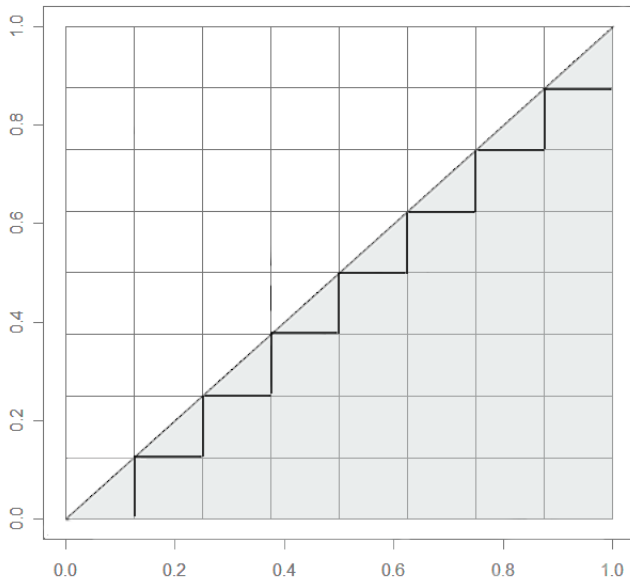


Figure 2: The shaded area is the true level set, the dark line is the boundary of the (not unique) optimal estimate in terms of symmetric difference

class of trees will never contain the true level set.

Consider Figure 3, where three values of  $\rho$  are considered for equally spaced data with no noise added. As a result, setting  $\rho = 0$  yields the optimal estimate. The effect of increasing  $\rho$  is seen as small boxes that are needed for a good estimate are pruned. In this particular case the estimate shrinks, though what happens in general depends on the orientation of the true boundary in the space as well as the behavior of the regression function near the boundary (taking the negative of the function and the threshold would clearly give an estimate that is expanding). The small boxes that are retained in the third panel are well within the level set and thus the risk is sufficient to overcome the size of the penalty.

Unfortunately, choosing an appropriate value of  $\rho$  is not straight forward. In the application given by Willett and Nowak (2007), the optimal tree used  $\rho = .0124$ , seemingly small despite the large sample size ( $n = 2^{18}$ ). This value was arrived at by comparing the (in this case known) true level set to the resulting estimators over a grid search of  $\rho$  values. The final value of  $\rho$  was chosen to minimize the risk between the two sets. In practice, however, the true level set will not be known and  $\rho$  must be selected by some other means. A proper selection of  $\rho$  depends on the behavior of the function near the boundary as well as the amount of noise in the data, among other things, most of which are generally unknown. Rather than attempting to come up with schemes to choose this parameter, we attempt to manipulate the data in such a way that the value of  $\rho$  does not have a great effect on the estimation of the level set.

We note that smoothing was done in this particular example, using a technique related to ‘voting over shifts’, where the data is shifted by a small amount repeatedly, with a new estimate obtained for each shift. The final estimate consists of all points for which the majority of shifted estimates contained those points. It is clear from Figure 3 however that for improper selection of  $\rho$ , this will not help much. Instead it will

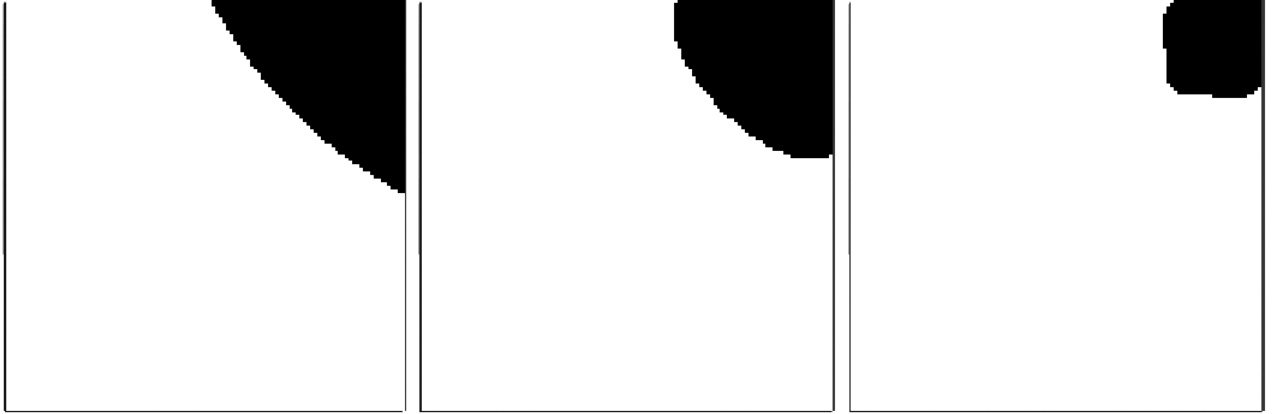


Figure 3: Three level set estimates with varying value of  $\rho$  (0,.06,.15). The true level set is essentially the first plot.

only return a smoothed version of the original estimate, which from the third panel of the figure, may not provide a satisfactory estimate of the true set.

### 3. Tree Warping

As further motivation beyond simple smoothing, the problems described above dealing with selection of  $\rho$  seem to relate to the fact that a union of large dyadic rectangles is unlikely to provide a reasonable approximation to the true level set. If however, the true level set was given as a union of large dyadic rectangles, the choice of  $\rho$  would not be so critical (simulation studies found that the range of  $\rho$  that gives ‘good’ estimates back is larger when the true level set is a union of large rectangles compared to sets consisting of smaller rectangles). Recalling the level set shown in figure 2, we might notice that if we rotated our tree, as in figure 4, not only does this class of trees contain the true level set, but the optimal tree is the simplest non-degenerate tree in the class, consisting of only one split.

Figure 4 only serves as motivation for what is to follow, as we should not expect to know before hand which rotation of the tree would be appropriate, or for there to even be a suitable rotation for our particular problem. However, the idea of changing the orientation of the tree to better fit the level set seems appealing. A priori, the appropriate orientation will not be known. However, after calculating a rough tree-based estimate, we will have some information about the orientation of the boundary, and thus a reasonable orientation of the tree we might consider. This is the premise of tree warping. After estimating a rough tree based estimate, we smooth the boundary of the estimate and take a collection of points near the boundary. These points are then mapped into a new space such that dyadic boxes in this space correspond to sets in the original space that should better align with the boundary of the true level set.

We assume that the true level set boundary does not intersect itself, and choose  $\rho$  so that the original estimate has this property. In the case that the set contains multiple disconnected regions, we can partition the space and run the algorithm for each region.

The algorithm is as follows:

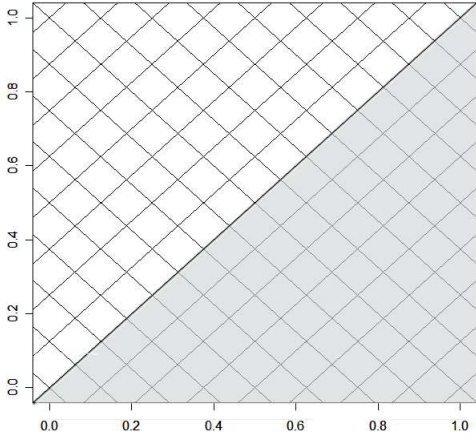


Figure 4: The tree rotated by  $\pi/4$ , the optimal estimated boundary is given by the darkened line.

1. Fit a rough estimate of the level set using a dyadic tree.
2. Smooth the estimated boundary.
3. Map a region of points close to the estimated boundary into  $[0, 1]^d$
4. Fit a dyadic tree in the “warped” space.
5. Map the estimated boundary in the “warped” space back into the original space.
6. Return to step 2.  $M$  times.

The final estimate is given as the smoothed estimate that has the smallest empirical risk, thus guaranteeing the resulting estimate is at least as good as the original. The exact details of the smoothing procedure and the map are discussed below.

### 3.1 Initial Smoothing

We continue to assume that  $d = 2$ . The boundary can be parameterized as a path in  $\mathfrak{R}^2$ , which we denote as  $z(t) = (z_1(t), z_2(t))$ ,  $t \in [0, 1]$ . The location of  $z(0)$  is arbitrary. Smoothing is done along the path to avoid selecting an orientation for the axis, so the smoothed path is written as  $\tilde{z}(t) = \frac{1}{2b} \int_{t-b}^{t+b} z(t) dz$  with  $z(t) = z(t-1)$  for  $t > 1$  and  $z(t) = z(1-t)$  for  $t < 0$  should the path form a loop. Here  $b$  is the bandwidth. To avoid selecting a global bandwidth, we select  $b$  to be relatively small and iterate the procedure. As a result, the estimate will tend to get smoother and smoother should the data allow it, as discussed below. Extensions to higher dimensions is straight forward.

### 3.2 The warping map

The key element of the warp transformation is the invertible map  $\ell_z$ .  $\ell_z$  maps a region around the boundary of the level set to  $[0, 1]^d$ , such that simple trees in the warped space correspond to good approximations in the original space. We use  $Z$  to be the  $d - 1$  dimensional boundary in  $\mathbb{R}^d$  of the true level set  $S \subset \mathbb{R}^d$  and

$\mathbf{z} : [0, 1]^{d-1} \rightarrow Z$  is an invertible map. We assume that the estimate is connected with the additional condition that  $z(t_1) = z(t_2) \Rightarrow t_1 = t_2$  for  $t_1, t_2 \in (0, 1)$  i.e. for instance the estimate cannot be connected only at corners. If  $C(z) \subseteq \mathbb{R}^d$  is the domain of  $\ell_z$ , the key properties of  $\ell_z : C(z) \rightarrow [0, 1]^d$  are (1)  $\ell_z(\mathbf{z}(\mathbf{t})) = (\mathbf{t}, 1/2)^T$  and (2)  $\ell_z$  maps points in  $C(z) \cap S^C$  to  $[0, 1]^{d-1} \times [0, 1/2)$  and points in  $C(z) \cap S$  to  $[0, 1]^{d-1} \times [1/2, 1]$ . Any transformation satisfying these properties will work. Here we will describe the specific approximation,  $\hat{\ell}_{z,w}$ , we use when  $d = 2$  and suggest a general approach to use for other  $d$ .  $w$  is a tuning parameter that is the 'width' of  $C(z)$ , which we will now denote  $C(z, w)$ .

Before we can describe our approach for approximating  $\ell_{z,w}$  when  $d = 2$ , we first must describe  $\ell_{z,w}$  for all  $d$ . For the purpose of defining  $\ell_{z,w}^{-1}$ ,  $\mathbf{z}$  needs to be differentiable, but as we shall see later on it does not need to be for our approximation,  $\hat{\ell}_{z,w}$ . We shall use  $\mathbf{z}(\mathbf{t}) = (z_1(\mathbf{t}), \dots, z_d(\mathbf{t}))^T$ .  $\mathbf{P}_z(\mathbf{x})$  is the projection of  $\mathbf{x} \in \mathbb{R}^d$  onto  $Z$ . Note that we use a functional definition for  $\mathbf{P}_z$  since  $Z$  will almost always be nonlinear, hence  $\forall \mathbf{x} \in \mathbb{R}^d \exists \mathbf{t}_x \in [0, 1]^{d-1}$  such that  $\mathbf{P}_z(\mathbf{x}) = \mathbf{z}(\mathbf{t}_x)$  and  $\forall \mathbf{t} \in [0, 1]^{d-1} \|\mathbf{z}(\mathbf{t}_x) - \mathbf{x}\| \leq \|\mathbf{z}(\mathbf{t}) - \mathbf{x}\|$ . Now define  $C(z, w) = \{\mathbf{x} \in \mathbb{R}^d \mid \|\mathbf{P}_z(\mathbf{x}) - \mathbf{x}\| \leq \frac{w}{2}\}$ . Thus  $C(z, w)$  is a region extending  $\frac{w}{2}$  out from  $Z$ . Now we can describe  $\ell_{z,w} : C(z, w) \rightarrow [0, 1]^d$ . For  $\mathbf{x} \in C(z, w)$

$$\ell_{z,w}(\mathbf{x}) = (\mathbf{z}^{-1}(\mathbf{P}_z(\mathbf{x})), \frac{1}{2} + \frac{\|\mathbf{P}_z(\mathbf{x}) - \mathbf{x}\|}{w} (-1)^{1-q})^T$$

where  $q = I(\mathbf{x} \in S)$ . To describe  $\ell_{z,w}^{-1}(\mathbf{t}, y)$  for  $\mathbf{t} \in [0, 1]^{d-1}$  and  $y \in [0, 1]$  we need the outward pointing unit vector surface normal of  $\mathbf{z}$  at  $\mathbf{t}$ ,  $\mathbf{r}(\mathbf{t})$ . If we define  $\mathbf{V}(\mathbf{t}^*) = (\frac{\partial z_i}{\partial t_j} |_{\mathbf{t}=\mathbf{t}^*})$  as the first derivative of  $\mathbf{z}(\mathbf{t})$  evaluated at  $\mathbf{t}^*$ , then  $\mathbf{r}(\mathbf{t})$  satisfies three conditions. (1)  $\|\mathbf{r}(\mathbf{t})\| = 1$ , (2)  $\mathbf{V}^T(\mathbf{t})\mathbf{r}(\mathbf{t}) = 0$  and (3)  $\exists \delta > 0$  such that  $\forall h \in (0, \delta)$ ,  $\mathbf{z}(\mathbf{t}) + h * \mathbf{r}(\mathbf{t}) \notin S$ . Condition (3) is necessary because we do not require  $S$  to be convex. For some shapes, such as a horseshoe style shape in  $\mathbb{R}^2$  a less precise definition would be incorrect. Now we can define  $\ell_{z,w}^{-1} : [0, 1]^d \rightarrow \mathbb{R}^d$  for  $\mathbf{t} \in [0, 1]^{d-1}$  and  $y \in [0, 1]$

$$\ell_{z,w}^{-1}(\mathbf{t}, y) = \mathbf{z}(\mathbf{t}) + w * (\frac{1}{2} - y) * \mathbf{r}(\mathbf{t})$$

Of course, since we are estimating  $\mathbf{z}$  none of the functions needed for calculating  $\ell_{z,w}$  or  $\ell_{z,w}^{-1}$  are known. We estimate  $\hat{\ell}_{z,w}$  with local linear approximations to  $Z$ . This has the advantage that  $\mathbf{z}$  does not need to be differentiable. For  $d = 2$   $Z$  will be a curve, possibly a closed loop. First we estimate  $b$  points on  $\mathbf{z}$ , defined by  $\mathbf{z}(t_0), \dots, \mathbf{z}(t_{b-1})$ . If we are estimating a closed loop we define one additional point  $\mathbf{z}(t_b) = \mathbf{z}(t_0)$ . For the sake of generality we will use  $\mathbf{z}(t_{b^*})$  to be the last point. Thus  $b^* = b$  for a loop and  $b^* = b - 1$  otherwise. We set  $t_0 = 0$  and for  $j = 1, \dots, b^*$ ,  $t_j = \frac{\sum_{i=1}^j \|\mathbf{z}(t_i) - \mathbf{z}(t_{i-1})\|}{\sum_{i=1}^{b^*} \|\mathbf{z}(t_i) - \mathbf{z}(t_{i-1})\|}$ , i.e.  $t_j$  represents the proportion of the arc length covered up to and including section  $j - 1$ . Using these line segments, we divide  $\hat{C}(z, w)$  into  $b^*$  regions,  $\hat{C}(z, w)_0, \dots, \hat{C}(z, w)_{b^*-1}$ .  $\hat{C}(z, w)_j$  corresponds to the line segment  $j$ , from  $\mathbf{z}(t_j)$  to  $\mathbf{z}(t_{j+1})$ . We estimate the  $\mathbf{r}(t_j)$ 's by choosing angles  $\hat{\theta}_j$  and setting  $\hat{\mathbf{r}}(t_j) = (\cos(\hat{\theta}_j), \sin(\hat{\theta}_j))^T$ , taking care that  $\hat{\mathbf{r}}(t_j)$  points outward. For  $j = 1, \dots, b^* - 1$   $\hat{\theta}_j$  bisects the angle between line segments  $j - 1$  and  $j$ . If  $Z$  is a closed loop,  $\hat{\theta}_0 = \hat{\theta}_{b^*}$  which bisects the angle between line segments  $b^* - 1$  and  $0$ . If  $Z$  is not a closed loop,  $\hat{\theta}_0$  and  $\hat{\theta}_{b^*}$  are chosen such that  $\hat{\mathbf{r}}(t_0)$  and  $\hat{\mathbf{r}}(t_{b^*})$  are orthogonal to the line segments  $0$  and  $b^* - 1$  respectively. Now let  $\mathbf{x}_0$  be any point in  $\mathbb{R}^2$  and  $j \in \{0, \dots, b^* - 1\}$  be an index to some  $\hat{C}(z, w)_j$ . To see if  $\mathbf{x}_0$  is transformed by  $\hat{\ell}_{z,w}$  in the region of  $\hat{C}(z, w)_j$

1. Project a line through  $\mathbf{x}_0$  parallel to line segment  $j$  and find the points  $\mathbf{x}_{0,j}$  and  $\mathbf{x}_{0,j+1}$  where this parallel line intersects the lines  $\hat{\mathbf{z}}(t_j) + h * \hat{\mathbf{r}}(t_j)$  and  $\hat{\mathbf{z}}(t_{j+1}) + h * \hat{\mathbf{r}}(t_{j+1})$  respectively.
2. If  $\mathbf{x}_0$  is not between  $\mathbf{x}_{0,j}$  and  $\mathbf{x}_{0,j+1}$  then  $\mathbf{x}_0 \notin \hat{C}(z, w)_j$ .
3. Otherwise set

$$\begin{aligned} \lambda_0 &= \frac{\|\mathbf{x}_0 - \mathbf{x}_{0,j}\|}{\|\mathbf{x}_{0,j} - \mathbf{x}_{0,j+1}\|} \\ \mathbf{z}_0 &= \mathbf{z}(t_j) + \lambda_0 * (\mathbf{z}(t_{j+1}) - \mathbf{z}(t_j)) \\ d_0 &= \|\mathbf{x}_0 - \mathbf{z}_0\|/w \end{aligned}$$

4. If  $d_0 > \frac{1}{2}$ ,  $\mathbf{x}_0 \notin \hat{C}(z, w)_j$ . Otherwise  $\mathbf{x}_0 \in \hat{C}(z, w)_j$  and

$$\hat{\ell}_{z,w}(\mathbf{x}_0) = (t_j + \lambda * (t_{j+1} - t_j), 1/2 + d * (-1)^{1-\hat{q}})^T$$

where  $\hat{q} = I(\mathbf{x}_0) \in \hat{S}$ .

Fortunately,  $\hat{\ell}_{z,w}^{-1}$  is much easier to define. For any point  $(t, y) \in [0, 1]^2$  find  $j$  such that  $t_j \leq t \leq t_{j+1}$ . Then set  $\lambda = \frac{t-t_j}{t_{j+1}-t_j}$  and  $\theta = \hat{\theta}_j + \lambda * (\hat{\theta}_{j+1} - \hat{\theta}_j)$ . Then

$$\hat{\ell}_{z,w}^{-1}(t, y) = \mathbf{z}(t_j) + \lambda * (\mathbf{z}(t_{j+1}) - \mathbf{z}(t_j)) + w * (1/2 - y) * \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix}$$

This approach can be generalized to  $d > 2$ . The line segments will become  $d - 1$  dimensional planes, defined by  $d$  points. Each of these 'tiles' would have  $d$   $\mathbf{r}(t_j)$ 's. Any reasonable linear interpolation method can be used for calculating the  $\hat{\theta}$ 's. By using piecewise linear components to approximate the surface, the map  $z: [0, 1]^{d-1} \rightarrow Z$  should not be overly difficult to compute.

### 3.3 Automatic Locally Adaptive Smoothing

The procedure calls for smoothing at each step along the path via a kernel type estimate with a fairly small bandwidth. As a result of iterating, the estimate will increase in smoothness unless roughness is added into the estimate through smaller boxes being used in the warped space. An absence of splits in the horizontal direction in the warped space will result in a highly smoothed estimate in the original space. A boundary in the warped space corresponding to a single vertical split at .5 will ultimately reduce the estimate to a single point if it is a closed path, though this shrinkage property will be reversed by vertical splits at values smaller than .5, causing the estimate to dilate (as seen in panel 8 of Figure 5).

As this procedure is iterative, the smoothness of the estimator will tend to increase towards the smoothness of the underlying set boundary, and then start to become over-smoothed. Given a reasonable value of  $\rho$ , at some point 'jumps' (splits in the horizontal direction) will be reinserted into the estimate in the warped space, causing sharp corners in the estimate in the original space. These corners will then be smoothed until the procedure deems jumps are needed again as the estimate has again become over-smoothed. By running the algorithm for a number of iterations, then choosing the estimate with the smallest empirical risk, we allow the data to select the smoothness level of the final estimate.

Furthermore, as jumps inserted into the estimate only effect smoothness of the estimate locally, the amount of smoothness in the estimate can differ along the path. In this sense, the procedure not only self selects the smoothness level, but does so locally, adapting an estimate that may have varying degrees of smoothness.

Figure 5 illustrates several iterations of the algorithm with a toy data set, in which no noise is added to the data. The smoothing parameter  $b$  is set rather high for expository purposes. As a result, the estimated level set in the warped space is constantly dilating the level set in our original space in order to counteract the contraction resulting from the large smoothing parameter.

## 4. Simulations

For the following simulations, the independent variables  $\mathbf{x}_i = (x_{i1}, x_{i2})$ ,  $i = 1, \dots, n$  ( $n = 1000$  for the first 4 simulations and  $n = 2000$  for the second) are generated as uniform in the unit square. The response variable has mean 2 in the level set and mean 0 outside the level set and is convoluted with standard normal white noise. While we only look at a 'nice' error distribution, we note that effects of other error distributions would manifest themselves not in the warping but rather in the original tree-based estimate itself. If the tree-based estimation procedure is not reasonable, then warping may not help. In each of the following

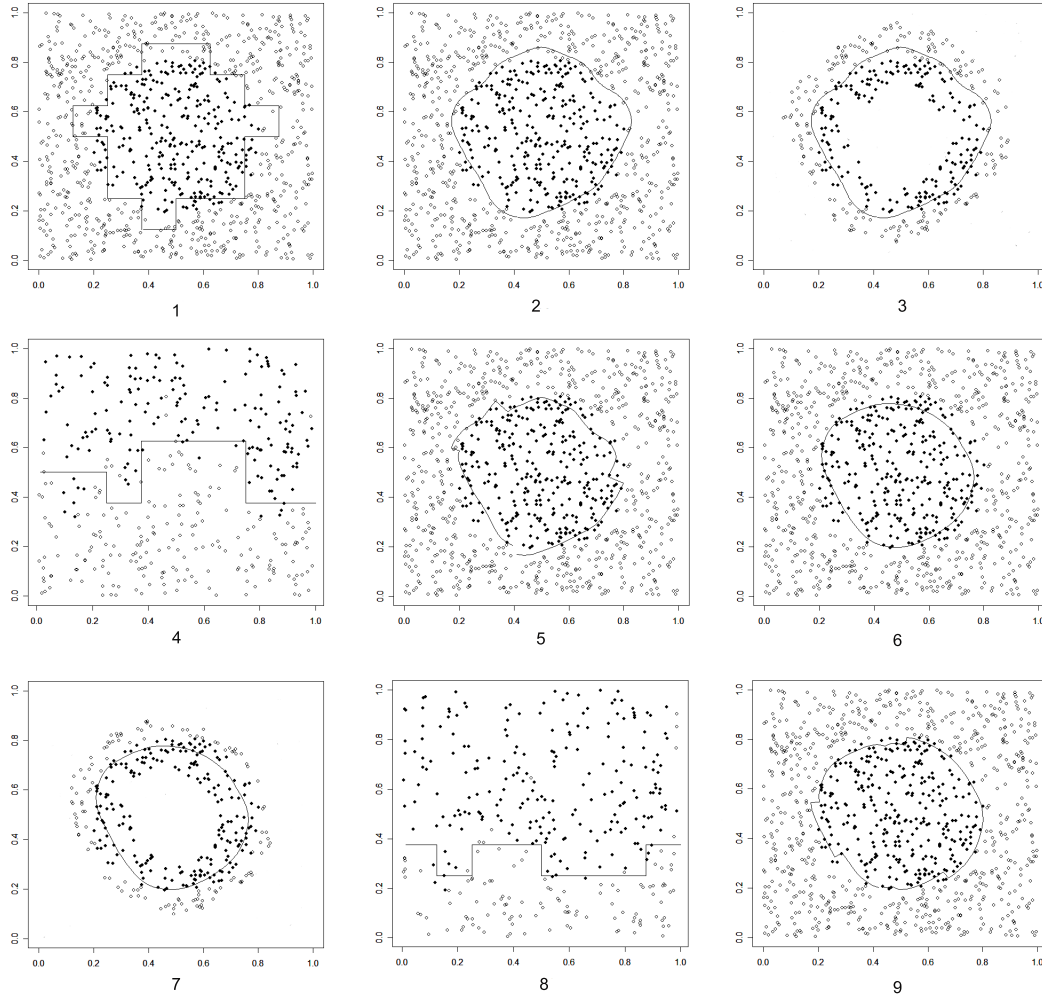


Figure 5: 1:  $z(t)$  2:  $\tilde{z}_1(t)$ . 3:  $\mathbf{x} \in C(z_1, w)$ . 4:  $\mathbf{y}$ . 5:  $z^*(t)$ . 6:  $z_2(t)$ . 7:  $\mathbf{x} \in C(z_2, w)$ . 8:  $\mathbf{y}$ . 9:  $z_3(t)$  (unsmoothed)

figures, the subset of the unit square representing the level set is pictured along with the data. We note that the assumptions used in the asymptotics of the original work of Willet and Nowak (2007) required the data to be bounded. While this is not satisfied with the simulated data, the figures provided serve to illustrate the performance of the tree warping method for a fixed sample size.

Due to the computational complexity of their method, especially for large  $k_{max}$ , we provide the following examples using relatively small values of  $k_{max}$ , either only 3 or 4. Certainly this does not provide much flexibility to the original tree based method to fit a smooth boundary. However, using a larger  $k_{max}$  introduces the problem of choosing an appropriate value of  $\rho$ . Repeatedly fitting trees with a small value of  $k_{max}$  is cheaper computationally than fitting a single large tree as each box requires 2 comparisons, with the number of boxes being  $4^{k_{max}}$ .

For all the examples, the estimate is smoothed along the path using a bandwidth of .02. One might consider using a smaller value, as this value was used to illustrate how quickly the warping improves the original estimate.

It would be possible to achieve near perfect simulation results using a large value of  $k_{max}$ , a well specified

value of  $\rho$  (as we know the true set), and many iterations. We purposefully avoid this in order to show the power of our method. For all the examples provided below, the algorithm was run for at most 10 iterations. The value of  $\rho$  used was on the order of .0001, but in general will depend on the sample size and the behavior of the function near the boundary, among other things. We simply started with a small value of  $\rho$ . If the resulting estimate was not ‘nice’ (connected with the condition  $z(t_1) = z(t_2) \Rightarrow t_1 = t_2$  for  $t_1, t_2 \in (0, 1)$ ), we simply increased  $\rho$  by a small amount until this was satisfied. Otherwise, there was no attempt to find ‘good’ values of  $\rho$ . Small changes in  $\rho$  were checked to confirm similar results would arise. A window of width .2 around the boundary was used, with points in this region being mapped into the warped space. Using a larger  $k_{max}$  would have allowed us to choose a smaller window, should we have faith in the value of  $\rho$ .

The first three examples (figures 6-8) have level set boundaries that are fairly smooth. For the second example, the smoothness is uniform across the boundary. With the first and the third however, as the number of iterations increases, jumps are continually inserted into the estimate in the warped space in locations where the boundary is less smooth, while allowing the estimate to continue to be smoothed elsewhere. Consequentially, the resulting estimates closely match the true level sets.

The fourth example uses a slightly more complicated level set boundary, as seen in figure 9. While the

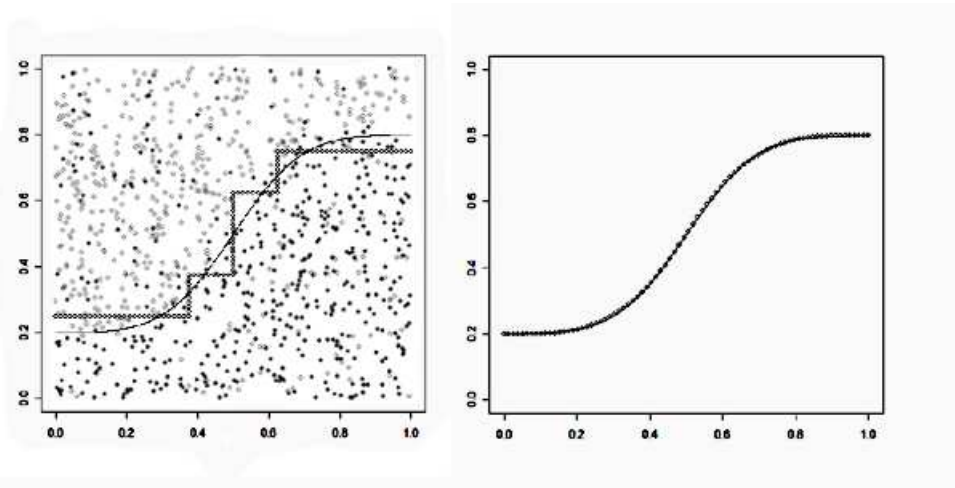


Figure 6: The left graph contains the boundary estimate using an unwarped tree along with the true boundary, the right is the warped tree estimate after 5 iterations. The darkened points are those which exceed the threshold.

estimate presented does not match the true level set as well as in the previous three simulations, it does not perform poorly. It clearly recognizes the lack of convexity in the true level set, and many of the regions where it deviates from the truth can be explained by the sparseness of the data in those regions.

In all cases, the estimates can be improved by using larger trees to allow the method more flexibility. However, this leaves us with a more difficult task of choosing a reasonable value of  $\rho$ . For the final simulation, shown in figure 10, the true level set exhibits a horseshoe shape. The resulting warped estimate is much closer to the true level set than the unwarped version, though some of this can be attributed to the small value of  $k_{max}$  allowed.

#### 4.1 Practical Issues

We mention some issues that arose as a result of the simulations. Firstly, the location of  $z(0)$  strongly determines the location of possible jumps in the warped space (as  $\text{arclength}(z)$  remains fairly constant through the iterative procedure), especially for small  $k_{max}$ . Especially when using small trees as we did above, it is

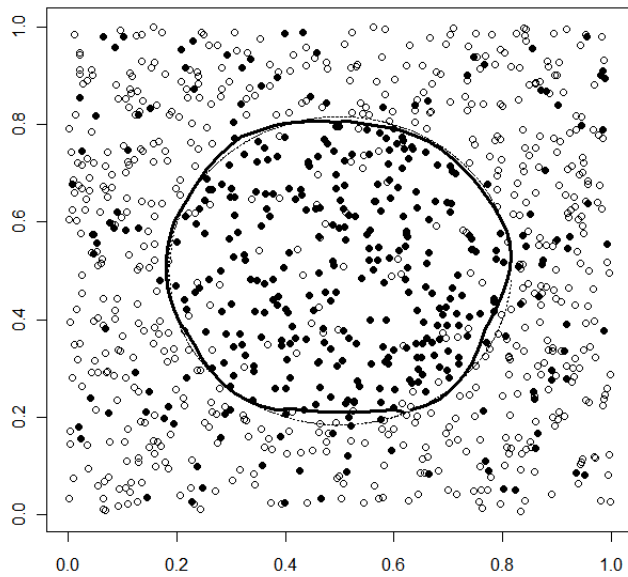


Figure 7: A circular level set with the tree-warped estimate. The estimate is given as the solid line. The darkened points are those which exceed the threshold.

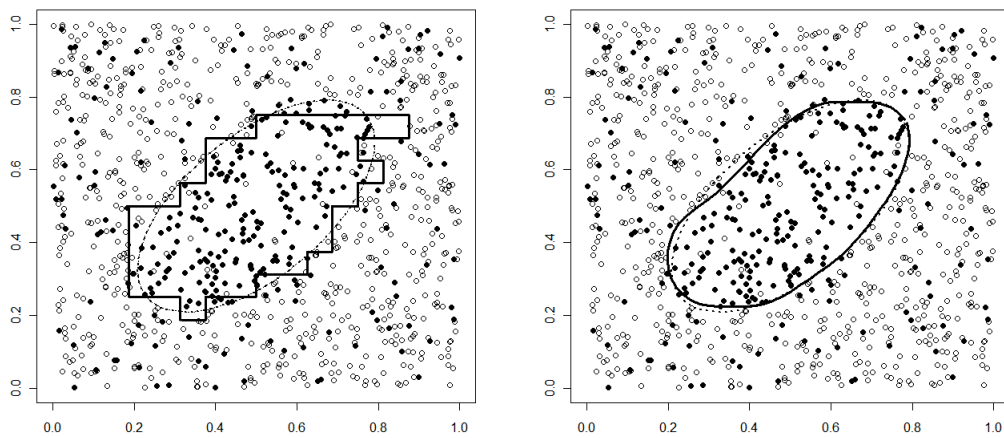


Figure 8: An elliptical level set with the unwarped (left) and tree-warped (right) estimates. The estimate is given as the solid line. The darkened points are those which exceed the threshold.

helpful to alter the location of  $z(0)$  from iteration to iteration, allowing jumps to occur at varying locations. As the location of  $z(0)$  was chosen arbitrarily to begin with, altering it from iteration to iteration does not hamper the algorithm in any way.

The second issue deals with the choice of  $w$ , the width of the region near the path to which the warping

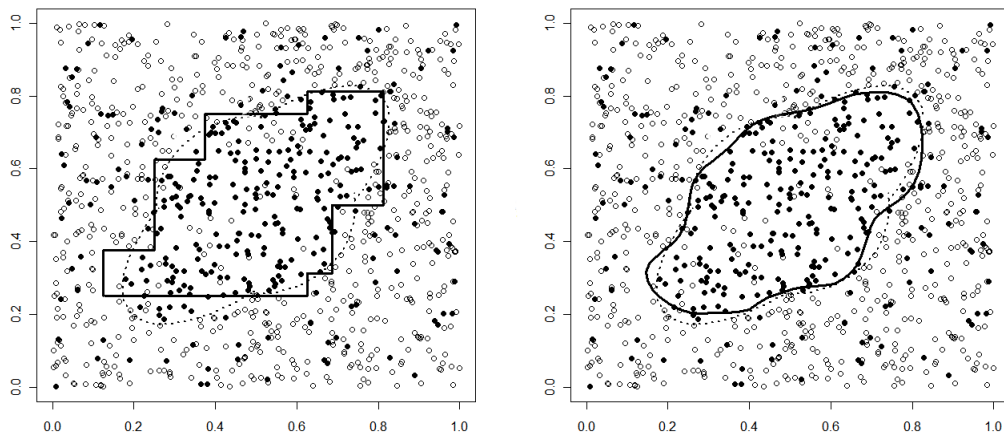


Figure 9: A non-convex level set with the tree-warped estimate. The estimate is given as the solid line. The darkened points are those which exceed the threshold.

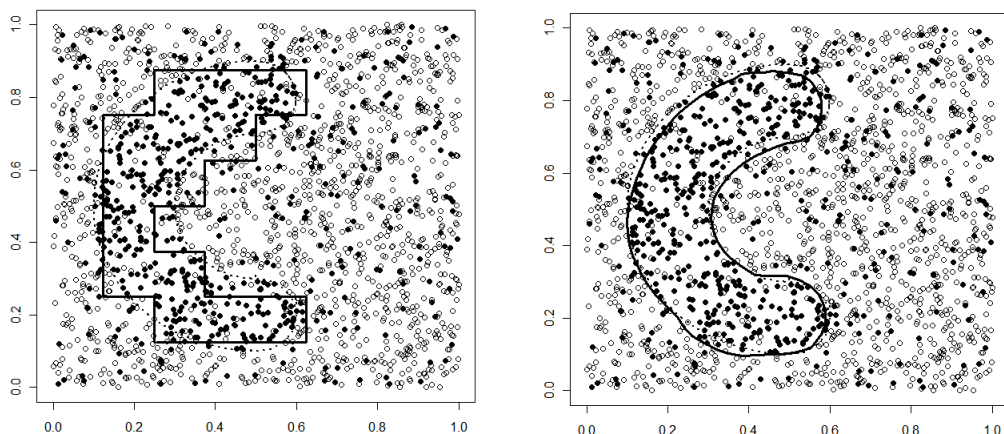


Figure 10: A horseshoe shaped level set with the tree-warped estimate. The estimate is given as the solid line. The darkened points are those which exceed the threshold.

map is applied. While it is preferable that the region is chosen large enough to ensure the true boundary is captured, the algorithm provides enough flexibility to eventually move the estimated boundary towards the true boundary over several iterations when it is not captured originally. This can be seen by referring to the third panel of Figure 3. While the original estimate is far from the truth, for a small enough value of  $w$ , all points mapped would lie in the true set. Thus, in the warped space, a degenerate tree would be fit, in which the entire warped space would be classified as lying in the set, thus expanding the estimate and approaching the truth. This would certainly be the case even if  $\rho$  is set too large as it is here, as the degenerate tree is the least complex.

In regards to a convergence criterion, our iterative process will not monotonically improve the estimate, for if at some point the estimate is optimal, the next iteration will make it worse, until the point that it

corrects the over-smoothing and begins to improve the estimate again. Thus, a standard convergence criterion is not appropriate here. One option is to keep track of the smallest empirical risk observed over groups of iterations. If successive groups of iterations fails to lower the empirical risk, we can assume that a near optimal warped estimate has been found.

The method is easily adapted to more interesting looking boundaries by partitioning the space and running the algorithm multiple times. This would be the case should the true set be disconnected, or have holes.

In the warped space, many of the possible splits are not of interest and are thus not necessary to explore, for instance those creating disconnected regions of the set away from the boundary. As such, it would be possible to ignore such splits, and lower the number of computations required.

## 5. Bibliography

Arias-Castro, E., Donoho, D. and Huo, X. (2006). Adaptive multiscale detection of filamentary structures in a background of uniform random points. *Ann. Statist.* **34** 326-349.

Blanchard, G., Schfer, C., and Rozenholc, Y. (2004), "Oracle bounds and exact algorithm for dyadic classification trees," *Proceedings of the 17th. Conference on Learning Theory (COLT 04), Springer Lecture Notes in Artificial Intelligence*, 3120, 378-392.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984), *Classification and Regression Trees*, Wadsworth International, Belmont, Ca.

Polonik, W., and Wang, Z. (2005), "Estimation of regression contour clusters: an application of the excess mass approach to regression," *Journal of Multivariate Analysis*, 94, 227-249.

Scott, C., and Davenport, M., (2007), "Regression level set estimation via cost-sensitive classification," *IEEE Transactions on Signal Processing*, 55, 2752-2757.

Willett, R., and Nowak, R. (2007), "Minimax Optimal Level Set Estimation," *IEEE Transactions on Image Processing*, 16, 2965-2979.