Automatic Locally Adaptive Smoothing for Tree-Based Set Estimation

Gabriel Chandler[1] and Leif Johnson[2]
[1]Department of Mathematics, Pomona College, Claremont, CA 91711-6312 USA
[2]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. In particular, we consider the problem of level set estimation for regression functions and the dyadic tree based method of Willet and Nowak (2007).

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-Castor, Donoho and Hou (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 will 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 γ. Formally, we suppose we have a response variable $y \in \mathbb{R}$ related to an independent variable $\mathbf{x} \in [0,1]^d$ via $y = f(\mathbf{x}) + \epsilon$ with ϵ being noise with mean 0 and finite variance. The level set is then $S_\gamma := \{\boldsymbol{x} : f(\boldsymbol{x}) \geq \gamma\}$. 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, which is chosen 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, see for example Hastie, Tibshirani and Friedman (2001). 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 Σ), the particular map used here is suggested 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 issues 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. Simulations and an application to real data are found in Section 4., with a discussion and concluding remarks found in Section 5.

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 gray scale value of each pixel. Thus, the flood plane at a water level γ is the complement of the regression level set. Polonik and Wang (2005) relates level set estimation to optimal operation conditions, noting that setting $\gamma = \sup f(\boldsymbol{x})$ 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). In this paper, we consider estimating the region of the United States for which the air quality falls below the safety limit set by the EPA with respect to a particular pollutant. The data is recorded at monitoring stations that are scattered across the country with non-uniform spacing.

Our goal is to estimate the S_γ based on observing data $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, n$. 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. $\hat{S}_\gamma := \{\boldsymbol{x} : \hat{f}(\boldsymbol{x}) \geq \gamma\}$. However, estimation of the entire regression function is a more general problem than level set estimation, which is likely to lead to a sub-optimal result, as density estimation is usually concerned with a global optimization criterion. 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 primary motivation for the methodology discussed below.

2.2 Dyadic trees

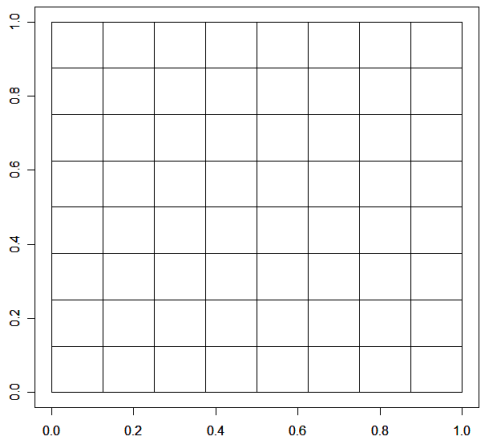


Figure 1: The largest unlabeled tree for $k_{max} = 3$

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 $2^{-k_{max}} = 1/8$.

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 Willet and Nowak (2007) is given as a linear combination of the empirical risk

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

where

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

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.

In particular, we seek to find the tree T that minimizes

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

where $0 < \rho < 1$ plays the role of a smoothing parameter. The larger the value of ρ, 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 γ. 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, possibly a degenerate tree (the estimated set is either the entire space or the empty 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.

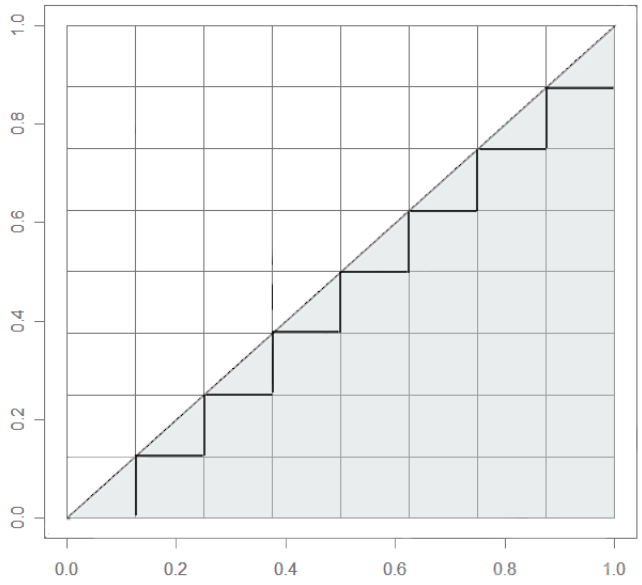


Figure 2: The shaded area is the true level set, the dark line is the boundary of the (not unique) optimal estimate in terms of symmetric difference

As a result, we realize that the precision of our estimate is both a function of k_{max} as well as ρ. Regardless of the size of k_{max}, if ρ 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 ρ, for any finite k_{max} the class of trees will never contain the true level set.

Consider Figure 3, where three values of ρ are considered for equally spaced data with no noise added. As a result, setting $\rho = 0$ yields the optimal estimate. The effect of increasing ρ 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. 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.

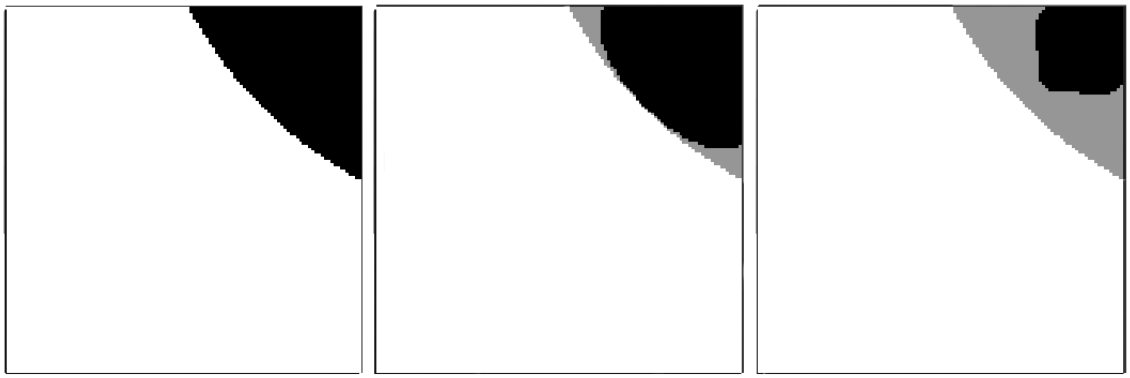


Figure 3: Three level set estimates with varying value of ρ (0,.06,.15). The true level set is essentially the first plot (gray in subsequent plots).

Unfortunately, choosing an appropriate value of ρ is not straight forward. In the application given by Willet 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 ρ values. The final ρ value was chosen to minimize the risk between the two sets. In practice, however, the true level set will not be known and ρ must be selected by some other means. A proper selection of ρ 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 ρ 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.' The premise is that the data is shifted slightly and a new estimate is attained. This is done repeatedly using different shifts. The final estimate consists of all points for which the majority of shifted estimates contained those point. It is clear from Figure 3 however that for improper selection of ρ, this will not help. Instead it will 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.

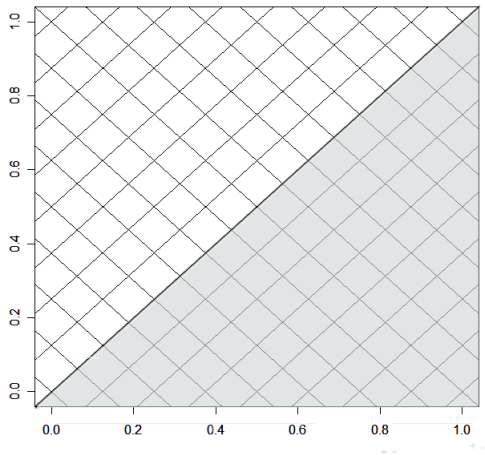


Figure 4: The tree rotated by $\pi/4$, the optimal estimated boundary is given by the darkened line.

3. Tree Warping

As further motivation beyond simple smoothing, the problems described above dealing with selection of ρ 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, we suspect the choice of ρ would not be so critical. 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 work under the assumption that the true level set is simply connected with a non-intersecting surface. The algorithm is as follows:

1. Fit a rough tree-estimate of the level set.

2. Smooth the estimated boundary.

3. Map a region of points close to the estimated boundary into $[0,1]^d$

4. Fit a 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.

Figure 5 illustrates nearly two complete iterations of the algorithm. A toy data set is used, with no noise added to the data. The true level set is a circle centered at the point $(0.5, 0.5)$. Panel (1) shows the data and a tree based estimate of the level set. Panel (2) shows the result of smoothing the estimate. In panel (3), only the points that lie within w of the boundary are pictured. These points are then mapped back into the unit square, and a tree based estimate is found in this space, as seen in panel (4). The tree based boundary in the warped space corresponds to the boundary pictured in Panel (5), which completes one iteration of the algorithm. The smoothing parameter, as discussed below, is set rather high for expository purposes. As a result, the algorithm is constantly dilating the level set in our original space in order to counteract the contraction resulting from the large smoothing parameter. This is seen in panel (8) with the estimated boundary being well below a split at 0.5 along the vertical axis, which would return the original boundary.

3.1 Initial Smoothing

We begin by assuming that $d = 2$. The boundary can be parametrized as a path in $\mathbb{R}^2$, 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. An extension to higher dimensions is straight forward.

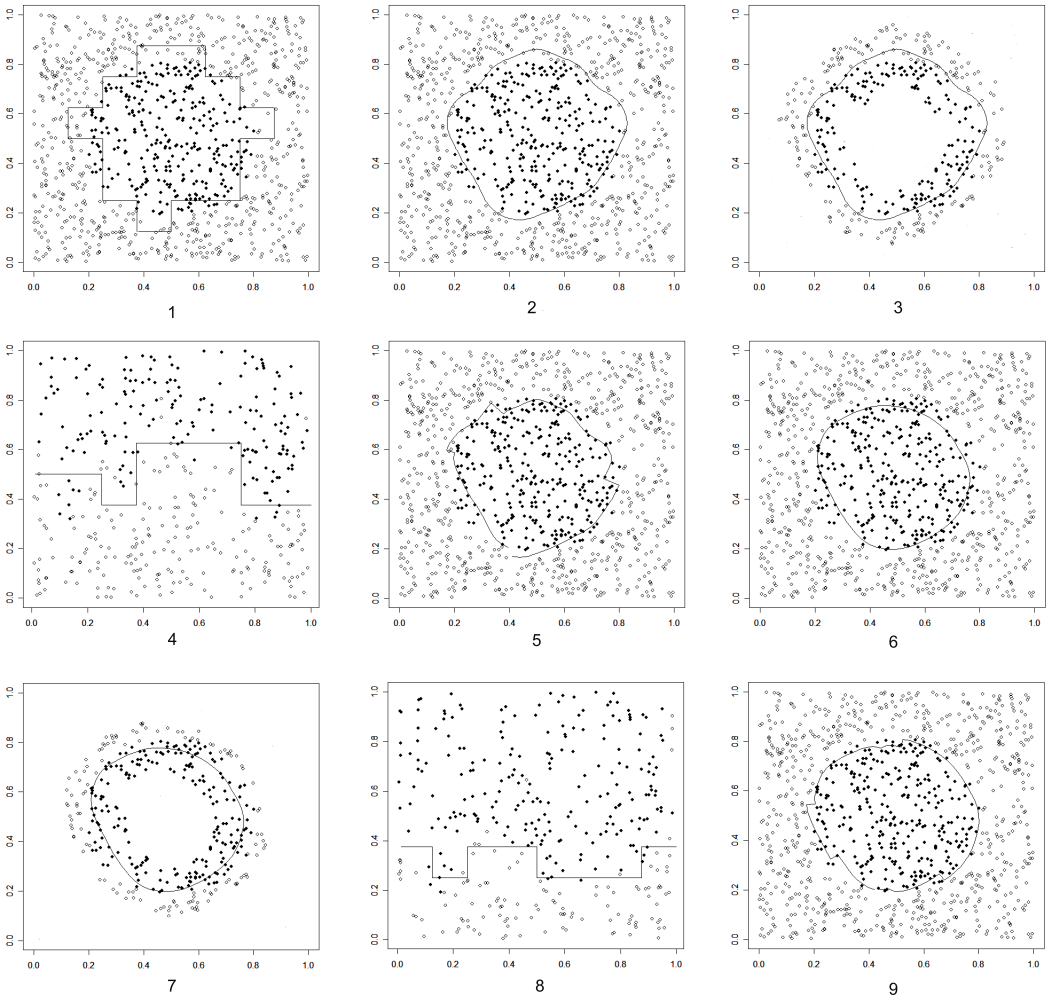


Figure 5: Darkened points exceed the threshold. 1: Initial Estimate. 2: $\tilde{z}_1(t)$. 3: $\mathbf{x} \in C(z_1, w)$. 4: Warped Data (with new tree estimate). 5: $z^*(t)$. 6. $z_2(t)$. 7. $\mathbf{x} \in C(z_2, w)$. 8: $\mathbf{y}$. 9. $z_3(t)$ (unsmoothed)

3.2 The warping map

The key element of the warp transformation is the invertible map ℓ_z. ℓ_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, for instance allowing us to use a small k_{max} in our current example. 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 $\mathbf{z} : [0,1]^{d-1} \to Z$ is an invertible map. If $C(z) \subseteq \mathbb{R}^d$ is the domain of ℓ_z, the key properties of $\ell_z \colon C(z) \to [0,1]^d$ are

i) $\ell_z(\mathbf{z}(\mathbf{t})) = (\mathbf{t}, 1/2)^T$.

ii) ℓ_z maps points in $C(z) \cap S^C$ to $[0,1]^{d-1} \times [0, 1/2)$.

iii) ℓ_Z maps 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 $\hat{\ell}_{z,w}$, the specific approximation we use when $d = 2$ and suggest a general approach to use for $d > 2$. 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 $\mathbf{z}(\mathbf{t}) = (z_1(\mathbf{t}), ..., z_d(\mathbf{t}))^T$. $\mathbf{P_Z}(\mathbf{x})$ 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 $\ell_{z,w} \colon C(z, w) \to [0,1]^d$. 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 $\mathbf{t}$, $\mathbf{r}(\mathbf{t})$. We will sometimes refer to $\mathbf{r}(\mathbf{t})$ as the "vector out." If we define $\mathbf{V}(\mathbf{t}^*) = (\frac{\partial z_i}{\partial t_j} \mid_{\mathbf{t}=\mathbf{t}^*})$ 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 $\ell_{z,w}^{-1} \colon [0,1]^d \to \mathbb{R}^d$ 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. The following procedure will work for any $d \geq 2$.

Our approximation depends on the use of Barycentric coordinates, see Warren (1996) for more detail. In short, if we have a $d-1$ dimensional simplex in $\mathbb{R}^d$ defined by the vertexes $p_1, \ldots, p_d$, we can specify a "plane" in $\mathbb{R}^d$, $A = \{\, \mathbf{x} \in \mathbb{R}^d \,|\, x = \sum_{i=1}^d \omega_i p_i \,\}$ where $\omega_i \in \mathbb{R}$. If we add the constraint $\omega_d = 1 - \sum_{i=1}^{d-1} \omega_i$ then each point in A can be represented by a unique $\omega = (\omega_1, \ldots, \omega_d)^T$. If $\mathbf{x} \in A$, and $\mathbf{x} = (p_1, \ldots, p_d)\omega$, then ω is called the Barycentric coordinates of $\mathbf{x}$. Barycentric coordinates have the useful property that if $0 \leq \omega_i \leq 1$

then $\mathbf{x}$ is inside (or on the boundary) of the simplex. If $f : \mathbb{R}^d \mapsto \mathbb{R}^q$ is some function, and p is some point inside of the simplex defined by $p_1, \ldots, p_d$ with Barycentric coordinates ω, then we can interpolate the value $f(p)$ with $\sum_{i=1}^{d} \omega_i f(p_i)$. I.e. we can linearly interpolate the value of a function across the simplex based on the function value at the corners.

We first estimate b points on $\mathbf{z}$, defined by $\hat{\mathbf{z}}(t_1), \ldots, \hat{\mathbf{z}}(t_b)$. Using these points as vertices, we tessellate the estimated surface with b^*-total $d-1$ dimensional simplexes. E.g. if $d = 2$, the surface is tessellated with line segments; if $d = 3$ the surface is tessellated with triangles; etc. Creating this tessellation can be tricky. Using these simplexes we divide $\hat{C}(z, w)$ into b^* regions, $\hat{C}(z, w)_1, \ldots, \hat{C}(z, w)_{b^*}$. If we use S_i to denote one of the $d-1$ dimensional simplexes, $\hat{C}(z, w)_i$ corresponds to the simplex S_i. Without loss of generality, let S_i be one of the $d-1$ dimensional simplexes. S_i consists of the points $p_1, \ldots, p_d$, and corresponding vectors $v_1, \ldots, v_d$. Each v_i is the estimated "vector out" at the point p_i. To see if a point $\mathbf{x} \in \mathbb{R}^d$ is transformed by $\hat{\ell}_{z,w}$ in the region $\hat{C}(z, w)_i$, first we calculate the plane

$$Y = \mathbf{x} + \begin{bmatrix} p_2 - p_1 & \cdots & p_d - p_1 \end{bmatrix} \beta$$

where $\beta \in \mathbb{R}^{d-1}$. Then calculate the points $p'_1, \ldots, p'_d$ where p'_i corresponds to the intersection of Y and the line $p_i + h_i\, v_i$ for some $h_i \in \mathbb{R}$. Next calculate ω, the Barycentric coordinates of $\mathbf{x}$ with respect to the simplex defined by $p'_1, \ldots, p'_d$. Finally if *all* $\omega_j \in [0, 1]$ and $\|\mathbf{x} - \begin{bmatrix} p_1 & \cdots & p_d \end{bmatrix} \omega\| \leq \frac{w}{2}$ then $\mathbf{x} \in \hat{C}(z, w)_i$ and $\mathbf{x} \notin \hat{C}(z, w)_i$ otherwise. If $\mathbf{x} \in \hat{C}(z, w)_i$ then

$$\hat{\ell}_{z,w}(\mathbf{x}) = \left(\begin{bmatrix} t_1 & \cdots t_d \end{bmatrix} \omega, 1/2 + \frac{\|\mathbf{x} - \begin{bmatrix} p_1 & \cdots & p_d \end{bmatrix} \omega\|}{w} * (-1)^{I(\mathbf{x} \notin \hat{S})} \right)^T.$$

Inverting a point from the transformed space is much simpler. Let $\mathbf{y} = (\mathbf{t}^T, h)^T$. Without loss of generality, let S_i denote the simplex with $p_1 = \hat{\mathbf{z}}(\mathbf{t_1}), \ldots, p_d = \hat{\mathbf{z}}(\mathbf{t_d})$ (and "vectors out" $v_1, \ldots, v_d$) such that $\mathbf{t}$ is in the simplex defined by $\mathbf{t_1}, \ldots, \mathbf{t_d}$. Let ω be the Barycentric coordinates of $\mathbf{t}$ with respect to $\mathbf{t_1}, \ldots, \mathbf{t_d}$. If we set

$$v' = \frac{\begin{bmatrix} v_1 & \cdots & v_d \end{bmatrix} \omega}{\| \begin{bmatrix} v_1 & \cdots & v_d \end{bmatrix} \omega \|},$$

then

$$\hat{\ell}^{-1}_{z,w}(\mathbf{y}) = \begin{bmatrix} p_1 & \cdots & p_d \end{bmatrix} \omega + v' \frac{w(h - 1/2)}{2}.$$

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 0.5 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 0.5, causing the estimate to dilate.

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 not too large value of ρ, at some point 'jumps' (splits in the horizontal direction) will be reinserted into the estimate in the warped space, as a more complicated tree will provide a better fit of the data in the warped space than the simple tree with a single split at 0.5. As a result, sharp corners in the estimate in the original space will be produced. 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.

4. Simulations and Applications

4.1 Regression Level Set Estimation

We consider applications of tree-warping to both the level set estimation problem discussed above as well as classification trees. Simulated data is used to show the versatility of the method, and a practical application follows. All programming was done in R (R Development Core Team, 2010). For the first simulations, we explore level set estimation in 2-dimensions. The independent variables $\mathbf{x}_i = (x_{i1}, x_{i2})$, $i = 1, \ldots, n$

($n = 1000$ for the first three simulations and $n = 2000$ for the fourth) are generated as uniform in the unit square. The response variable is generated according to the model as

$$Y = 2\ \mathrm{I}(\mathbf{x} \in S) + \epsilon,$$

with ϵ as standard normal, S is the set of interest and $\mathrm{I}(\cdot)$ is the indicator function. Letting the level $\gamma = 1$, S then is the level set, where the response variable has mean 2 in the set and mean 0 outside the level set. In each of the following figures, the subset of the unit square representing the level set is pictured along with the simulated data. The black points represent observations for which the response exceeds the threshold of 1 and the white points represent those whose response is below 1. Pictured along with these are the tree-based estimate of the level set and the warped estimate. 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 ρ. Repeatedly fitting trees with a small value of k_{max} seems to be significantly 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 0.02. 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 results from warping by using a large value of k_{max}, a well specified value of ρ (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 a number of iterations between 5 and 10. The value of ρ used was on the order of 0.0001, but in general will depend on the sample size and the behavior of the function near the boundary, among other things. A window of width 0.2 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 ρ.

The first two examples (Figures 6 and 7) have level set boundaries that are fairly smooth. For the first example, the smoothness is uniform across the boundary. With the second 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.

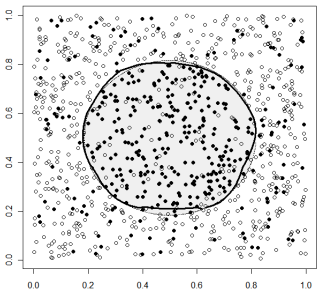


Figure 6: A circular level set with the tree-warped estimate. The estimate is given as the solid line. The darkened points are those which exceed the threshold.

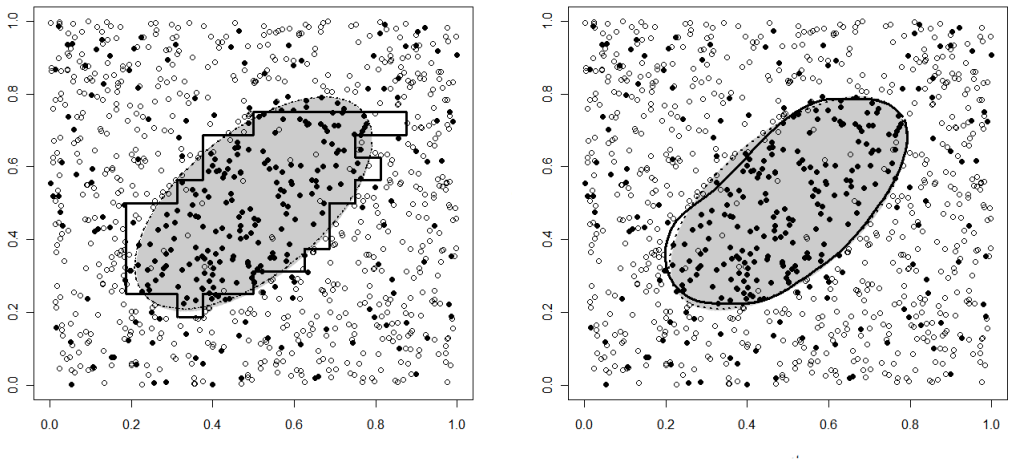


Figure 7: An elliptical level set with the unwarped (left) and tree-warped (right) estimates. The estimate is given as the solid line. The darkened points are those which exceed the threshold.

The fourth example uses a slightly more complicated level set boundary, as seen in Figure 8. While the 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.

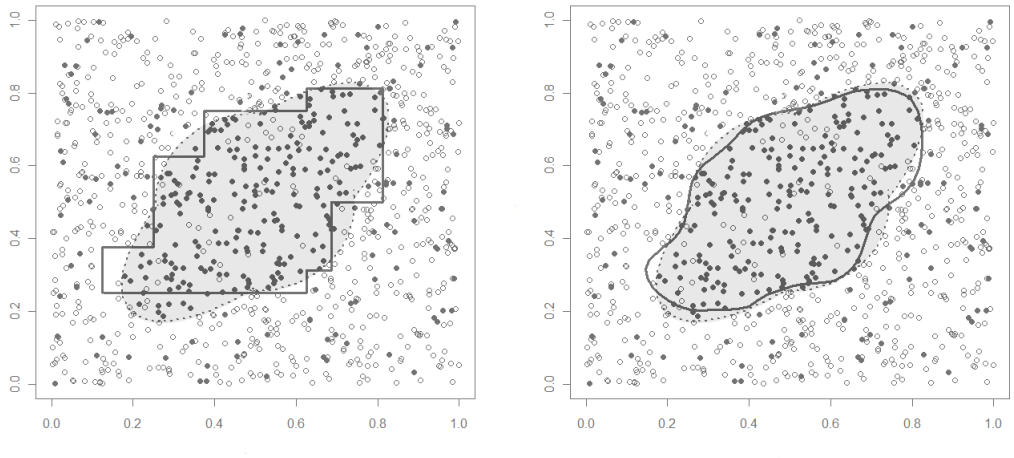


Figure 8: A non-convex level set with the tree-warped estimate. The estimate is given as the solid line. The darkened points are those which exceed the threshold.

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 ρ.

For the final 2-D level set simulation, shown in Figure 9, 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.

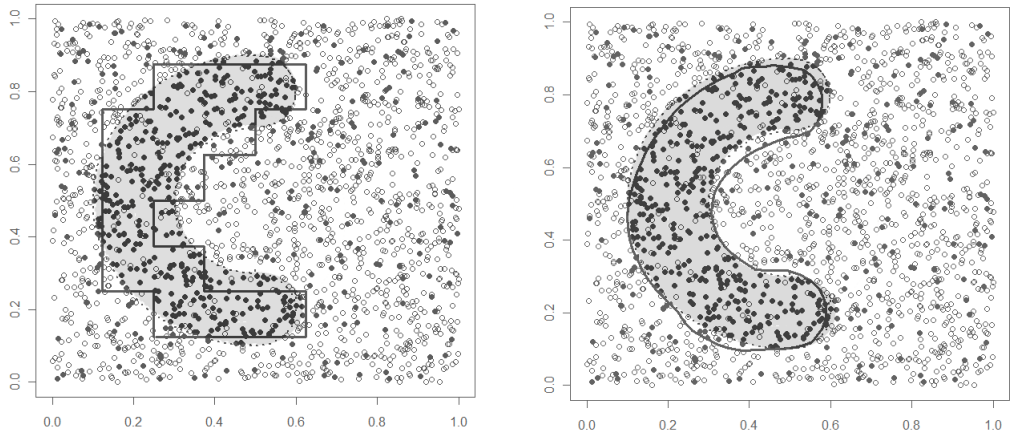


Figure 9: A horseshoe shaped level set with the tree-warped estimate. The estimate is given as the solid line. The darkened points are those which exceed the threshold.

4.2 Classification Trees

We also apply the warping methodology to classification trees using the *tree* function from the R library of the same name (Ripley, 2009). 1000 random points were generated uniformly in the unit square and assigned a label of 0. An additional 1000 points were generated uniformly on the subset of the unit square falling below the curve $y = \frac{sin(5x)+4}{8}$ and assigned a label of 1. The default choice for the complexity parameter for in pruning was used. Our goal was to estimate this boundary curve. To illustrate the power of the proposed method, Figure 11 shows the results after a single iteration of the algorithm. The bottom panel shows how well the warped boundary is approximated by a union of 'boxes', as desired.

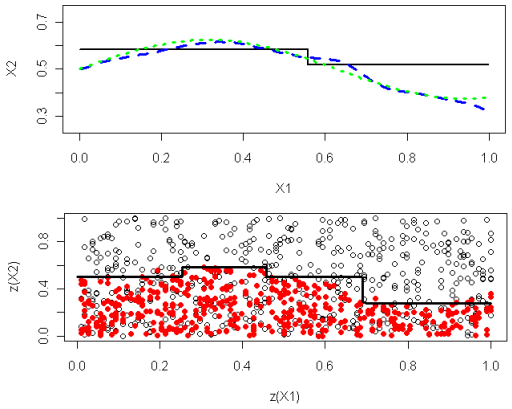


Figure 10: Top panel: The classification tree estimate (solid), the true curve (short dashed line) and the resulting warped estimate after one iteration (long dashed line). Bottom panel: The warped data (solid points have label 1) and classification tree estimate of the warped boundary.

Table 1 contains simulation results looking at the area of the symmetric difference between the estimated set for which the points with label 1 were generated, and its estimate, both by using a classification tree and using a classification tree and one warping iteration followed by an additional smoothing. As can be seen, one iteration resulted in, on average, a 58 percent reduction in the area of the symmetric difference. As the true boundary is quite smooth and the problem is somewhat 'easy', multiple iterations result in a very good estimate. In practice, the warping methodology is only as good as the underlying estimation methods the warping is applied to, though tree based methods are known to have a great many applications.

Table 1: Simulation results for classification tree estimation of set pictured in Figure 11. The measure used is the area of the symmetric difference between the true and estimated set.

Method	Mean	Standard Deviation
Classification Tree	.0576	.0195
Warped Tree	.0243	.0087

4.3 3D Demonstration

We also applied the warping method when $d = 3$ using a sphere of radius 0.25 centered at $(1/2, 1/2, 1/2)^T$ as the true level set using the function

$$f(x) = .6 * \mathrm{I}\left(\sqrt{(x-1/2)^2 + (y-1/2)^2 + (z-1/2)^2} \leq .25\right)$$

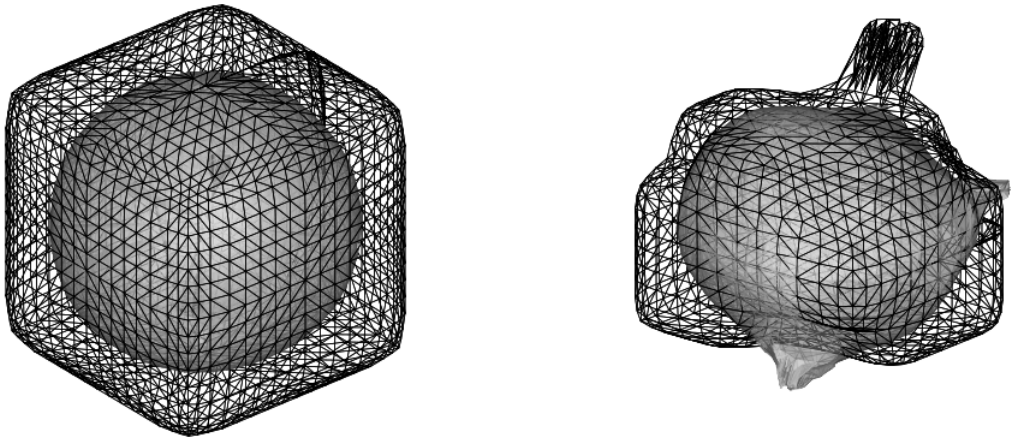
with $\gamma = 0.3$. We used 6 iterations of the warping method to illustrate the speed at which warping improves the estimates. Figure 11(a) shows the final surface estimation from a run when no noise was added to the image, the volume of the symmetric difference between the estimated set and the true level set is 0.0456 as opposed to 0.2371 for the (smoothed) estimate from the dyadic trees. We ran a simulation ($n_{sims} = 120$) with iid Normal noise ($\mu = 0$, $\sigma = 0.2$) added to the response at each data point. Each simulation consisted of drawing x, y and z independently from Uniform$(0, 1)$ distributions, accepting 2000 points inside the level set and 2000 points outside of the level set. Trees were fit using the dyadic tree estimation method of Willet and Nowak (2007). We used 6 iterations of the warping method. Table 2 summarizes the results. Tree warping resulted in an average decrease of 36% in the volume of the symmetric difference.

Table 2: Simulation results for 3D Demonstration. Gives the volume of the symmetric difference between the estimated and true level set.

Method	Mean	Standard Deviation
Original Tree	0.2549	0.0480
Warped	0.1602	0.0608

4.4 Application: Particulate Matter 2.5

Particulate Matter 2.5 ($PM_{2.5}$) is a pollutant consisting of "fine particles", particles that are 2.5 micrometers in diameter or smaller according to the EPA (2010a). The primary standard for $PM_{2.5}$ is $35\mu g/m^3$ in a 24-hour average (EPA, 2010b). $PM_{2.5}$ is measured at monitoring stations throughout the U.S. Interest in level-sets arises naturally from the question of what regions of the U.S. are near or over the safety limit.



(a) (b)

Figure 11: 3D Tree Warping demonstration. In each panel, the black wire frame shows the initial (smoothed) tree based estimate, the gray (slightly transparent) surface shows the estimate based on tree warping. (a) True level set is a sphere, no noise added. (b) True Level set is a sphere, noise added.

The monitoring stations are not located on a regular grid necessitating the use of advanced techniques for level-set estimation. Here we will estimate the level set S_{25}, the region of the U.S. where $PM_{2.5}$ met or exceeded $25\mu g/m^3$ on May 27, 2007, using the measurements available from the EPA (2010c).

In Figures 12 and 13, each point on the U.S. map represents a measurement station, the station is plotted with a circle if the 24 hour average of $PM_{2.5}$ is less than $25\mu g/m^3$ and a triangle if the 24 hour average of $PM_{2.5}$ is greater than or equal to $25\mu g/m^3$. The best possible empirical risk for this data set is -0.0551, calculated by estimating a point to be inside S_{25} if the measurement was greater than or equal to $\gamma = 25\mu g/m^3$ and outside otherwise. We estimated the level set using the dyadic trees of Willet and Nowak (2007) using $k_{max} = 3, \ldots, 6$ and using our tree warping method (using the dyadic trees with $k_{max} = 3$ as the base estimate). The empirical risk of the estimated level set using $k_{max} = 6$ is -0.0540 and as can be seen in Table 3, this is the first estimate to beat the empirical risk for the warped tree. However, if we visually compare Figure 12(d) to Figure 13(b) we can see that while the empirical risk for the dyadic tree with $k_{max} = 6$ is slightly lower, the complexity of the estimate is much higher and it seems likely that it is overfit. The smooth contour in Figure 13(b) achieves an empirical risk that is nearly as good (-0.0540 vs. -0.544) and is visually a much easier set to comprehend.

In order to substantiate our claims that our method is more robust to the choice of the smoothing

Table 3: Empirical Risks for level set estimates of the region where $PM_{2.5} \geq 25\mu g/m^3$.

Method	Empirical Risk
$k_{max} = 3$	-0.0418
$k_{max} = 4$	-0.0531
$k_{max} = 5$	-0.0539
Warped Tree	-0.0540
$k_{max} = 6$	-0.0544
Optimal	-0.0551

Table 4: Results for the $PM_{2.5}$ data set as a function of ρ and w. Original risk is the (empirical) risk resulting from the tree based estimate. $k_{max} = 4$ for all estimates.

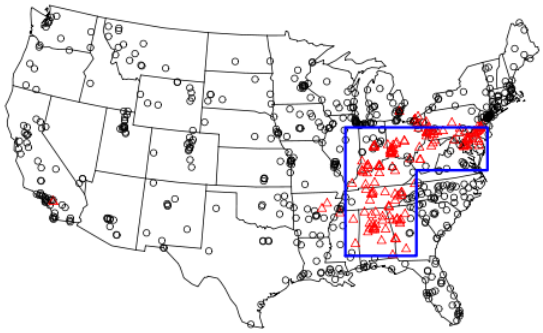
	Original Risk	Warped Risk	ρ	w
1	-0.05321	-0.05289	0.0002	0.05
2	-0.05319	-0.05360	0.0100	0.05
3	-0.04927	-0.04951	0.5000	0.05
4	-0.05321	-0.05349	0.0002	0.10
5	-0.05319	-0.05362	0.0100	0.10
6	-0.04927	-0.05286	0.5000	0.10
7	-0.05321	-0.05268	0.0002	0.15
8	-0.05319	-0.05323	0.0100	0.15
9	-0.04927	-0.05348	0.5000	0.15

parameter ρ, we ran the algorithm on the $PM_{2.5}$ data set with varying values of both ρ and the tunnel width w. Results for k_{max}=4 are provided in Table 4. The standard deviation in the empirical risk for the three tree estimates is 1.75 times higher than the resulting risks from the warped estimates, suggesting that the choice of ρ is more important for the tree estimate. We draw attention to the third run, where the warped estimate performed substantially worse than any other run. The combination of a large smoothing parameter and a small tunnel about the estimated boundary resulted in a boundary that tended to be over-smoothed (jumps are discouraged with a large value of ρ) and an inability to see the true boundary in the warped space, as the over-smoothing caused the estimated boundary and the true boundary to be different by more than w quite often. Larger values of w dealt with the large value of ρ much better. Removing this observation, the standard deviation of the empirical risk for the trees is 6 times that of the warped versions.

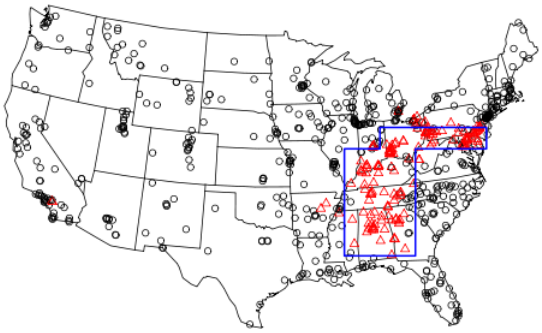
5. Discussion

5.1 Practical Issues

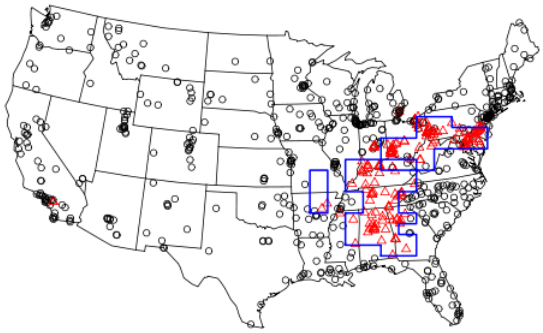
We mention a couple of issues that arose as a result of the simulations. Firstly, in the case of d=2, the location of $z(0)$ strongly determines the location of possible jumps in the warped space (as arclength(z)



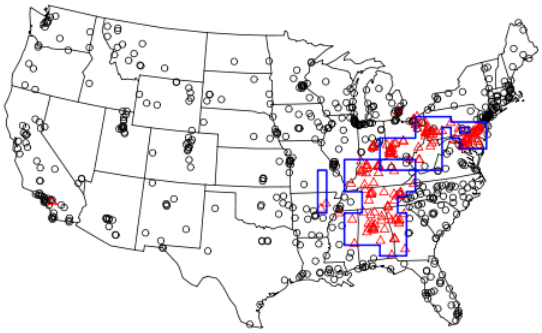
(a) $k_{max} = 3$



(b) $k_{max} = 4$



(c) $k_{max} = 5$

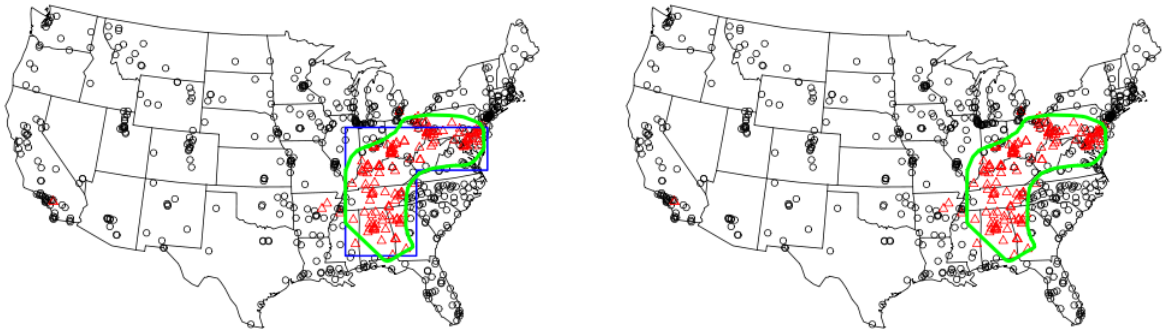


(d) $k_{max} = 6$

Figure 12: Estimated level set ($\text{PM}_{2.5}\text{g} \geq 25\mu/\text{m}^3$) for different values of k_{max}.

remains fairly constant through the iterative procedure), especially for small k_{max}. Especially when using small trees as we did above, it is 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. Similarly, it is helpful to alter the location of 'cuts' in higher dimensions.

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



(a) (b)

Figure 13: The warped estimate of the level set, displayed with the original tree estimate (a) and without (b)

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, assuming that the value of the smoothing parameter ρ is not chosen so large that it prohibits flexibility in the tree. One might also consider randomizing the value of w from iteration to iteration, adding extra flexibility into the method.

Thirdly, while this method seems to relax the importance of choosing the parameter ρ, some thought still needs to be given here. As we assume that the underlying set is connected, the parameter needs to be set large enough so that all tree estimates (the original as well as in the warped space) are themselves connected (its likely that a larger value of ρ is needed in the warped space, as we are only looking at a subset of the data here). A reasonable solution then seems to be to choose the parameter just large enough to ensure a connected estimate, as this will allow the method enough flexibility to adjust to the data.

One might then wonder about the method over-fitting the data. We point out that our suggestion, in the case of level set estimation, is to use relatively small values of k_{max} (3 or 4). As such, there is little opportunity for the estimate to pick out individual data points, as the result of jumps in the estimate are that large parts of the estimated boundary are adjusted.

We mention above that the smoothing bandwidth b should be set to a 'small' value, and beyond this the

choice of the actual smoothness level is automatic. The smaller one chooses b to be, the longer the algorithm is going to need to be run before a suitable estimate of the set is attained, so computationally this choice becomes important. While we lack firm guidance on choosing this value, its clear that b should be chosen to be smaller than the minimum amount of smoothness desired in your final estimate.

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 largest issue with estimating $\hat{\ell}_{z,w}$ after obtaining the points to estimate the surface is estimating the values of t and creating the tessellation. That is, once we have estimated $\hat{\mathbf{z}}(t_j)$, what is the best value of t_j, and what is the "best" tessellation of the estimated surface? For $d = 2$, tessellation is usually straightforward, as the points should be organized into a curve. And the t_js can be chosen using the arclength of the estimated surface, i.e.

$$t_{j+1} - t_j = \frac{\|\hat{\mathbf{z}}(t_{j+1}) - \hat{\mathbf{z}}(t_j)\|}{\text{Total Arclength}}$$

When $d = 3$, the choice of t_j is less straightforward. Our implementation used the spherical coordinates of $\hat{\mathbf{z}}(t_j)$ to establish the value of t_j, and Delaunay Triangulation (see Hjelle and Daehlen (2006)) to create the surface tessellation. For $d \geq 4$, there is no useable method for establishing the values of t_j.

5.2 Complexity

Computationally, our method uses an underlying tree based method in conjunction with locally 'planar' approximations to the smoothed estimate, repeated across iterations. At each iteration, a tree estimation is performed, and a warping is preformed. Thus the computational complexity for an iteration is the sum of the two. This is repeated M times, so the total computational complexity is $O(M(treecomplexity+warpcomplexity))$. The overall complexity will depend on the underlying complexity of the two components. In the case of the dyadic tree based of Willet and Nowak (2007), estimating the tree has complexity $O(ndk_{max}^d \log(nk_{max}^d))$. The naive implementation of our barycentric oriented warping math has complexity of $O(snd^3)$ where s is the number of simplexes in the tesselation, because each point has to be checked against each simplex, and doing so requires a matrix inversion. However, in reality the warp complexity is much better than $O(snd^3)$

for $d = 2$ and $d = 3$ since the implementation does not require matrix inversions for each simplex and point check. The barycentric computations can be done using dot products if $d = 2$ and using cross products if $d = 3$, both of which remove the d^3 from the above estimate. Since the tree warping approach allows us to keep k_{max} relatively small, we may be able to efficiently achieve a better estimate in higher dimensions.

The only major hurdle for our tree warping map is determining the surface parameterization. Surface parameterization is still a difficult problem with much active research, as can be seen in Floater and Hormann (2005). As such, we have thus far only implemented our approach for $d = 2, 3$.

5.3 Conclusion

We propose a method for creating smooth estimates of set boundaries based on tree-based methods and an idea similar to the kernel 'trick' used in support vector machines. The resulting estimate has smoothness that is both locally adaptive and chosen automatically by the data. The goal is to lessen the dependency on the choice of tuning parameters, a problem that is not unique to tree based methods. Support vector machines, radial basis function networks and other ideas all seem to depend heavily on the choice of these parameters, as well as, for instance, the kernel itself in the case of SVMs. While our method has some dependency on the selection of tuning parameters, the dependency appears to be less than the underlying tree based methods we study.

References

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., Schäfer, 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.

EPA (2010a). Particulate Matter. `http://www.epa.gov/pm/`

EPA (2010b). National Ambient Air Quality Standards (NAAQS). `http://epa.gov/air/criteria.html`

EPA (2010c). Download Detailed AQS Data. http://www.epa.gov/ttn/airs/airsaqs/detaildata/downloadaqsdata.htm

Floater, M.S., and Hormann, K. (2005). Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling*, N. A. Dodgson, M. S. Floater, and M. A. Sabin, Eds., Mathematics and Visualization pages 157–186. Springer, Berlin, Heidelberg.

Hastie, T., Tibshirani, R. and Friedman, J. (2001). *The Elements of Statistical Learning - Data Mining, Inference, and Prediction*, New York: Springer.

Hjelle, Ø. and Daehlen, M. (2006). *Triangulations and applications*, Berlin: Springer-Verlag.

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.

R Development Core Team (2010). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. Vienna, Austria. http://www.R-project.org

Brian Ripley. (2009). tree: Classification and regression trees. R package version 1.0-27. http://CRAN.R-project.org/package=tree

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

Warren, J. (1996). Barycentric Coordinates for Convex Polytopes. In *Advances in Computational Mathematics*, Twizel, E. H. editor, volume 6, pages 97–108. Springer U.S..

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