Nonparametric Regression on a Graph

Arne Kovac and Andrew D.A.C. Smith

Abstract

The ‘Signal plus Noise’ model for nonparametric regression can be extended to the case of observations taken at the vertices of a graph. This model includes many familiar regression problems. This article discusses the use of the edges of a graph to measure roughness in penalized regression. Distance between estimate and observation is measured at every vertex in the L_2 norm, and roughness is penalized on every edge in the L_1 norm. Thus the ideas of total-variation penalization can be extended to a graph. The resulting minimization problem presents special computational challenges, so we describe a new, fast algorithm and demonstrate its use with examples.

The examples include image analysis, a simulation applicable to discrete spatial variation, and classification. In our examples, penalized regression improves upon kernel smoothing in terms of identifying local extreme values on planar graphs. In all examples we use fully automatic procedures for setting the smoothing parameters. Supplemental materials are available online.

KEY WORDS: Penalized regression; Total variation; Image analysis; Active set algorithm

Arne Kovac is Lecturer, Department of Mathematics, University of Bristol, Bristol BS8 1TW, UK (E-mail: A.Kovac@bristol.ac.uk). Andrew D.A.C. Smith is PhD student, Department of Mathematics, University of Bristol, Bristol BS8 1TW, UK (E-mail: Andrew.D.Smith@bristol.ac.uk). Andrew Smith’s research is funded by EPSRC.

1 INTRODUCTION

There are a number of statistical models that contain some sort of graphical structure. Examples include image analysis, disease risk mapping and discrete spatial variation. We focus on those for which penalized regression is appropriate, and can be thought of in terms of the 'signal + noise' framework.

We consider the regression of a continuous response variable on one or more explanatory variables. Often there is some sort of graphical structure in and between the observations, or some obvious neighboring scheme that gives rise to a graph. We think of the locations of the observations as the vertices of the graph. The edges may be suggested by the neighboring scheme or by explanatory observations, if they exist. We will see some examples in this section.

A model for data on the graph $(\mathcal{V}, \mathcal{E})$, which has vertices in the set $\mathcal{V}$ and edges in the set $\mathcal{E}$, is

$$\begin{array}{ccccccc} \text{Data} & = & \text{Signal} & + & \text{Noise} & & \\ y_i & = & f_i & + & \sigma z_i, & & i \in \mathcal{V}. \end{array}$$

The noise terms, z_i, are usually assumed to be independent realizations of a random variable with zero mean and unit variance. Under this model regression on a graph involves estimating the underlying signal values f_i, based on the observations y_i, at all vertices i in the set $\mathcal{V}$. We assume that f describes a smooth function on the graph, e.g. $\sum_{(i,j)\in\mathcal{E}} |f_j - f_i|$ is small. Hence we use the edges to measure the complexity of the estimate.

Figure 1 shows an example of regression on a graph: a small, noisy image with 64 pixels. The responses are the grey levels of the pixels, so each pixel is a vertex of the graph. A natural choice of edges connects each pixel with its neighbors, resulting in the graph superimposed on the left-hand image in Figure 1. Regression on this graph involves estimating the underlying signal image, which is

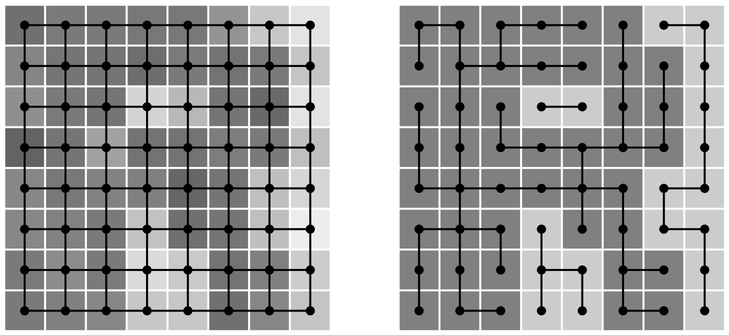

Figure 1: Example of a graphical structure present in a regression situation. The noisy image (left) shows a suitable graph for regression, based on the 4-neighborhood. On the noiseless version (right) only the edges in the active set are shown.

displayed in the right-hand image.

In this article we discuss penalized regression on the graph $(\mathcal{V}, \mathcal{E})$ and the estimate that minimizes

$$Q(f) := \frac{1}{2} \sum_{i \in \mathcal{V}} w_i (f_i - y_i)^2 + \sum_{(i,j) \in \mathcal{E}} \lambda_{i,j} |f_j - f_i|$$

for appropriate weights $w_i \geq 0$, for $i \in \mathcal{V}$, and smoothing parameters $\lambda_{i,j} > 0$, for $(i, j) \in \mathcal{E}$. This is the sum of a term that penalizes distance from the data plus a term that penalizes roughness. The first term is the distance from the data, measured at every vertex in the L_2 norm. The second term is the weighted sum of roughness at every edge, measured in the L_1 norm. Our model allows for a different weight or smoothing parameter at each vertex and each edge.

Although it is usual, in graph theory, to denote the edges by unordered pairs, we will treat $\mathcal{E}$ as a set of ordered pairs for convenience of notation. This does not mean that $(\mathcal{V}, \mathcal{E})$ is a directed graph, since the ordering can be completely arbitrary. We do, however, consider there to be at most one edge that joins any pair of vertices. This is because it makes no sense to split the penalty between two vertices over more than one edge.

It is possible to have multiple observations at every vertex. Suppose that we have n_i observations $y_{i1}, \dots, y_{in_i}$ with associated weights $w_{i1}, \dots, w_{in_i}$ for all vertices $i \in \mathcal{V}$. Then minimizing

$$\frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{k=1}^{n_i} w_{ik}(f_i - y_{ik})^2 + \sum_{(i,j)\in\mathcal{E}} \lambda_{i,j}|f_j - f_i|$$

is equivalent to minimizing $Q(f)$ with weights $w_i = \sum_{k=1}^{n_i} w_{ik}$ and observations $y_i = \sum_{k=1}^{n_i} w_{ik}y_{ik}/w_i$. This includes the special case where $n_i = 0$, which might happen if there are missing observations, or vertices at which we wish to predict the response. Therefore vertices without observations are given zero weight and the minimization of $Q(f)$ provides an estimate at all vertices that have observations and a prediction at all vertices that do not.

1.1 Motivating examples

As a first motivating example, we consider the problem of nonparametric regression between two continuous variables. Suppose we have response observations $y_1, \dots, y_n$ taken at strictly ordered design points. There is a natural neighboring structure: the first observation is adjacent to the second, the second next to the third, and so on. Hence a natural graphical structure is given by $(\mathcal{V}_2, \mathcal{E}_2)$, where

$$\mathcal{V}_2 = \{1, 2, \dots, n\} \text{ and } \mathcal{E}_2 = \{(1,2), (2,3), \dots, (n-1, n)\}.$$

The minimization of $Q(f)$ provides an estimate of f_i at every observation. If we let $w_i = 1$ for all $i \in \mathcal{V}_2$ and use the convenient shorthand $\lambda_i = \lambda_{i,i+1}$, then $Q(f)$ becomes

$$\frac{1}{2}\sum_{i=1}^{n}(y_i - f_i)^2 + \sum_{i=1}^{n-1}\lambda_i|f_{i+1} - f_i| \tag{1}$$

and the roughness penalty is the weighted total variation of the estimate.

Total variation can be extended to higher dimensions to tackle, for example, image analysis. An image can be thought of as an $n_1 \times n_2$ grid of pixels, with observations at each pixel. Then the set of vertices of the graph is the set of pixels

$$\mathcal{V}_4 = \{(i_1, i_2) : i_1 = 1, \ldots, n_1, i_2 = 1, \ldots, n_2\}.$$

There are a number of neighboring structures in use in image analysis. The simplest is the 4-neighborhood (Winkler 2003, p. 57) in which a pixel has neighbors immediately above, below, to the left and to the right. This neighboring scheme suggests the set of edges

$$\mathcal{E}_4 = \left\{((i_1, i_2), (i_1, i_2 + 1)) \in \mathcal{V}_4^2\right\} \cup \left\{((i_1, i_2), (i_1 + 1, i_2)) \in \mathcal{V}_4^2\right\}.$$

Figure 1 shows a picture of this graph. Using $(\mathcal{V}_4, \mathcal{E}_4)$, we can find a denoised image by minimising $Q(f)$. Now the roughness penalty is a measure of the total variation in the horizontal direction plus the total variation in the vertical direction.

1.2 Review of existing methods

Mammen and van de Geer (1997) first discussed the estimator obtained by minimising (1) where λ is a global smoothing parameter. Some authors have allowed the smoothing parameters to differ. For example Davies and Kovac (2001) alter them during their local squeezing procedure. There are fast algorithms that find the solution to this specific minimization problem, in particular the taut string algorithm of Davies and Kovac (2001), which has $O(n)$ computational complexity.

The estimator that minimizes (1), in which error is measured in the L_2 norm and roughness in the L_1 norm, is a nonparametric version of the lasso (least absolute shrinkage and selection operator) estimator (Tibshirani 1996). Therefore the estimator that minimizes $Q(f)$ can be seen as a generalization of the non-parametric lasso to any graph. There are other methods of penalized regression, with different roughness measures, that have been applied to observations on a graph. Belkin, Matveeva and Niyogi (2004) describe an algorithm for Tikhonov regularization. Their algorithm measures roughness at every edge in the L_2 norm.

Koenker and Mizera (2004) employ a penalty term for triograms. Given irregularly-spaced observations, they create a graph by computing a Delaunay triangulation of the observations. Their penalty term is also a weighted sum over all edges of the triangulation. However they measure roughness as the squared (L_2) differences between gradients. Jansen, Nason and Silverman (2009) have discussed wavelet lifting as a method for regression on a graph. Like Koenker and Mizera, the authors use a Delaunay triangulation.

All of the methods above require the selection of a smoothing parameter. In the case of minimizing $Q(f)$ this means choosing w_i for $i \in \mathcal{V}$ and $\lambda_{i,j}$ for $(i, j) \in \mathcal{E}$, but simpler estimators may have only one smoothing parameter. There are many different automatic ways to choose the smoothing parameter, including cross-validation and the multiresolution criterion (Davies and Kovac, 2001). In Section 4.1 we describe an automatic choice attributed to Rudin, Osher and Fatemi (1992).

In the context of Bayesian statistics, Besag, Green, Higdon and Mengersen (1995) examine pairwise interaction Markov random fields, which describe a prior distribution on the edges of an undirected graph. The special case of an L_1 prior was discussed by Besag (1989) and leads to the minimization of $Q(f)$, which is solved by a probabilistic algorithm, such as Markov chain Monte Carlo.

Observations on the vertices of a graph have been studied in the literature on semi-supervised or transductive learning. Regression on a graph is a version of the metric labeling problem (Kleinberg and Tardos 2002). Most graph-labeling methods are concerned with classification, such as graph mincuts (Blum and Chawla 1998), label propagation (Zhu and Ghahramani 2002) and the perceptron (Herbster and Pontil 2006). However there are some methods, such as spectral graph transducer (Joachims 2003), harmonic energy minimization (Zhu, Ghahramani and Lafferty 2003), label spreading (Zhou, Bousquet, Lal, Weston and Scholkopf 2004) and the work of Culp and Michailidis (2008) that may be adapted to regression. Indeed the work by Belkin, Niyogi and Sindhwani (2006) on manifold regularization and Culp, Michailidis and Johnson (2009) on self-training is formulated within the regression framework, although the examples are of classification.

Our algorithm is based on ideas similar to active set methods, which features in a number of algorithms, including that of Goldfarb and Idnani (1983).

2 OPTIMIZATION ALGORITHM

In Theorem 1 below we give necessary and sufficient conditions for f to minimize $Q(f)$ and in Subsection 2.2 we present a fast algorithm for finding such a minimizer. The minimum exists because $Q(f)$, as a sum of convex functions, is convex itself although not necessarily strictly convex. Therefore any local minimum of $Q(f)$ will be a global minimum, and the set of all global minima will be a convex set. In the important case where all the weights w_i are strictly positive a unique global minimum exists, because $Q(f)$ is strictly convex.

2.1 Necessary and sufficient condition for minimization

The solution to the minimization problem is characterized by *regions of constant value*, that is, sets of neighboring vertices that share the same value of f. We define such regions by use of a special *active set* of edges, indexed by $\mathcal{A}$. This consists of edges $(i, j) \in \mathcal{E}$ for which $f_i = f_j$, such that the graph $(\mathcal{V}, \mathcal{A})$ is acyclic. Note that, unlike the definition of active set used in many optimization algorithms, there can still be edges $(i,j) \notin \mathcal{A}$ such that $f_i = f_j$.

We will denote by $\mathcal{R}(k)$ the entire region of constant value that contains the vertex k. More formally let

$$\mathcal{R}(k) = \{i \in \mathcal{V} : i \text{ is connected to } k \text{ in } (\mathcal{V}, \mathcal{A})\}.$$

We will also denote by $\mathcal{A}(k)$ that subset of the active set that holds the region $\mathcal{R}(k)$ together, so

$$\mathcal{A}(k) = \{(i,j) \in \mathcal{A} : i \in \mathcal{R}(k), j \in \mathcal{R}(k)\}.$$

Figure 1 shows an example of an active set in the graph $(\mathcal{V}_4, \mathcal{E}_4)$. Note how the edges in the active set join together vertices that share the same value, thus holding together regions of constant value.

Since $(\mathcal{V}, \mathcal{A})$ is acyclic, the graph $(\mathcal{R}(k), \mathcal{A}(k))$ is a connected, acyclic graph. This feature is crucial as it allows the region $\mathcal{R}(k)$ to be split into two subregions by removing just one edge (I, J) from $\mathcal{A}(k)$. We will denote these two subregions by $\mathcal{R}(I, J)$ and $\mathcal{R}(J, I)$, where

$$\mathcal{R}(I,J) = \{i \in \mathcal{R}(I) : i \text{ is connected to } I \text{ in } (\mathcal{V}, \mathcal{A} \setminus (I,J))\}.$$

We associate with the (sub)region $\mathcal{R}(a)$ (where $a = k$ or $a = I, J$) the quantities

$$m_a = \sum_{i \in \mathcal{R}(a)} \left(w_i y_i + \sum_{j:(i,j) \in \mathcal{E}} c_{i,j} \lambda_{i,j} - \sum_{j:(j,i) \in \mathcal{E}} c_{j,i} \lambda_{j,i} \right) \text{ and } u_a = \sum_{i \in \mathcal{R}(a)} w_i.$$

Theorem 1 *A fit f minimizes $Q(f)$ if and only if there are values $c_{i,j}$ and a set of edges $\mathcal{A}$, such that $(\mathcal{V}, \mathcal{A})$ is acyclic, $f_i = f_j$ for all $(i,j) \in \mathcal{A}$ and the following* *conditions hold:*

$$c_{i,j} = \operatorname{sign}(f_j - f_i) \text{ or } f_i = f_j \text{ for all } (i,j) \in \mathcal{E}, \tag{2}$$

$$|c_{i,j}| = 1 \text{ if } (i,j) \in \mathcal{A}, \tag{3}$$

$$u_k f_k = m_k \text{ for all } k \in \mathcal{V}, \tag{4}$$

$$\text{and} \quad |u_{I,J} f_I - (m_{I,J} - c_{I,J}\lambda_{I,J})| \leq \lambda_{I,J} \text{ for all } (I,J) \in \mathcal{A}. \tag{5}$$

A proof is given in the Appendix. These conditions can be shown to be similar to the taut string of Davies and Kovac (2001). When the graph is $(\mathcal{V}_2, \mathcal{E}_2)$ the condition (5) describes a tube and (4) describes a string threaded through the tube and pulled taut (Mammen and van de Geer 1997).

2.2 Algorithm

The algorithm that we describe below can be considered to search for the graph $(\mathcal{V}, \mathcal{A})$ and vector c described in Theorem 1, and hence the minimizer f of $Q(f)$. To explain it fully we must define the working objective function

$$Q(f;c) := \frac{1}{2}\sum_{i\in\mathcal{V}} w_i(f_i - y_i)^2 + \sum_{(i,j)\in\mathcal{E}} |c_{i,j}|\lambda_{i,j}|f_j - f_i|,$$

which is parameterized by c. A slightly modified version of Theorem 1 says that for f to minimize $Q(f;c)$ we must have

$$\operatorname{sign}(c_{i,j}) = \operatorname{sign}(f_j - f_i) \text{ when } f_i \neq f_j, \tag{6}$$

(3) and (4) must hold, and

$$0 \leq -\operatorname{sign}(c_{I,J})(u_{I,J} f_I - m_{I,J}) \leq 2|c_{I,J}|\lambda_{I,J} \text{ for all } (I,J) \in \mathcal{A}. \tag{7}$$

During the course of the algorithm, the current value of f will always minimize $Q(f;c)$. The algorithm changes c until (2), a stronger version of (6), is satisfied.

When this occurs (7) becomes equivalent to (5) and the value of f that minimizes $Q(f;c)$ will also minimize $Q(f)$. This event will occur in a finite time, as stated by Theorem 2 below, which is proven in the Appendix.

Theorem 2 *The algorithm described below will terminate in a finite time, and find a minimizer of $Q(f)$, for any graph, data, weights and smoothing parameters.*

Step 0 Set $c = 0$. Then $Q(f;c) = \frac{1}{2}\sum_{i\in\mathcal{V}} w_i(f_i - y_i)^2$, so set $f = y$ as this is a minimizer. Let $\mathcal{A}$ be empty.

Step 1 Choose an edge for which (2) is not satisfied, i.e. choose an edge $(k,l) \in \mathcal{E}$ such that $f_k \neq f_l$ and $c_{k,l} = 0$.

The current value of f minimizes $Q(\,\cdot\,;c)$.

Step 1.1 For every possible change to the active set (listed in Subsections 2.2.1–4; no change, merging, amalgamation, splitting) calculate the corresponding step sizes δf_k and δf_l.

Step 1.2 Choose the event for which $|\delta f_k|$ and $|\delta f_l|$ are both smallest. Let $\Delta f_i = \delta f_k$ for $i \in \mathcal{R}(k)$, $\Delta f_i = \delta f_l$ for $i \in \mathcal{R}(l)$ and $\Delta f_i = 0$ otherwise, and $\Delta c_{i,j} = 0$ for $(i,j) \neq (k,l)$ and $\Delta c_{k,l} = u_k \delta f_k/\lambda_{k,l}$.

At this moment $f + \delta f$ minimizes $Q(\,\cdot\,;c+\delta c)$.

Step 1.3 Update $f \leftarrow f + \delta f$ and $c \leftarrow c + \delta c$. If $\mathcal{R}(k)$ and $\mathcal{R}(l)$ have been merged, add (k,l) to $\mathcal{A}$ and let $c_{k,l} = 1$. If $\mathcal{R}(k)$ or $\mathcal{R}(l)$ has been amalgamated with a neighboring region, then add the relevant edge to $\mathcal{A}$. If $\mathcal{R}(k)$ or $\mathcal{R}(l)$ has been split at an edge (I,J), remove this edge from $\mathcal{A}$ and let $c_{I,J} = \operatorname{sign}(f_J - f_I)$.

Now f minimizes $Q(\,\cdot\,; c)$.

Repeat Steps 1.1–3 until $|c_{k,l}| = 1$. This means that (2) is satisfied for (k, l), and will remain satisfied for this edge.

Repeat Step 1 until there is no such edge. Now (2) is fully satisfied, and we have $Q(f; c) \equiv Q(f)$. Since f minimizes $Q(f; c)$ it also minimizes $Q(f)$.

Our algorithm gradually increases the penalty on each edge. As $c_{k,l}$ changes, the penalty on the edge (k, l) increases, so we must reduce $|f_l - f_k|$ in order to move to the minimum of $Q(\,\cdot\,; c + \delta c)$. This change must take place within the constraints of the active set. Therefore we alter f_k and f_l uniformly on the whole of the regions $\mathcal{R}(k)$ and $\mathcal{R}(l)$.

As the regions move closer together there may need to be changes to the active set. To make sure that these changes happen we increase the penalty on (k, l) in small steps. We alter f and c only enough to trigger the first change in the active set. We describe the possible changes below, giving the associated values of δf_k and δf_l and conditions for each event to be possible. The Appendix contains proofs of these values.

2.2.1 No change to active set

There may be no disruption necessary to the active set before $c_{k,l} + \delta c_{k,l} = \text{sign}(f_l - f_k)$ is satisfied. This event can only occur if $u_k > 0$ and $u_l > 0$. The associated changes in f_k and f_l are

$$\delta f_k = \frac{(\text{sign}(f_l - f_k) - c_{k,l})\lambda_{k,l}}{u_k} \text{ and } \delta f_l = \frac{(\text{sign}(f_k - f_l) + c_{k,l})\lambda_{k,l}}{u_l}.$$

2.2.2 Merging of the two regions

Before we reach the target value of $c_{k,l} = \operatorname{sign}(f_l - f_k)$, the regions $\mathcal{R}(k)$ and $\mathcal{R}(l)$ might meet each other in value. This would mean that $|f_l - f_k|$ can be decreased no further. The changes in f_k and f_l are

$$\delta f_k = \frac{u_l}{u_k + u_l}(f_l - f_k) \text{ and } \delta f_l = \frac{u_k}{u_k + u_l}(f_k - f_l). \tag{8}$$

If $u_k = u_l = 0$ then we can choose $\delta f_k = (f_l - f_k)/2$ and $\delta f_l = (f_k - f_l)/2$.

Since we now have $f_k = f_l$ we merge the two regions $\mathcal{R}(k)$ and $\mathcal{R}(l)$ by adding (k, l) to the active set. If there are other edges that join $\mathcal{R}(k)$ and $\mathcal{R}(l)$, then they will not be added to $\mathcal{A}$, even though they share the same value of f. This will ensure that the graph $(\mathcal{V}, \mathcal{A})$ remains acyclic.

2.2.3 Amalgamation of a neighboring region

Before we reach the minimizer of $Q(\,\cdot\,; c + \delta c)$, the value of f in the region $\mathcal{R}(k)$ may meet the value in a neighboring region that is not $\mathcal{R}(l)$. More formally there may be a vertex $i \in \mathcal{R}(k)$ and $K \notin \mathcal{R}(k) \cup \mathcal{R}(l)$ for which $c_{i,K} \neq 0$ or $c_{K,i} \neq 0$, and $f_k \leq f_K < f_l$ or $f_k \geq f_K > f_l$.

This event is only possible if $u_l > 0$, or if $u_l = u_k = 0$, or if $u_l = 0$ and $f_K = f_k$. The changes to f associated with this event are

$$\delta f_k = f_K - f_k \text{ and } \delta f_l = \begin{cases} u_k(f_k - f_K)/u_l & u_l > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

We now have $f_i = f_K$ and if we proceed to alter f we may break the constraint (6) at the edge (i, K) or (K, i). Therefore, if $\operatorname{sign}(c_{i,K}) = \operatorname{sign}(\delta f_k)$ or $\operatorname{sign}(c_{K,i}) = -\operatorname{sign}(\delta f_k)$, we add this edge to the active set. This will amalgamate the region $\mathcal{R}(K)$ into $\mathcal{R}(k)$. If there are other edges that join $\mathcal{R}(k)$ and $\mathcal{R}(K)$ then they will not be added to $\mathcal{A}$. This ensures that the graph $(\mathcal{V}, \mathcal{A})$ remains acyclic. Of course a similar amalgamation might occur with a neighbor of $\mathcal{R}(l)$.

2.2.4 Splitting a region

Before arriving at the minimizer of $Q(\,\cdot\,; c + \delta c)$ we must test whether an edge $(I, J) \in \mathcal{A}(k) \cup \mathcal{A}(l)$ should be removed from the active set. This will split the region $\mathcal{R}(k)$ or $\mathcal{R}(l)$ into two subregions. If the split takes place it may be necessary to swap the sign of $c_{I,J}$, in order to preserve the constraint (6) at (I, J). This will not affect $Q(f; c)$. We use condition (7) to tell us when an edge should be removed, once we have accounted for the possible sign change.

This event can only occur if $u_k > 0$ and $u_l > 0$. The values of f and c at which $(I, J) \in \mathcal{A}(k)$ should be removed are given by

$$\delta f_k = \frac{m_{I,J} - u_{I,J} f_k - c_{I,J}\lambda_{I,J} \pm \operatorname{sign}(f_l - f_k)|c_{I,J}|\lambda_{I,J}}{u_{I,J}} \text{ and } \delta f_l = -\frac{u_k}{u_l}\delta f_k,$$

with $+$ for $k \in \mathcal{R}(J, I)$ and $-$ for $k \in \mathcal{R}(I, J)$. The corresponding values for $(I, J) \in \mathcal{A}(l)$ are obtained by swapping k and l.

3 COMPUTATIONAL COMPLEXITY

We now discuss the computational complexity of our algorithm in the setting of image analysis, in which the graph, $(\mathcal{V}_4, \mathcal{E}_4)$, is planar. For the sake of simplicity we consider a square image, letting $\mathcal{V}_4$ be an $\eta \times \eta$ grid of vertices. We are interested in the computational complexity in terms of the number of observations $n = \eta^2$.

Suppose we were to use a generic active set method, such as that of Goldfarb and Idnani (1983), to minimize $Q(f; c)$ subject to (2). This would be very computationally expensive, mainly because we may need to try all possible combinations of c in $\{-1,1\}^{2n-2n^{1/2}}$, which leads to exponential complexity. Our algorithm does not do this, and has polynomial complexity in the number of edges, even for non-planar graphs.

We can reduce the computational complexity even further by working with small sub-images that gradually increase in size. We control the order in which the edge constraints (2) are satisfied in order to keep $|\mathcal{R}(k)|$ and $|\mathcal{R}(l)|$ as small as possible. Here we describe an implementation of our algorithm in which the maximum size of a region grows dyadically. For the sake of simplicity we will consider η to be an integer power of 2. It is easy to adapt this method for other values of η, and for non-square images.

The edge constraints are satisfied in stages, there being $\log_2 \eta$ stages in total. At stage p we consider those edges in the set

$$\left\{\left((i, 2^p q - 2^{p-1}), (i, 2^p q - 2^{p-1} + 1)\right) \in \mathcal{E}_4 : q = 1, \ldots, \eta/2^p\right\}$$

followed by those in the set

$$\left\{\left((2^p q - 2^{p-1}, i), (2^p q - 2^{p-1} + 1, i)\right) \in \mathcal{E}_4 : q = 1, \ldots, \eta/2^p\right\}.$$

The effect is that as the edges are considered the graph of satisfied edges grows dyadically. At the first stage the graph looks like pairs of vertices, followed by squares of 2×2 vertices. At the second stage the graph looks like connected rectangles of 2×4 vertices, followed by squares of 4×4 vertices. The process continues until all edges are satisfied and the whole $\eta \times \eta$ image is connected.

The advantage of this implementation is that our algorithm will never allow an edge (k, l) in the active set if $c_{k,l} = 0$. Therefore $\mathcal{R}(k)$ and $\mathcal{R}(l)$ can never be larger than the rectangle connected by satisfied edges that contains k and l. At stage p this rectangle will contain at most 2^{2p} vertices and $2^{p+1}(2^p - 1)$ edges.

At each stage we perform Step 1 on $O(\eta^2 2^{-p})$ edges. During each iteration of Step 1 we may need to change the active set many times, through repeated splitting or amalgamation in Step 1.3. Since $|f_l - f_k|$ decreases monotonically, once an edge has been removed from $\mathcal{A}(k)$ or $\mathcal{A}(l)$ it cannot be included again during this iteration of Step 1. Therefore during each iteration the algorithm will consider at most $2^{p+2}(2^p - 1)$ active sets and perform Steps 1.1–3 $O(2^{2p})$ times.

There are some calculations to make in Steps 1.1–3. It is possible to calculate all necessary values u_a and m_a without visiting a vertex in $\mathcal{R}(k) \cup \mathcal{R}(l)$ more than twice. The algorithm must check for possible neighboring regions to amalgamate with. It must also check condition (7) at every edge in $\mathcal{A}(k)$ and $\mathcal{A}(l)$. Since $(\mathcal{R}(k), \mathcal{A}(k))$ and $(\mathcal{R}(l), \mathcal{A}(l))$ are connected, acyclic graphs, there will only be $|\mathcal{R}(k)| - 1$ and $|\mathcal{R}(l)| - 1$ edges to check. Therefore the complexity of Steps 1.1–3 is $O(|\mathcal{R}(k)| + |\mathcal{R}(l)|)$. We know that $|\mathcal{R}(k)|$ and $|\mathcal{R}(l)|$ are contained inside the connected rectangle with 2^{2p} vertices, so Steps 1.1–3 have $O(2^{2p})$ computational complexity. The total computational complexity of this implementation is therefore

$$O\left(\sum_{p=1}^{\log_2 \eta} \eta^2 2^{-p} 2^{2p} 2^{2p}\right) = O(\eta^5) = O(n^{5/2}).$$

4 EXAMPLES

4.1 Image analysis

Figure 2 shows, on the left, a noisy image that was used as an example by Polzehl and Spokoiny (2000). This example demonstrates the use of our algorithm in the case where the graph is $(\mathcal{V}_4, \mathcal{E}_4)$, which is suggested by the 4-neighborhood.

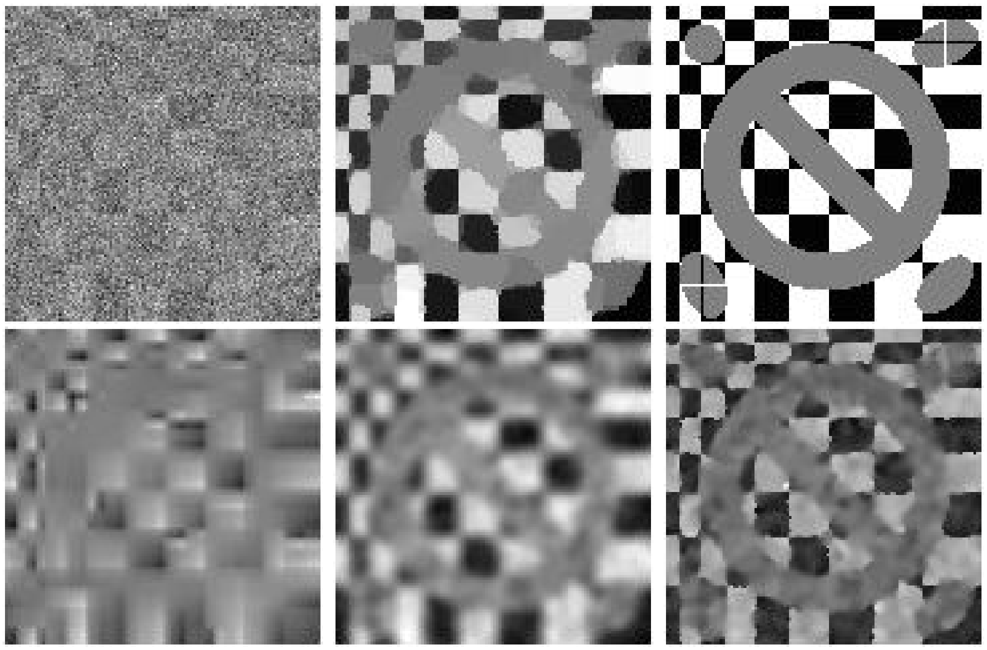

Figure 2: Noisy (top left) and denoised versions of the signal image (top right) of Polzehl and Spokoiny (2000). The estimates shown are the minimizer of $Q(f)$ (top center), wavelet thresholding (bottom left), kernel smoothing (bottom center) and adaptive weights smoothing (bottom right).

This particular image exhibits areas of solid colour, with sharp discontinuities between them, as is typical of many images (Polzehl and Spokoiny 2003). Our algorithm works well on this kind of image, because the areas of solid colour can be represented by regions of constant value.

There are many proposed methods for choosing the smoothing parameters. As, at this point, we are only interested in demonstrating our algorithm, we have employed a simple method suggested by Rudin, Osher and Fatemi (1992). It uses a global smoothing parameter, λ, and is based around an estimate of the global variance, σ^2. Of course our algorithm allows different smoothing parameters at every edge, so we can make use of more elaborate methods if we wish.

In order to find the simplest image for which the residuals behave as expected, we increase λ until $\sum_{i \in \mathcal{V}_4} (f_i - y_i)^2 = \hat{\sigma}^2 |\mathcal{V}_4|$. According to Chambolle (2004) this value of λ will always exist. Of course we require an estimate of σ^2 that is independent of the residuals. We can use, for example, one similar to that proposed by Davies and Kovac (2001):

$$\hat{\sigma} = \frac{1.48}{\sqrt{2}} \operatorname{median} \left(|y_j - y_i| : (i, j) \in \mathcal{E}_4 \right).$$

In Table 1 we compare this implementation of our algorithm with some other automatic image denoising algorithms: kernel smoothing, adaptive weights smoothing and wavelet thresholding. The kernel estimate uses a Gaussian kernel with bandwidth chosen by cross-validation. The adaptive weights smoothing and wavelet thresholding estimates are, respectively, the defaults of the `aws` (see Polzehl and Spokoiny 2000) and `wavethresh` (see Nason 2008, p. 143) R packages. Adaptive weights smoothing provides the only estimate that is closer to the signal image, g, than the minimizer of $Q(f)$. However, the minimizer of $Q(f)$ is not as rough as the other estimates. The output of our algorithm, the image estimated by use of the graph $(\mathcal{V}_4, \mathcal{E}_4)$, is shown in the top center image of Figure 2.

4.2 Irregularly-spaced data

We compare our algorithm to some other regression methods in a simulation experiment. In each simulation we generated 1000 points uniformly on $[0, 1]^2$ and connected them via the Delaunay triangulation. At each of these points we calculated a value from one of the following functions

$$g_1(x_1, x_2) = \exp\left(-100\left((x_1 - 0.5)^2 + (x_2 - 0.5)^2\right)\right),$$

$$\begin{aligned} g_2(x_1, x_2) &= I_{[0,1]}\left(10(x_1 - 0.5)^2 + 10(x_2 - 0.5)^2\right), \\ g_3(x_1, x_2) &= I_{[0,0.5]}(x_2), \\ g_4(x_1, x_2) &= (I_{[0,0.5]}(x_2) - 1)x_1 + 1. \end{aligned}$$

To each of these values we added Gaussian noise with zero mean and standard deviation 0.05, to make 1000 noisy response observations. We then removed half the observations at random to simulate 500 missing values.

In Table 2 we examine the performance of our estimate, with global smoothing parameter selected by the same automatic method as the image analysis example. As there are missing values and hence vertices with zero weight, the minimizer of $Q(f)$ is not unique. Therefore we let the estimate at every missing value equal the mean of the estimate at neighboring values. This can be thought of as the limit of a nonparametric elastic net estimator (Zou and Hastie 2005) as the L_2 smoothing parameter tends to zero. Since we are interested in judging the performance in terms of both prediction and estimation, Table 2 reports the average, over 100 simulations, mean square error MSE $= \frac{1}{1000}\sum_{i\in\mathcal{V}}(f_i - g(x_{1i}, x_{2i}))^2$. Davies and Kovac (2001) mention a mean correction that we have also applied to our estimator.

	$\sum_{i\in\mathcal{V}_4}(f_i - g_i)^2$	$\sum_{(i,j)\in\mathcal{E}_4}\lvert f_j - f_i\rvert$
Noisy image	65146	147769
Wavelet thresholding	5193	3462
Gaussian kernel estimate	3650	2371
Minimizer of $Q(f)$	2896	1696
Adaptive weights smoothing	1907	4762
Clean image	0	3787

Table 1: Performance measurements for estimates of the image in Figure 2.

It typically results in a decrease in MSE. We have compared our estimator with the graph-based regularization method of Belkin et al. (2004), which uses an L_2 penalty term for roughness, and with a kernel estimator (Nadaraya, 1964). These competing methods had smoothing parameter and bandwidth chosen by 10-fold cross-validation and also chosen to minimize the average MSE.

Table 2 shows that our algorithm is competitive when compared with other estimators. The minimizer of $Q(f)$, with an automatic choice of the smoothing parameter, gives smaller MSE than the estimator with the L_2 roughness penalty, in all examples. Once the mean correction has been applied the minimizer of $Q(f)$ also has smaller MSE than the kernel smoother.

	g_1	g_2	g_3	g_4
Minimizer of $Q(f)$				
automatic global smoothing parameter	1.14	11.7	6.43	3.17
mean correction	0.59	11.1	6.18	2.60
L_2 roughness penalty				
10-fold cross-validation	1.80	12.8	8.04	3.65
minimum MSE	1.31	12.8	7.98	3.47
Gaussian kernel estimator				
10-fold cross-validation	0.88	14.8	8.02	3.20
minimum MSE	0.87	12.8	7.18	2.97
Minimizer of $Q(f)$				
automatic edge length smoothing parameter	0.96	9.8	5.23	2.55
mean correction	0.50	9.3	5.01	2.12

Table 2: MSE $\times 10^{-3}$ for different estimators, evaluated for four functions.

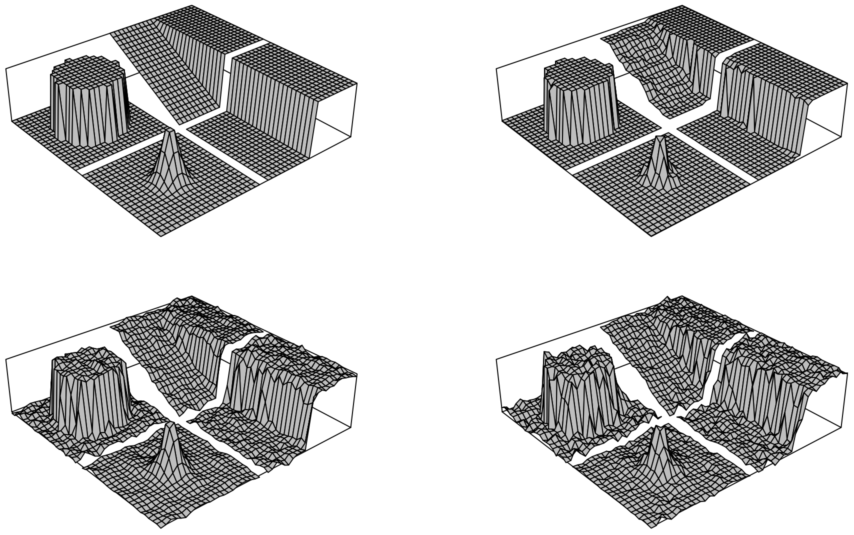

Figure 3: Estimates for one simulation. The original functions (top left) are shown compared with the estimate obtained by minimising $Q(f)$ (top right). The kernel estimate (bottom left) and estimate with L_2 roughness penalty (bottom right) have bandwidth and smoothing parameters that minimize MSE.

The kernel estimator performs very well when compared to the graph-based estimators. This is to be expected since the kernel estimator knows the distance between observations, not just their neighbors. To demonstrate how our algorithm performs when these distances are known, we let the smoothing parameter at each edge be proportional to the reciprocal of the length of that edge. The last section of Table 2 compares the MSE for this enhanced estimator with the kernel estimator with optimum global bandwidth. Incorporating the edge lengths results in an improvement over both the original L_1 penalty estimator and the kernel estimator.

A further advantage of using the L_1 roughness penalty over the L_2 roughness penalty and kernel smoothing is qualitative. Figure 3 shows the estimated surfaces from one simulation. When the L_2 norm is used as a roughness penalty, very small smoothing parameters are required to minimize the MSE. Hence the L_2 roughness penalty estimate exhibits many additional bumps in locations where the true function is flat. This is also a problem for the kernel estimator. However the minimizer of $Q(f)$ produces much simpler functions, without these extra bumps. There are large regions of constant value where the true functions are flat, so the estimate is also flat in these locations.

4.3 Classification

If the response observations are binary then a straightforward argument shows that the minimizer of $Q(f)$ provides an estimate for penalized logistic regression, and hence can be used for classification (Dümbgen and Kovac 2009). We feed the algorithm values in $\{0, 1\}$, with unit weight, according to the class of the training data, and give vertices with missing values zero weight, then classify depending on whether or not the resulting minimizer of $Q(f)$ is greater than 1/2.

We demonstrate our algorithm on the Ionosphere dataset (Frank and Asuncion 2010), which consists of 341 observations of a binary class label and 34 explanatory variables. We constructed a non-planar graph in which each vertex is connected to its 6 nearest neighbors in the 34 dimensional space. This graph was used to demonstrate the L_2 penalty method of Belkin et. al (2004) and we will compare our method with this, and the label spreading method of Zhou et. al (2004).

We performed 100 simulations each for probabilities of missingness between 0.1 and 0.9. Figure 3 shows the test error for the three methods at different probabilities of missing. The L_2 penalty method and label spreading have both been given the smoothing parameter that minimizes the test error. However the results of our algorithm are based on a crude automatic method of choosing the largest smoothing parameter that gives 5% training error. Our algorithm, with L_1 roughness penalty, shows an improvement over the method with L_2 roughness penalty and also performs quite well when compared with label spreading, which is better than both regression-based methods when there are lots of missing observations.

ACKNOWLEDGEMENTS

We thank an anonymous referee for suggestions about an earlier revision of this paper that lead to considerable improvements. The Ionosphere dataset was made available by the UCI Machine Learning Repository.

SUPPLEMENTAL MATERIALS

Appendix: Proofs of all the theorems above, and also proofs of the values associated with the events described in Subsection 2.2. (Regression_on_a_Graph

	Probability of missingness								
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
L_2 roughness penalty	0.19 (.07)	0.20 (.04)	0.20 (.03)	0.20 (.03)	0.21 (.02)	0.23 (.03)	0.25 (.04)	0.28 (.06)	0.35 (.03)
Minimizer of $Q(f)$	0.14 (.07)	0.15 (.04)	0.15 (.04)	0.15 (.03)	0.15 (.03)	0.15 (.02)	0.15 (.02)	0.17 (.06)	0.32 (.09)
Label spreading	0.15 (.07)	0.15 (.04)	0.15 (.03)	0.15 (.03)	0.15 (.02)	0.15 (.02)	0.15 (.02)	0.16 (.03)	0.19 (.03)

Table 3: Test errors (standard deviations) at different probabilities of missingness for three classification methods.

Appendix.pdf, portable document file)

Computer Code: R and C code that implements our algorithm for all the above examples. See the file readme.txt for further details. (roag.zip, zip archive)

REFERENCES

Belkin, M., Matveeva, I., and Niyogi, P. (2004), "Regularization and Semi-supervised Learning on Large Graphs," in *Learning Theory*, eds. J. Shawe-Taylor and Y. Singer, Berlin: Springer-Verlag, pp. 624–638.

Belkin, M., Niyogi, P., and Sindhwani, V. (2006), "Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples," *Journal of Machine Learning Research*, **7**, 2399–2434.

Besag, J. (1989), "Towards Bayesian Image Analysis," *Journal of Applied Statistics*, **16**, 395–407.

Besag, J., Green, P., Higdon, D., and Mengersen, K. (1995), "Bayesian Computation and Stochastic Systems," *Statistical Science*, **10**, 3–66.

Blum, A., and Chawla, S. (2001), "Learning from Labeled and Unlabeled Data using Graph Mincuts," in *Proceedings of the Eighteenth International Conference on Machine Learning.*

Chambolle, A. (2004), "An Algorithm for Total Variation Minimization and Applications," *Journal of Mathematical Imaging and Vision*, **20**, 89–97.

Culp, M., and Michailidis, G. (2008). "Graph-Based Semi-Supervised Learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **30**, 174–179.

Culp, M., Michailidis, G., and Johnson, K. (2009). "On Multi-view Learning with Additive Models," *The Annals of Applied Statistics*, **3**, 292–318.

Davies, P. L., and Kovac, A. (2001), "Local Extremes, Runs, Strings and Multiresolution," *The Annals of Statistics*, **29**, 1–65.

Dümbgen, L., and Kovac, A. (2009), "Extensions of Smoothing via Taut Strings," *Electronic Journal of Statistics*, **3**, 41–75.

Frank, A., and Asuncion, A. (2010), UCI Machine Learning Repository.

Goldfarb, D., and Idnani, A. (1983), "A Numerically Stable Dual Method for Solving Strictly Convex Quadratic Programs," *Mathematical programming*, **27**, 1–33.

Hastie, T., and Zou, H. (2005), "Regularization and Variable Selection via the Elastic Net," *Journal of the Royal Statistical Society, Series B*, **67**, 301–320.

Herbster, M. and Pontil, M. (2006), "Prediction on a Graph with a Perceptron," in *Advances in Neural Information Processing Systems 19*, eds. B. Sch¨olkopf, J. Platt and T. Hofmann, Cambridge, MA: MIT Press, pp. 577–584.

Jansen, M., Nason, G. P., and Silverman, B. W. (2009), "Multiscale Methods for Data on Graphs and Irregular Multidimensional Situations," *Journal of the Royal Statistical Society, Series B*, **71**, 97–125.

Joachims, T. (2003), "Transductive Learning via Spectral Graph Partitioning," in *Proceedings of the Twentieth International Conference on Machine Learning.*

Kleinberg, J., and Tardos, E. (2002), "Approximation Algorithms for Classification Problems with Pairwise Relationships: Metric Labeling and Markov Random Fields," *Journal of the ACM*, **49**, 616–639.

Koenker, R., and Mizera, I. (2004), "Penalized Triograms; Total Variation Regularization for Bivariate Smoothing," *Journal of the Royal Statistical Society, Series B*, **66**, 145–163.

Mammen, E., and van de Geer, S. (1997), "Locally Adaptive Regression Splines," *The Annals of Statistics*, **25**, 387–413.

Nadaraya, E. A. (1964), "On Estimating Regression," *Theory of Probability and its Applications*, **9**, 141–142.

Nason, G. P. (2008), *Wavelet Methods in Statistics with R*, New York: Springer.

Polzehl, J., and Spokoiny, V. G. (2000), "Adaptive Weights Smoothing With Applications to Image Restoration," *Journal of the Royal Statistical Society, Series B*, **62**, 335–354.

Polzehl, J., and Spokoiny, V. (2003), "Image Denoising: Pointwise Adaptive Approach," *The Annals of Statistics*, **31**, 30–57.

Rudin, L. I., Osher, S., and Fatemi, E. (1992), "Nonlinear Total Variation Based Noise Removal Algorithms," *Physica D*, **60**, 259–268.

Tibshirani, R. (1996), "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society, Series B*, **58**, 267–288.

Winkler, G. (2003), *Image Analysis, Random Fields and Markov Chain Monte Carlo Methods*, Berlin: Springer-Verlag.

Zhou, D., Bousquet, O., Lal, T.N., Weston, J., and Scholkopf, B. (2004), "Learning with Local and Global Consistency," in *Advances in Neural Information Processing Systems 16*, eds. S. Thrun, L. K. Saul and B Schölkopf, Cambridge, MA: MIT Press, pp. 321–328

Zhu, X., and Ghahramani, Z. (2002), "Learning from Labeled and Unlabeled Data with Label Propagation," Technical Report CMU-CALD-02-107, Carnegie Mellon University.

Zhu, X., Ghahramani, Z., and Lafferty, J. (2003), "Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions," in *Proceedings of the Twentieth International Conference on Machine Learning.*