Working Paper 2006-6

Fast Kernel Regression

Niels Stender

ISSN 1398-6228

Fast Kernel Regression

Niels Stender
University of Aarhus
niels@innohead.com

May 10, 2006

Abstract

A conjectured $O(n)$-method for computation of unconditional kernel density estimates is extended to conditional density estimation and regression. Empirical calculation time is investigated on simulated data with a limited dependent variable.

Contents

1 Introduction **66**
1.1 Note on Dimensionality 67

2 Kernel Methods **67**

3 Algorithm **68**
3.1 Unconditional Density Algorithm 69
3.2 Conditional Density Algorithm 70
3.3 Outliers 70

4 Experiment **71**
4.1 Results 71

5 Conclusion **73**

1 Introduction

When conducting many types of inference in the presence of a large number of observations, kernel density estimation and regression can be an attractive choice of method due to its nearly assumption-free nature. However, computing time is a significant barrier to usage of the method. For a naive implementation computing time scales with the number of observations n as $O(n^2)$.

In Gray & Moore (2001, 2003*a*,*b*) a new algorithm is introduced for unconditional kernel density estimation that is conjectured to scale $O(n)$ and seems to do so on a range of simulated and real-world datasets. In this paper, the Gray algorithm is extended to *conditional* kernel density estimation.

1.1 Note on Dimensionality

It is sometimes argued that nonparametric methods are useless in higher dimensions, since the number of observations needed to attain a fixed level of estimate confidence grows exponentially with the number of dimensions. The empty space phenomena, Scott & Thomsen (1983), illustrates the problem. A ten-dimensional multivariate normal distribution will on average have some 99% of observations falling outside the one standard deviation hypercube. However, many real-world high-dimensional datasets are spanned by lower-dimensional manifolds, and so can be said to be of a lower intrinsic dimensionality.

2 Kernel Methods

This section is based on Pagan & Ullah (1999) and Racine & Li (2004).

The goal of density estimation is to approximate a proper density function $f : \mathcal{R}^D \to \mathcal{R}_0^+, D \in \mathcal{N}$, given a set of n realisations $\{x_i\}_{i=1}^n$ from f. The main idea behind *kernel* density estimation is as follows. Given a point of interest $x^* \in \mathcal{R}^D$ we estimate $f(x^*)$ by calculating the proportion of observations falling in the neigborhood of x^*, each observation x_i weighted proportionally to its distance to x^*.

Given a weight function, also called a kernel, $K(\cdot)$, the kernel density estimate $\hat{f}$ for f is calculated by eq. (1).

$$\hat{f}(x^*) = \frac{1}{n}\sum_{i=1}^{n} K(x^* - x_i) \tag{1}$$

A kernel is any function for which $\int_{\mathcal{R}^D} K(x)\,dx = 1$ and is often taken to be a product of univariate smooth functions $K_d(\cdot)$, such as gaussians. The distance metric is scaled by a bandwidth h_d along each dimension, resulting in eq. (2). Bandwidth scaling of distance necessitates the inclusion of additional normalization terms $\frac{1}{h_d}$, to ensure $\int_{\mathcal{R}^D} \hat{f}(x)\,dx = 1$.

$$\hat{f}(x^*|h) = \frac{1}{n}\sum_{i=1}^{n}\prod_{d=1}^{D}\frac{1}{h_d}K_d\left(\frac{x^{*,d} - x_i^d}{h_d}\right), \tag{2}$$

$x^{*,d}$ and x_i^d being the d'th element of vectors x^* and x_i, respectively.

Conditional kernel density estimation is a straightforward extension of plain estimation. Given a point of interest $z^* = (y^*, x^*)$ we can estimate $f(y^*|x^*)$ by eq. (3).

$$\hat{f}(y^*|x^*, h) = \begin{cases} \frac{\hat{f}(z^*|h)}{\hat{f}(x^*|h)} & \text{for } \hat{f}(x^*|h) > 0 \\ 0 & \text{for } \hat{f}(x^*|h) = 0 \end{cases} \tag{3}$$

The possibility of zero probability in eq. (3) can in practice lead to optimization problems, so a modified version will often be used as described in section 3.3, p. 70.

Limited dependent variables can be accomodated by discrete kernels. The simplest possible is the counting measure, $K_{\text{count}}(y_i, y_j) = \mathbf{1}(y_i = y_j)$. The unordered categorical kernel shown in eq. (4) is a more sophisticated choice in allowing for a variable degree of smoothing.

$$K_{\text{uc}}(y_i, y_j|\lambda) = \begin{cases} \lambda & \text{for } y_i = y_j \\ \frac{1-\lambda}{c-1} & \text{for } y_i \neq y_j \end{cases}, \ \lambda \in [\frac{1}{c}; 1],\ y_i, y_j \in \{0, \ldots, c-1\}, \tag{4}$$

c being the number of categories for y_i. For $\lambda = 1$ eq. (4) equals the counting measure, while $\lambda = \frac{1}{c}$ results in an indiscriminate kernel giving equal weight to matching and non-matching observations.

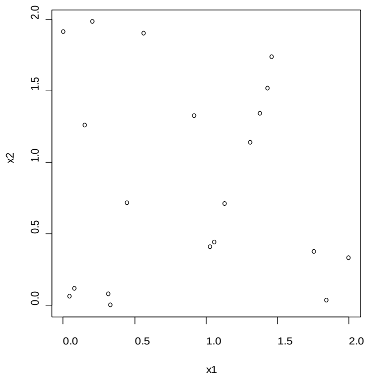
3 Algorithm

The following is a simplified walk-through of the mechanisms of the unconditional kernel density algorithm, based on Gray & Moore (2003a).

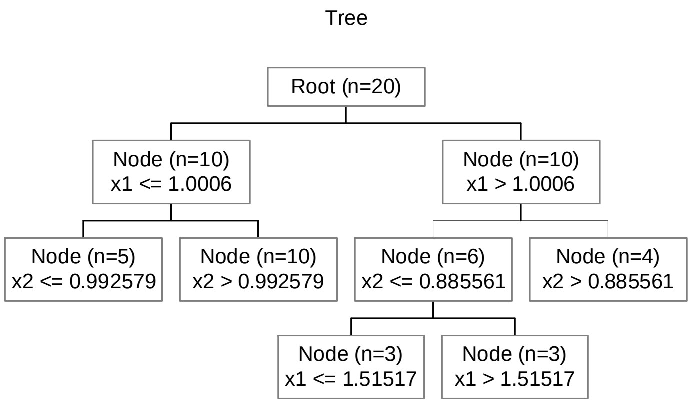
Many previous kernel-estimation algorithms achieved calculation time speedups through binning, the process of grouping data. The new method is based on a form of sophisticated binning. Instead of clustering observations on an evenly spaced grid, a binary tree stores observations hierachially in clusters of similar observations.

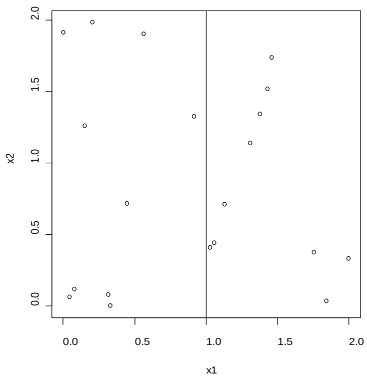
In the root node, all observations are available. Then along the most wide dimension, data is split along the middle of this dimension into two groups, thus creating two new nodes. Each node is decorated with information on the bounds of all data it contains.

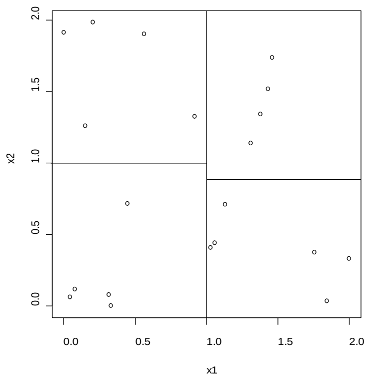
The process is iterated until each node contains a preset maximum of observations. Thus the higher up the tree one moves, the more similar observations will be. In computer science such a tree is called a kd-tree, for k-dimensional tree.

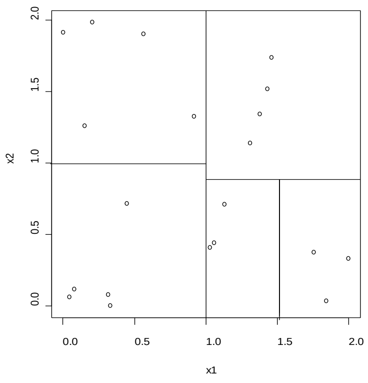


Root




Level 2


Level 3


Level 4

Assume the existence of two datasets: a query set and a training set. Suppose we would like to calculate the likelihood of the query set, using the training set to form the kernel density estimate at each query point. First, a *kd*-tree is generated for each set. Each pair of nodes in the trees are now processed iteratively. Given a node Q from the query tree and a node T from the training tree, we can infer about the lower bound dl and upper bound du of the likelihood contribution of T to Q, by comparing the pre-calculated data boundaries saved in the nodes. If $|du - dl|$ is below some preset thresold, the likelihood contribution is approximated by letting each $x_t \in T$ contribute, say, $\frac{|du-dl|}{2}$ to each $x_q \in Q$. Now all pairwise comparisons of the child nodes of T and Q can be pruned. Chunks of data in Q is compared against chunks of data in T. Pairs of chunks with high relative heterogeneity is examined more closely in smaller chunks, while those with low heterogeneity can be put aside.

3.1 Unconditional Density Algorithm

Below is a simplified version of the Gray-algorithm, while the conditional density version is presented in the following section. The iterative scheme is started out by calling **dualtree** on the two root nodes of the training and query tree.

```
dualtree($Q, T$)
    $dl = N_T K(\text{maxdist}(Q, T))$
    $du = N_T K(\text{mindist}(Q, T))$
    if $|du - dl| < \delta$
        for each $q \in Q$
        $l_q = l_q + dl$
        $u_q = u_q + du - N_T$
        return
    if leaf($Q$) and leaf($T$)
        dualtree_base($Q, T$)
        return
    dualtree($Q$.left, $T$.left)
    dualtree($Q$.left, $T$.right)
    dualtree($Q$.right, $T$.left)
    dualtree($Q$.right, $T$.right)
end dualtree

dualtree_base($Q, T$)
    for each $q \in Q$
        for each $t \in T$
            $c = K(x_q, x_t)$
            $l_q = l_q + c$
            $u_q = u_q + c - 1$
end dualtree_base
```

Notation. **maxdist** and **mindist** calculates the maximum and minimum distances between points in supplied nodes, using the boundary information of supplied nodes. **leaf** is true if the supplied node has no children.

l_q, u_q lower and upper bounds on the likelihood for individual observation q.

N_T number of observations contained in node T.

3.2 Conditional Density Algorithm

Each node in the training tree is now decorated with an additional piece of information: The empirical density of the dependent variable based on the information available at the node-level $f_{Y,T}(y)$.

```
**dualtree**($Q, T$)
    $dl = N_T K_X (\text{maxdist}(Q,T))$
    $du = N_T K_X (\text{mindist}(Q,T))$
    if $|du - dl| < \delta$
        for each $q \in Q$
        $l_{X,q} = l_{x,q} + dl$
        $u_{x,q} = u_{x,q} + du - N_T$
        for each $y^* \in Y$
            $c_{xy} = K_Y(y_q, y^*) \times f_{Y,T}(y^*)$
            $l_{xy,q} = dl \times c_{xy}$
            $u_{xy,q} = du \times c_{xy}$
        return
    if leaf($Q$) and leaf($T$)
        dualtree_base($Q, T$)
        return
    dualtree($Q$.left, $T$.left)
    dualtree($Q$.left, $T$.right)
    dualtree($Q$.right, $T$.left)
    dualtree($Q$.right, $T$.right)
**end dualtree**

**dualtree_base**($Q, T$)
    for each $q \in Q$
        for each $t \in T$
            $c_x = K_X(x_q, x_t)$
            $l_{x,q} = l_{x,q} + c_x$
            $u_{x,q} = u_{x,q} + c_x - 1$
            $c_{yx} = c_x \times K_Y(y_q, y_t)$
            $l_{xy,q} = l_{xy,q} + c_{xy}$
            $u_{xy,q} = u_{xy,q} + c_{xy}$
**end dualtree_base**
```

Lower and upper bounds on the global log-likelihood are calculated as a function of the individual boundaries $l_{x,i}, u_{x,i}, l_{xy,i}$ and $u_{xy,i}$.

$$(L_l, L_u) = \left(\sum_{i=1}^{N} \log\left(\frac{l_{x,i}}{u_{xy,i}}\right), \sum_{i=1}^{N} \log\left(\frac{u_{x,i}}{l_{xy,i}}\right) \right). \tag{5}$$

3.3 Outliers

A situation can arise in which one or more observations in Q has no neighbours in T. One can choose to ignore a fixed proportion of far-flung observations in the hope that no outliers will be included, as is often done in practice and in the literature. As an alternative you can assume a mixed model generates the data. Assume probability α of observing an instance of the non-parametric model f_{np}, and a small probability $(1-\alpha)$ of observing an instance from a multivariate uniform distribution f_{unif} bounded by the range of T.

$$f_{mix}(x) = \begin{cases} \alpha f_{np}(x) & \text{with probability } \alpha \\ (1-\alpha) f_{unif}(x) & \text{with probability } (1-\alpha) \end{cases}$$

Note that α is fixed in advance and not considered a part of the estimation. Now all observations will be assigned some probability, though it will be very small for observations without neighbours.

A third alternative exists for conditional density estimation. When an observation turns up with no probability mass, a crude nearest neighbour-principle can be used. Since a tree has already been generated, a "tree-nearest-neighbour" search is carried out by sweeping the parent leaf and maybe a few neighbouring leafs.

4 Experiment

Artificial data is generated to test how algorithmic time performance scales with number of observations, namely how long it takes to calculate a leave-one-out cross-validation statistic at the optimal bandwidth.

A kernel estimator is preferred in situations with a lot of local structure and a high number of observations, while other methods such as spline smoothing might be preferred in scenarios with smoother surfaces and a lower number of observations. The simulated data is structured as an egg tray, a smooth surface with peaks and valleys distributed uniformly as seen in fig. 1, p. 72. To someone ignorant about the global structure of the surface, this surface seems to have a lot of local structure.

A binary variable y is generated as a realization from a density g in eq. 7, a function of two continous variables x_1 and x_2.

$$g^*(x_1, x_2) = \frac{1}{4}(\sin(\pi x_1) + \sin(\pi x_2)) + \frac{1}{2} \tag{6}$$

$$g(x_1, x_2) = \begin{cases} 1 & \text{with probability } g^*(x_1, x_2) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

A realization of y is shown in fig. 2, p. 72. Both figures are zoomed versions of the actual dataset, where $(x_1, x_2) \in [0; 128]^2$. Six datasets of size n are simulated five times each, letting $n \in 1000 \times \{64, 128, 256, 512, 1024, 2048\}$. The experiment is carried out using a 2 GHz Intel Centrino processor on a 2 GB ram system running Windows XP.

An approximate optimal bandwidth is found by grid-searching on a shrinking interval as n grows, examining the cross-validation log-likelihood at each iteration. A relation between CV log-likelihood and a large bandwidth range is shown in fig. 6 and is used to set the intial range. A large interval is set initially at the smallest sample size, and the interval is then shrunk around the expected optimal bandwidth according to table 3.

4.1 Results

The average leave-one-out cross-validation log likelihood as a function of bandwidth is displayed in fig. 5 for 64k observations, while figure 7 displays the CV log-likelihood for the largest sample size. The identification of an optimal bandwidth is easier as n grows.

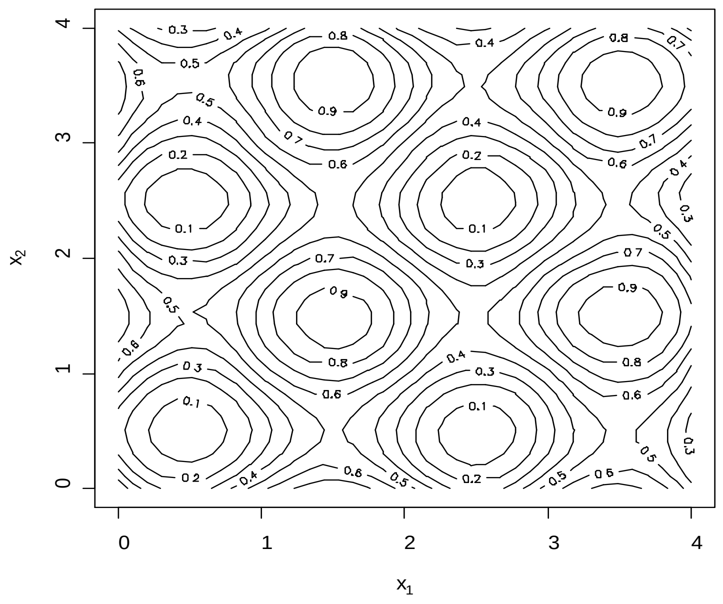


Figure 1: Contour plot of function g^*

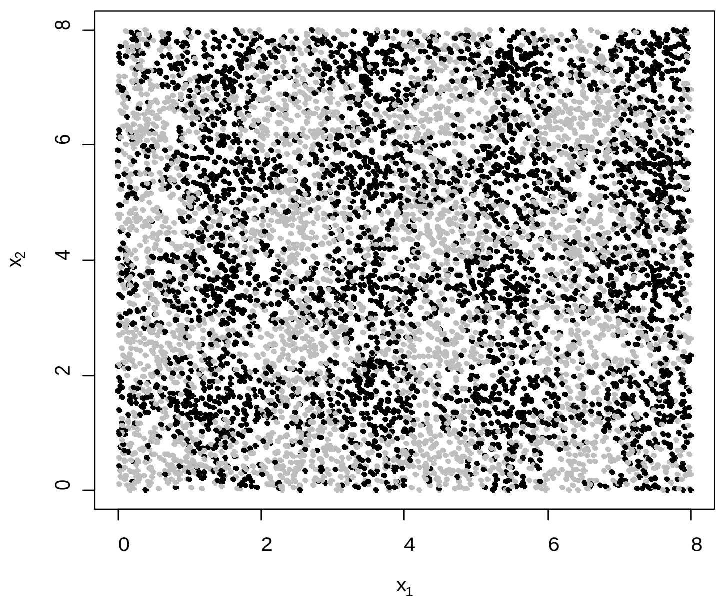


Figure 2: A realization of $y = g\,(x_1, x_2)$. Grey dots: $y = 0$, black dots: $y = 1$

n	min	max
32,000	0.60	8.00
64,000	0.42	5.66
128,000	0.30	4.00
256,000	0.21	2.83
512,000	0.40	1.00
1,024,000	0.28	0.71
2,048,000	0.20	0.50

Figure 3: Ranges for bandwidth grid search

Fig. 8 and table 9 display the average calculation time at the optimal average bandwidth for each of the dataset sizes. The speeds are shown next to the estimated time of a naive ckde implementation.

For the dualtree algorithm, the growth in time consumption as n increases is steeper than linear, but much less than quadratic. The speed ratio degrades significantly when moving from 1M to 2M datapoints, but this could be a hardware effect of working memory being squeezed out of the memory cache. Nonetheless, with the number of observations ranging in the millions and calculation time ranging in seconds and minutes, conditional kernel density estimation is feasible on large datasets.

From fig. 6 it is seen how time consumption increases with bandwidth. Time consumption will reach an upper plateau when the bandwidth makes the kernel span the entire range of the data, and then gradually fade again as approximation can kick in as the relative distance between observation points converge to zero. In Gray & Moore (2003a) it is conjectured that time consumption reaches a maximum at the optimal bandwidth, but for the conditional kernel density case, this is clearly not the case. A real-world dataset is likely to display some local structure along one or more dimensions, thus ensuring that the optimal bandwidth along those dimensions is unlikely to span the range.

Another experiment is carried out to look at algorithmic time consumption versus increased dimensionality. A dataset of size $n = 64k$ is generated, letting $(x_1, ..., x_d) \in [0; 16]^d$. Time consumption is recorded at a fixed bandwidth of $0.3\dot{5}$. As seen from fig. 10, time consumption seems nearly linear. However, this example does not give a clear indication dimensional performance on real-world datasets, since the optimal bandwidth is increasing in the number of dimensions at fixed dataset sizes, and the true dimensionality data. In Gray & Moore (2003a) experiments on real-world data suggests that time grows in polynomial time as dimensionality is increased.

5 Conclusion

Conditional kernel density estimation is a statistically well understood non-parametric estimator. Computational feasibility of the method in the presence of large datasets has been demonstrated in this paper, so any researcher with access to standard computing resources has little reason to avoid analyzing data with suspected non-linearities. A similar approach will work for continous and ordered categorical dependent variables, while presence of categorical independent variables must be given more thought.

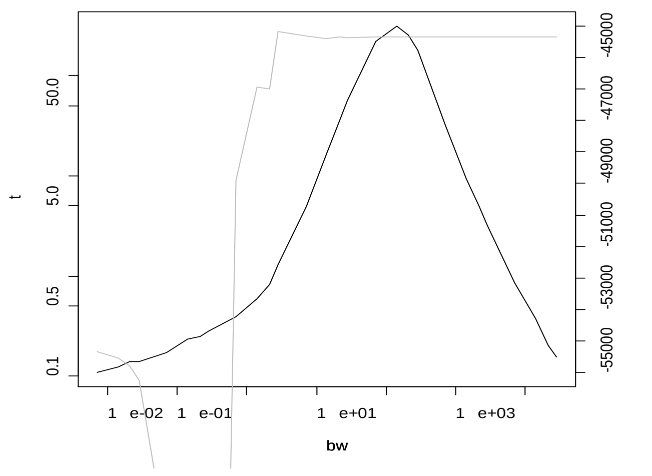


Figure 4: Black line-left axis: Time consumption in seconds. Grey line-right axis: CV log-likelihood. Both are based on one realization of a dataset of size $n = 64k$. Log-scales on both axes.

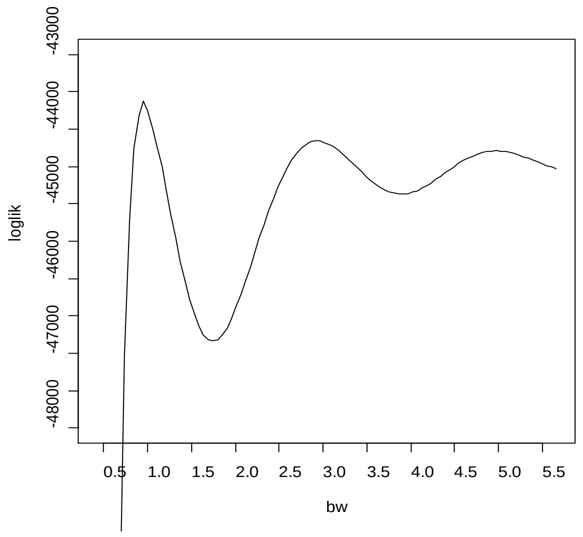


Figure 5: Average CV log-likelihood from five realizations of a dataset of size $n = 64k$

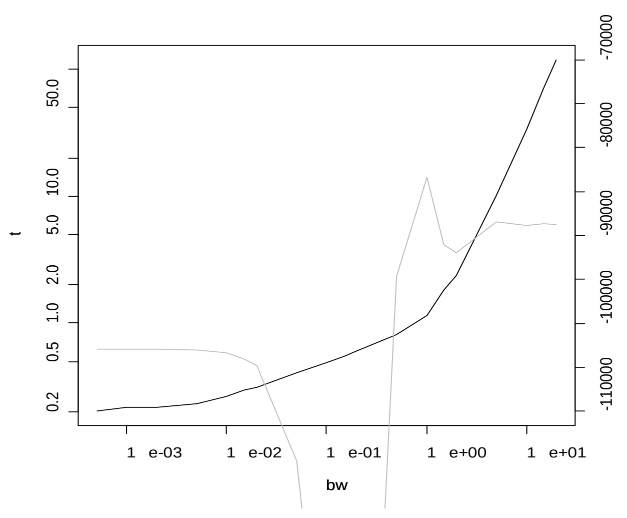


Figure 6: Black line-left axis: Time consumption in seconds. Grey line-right axis: CV log-likelihood. Both are based on one realization of a dataset of size $n = 128k$

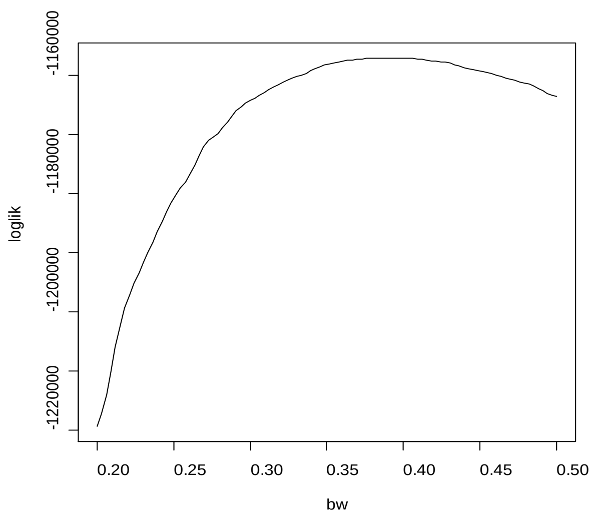


Figure 7: Average CV log-likelihood from five realizations of a dataset of size $n = 2,048k$

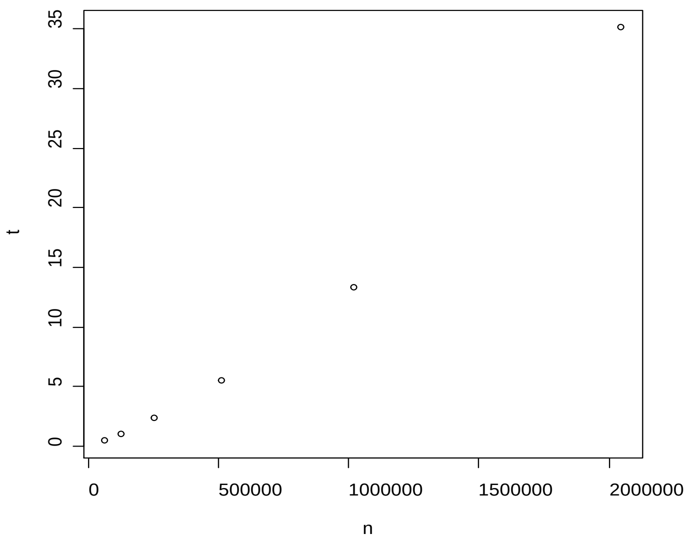


Figure 8: Calculation time in seconds for CV log likelihood at the optimal bandwidth

n	**Dualtree CKDE** **t** (sec.)	**Naive CKDE** **t approx.**
64,000	0.4	4.2 m
128,000	1.0	16.9* m
256,000	2.3	1.1* h
512,000	5.4	4.5* h
1,024,000	13.3	18.0* h
2,048,000	35.1	71.9* h

Figure 9: Calculation time in seconds for CV log likelihood at the optimal bandwidth

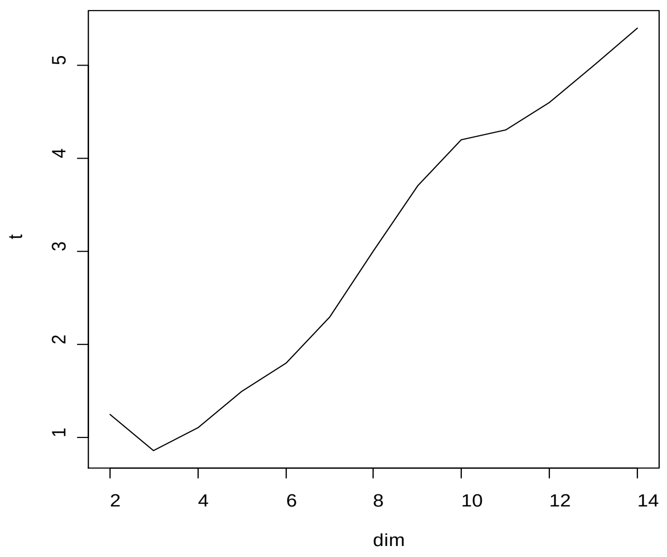


Figure 10: Time consumption in seconds at varying independent dimensions

References

Gray, A. G. & Moore, A. W. (2001), N-body problems in statistical learning, *in* T. K. Leen & T. G. Dietterich, eds, 'Advances in Neural Information Processing Systems', MIT Press.

Gray, A. G. & Moore, A. W. (2003*a*), Nonparametric density estimation: Toward computational tractability, *in* D. Barbar & C. Kamath, eds, '2003 SIAM International Conference on Data Mining'.

Gray, A. G. & Moore, A. W. (2003*b*), Rapid evaluation of multiple density models, *in* 'Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics'.

Pagan, A. & Ullah, A. (1999), *Nonparametric Econometrics*, Themes in Modern Econometrics, Cambridge University Press.

Racine, J. & Li, Q. (2004), 'Nonparametric estimation of regression functions with both categorical and continuous data', *Journal of Econometrics* **119**(1), 99–130.

Scott, D. & Thomsen, J. (1983), Probability density estimation in higher dimensions, *in* J. Gentle, ed., 'Computer Science and Statistics: Fifteenth Symposium on the Interface', North Holland-Elsevier Science Publishers, pp. 173–179.