# Conditional Density Estimation with Class Probability Estimators

Eibe Frank and Remco R. Bouckaert

Department of Computer Science, University of Waikato, New Zealand
{eibe,remco}@cs.waikato.ac.nz

**Abstract.** Many regression schemes deliver a point estimate only, but often it is useful or even essential to quantify the uncertainty inherent in a prediction. If a conditional density estimate is available, then prediction intervals can be derived from it. In this paper we compare three techniques for computing conditional density estimates using a class probability estimator, where this estimator is applied to the discretized target variable and used to derive instance weights for an underlying univariate density estimator; this yields a conditional density estimate. The three density estimators we compare are: a histogram estimator that has been used previously in this context, a normal density estimator, and a kernel estimator. In our experiments, the latter two deliver better performance, both in terms of cross-validated log-likelihood and in terms of quality of the resulting prediction intervals. The empirical coverage of the intervals is close to the desired confidence level in most cases. We also include results for point estimation, as well as a comparison to Gaussian process regression and nonparametric quantile estimation.

## 1 Introduction

In this paper we investigate methods for performing conditional density estimation using class probability estimators. Conditional density estimation makes it possible to quantify and visualize the uncertainty associated with the prediction of a continuous target variable. Given an observed vector of attribute values, a conditional density estimator provides an entire density function for the target variable, rather than a point estimate consisting of a single value. This function can then be visualized, or it can be summarized in prediction intervals that contain the true target value with a certain pre-specified probability.

As an example, consider the artificial data shown in Figure 1, consisting of two superimposed Mexican hats, each with Gaussian noise that exhibits constant variance. In this problem, there is a single attribute (shown on the $x$ axis) that is used to predict the target variable (shown on the $y$ axis). For each attribute value $x$, there is a conditional density function $f(y|x)$. Figure 2 shows conditional density functions for three values of $x$, namely 0, 3, and 8. These density functions are obviously unknown for real-world data. However, if they were available, we could use them to quantify predictive uncertainty, e.g. in the form of prediction

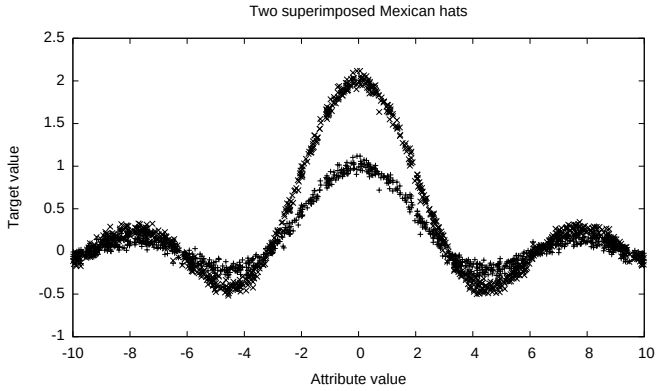


**Fig. 1.** Example data used to illustrate the output of CDE

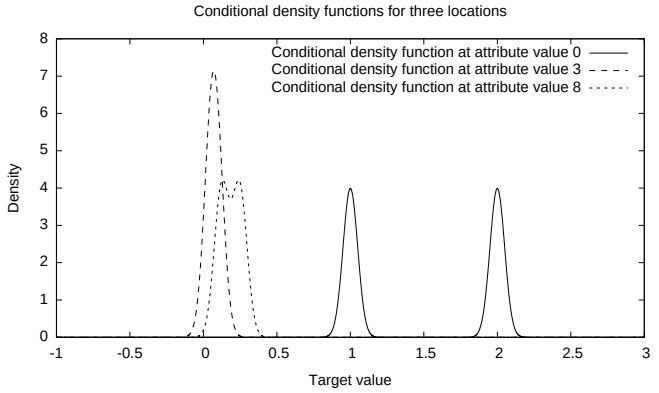


**Fig. 2.** Three conditional density functions for superimposed Mexican hats

intervals. The aim of conditional density estimation is to accurately estimate conditional density functions like these.

Although information on predictive uncertainty is very useful in interactive applications of prediction methods, there has been comparatively little work on conditional density estimation in statistics [1] and machine learning [2] outside the area of Bayesian models for regression, e.g. Gaussian process regression [3]. A significant amount of related material can be found in the economics literature [4], but this work has a focus on time series problems. In machine learning, a number of publications discuss conditional density estimation for particular types of neural networks, see e.g. [5–12]. Excluding Bayesian regression and neural networks, work on conditional density estimation in machine learning appears to be rare. One exception is [13], which investigates methods for learning conditional density trees in the context of Bayesian networks. Another is [2],

which describes a computationally efficient version of kernel conditional density estimation—a popular method in statistics [1].

The aim of this paper is to investigate a generic discretization-based technique for conditional density estimation that wraps around an existing machine learning algorithm—more specifically, a class probability estimator. A key advantage of generic techniques is that they can leverage existing implementations of machine learning algorithms in software suites like the Weka workbench [14]. It is well known that the most accurate algorithm for a particular prediction problem cannot in general be determined a priori and that experimentation with a collection of algorithms is necessary. Thus it is useful to have access to generic techniques that can be used to generate conditional density estimates in conjunction with existing class probability estimation techniques.

The idea of using class probability estimation for conditional density estimation is not new, and has been used in the context of modeling auction price uncertainty using boosting [15]. The contribution of this paper is that we show how to combine class probability estimation with univariate normal and kernel density estimators to improve on the histogram-based estimator used in [15]. Empirical evaluation is used to ascertain that progress has been made. We use two approaches for evaluation, both based on computing performance measures on independent test data. First, we compute log-likelihood values based on the predicted density estimates [9]. Second, we compute prediction intervals from the conditional density estimates based on pre-specified confidence levels, and measure their size and coverage [2]. The aim is to achieve small intervals with empirical coverage that is close to the desired confidence level.

The paper is structured is follows. In the next section, we discuss how class probability estimation can be combined with univariate density estimators to obtain conditional density estimates. Section 3 presents experimental results comparing the performance of histogram-based estimation with that obtained from normal and kernel estimators, based on two underlying class probability estimators. We also include results for point estimation, as well as a comparison to Gaussian process regression and nonparametric quantile estimation. Section 4 summarizes our findings and concludes the paper.

## 2 Conditional Density Estimation via Class Probabilities

We assume access to a class probability estimation scheme—e.g. an ensemble of class probability estimation trees—that can provide class probabilities $p(c|X)$ based on some labeled training data, where $c$ is a class value and $X$ an instance described by some attribute values. The basic idea is to discretize the continuous target values $y$ into intervals that can be treated as class values $c$ and use $p(c|X)$ to obtain a weight for each $y$, conditioned on $X$. A univariate density estimate—e.g. a normal density—constructed from these weighted target values in the training data, constitutes a conditional density estimate.

### 2.1 Three Density Estimators based on Weighted Target Values

For the discussion that follows, we assume that the target variable has been discretized into non-overlapping intervals ("bins"), for example, by applying equal-frequency discretization to the target values in the training data.[1] Regardless of which discretization method is applied, once the target values have been discretized, a class probability estimator can be used to estimate $p(c|X)$. Note that the class values created in this fashion exhibit a natural order, and it may be advantageous to exploit this by choosing an appropriate underlying class probability estimator.

Let $c_y$ be the bin (i.e. class) that contains the target value $y$ and let $p(c_y|X)$ be the predicted probability of that class given $X$, which is obtained from the class probability estimator. Let $n$ be the total number of target values in the training data and $n_c$ be the number of target values in bin $c$.

We compute a conditional density estimate by weighting the target values in the training data and then using these weights in a standard univariate density estimator to obtain a conditional density estimate. A weight $w(y_i|X)$ for a particular target value $y_i$ given an instance $X$ is computed by "spreading" the predicted class probability for bin $c_{y_i}$ across all $n_{c_{y_i}}$ target values in that bin:

$$w(y_i|X) = n\frac{p(c_{y_i}|X)}{n_{c_{y_i}}},$$

where the multiplier $n$ ensures that the sum of weights is $n$.

The weight $w(y_i|X)$ can be viewed as an estimate of how likely it is that a future observed target value associated with $X$ will be close to $y_i$, based on the class probability estimation model that has been inferred from the discretized training data. Given weights for all target values in the training data, we can then use a univariate density estimator, applied to these weighted values, to obtain a conditional density estimate $f(y|X)$.

The simplest estimator is the standard univariate normal estimator:

$$f_{normal}(y|X) = \mathcal{N}(y; \mu_X, \sigma_X^2),$$

where $\mu_X$ and $\sigma_X^2$ are the mean and variance respectively, computed based on the weighted target values; i.e. the mean is defined as $\mu_X = \frac{1}{n}\sum_{i=0}^{n} w(y_i|X)y_i$ and the variance as $\sigma_X^2 = \frac{1}{n}\sum_{i=0}^{n} w(y_i|X)(y_i - \mu_X)^2$.

Although the normal estimator is useful when the data is approximately normal, it is not flexible. A popular non-parametric alternative is a kernel density estimator. Using Gaussian kernels with kernel bandwidth $\sigma_{kernel}$, the weighted kernel estimator is:

$$f_{kernel}(y|X) = \frac{1}{n}\sum_{i=0}^{n} w(y_i|X)\mathcal{N}(y; y_i, \sigma_{kernel}^2)$$

[1] Note that it is also possible to create a bin for each unique target value that occurs in the training data, assuming this is feasible given the computational resources that are available, and assuming an appropriate class probability estimator is applied.

The bandwidth parameter determines how closely the estimator fits the (weighted) data points. We use a data-dependent value based on the global (weighted) standard deviation and the number of data points, namely $\sigma_{kernel} = \frac{\sigma_X}{n^{1/4}}$ This is based on [16], which advocates a bandwidth chosen at $O(n^{-1/4})$.

Note that the kernel estimator is a "lazy" estimator and requires more computational effort than the normal estimator. However, the procedure for computing $f_{kernel}(y|X)$ can be sped up significantly using a binary search for the kernel closest to $y$ and scanning the sorted list of kernels in both directions until the overall contribution from visiting additional kernels becomes negligible. Hence, the number of kernels that have to be evaluated to compute $f_{kernel}(y|X)$ is usually much smaller than $n$.

Another possible univariate density estimator, and perhaps the most obvious one in this context, is a histogram estimator based on the bins provided by the discretization. In this case, the density is assumed to be constant in each bin $c$ and based on the bin's width $r_c$ —the difference between the upper and lower boundary of the bin—and the class probability assigned to it:

$$f_{bins}(y|X) = \frac{p(c_y|X)}{r_{c_y}}$$

This estimator has been used for conditional density estimation in [15].

It is instructive to write the histogram estimator in a form analogous to the kernel estimator, based on weighted target values:

$$f_{bins}(y|X) = \frac{1}{n}\sum_{i=0}^{n} w(y_i|X)\frac{I(c_{y_i} = c_y)}{r_{c_y}},$$

where $I(a)$ takes on value 1 if the proposition $a$ is true and 0 otherwise. This formulation shows that the histogram estimator can be viewed as a kernel estimator where the kernel function is identical for all target values in a bin, with kernel value $\frac{1}{r_{c_y}}$ inside the bin, and value 0 outside. Note that this means that potentially valuable information about the relative position of $y$ with respect to the individual $y_i$ in a bin is discarded.

Another drawback of the histogram estimator is that it can return density 0 if $y$ falls into a bin that receives weight 0 or if it is located outside the range of all bins. To circumvent this problem we use the following adjusted version in our experiments:

$$f'_{bins}(y|X) = \frac{1}{n+2}\left(nf_{bins}(y|X) + \begin{cases} \frac{1}{max-min} : y \in [min, max] \\ \mathcal{N}(y; max, \sigma^2_{kernel}) : y > max \\ \mathcal{N}(y; min, \sigma^2_{kernel}) : y < min \end{cases}\right),$$

where $min$ is the smallest bin boundary of all bins, and $max$ the largest one. This means one data point is notionally spread out across the full $[min, max]$ range, and half a kernel function from the kernel estimator is attached to the left- and right-most bin respectively. Appropriate normalization ensures that the overall estimate integrates to one.

### 2.2 Examples

The Mexican hat data from Figure 1 can be used to illustrate the behavior of the three estimators discussed above. To this end, we discretized the 2,000 target values in Figure 1 into 20 bins using equal-frequency discretization and then applied a random-forest-based method, described in more detail in Section 3, to estimate class probabilities for the 20 discretization intervals.

The true conditional density function at attribute value 0 exhibits two well-separated peaks at target values 1 and 2 respectively (cf. Figure 2). Figure 3 shows the three different conditional density estimates for attribute value 0. It also shows an (unconditional) density estimate that was generated by applying a kernel density estimator to the unweighted target values. This latter estimate is labeled "prior kernel" in the figure and has a peak close to target value 0 because that is where most of the data in Figure 1 is located.

The figure shows that both the histogram estimator and the (conditional) kernel estimator reflect the two peaks in the true conditional density function quite accurately—as a unimodal estimator, the normal estimator is obviously not able to do so. However, a visual comparison of the estimates to the true conditional density shows that they do not model the height of the two peaks perfectly: they should be of the same height. This is due to the influence of discretization: the histogram estimate shows that the first peak is represented by two bars (corresponding to two discretization intervals) that together cover a wider range of target values than the single bar corresponding to the second peak. Thus the predicted class probability is spread across a wider range of target values and the height of the peak is reduced.

Let us now consider a situation where the two peaks in the true conditional density function are closer together. This is the case for attribute value 8 (cf. Figure 2). Figure 4 shows the conditional density estimates for this value. As before, the normal estimator is problematic, but less so as in the previous example. The kernel estimator models the two peaks quite well. The histogram estimator also has two main peaks, but additionally a third narrow peak that is a consequence of under-smoothing.

The data also exhibits attribute values where the normal estimator is appropriate, e.g. attribute value 3. The two Mexican hats intersect in the vicinity of this value, giving rise to a unimodal conditional density (cf. Figure 2). Figure 5 shows that the normal estimator yields the most accurate representation of the true density in this case. Both the histogram estimator and the kernel estimator overfit; however, the kernel estimator exhibits less under-smoothing.

### 2.3 Computing Prediction Intervals

An important application of conditional density estimation is the generation of prediction intervals. For a given confidence level $\alpha$ and an instance $X$, we would like to obtain an $\alpha\%$ prediction interval that contains the target value for $X$ with probability $\alpha$.

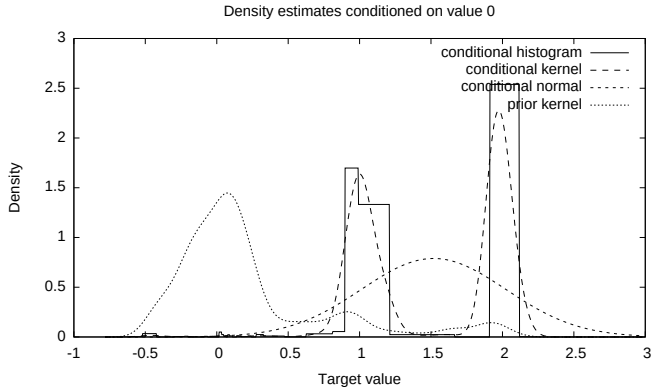


**Fig. 3.** Conditional density estimates for attribute value 0; prior estimate also included

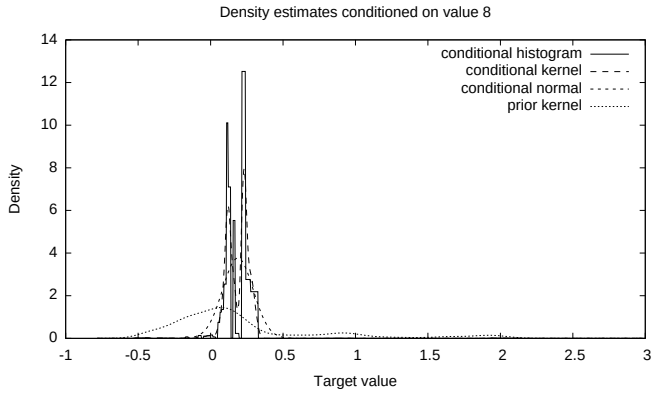


**Fig. 4.** Conditional density estimates for attribute value 8; prior estimate also included

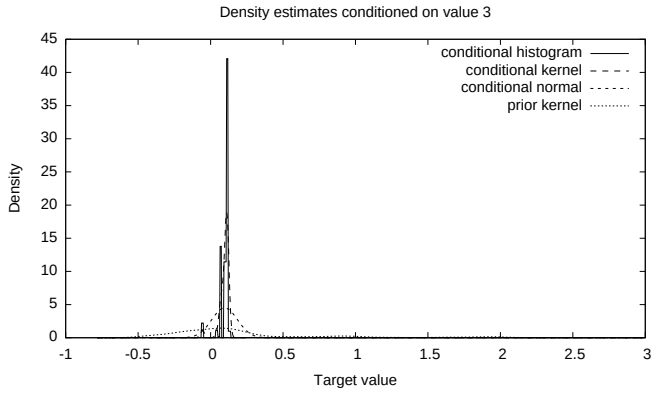


**Fig. 5.** Conditional density estimates for attribute value 3; prior estimate also included

Assuming we know the true conditional density $f(y|X)$, we can determine appropriate intervals—note that there is not necessarily a single interval—by choosing interval boundaries $b_i$ such that $\sum_i \int_{b_i}^{b_{i+1}} f(y|X)dy = \alpha$, i.e. the area under the chosen segments of the density function equals $\alpha$.

In practice, the true density function is not known and hence replaced by the estimators introduced above. This is particularly straightforward given the simple normal estimator $f_{normal}(y|X)$: the interval boundaries are easily obtained using the inverse of the cumulative distribution function for the normal density.

It is not so straightforward to obtain appropriate boundaries from a kernel density estimator $f_{kernel}(y|X)$. We adopt an approximate method, which assumes that the aim is to find interval boundaries such that the sum of all interval widths for the given confidence level is minimized. To this end, the range of the target variable is first segmented into 1,000 very small equal-width intervals. The density estimator is applied to the two end points of each interval and the two resulting density estimates are averaged. This average is then multiplied by the interval width to get an estimate of the area under the density function for that interval. The intervals are subsequently sorted in descending order according to their corresponding areas $a_j$, and the top $k$ intervals chosen. More precisely, $k$ is minimized subject to the constraint $\sum_{j=1}^{k} a_j \geq \alpha$. In most cases, many of the resulting intervals will be adjacent. Adjacent intervals are merged and the resulting merged intervals are output as prediction intervals.

Exactly the same process can also be applied in conjunction with the histogram estimator $f_{bins}(y|X)$. Using the same process in both cases facilitates a fair comparison based on the quality of the prediction intervals that are obtained. Like the kernel estimator, the histogram estimator can produce multiple prediction intervals when the density estimate is multi-modal (see, e.g., Figure 3 for a multi-modal estimate).

## 3 Experiments

In this section, we compare the performance of the conditional density estimators introduced in Section 2. We also evaluate point estimation performance and compare to Gaussian process regression and nonparametric quantile estimation.

### 3.1 Comparison of Density Estimators

We present results for two underlying class probability estimators: linear support vector machines, with Platt scaling [17] to obtain useful probability estimates,[2] and random forests, with 100 trees in the ensemble.[3] Performance estimates are obtained using 10 times 10-fold cross-validation. The corrected resampled paired $t$-test [18] is used to test for significant differences, at a 5% significance level.

[2] SMO with options -V 5 -M in Weka.

[3] To eliminate bias, each tree was grown from 66% of the training data, sampled w/o replacement, and the remainder used to estimate class probabilities.

Because a discretized regression target yields an ordinal variable, the method introduced in [19] was used to exploit this ordering information by creating multiple two-class problems. The base learners (i.e. SVMs and random forests) were applied to these two-class problems. This method was also used for conditional density estimation in [15], using an additional monotonizing step that we also applied for the results shown here. All results are based on the data from [20].

One method of measuring the performance of a conditional density estimator is to measure average log-likelihood (the average of $\log(f(y|X))$ across all test instances, where $y$ is the actual target value in the test instance). We use base 2 logarithm. To make this measure more informative, we subtract from this the log-likelihood obtained for an unconditional kernel density estimator (i.e. the "prior kernel" in Figures 3, 4, and 5). A positive value means that the conditional estimator improves on the prior estimator.

Table 1 shows results obtained using random forests.[4] Scores are shown for all three density estimators from Section 2. For the first set of results, the target variable was discretized into 10 bins, using equal-frequency discretization; for the second set, it was discretized into 20 bins. For each set of results, statistical significance is measured with respect to the histogram-based method.

The results show that both the kernel estimator and the normal estimator improve on the simple histogram estimator used previously in [15]. There are only a handful of cases were the histogram estimator significantly outperforms the kernel estimator. In spite of the fact that the normal estimator is not flexible, it performs well compared to the histogram estimator. However, there are also a number of cases were it performs significantly worse and also worse than the kernel estimator, indicating that additional flexibility can be important.

Table 2 shows the results obtained using SVMs. These results are even more strongly in favor of the kernel estimator and the normal estimator, e.g. there is only a single case where the histogram estimator yields significantly higher improvement in log-likelihood vs. the prior estimator than the kernel estimator.

It is noteworthy that the effect of the number of discretization intervals used to obtain class probability estimates is relatively minor for the kernel estimator and the normal estimator in most cases. However, the likelihood scores indicate that increasing the number of intervals from 10 to 20 renders the histogram estimator more prone to overfitting.

In practice, conditional density estimation is often used to obtain prediction intervals. Hence, we also evaluate the quality of these prediction intervals. To this end, we measured empirical coverage of target values in the test data as well as average width of the predicted intervals. The aim is to obtain narrow intervals with empirical coverage close to the chosen confidence level. For the results shown here, we chose a 95% confidence level. Empirical coverage is expressed as the average percentage of target values in the test data that were in the range of the predicted intervals. Interval width is expressed relative to the full range of target values in the training data: a relative width of 100 means that the

[4] The meta and schlvote data are excluded from this and the next table, due to the high variance in the estimates. No significant differences in performance were obtained.

**Table 1.** Mean improvement in log-likelihood vs. prior estimator, using random forests.

| | 10 bins | | | 20 bins | | |
|---|---|---|---|---|---|---|
| Dataset | Histogram | Kernel | Normal | Histogram | Kernel | Normal |
| auto93 | 0.21±0.79 | 0.90±0.28 ○ | 0.75±0.64 | -0.36±1.17 | 0.87±0.38 ○ | 0.75±0.62 ○ |
| autoHorse | 2.23±0.47 | 1.96±0.21 ● | 1.23±0.13 ● | 2.87±0.61 | 2.20±0.25 ● | 1.28±0.15 ● |
| autoMpg | 0.72±0.44 | 1.37±0.19 ○ | 1.37±0.10 ○ | 0.02±0.64 | 1.39±0.23 ○ | 1.40±0.11 ○ |
| autoPrice | 1.10±0.60 | 1.46±0.16 ○ | 0.90±0.20 | 0.43±0.81 | 1.53±0.19 ○ | 0.95±0.22 |
| baskball | -1.33±0.85 | 0.02±0.39 ○ | 0.43±0.66 ○ | -2.08±1.22 | 0.00±0.43 ○ | 0.43±0.65 ○ |
| bodyfat | 2.56±0.24 | 2.38±0.17 ● | 1.59±0.14 ● | 3.38±0.25 | 2.77±0.16 ● | 1.62±0.16 ● |
| bolts | 0.99±1.11 | 1.28±0.34 | 0.73±0.29 | 0.10±1.58 | 1.28±0.40 ○ | 0.78±0.27 |
| breastTumor | -0.78±0.60 | -0.27±0.31 ○ | -0.13±0.14 ○ | -0.64±0.70 | -0.26±0.25 | -0.11±0.14 ○ |
| cholesterol | -0.99±0.43 | -0.08±0.14 ○ | 0.28±0.84 ○ | -1.68±0.72 | -0.02±0.16 ○ | 0.31±0.89 ○ |
| cleveland | 0.48±0.22 | 0.64±0.19 ○ | -0.33±0.16 ● | 0.48±0.22 | 0.64±0.19 ○ | -0.33±0.16 ● |
| cloud | 0.39±0.95 | 1.11±0.31 ○ | 0.74±0.32 | -0.46±1.21 | 1.14±0.35 ○ | 0.77±0.43 ○ |
| cpu | 2.72±0.77 | 2.23±0.51 ● | 1.07±0.73 ● | 2.51±0.98 | 2.16±0.62 | 0.88±0.83 ● |
| detroit | 0.58±1.56 | -0.86±5.55 | 0.16±2.03 | 0.35±1.82 | -0.52±4.52 | 0.45±1.32 |
| echoMonths | 0.55±0.61 | 0.58±0.23 | 0.33±0.12 | -0.03±0.83 | 0.59±0.25 ○ | 0.33±0.13 |
| elusage | -1.16±1.45 | 0.69±0.73 ○ | 0.88±0.38 ○ | -1.89±1.51 | 0.48±1.00 ○ | 0.87±0.49 ○ |
| fishcatch | 1.89±0.71 | 2.25±0.23 | 1.55±0.15 | 1.66±0.76 | 2.37±0.24 ○ | 1.54±0.17 |
| fruitfly | -0.99±0.68 | -0.13±0.15 ○ | -0.35±0.34 ○ | -1.96±0.98 | -0.14±0.15 ○ | -0.33±0.34 ○ |
| gascons | 0.67±1.46 | 1.43±0.50 | 1.24±0.33 | 0.23±1.90 | 1.59±0.57 ○ | 1.40±0.36 |
| housing | 1.04±0.30 | 1.39±0.14 ○ | 1.02±0.13 | 0.55±0.47 | 1.46±0.17 ○ | 1.06±0.13 ○ |
| hungarian | -0.38±0.13 | 0.89±0.17 ○ | -1.12±0.26 ● | -0.38±0.13 | 0.89±0.17 ○ | -1.12±0.26 ● |
| longley | 0.41±1.82 | 1.21±0.65 | 1.09±0.30 | 0.18±1.96 | 1.28±0.79 | 1.12±0.25 |
| lowbwt | 0.22±0.48 | 0.72±0.22 ○ | 0.74±0.16 ○ | -0.33±0.67 | 0.73±0.22 ○ | 0.76±0.16 ○ |
| mbagrade | -0.99±1.09 | -0.01±0.43 ○ | 0.00±0.43 ○ | -1.83±1.37 | -0.04±0.47 ○ | 0.03±0.44 ○ |
| pbc | 0.09±0.30 | 0.38±0.12 ○ | 0.23±0.10 | -0.63±0.49 | 0.39±0.11 ○ | 0.23±0.10 ○ |
| pharynx | -1.35±0.79 | 0.21±0.21 ○ | 0.06±0.28 ○ | -2.34±0.91 | 0.23±0.21 ○ | 0.06±0.25 ○ |
| pollution | -0.84±0.92 | 0.26±0.23 ○ | 0.46±0.34 ○ | -1.56±1.05 | 0.24±0.22 ○ | 0.44±0.34 ○ |
| pwLinear | 0.24±0.58 | 1.00±0.20 ○ | 0.94±0.13 ○ | -0.50±0.79 | 1.05±0.23 ○ | 0.97±0.13 ○ |
| quake | -0.53±0.13 | -0.22±0.09 ○ | -0.86±0.07 ● | -0.54±0.13 | -0.23±0.10 ○ | -0.86±0.07 ● |
| sensory | -0.77±0.30 | -0.13±0.21 ○ | -0.04±0.07 ○ | -0.76±0.31 | -0.13±0.21 ○ | -0.04±0.07 ○ |
| servo | 0.30±0.91 | 1.32±1.48 ○ | 1.08±0.32 ○ | -0.90±1.15 | 1.46±0.75 ○ | 1.09±0.32 ○ |
| sleep | -0.30±0.97 | 0.36±0.35 ○ | 0.35±0.22 ○ | -0.89±1.07 | 0.35±0.36 ○ | 0.33±0.25 ○ |
| strike | 0.42±1.90 | 0.51±1.79 | -1.30±2.93 ● | -1.28±3.49 | -0.15±3.39 ○ | -2.16±5.10 |
| veteran | -0.35±1.30 | 0.39±0.83 ○ | -0.17±1.35 | -1.41±1.22 | 0.30±0.45 ○ | -0.28±1.26 ○ |
| vineyard | -0.10±1.26 | 0.66±0.97 ○ | 0.67±1.24 | -0.28±1.43 | 0.49±1.18 | 0.73±1.15 |

○/● statistically significant improvement/degradation

total interval width is equal to the full range of target values in the training data, yielding no useful information. For the results shown here, we used 10 discretization intervals. Results for 20 intervals are not shown due to lack of space, but yield comparable relative performance.

Table 3 shows results for random forests, and Table 4 those for linear SVMs. When interpreting these results, it is important to keep in mind that there is a trade-off between maximizing coverage and minimizing interval width: it is trivial to achieve high coverage by predicting very large intervals and vice versa. Hence, it is useful to pay attention in particular to those cases where a method dominates another one based on both criteria.

The results show that both the kernel estimator and the normal estimator often produce wider intervals than the histogram estimator. This is because they generally perform more smoothing, as could also be seen in the examples in Section 2.2. However, in contrast to the histogram estimator, they achieve empirical coverage close to, or above, the chosen 95% confidence level for almost all datasets, which is important for practical applications where the end user expects reliable prediction intervals. Moreover, there are several cases where the

**Table 2.** Mean improvement in log-likelihood vs. prior estimator, using linear SVMs.

| | 10 bins | | | 20 bins | | |
|---|---|---|---|---|---|---|
| Dataset | Histogram | Kernel | Normal | Histogram | Kernel | Normal |
| auto93 | -0.59±1.32 | 0.88±0.70 ○ | 1.12±0.65 ○ | -1.38±1.67 | 0.71±1.35 ○ | 1.10±0.48 ○ |
| autoHorse | 2.03±0.75 | 2.33±0.51 | 1.82±0.65 | 2.58±0.78 | 2.50±1.19 | 1.82±0.53 ● |
| autoMpg | 0.18±0.50 | 1.15±0.22 ○ | 1.33±0.14 ○ | -0.39±0.64 | 1.14±0.25 ○ | 1.37±0.14 ○ |
| autoPrice | 0.71±0.69 | 1.36±0.24 ○ | 1.24±0.21 ○ | 0.31±0.79 | 1.35±0.34 ○ | 1.34±0.26 ○ |
| baskball | -0.95±0.76 | -0.03±0.44 ○ | 0.43±0.59 ○ | -1.53±0.95 | 0.02±0.29 ○ | 0.45±0.64 ○ |
| bodyfat | 0.68±1.82 | 1.06±1.75 ○ | 1.63±0.26 | 0.22±2.34 | 0.92±2.28 ○ | 1.67±0.37 |
| bolts | 0.01±1.23 | 0.87±0.42 ○ | 0.42±0.33 | -0.85±1.33 | 0.82±0.51 ○ | 0.42±0.32 ○ |
| breastTumor | 0.08±0.29 | -0.05±0.11 | -0.04±0.06 | 0.37±0.35 | -0.04±0.07 ● | -0.04±0.06 ● |
| cholesterol | -0.51±0.33 | -0.02±0.10 ○ | 0.27±0.85 ○ | -0.68±0.46 | 0.01±0.14 ○ | 0.28±0.88 ○ |
| cleveland | 0.42±0.26 | 0.67±0.27 ○ | -0.34±0.18 ● | 0.42±0.26 | 0.67±0.27 ○ | -0.34±0.18 ● |
| cloud | -0.10±0.93 | 0.74±1.78 | 0.91±0.53 ○ | -1.13±1.07 | 0.82±0.44 ○ | 0.87±0.51 ○ |
| cpu | 0.97±0.94 | 2.03±0.55 ○ | 1.81±0.70 ○ | 0.24±1.25 | 2.05±0.66 ○ | 1.98±0.77 ○ |
| detroit | 0.14±2.96 | 0.69±2.18 | 0.88±0.83 | -0.48±3.10 | 0.45±2.33 | 0.84±0.85 |
| echoMonths | 0.06±0.58 | 0.37±0.21 | 0.24±0.11 | -0.35±0.80 | 0.36±0.23 ○ | 0.24±0.12 ○ |
| elusage | -1.32±1.44 | 0.38±0.51 ○ | 0.41±0.29 ○ | -1.75±1.48 | 0.35±0.56 ○ | 0.46±0.32 ○ |
| fishcatch | 1.26±0.83 | 2.15±0.39 ○ | 2.04±0.30 ○ | 0.60±0.94 | 2.16±0.39 ○ | 2.12±0.30 ○ |
| fruitfly | -0.58±0.55 | -0.01±0.05 ○ | -0.27±0.32 | -1.07±0.78 | -0.02±0.07 ○ | -0.28±0.32 ○ |
| gascons | 0.23±1.94 | 0.85±1.99 | 1.08±0.76 | -0.02±2.16 | 0.82±1.88 | 0.99±0.97 |
| housing | 0.98±0.30 | 1.21±0.19 ○ | 0.95±0.45 | 0.68±0.38 | 1.22±0.21 ○ | 1.01±0.42 |
| hungarian | -0.36±0.16 | 1.06±0.22 ○ | -1.17±0.60 ● | -0.36±0.16 | 1.06±0.22 ○ | -1.17±0.60 ● |
| longley | -0.01±2.27 | 0.98±1.39 ○ | 1.27±0.47 | -0.19±2.07 | 1.14±1.25 ○ | 1.29±0.47 ○ |
| lowbwt | -0.01±0.54 | 0.63±0.22 ○ | 0.68±0.16 ○ | -0.28±0.60 | 0.62±0.24 ○ | 0.68±0.15 ○ |
| mbagrade | -0.62±0.79 | 0.02±0.27 ○ | 0.04±0.19 ○ | -1.06±0.93 | 0.02±0.21 ○ | 0.03±0.21 ○ |
| pbc | -0.36±0.39 | 0.24±0.15 ○ | 0.22±0.10 ○ | -1.00±0.49 | 0.25±0.15 ○ | 0.24±0.10 ○ |
| pharynx | -0.40±0.49 | 0.33±0.19 ○ | 0.12±0.24 ○ | -1.35±0.81 | 0.32±0.21 ○ | 0.13±0.21 ○ |
| pollution | -1.16±1.43 | 0.07±1.01 ○ | 0.52±0.35 ○ | -1.66±1.37 | 0.10±0.79 ○ | 0.59±0.28 ○ |
| pwLinear | 0.19±0.64 | 0.99±0.29 ○ | 1.14±0.28 ○ | -0.51±0.88 | 0.96±0.37 ○ | 1.14±0.32 ○ |
| quake | -0.19±0.04 | -0.01±0.02 ○ | -0.86±0.05 ● | -0.19±0.04 | -0.02±0.03 ○ | -0.86±0.05 ● |
| sensory | -0.47±0.19 | -0.09±0.12 ○ | -0.17±0.03 ○ | -0.45±0.19 | -0.09±0.12 ○ | -0.17±0.03 ○ |
| servo | 0.31±1.00 | 1.44±0.30 ○ | 0.90±0.27 | -0.21±1.13 | 1.49±0.35 ○ | 0.92±0.28 ○ |
| sleep | -0.42±1.02 | 0.42±0.40 ○ | 0.43±0.27 ○ | -0.84±1.08 | 0.47±0.38 ○ | 0.43±0.27 ○ |
| strike | -0.58±0.75 | 0.38±0.21 ○ | -0.97±0.68 | -1.29±0.77 | 0.33±0.44 ○ | -0.97±0.81 |
| veteran | -0.39±0.92 | 0.23±0.23 ○ | -0.57±1.09 | -0.90±0.97 | 0.21±0.16 ○ | -0.57±1.17 |
| vineyard | -0.36±1.31 | 0.21±1.03 | 0.63±0.99 ○ | -0.56±1.36 | 0.28±0.62 ○ | 0.65±0.83 ○ |

○/● statistically significant improvement/degradation

kernel estimator and the normal estimator dominate the histogram estimator regarding both width and coverage (e.g. baskball, cholesterol, lowbwt, meta). The opposite is never the case. This is most likely due to the loss of location information in the histogram estimator. Considering the relative performance of the normal estimator and the kernel estimator, we can see that the former generally creates narrower intervals. As both exhibit acceptable coverage in most cases, the kernel estimator appears preferable.

### 3.2 Point Estimation Performance

The results so far show that the proposed methods can produce useful conditional density estimates and corresponding prediction intervals. However, it is important to consider the quality of the point estimates that they correspond to. To compute point estimates that are designed to minimize the squared error, we can compute the expected value of the target variable based on the conditional density estimate. In the case of the normal estimator and the kernel estimator, this is simply the weighted sum of the target values used to construct the estimator (i.e. the expected value is identical in both cases).

**Table 3.** Quality of prediction intervals, using random forests and 10 bins.

| | Coverage | | | Relative width | | |
|---|---|---|---|---|---|---|
| Dataset | Histogram | Kernel | Normal | Histogram | Kernel | Normal |
| auto93 | 89.4± 9.2 | 97.4± 5.3 ○ | 97.5± 5.1 ○ | 55.3± 6.7 | 52.9± 6.0 | 55.3± 5.4 |
| autoHorse | 96.8± 4.3 | 97.8± 3.4 | 98.9± 2.2 | 35.4± 6.6 | 31.6± 4.3 ○ | 35.6± 3.5 |
| autoMpg | 88.9± 5.2 | 95.3± 3.4 ○ | 98.7± 1.8 ○ | 33.8± 2.0 | 35.6± 2.0 ● | 39.3± 1.6 ● |
| autoPrice | 93.7± 6.0 | 98.6± 3.2 ○ | 99.1± 2.5 ○ | 38.8± 6.7 | 40.5± 6.7 ● | 46.8± 4.3 ● |
| baskball | 75.0± 12.2 | 94.3± 7.6 ○ | 95.4± 7.0 ○ | 73.6± 3.5 | 62.3± 6.7 ○ | 60.7± 5.9 ○ |
| bodyfat | 98.8± 2.0 | 99.2± 1.8 | 99.2± 1.7 | 37.8± 2.4 | 37.6± 2.5 | 37.0± 2.3 |
| bolts | 88.0± 15.7 | 100.0± 0.0 ○ | 100.0± 0.0 ○ | 66.0± 7.5 | 73.6± 9.8 ● | 79.4± 8.7 ● |
| breastTumor | 76.2± 7.7 | 94.0± 4.9 ○ | 91.4± 5.2 ○ | 59.6± 2.9 | 75.6± 3.6 ● | 76.1± 3.3 ● |
| cholesterol | 90.0± 5.3 | 94.7± 4.5 ○ | 95.6± 4.0 ○ | 68.7± 3.9 | 46.0± 6.2 ○ | 47.9± 5.6 ○ |
| cleveland | 90.1± 5.0 | 97.2± 3.1 ○ | 97.3± 3.0 ○ | 67.6± 3.2 | 84.6± 3.8 ● | 98.2± 3.5 ● |
| cloud | 90.0± 10.1 | 96.1± 5.8 | 97.9± 4.1 ○ | 43.6± 7.0 | 44.5± 7.0 | 50.0± 5.6 ● |
| cpu | 96.2± 3.8 | 97.2± 3.9 | 98.5± 2.4 | 27.0± 7.9 | 17.8± 6.0 ○ | 25.4± 6.2 |
| detroit | 85.5± 32.8 | 87.5±32.1 | 87.5±32.1 | 87.2± 10.7 | 88.4±33.0 | 80.7±28.6 |
| echoMonths | 87.8± 10.4 | 96.5± 4.6 ○ | 96.0± 5.2 ○ | 71.5± 3.7 | 87.9± 2.7 ● | 86.3± 3.0 ● |
| elusage | 61.8± 20.8 | 89.6±14.8 ○ | 96.2± 9.3 ○ | 38.0± 5.9 | 46.9± 8.3 ● | 54.8± 7.0 ● |
| fishcatch | 91.6± 7.7 | 98.2± 3.4 ○ | 98.8± 3.0 ○ | 26.1± 5.7 | 30.2± 5.9 ● | 37.5± 4.3 ● |
| fruitfly | 82.5± 9.8 | 93.1± 7.9 ○ | 93.9± 7.6 ○ | 70.3± 3.6 | 76.4± 5.7 ● | 75.8± 4.3 ● |
| gascons | 88.7± 20.4 | 99.0± 7.0 | 100.0± 0.0 | 59.7± 6.3 | 63.6±11.2 | 67.0± 8.7 ● |
| housing | 94.8± 3.0 | 98.7± 1.5 ○ | 99.2± 1.3 ○ | 41.5± 1.7 | 44.9± 2.1 ● | 48.8± 2.0 ● |
| hungarian | 55.8± 10.3 | 100.0± 0.0 ○ | 96.1± 3.5 ○ | 78.7± 3.0 | 64.2± 4.7 ○ | 139.2± 8.1 ● |
| longley | 87.5± 32.1 | 100.0± 0.0 | 100.0± 0.0 | 79.7± 8.4 | 97.7±21.1 ● | 91.2±18.2 ● |
| lowbwt | 89.9± 6.9 | 96.1± 4.8 ○ | 96.7± 4.0 ○ | 49.6± 1.8 | 46.8± 2.2 ○ | 46.9± 2.1 ○ |
| mbagrade | 69.1± 20.2 | 90.8±10.1 ○ | 90.1±10.5 ○ | 67.7± 4.8 | 84.0± 5.2 ● | 79.0± 5.7 ● |
| meta | 87.6± 4.8 | 96.7± 2.2 ○ | 96.4± 2.1 ○ | 60.6± 7.1 | 6.7± 1.9 ○ | 18.6± 2.7 ○ |
| pbc | 91.9± 4.0 | 97.1± 2.7 ○ | 96.3± 2.9 ○ | 72.6± 1.5 | 79.1± 1.6 ● | 80.5± 1.8 ● |
| pharynx | 72.5± 10.5 | 93.8± 5.4 ○ | 94.4± 5.2 ○ | 57.1± 3.5 | 69.9± 3.7 ● | 72.7± 3.8 ● |
| pollution | 79.5± 15.1 | 94.8± 9.7 ○ | 95.8± 8.3 ○ | 80.5± 2.6 | 79.6± 4.3 | 72.4± 4.9 ○ |
| pwLinear | 89.8± 6.8 | 98.1± 3.2 ○ | 98.6± 2.8 ○ | 54.8± 1.9 | 57.5± 2.0 ● | 53.9± 2.2 |
| quake | 90.0± 2.1 | 93.8± 1.5 ○ | 94.3± 1.5 ○ | 46.3± 0.5 | 53.4± 0.6 ● | 65.7± 0.6 ● |
| schlvote | 87.0± 18.4 | 91.9±15.8 | 95.4±13.2 | 54.1± 14.8 | 42.8±16.4 ○ | 51.8±16.8 |
| sensory | 86.5± 4.6 | 89.9± 4.1 ○ | 95.8± 2.5 ○ | 58.6± 4.5 | 56.7± 1.5 | 58.9± 1.4 |
| servo | 83.7± 9.0 | 96.2± 5.1 ○ | 96.0± 4.5 ○ | 30.3± 7.9 | 33.3± 8.8 ● | 38.5± 9.2 ● |
| sleep | 80.5± 16.5 | 96.4± 8.9 ○ | 94.3±10.0 ○ | 74.5± 5.3 | 90.0± 5.3 ● | 84.8± 6.4 ● |
| strike | 89.2± 4.0 | 95.6± 2.8 ○ | 97.4± 2.1 ○ | 51.0± 5.9 | 13.9± 1.6 ○ | 23.6± 2.5 ○ |
| veteran | 85.1± 11.1 | 94.9± 6.3 ○ | 95.9± 5.6 ○ | 51.9± 6.9 | 44.6± 5.5 ○ | 54.6± 5.8 |
| vineyard | 81.6± 16.9 | 93.7±10.3 ○ | 94.0±10.1 ○ | 56.8± 10.9 | 59.7±10.8 | 54.9± 9.3 |

○/● statistically significant improvement/degradation

Table 5 shows the root relative squared error for point estimates obtained this way, for the two base learners used previously (applied in conjunction with the wrapper for ordinal classification), based on 20 discretization intervals, and compared to two dedicated regression methods: 100 bagged unpruned regression trees and linear support vector machines for regression. These results show that the point estimates produced using regression by discretization are generally competitive with those produced by dedicated regression schemes.

### 3.3 Comparison to Gaussian Process Regression and Nonparametric Quantile Regression

In this section, we compare the performance of discretization-based density estimation to Gaussian process regression and techniques for quantile regression. To this end, we use SMO as the base learner, in conjunction with the ordinal wrapper and univariate kernel density estimation with 20 discretization bins. To yield comparable results, RBF kernels were used in SMO, and the $\gamma$ parameter

**Table 4.** Quality of prediction intervals, using linear SVMs and 10 bins.

| | Coverage | | | Relative width | | |
|---|---|---|---|---|---|---|
| Dataset | Histogram | Kernel | Normal | Histogram | Kernel | Normal |
| auto93 | 75.5± 14.0 | 92.8± 9.0 ○ | 96.4± 6.6 ○ | 35.8± 7.1 | 38.4± 6.5 | 42.1± 6.3 ● |
| autoHorse | 91.5± 6.8 | 95.1± 5.1 | 96.2± 4.3 ○ | 21.6± 5.3 | 19.6± 4.1 ○ | 23.3± 4.3 ● |
| autoMpg | 83.4± 6.3 | 94.8± 3.7 ○ | 97.9± 2.2 ○ | 37.6± 2.5 | 39.2± 2.6 ● | 38.9± 2.5 ● |
| autoPrice | 87.3± 8.4 | 95.7± 5.8 ○ | 98.4± 3.6 ○ | 32.2± 6.2 | 32.8± 5.3 | 36.2± 4.9 ● |
| baskball | 83.4± 11.3 | 94.4± 7.0 ○ | 97.8± 4.3 ○ | 75.6± 3.6 | 63.4± 6.2 ○ | 61.7± 5.5 ○ |
| bodyfat | 91.4± 5.7 | 94.3± 4.7 | 96.6± 3.4 ○ | 27.9± 2.1 | 25.5± 1.3 ○ | 25.8± 1.3 ○ |
| bolts | 79.3± 20.1 | 98.0± 6.8 ○ | 99.0± 6.1 ○ | 55.4± 9.8 | 77.6±11.7 ● | 88.7± 9.7 ● |
| breastTumor | 90.0± 5.2 | 95.8± 4.4 ○ | 94.1± 4.1 ○ | 76.5± 3.0 | 85.6± 2.7 ● | 80.5± 1.6 ● |
| cholesterol | 94.6± 4.6 | 95.5± 4.5 | 95.9± 3.9 | 72.9± 2.9 | 46.3± 6.1 ○ | 48.5± 5.5 ○ |
| cleveland | 85.6± 6.9 | 96.5± 3.4 ○ | 94.2± 3.9 ○ | 56.0± 4.3 | 71.8± 5.1 ● | 91.5± 4.7 ● |
| cloud | 81.9± 10.8 | 94.8± 6.6 ○ | 96.1± 5.3 ○ | 31.5± 7.0 | 35.6± 7.0 ● | 37.8± 7.0 ● |
| cpu | 87.5± 7.4 | 96.5± 4.3 ○ | 98.9± 2.1 ○ | 22.1± 7.0 | 16.6± 5.4 ○ | 20.1± 6.1 |
| detroit | 82.0± 35.2 | 90.0±28.4 | 93.0±24.6 | 78.3± 12.1 | 78.4±19.0 | 74.8±15.9 |
| echoMonths | 85.0± 8.9 | 96.5± 4.7 ○ | 94.9± 6.2 ○ | 70.2± 4.0 | 96.3± 2.2 ● | 91.8± 3.6 ● |
| elusage | 64.3± 22.8 | 91.0±11.8 ○ | 95.3± 9.3 ○ | 60.3± 6.3 | 83.6± 8.3 ● | 80.6± 8.6 ● |
| fishcatch | 86.8± 10.1 | 95.6± 5.8 ○ | 97.4± 3.9 ○ | 27.1± 5.0 | 27.0± 4.7 | 30.4± 5.2 ● |
| fruitfly | 88.5± 9.5 | 93.4± 7.1 | 94.2± 7.1 | 75.4± 3.9 | 76.7± 5.2 | 75.3± 3.2 |
| gascons | 82.0± 23.7 | 91.7±18.9 | 92.8±17.9 | 53.8± 10.2 | 63.6±18.5 ● | 62.9±18.0 ● |
| housing | 89.5± 4.2 | 94.8± 2.9 ○ | 96.4± 2.4 ○ | 34.0± 1.8 | 35.9± 2.2 ● | 39.3± 2.6 ● |
| hungarian | 59.7± 12.2 | 99.7± 0.9 ○ | 94.5± 5.2 ○ | 76.4± 4.2 | 57.7± 6.0 ○ | 127.7±10.7 ● |
| longley | 73.0± 38.5 | 94.5±22.4 | 98.5±11.1 | 66.3± 8.3 | 73.5±18.8 | 72.2±14.8 |
| lowbwt | 89.4± 7.2 | 95.9± 4.6 ○ | 96.9± 3.9 ○ | 52.1± 2.8 | 48.0± 2.7 ○ | 48.9± 2.7 ○ |
| mbagrade | 80.2± 14.5 | 97.1± 7.1 ○ | 92.4±10.4 ○ | 82.3± 4.9 | 97.7± 3.4 ● | 90.9± 5.5 ● |
| meta | 86.3± 7.2 | 98.2± 1.8 ○ | 99.4± 1.1 ○ | 65.5± 11.2 | 5.3± 1.1 ○ | 17.8± 2.9 ○ |
| pbc | 85.7± 5.7 | 96.1± 2.8 ○ | 96.7± 2.9 ○ | 70.4± 3.0 | 79.1± 3.0 ● | 79.7± 3.2 ● |
| pharynx | 83.3± 7.7 | 94.9± 4.8 ○ | 95.8± 4.5 ○ | 59.7± 3.4 | 70.3± 4.0 ● | 75.6± 3.6 ● |
| pollution | 76.3± 18.2 | 90.7±13.3 ○ | 91.8±12.0 ○ | 61.0± 6.9 | 64.6± 5.7 | 59.1± 5.2 |
| pwLinear | 84.5± 7.8 | 94.3± 5.0 ○ | 95.0± 5.3 ○ | 36.8± 3.0 | 40.0± 3.3 ● | 39.8± 3.5 ● |
| quake | 94.5± 1.4 | 98.0± 1.3 ○ | 94.6± 1.3 | 51.6± 1.0 | 58.1± 0.9 ● | 67.5± 0.7 ● |
| schlvote | 82.2± 20.6 | 89.0±18.2 | 93.9±14.3 | 56.2± 17.9 | 50.7±21.5 | 54.2±21.4 |
| sensory | 93.1± 3.5 | 95.2± 3.0 ○ | 96.3± 2.4 ○ | 72.1± 3.4 | 66.1± 1.9 ○ | 64.8± 1.5 ○ |
| servo | 81.0± 9.2 | 95.7± 5.4 ○ | 98.0± 3.6 ○ | 29.0± 7.2 | 32.8± 7.5 ● | 40.6± 8.2 ● |
| sleep | 79.4± 17.8 | 98.6± 4.9 ○ | 94.3±10.1 ○ | 68.8± 6.4 | 86.1± 7.8 ● | 81.8± 6.5 ● |
| strike | 85.5± 5.9 | 95.4± 3.2 ○ | 97.7± 2.3 ○ | 62.6± 14.4 | 20.6± 1.7 ○ | 30.1± 2.3 ○ |
| veteran | 88.6± 8.7 | 96.1± 5.6 ○ | 95.8± 6.0 ○ | 65.6± 4.9 | 51.1± 3.9 ○ | 60.7± 4.4 ○ |
| vineyard | 81.8± 19.2 | 91.8±13.1 | 92.5±11.4 | 56.1± 8.3 | 60.0±11.1 | 56.0±10.6 |

○/● statistically significant improvement/degradation

for the kernel and the $C$ parameter for regularization were optimized using *internal* 10-fold cross-validation, choosing from $10^i$ with $i \in \{-5, 4, ..., 4, 5\}$ for $\gamma$ and $10^i$ with $i \in \{-2, -1, 0, 1, 2\}$ for $C$.[5]

Table 6 shows the relative performance of discretization-based density estimation and Gaussian process regression (GPR) with RBF kernels. Gaussian process regression was applied with inputs normalized to $[0, 1]$, as was SMO. The target was also normalized for GPR, to make choosing an appropriate noise level $\sigma$ easier. Predictions were transformed back into the original range of the target variable. The noise level $\sigma$ for GPR, and $\gamma$ for the RBF kernel, were optimized using internal 10-fold cross-validation, with the same values for $\gamma$ as in the case of SMO, and values $10^i$ with $i \in \{-2, -1.8, -1.6, ..., -0.4, -0.2, 0\}$ for $\sigma$.

The log-likelihood scores exhibit 11 significant differences, three in favor of GPR and eight against. Comparing the relative scores for GPR to those for the normal estimator in Table 2 shows that there are four cases where the gain for

[5] To reduce runtime, the tolerance parameter in SMO was increased from 0.001 to 0.1, which did not significantly impact performance on the datasets investigated.

**Table 5.** Root relative squared error: regression by discretization (20 bins) vs. dedicated regression methods.

| Dataset | Random forests (20 bins) | Bagged regression trees | Linear SVMs (20 bins) | Linear SVMs for regression |
|---|---|---|---|---|
| auto93 | 58.6± 13.6 | 93.8± 21.4 ○ | 53.1± 16.2 | 62.5± 18.8 |
| autoHorse | 33.3± 11.2 | 40.3± 11.7 ○ | 30.6± 11.2 | 31.5± 8.7 |
| autoMpg | 37.0± 5.3 | 40.2± 7.2 | 38.1± 5.7 | 39.0± 5.6 |
| autoPrice | 37.8± 8.6 | 37.9± 11.3 | 39.5± 10.8 | 45.6± 8.9 ○ |
| baskball | 84.5± 12.5 | 85.5± 21.3 | 84.3± 9.9 | 82.2± 17.9 |
| bodyfat | 22.4± 5.8 | 13.8± 11.0 ● | 29.1± 6.0 | 9.7± 12.7 ● |
| bolts | 33.2± 13.8 | 35.8± 29.3 | 49.0± 22.2 | 32.8± 23.1 ● |
| breastTumor | 100.1± 7.3 | 107.8± 11.6 ○ | 97.9± 1.6 | 98.1± 7.7 |
| cholesterol | 99.0± 3.2 | 103.4± 10.0 | 99.6± 1.2 | 103.7± 9.2 |
| cleveland | 72.5± 6.8 | 74.1± 10.3 | 68.9± 9.1 | 71.5± 9.0 |
| cloud | 48.6± 14.0 | 45.7± 14.7 | 50.0± 12.9 | 37.2± 13.3 ● |
| cpu | 40.3± 17.5 | 27.2± 12.6 | 35.7± 19.1 | 29.2± 14.4 |
| detroit | 143.2±229.4 | 133.0±145.5 | 149.9±241.1 | 103.8±119.9 |
| echoMonths | 71.7± 11.5 | 76.3± 14.8 ○ | 74.0± 9.5 | 74.1± 14.2 |
| elusage | 50.1± 17.1 | 57.5± 22.6 ○ | 62.3± 14.4 | 60.5± 21.7 |
| fishcatch | 24.6± 7.5 | 20.8± 6.3 | 22.4± 5.9 | 25.8± 5.5 |
| fruitfly | 104.1± 5.7 | 116.9± 15.5 ○ | 100.7± 2.6 | 102.4± 9.7 |
| gascons | 25.2± 10.2 | 24.2± 13.2 | 26.3± 10.6 | 24.1± 16.2 |
| housing | 41.6± 8.1 | 38.1± 8.6 | 45.2± 10.8 | 54.5± 11.2 ○ |
| hungarian | 73.6± 10.3 | 78.0± 11.5 ○ | 71.6± 11.5 | 88.6± 16.8 ○ |
| longley | 49.9± 46.3 | 53.2± 71.6 | 56.1± 58.9 | 15.0± 14.0 ● |
| lowbwt | 61.5± 8.2 | 64.2± 10.3 | 64.0± 6.7 | 64.9± 10.9 |
| mbagrade | 97.6± 19.3 | 99.0± 23.4 | 96.2± 9.7 | 88.4± 23.8 |
| meta | 95.7± 22.2 | 157.1± 70.3 ○ | 108.0± 34.3 | 81.6± 19.6 |
| pbc | 83.2± 5.9 | 83.0± 9.3 | 81.5± 6.0 | 82.4± 8.5 |
| pharynx | 78.5± 9.2 | 103.9± 6.7 ○ | 76.2± 8.9 | 83.6± 13.3 ○ |
| pollution | 78.0± 8.8 | 69.1± 17.9 | 69.3± 17.3 | 68.6± 25.8 |
| pwLinear | 46.2± 6.9 | 40.4± 7.8 ● | 42.0± 8.3 | 50.7± 10.1 ○ |
| quake | 100.0± 1.7 | 100.9± 2.8 | 100.0± 0.2 | 107.7± 1.7 ○ |
| schlvote | 77.0± 26.5 | 72.3± 29.4 | 91.5± 36.3 | 89.7± 39.4 |
| sensory | 86.7± 4.2 | 89.0± 6.9 | 95.2± 1.8 | 94.7± 5.9 |
| servo | 38.9± 14.4 | 34.3± 16.5 | 37.2± 14.5 | 62.7± 23.6 ○ |
| sleep | 79.0± 18.3 | 79.6± 27.1 | 77.0± 22.8 | 74.7± 22.0 |
| strike | 79.8± 13.2 | 91.3± 15.6 ○ | 85.2± 8.6 | 82.3± 12.4 |
| veteran | 90.7± 15.0 | 105.8± 26.4 ○ | 92.3± 6.4 | 89.1± 13.7 |
| vineyard | 60.7± 23.5 | 66.1± 31.1 | 70.6± 23.0 | 72.6± 28.9 |

○/● statistically significant improvement/degradation

these techniques is well below zero while it is well above zero for the kernel-density-based estimate: cleveland, hungarian, strike, and veteran. This indicates that it is not appropriate to model the predictive distribution with a normal density in these cases.

Considering the quality of the prediction intervals, there is one case where GPR performs significantly better than the discretization-based approach according to both coverage and width (bodyfat), and two where it is significantly worse (hungarian, quake). Ignoring significance, there are four additional cases where GPR performs better according to both statistics (auto93, autoMpg, cpu, lowbwt) and three more cases where it is worse (cleveland, echoMonths, servo). It is noticeable that the observed coverage is generally closer to the desired confidence level (95%) for the discretization-based approach.

Due to space constraints, we cannot show detailed results for the root relative squared error here. In most cases, the error is comparable. However, there are three very small datasets with 40 or less instances (bolts, gascons, longley) where

**Table 6.** Comparison of discretization-based conditional density estimation (SVMs, kernel density estimator, 20 bins) with Gaussian process regression (GPR), using RBF kernels in both cases.

| | Mean improvement in log-likelihood | | Coverage | | Relative width | | |
|---|---|---|---|---|---|---|---|
| Dataset | RBF SVMs (20 bins) | RBF GPR | RBF SVMs (20 bins) | RBF GPR | RBF SVMs (20 bins) | RBF GPR | |
| auto93 | 0.68± 0.95 | 1.00± 0.77 | 92.70± 9.98 | 96.18± 6.93 | 38.43± 7.81 | 37.49± 6.32 | |
| autoHorse | 2.63± 0.63 | 1.48± 1.80 | 94.46± 5.41 | 95.86± 4.48 | 16.20± 4.58 | 16.44± 2.69 | |
| autoMpg | 1.29± 0.27 | 1.45± 0.23 | 93.56± 4.28 | 96.38± 3.61 | 32.71± 2.54 | 31.88± 2.72 | |
| autoPrice | 1.27± 0.46 | 1.08± 0.44 | 92.83± 5.87 | 91.32± 6.98 | 28.46± 5.22 | 25.95± 4.85 | |
| baskball | -0.10± 0.56 | 0.50± 0.74 | 93.11± 8.55 | 97.61± 5.09 | 60.16± 6.45 | 64.79± 1.04 | ○ |
| bodyfat | 1.57± 1.77 | 1.74± 3.72 | 94.83± 4.49 | 98.41± 2.60 | ● | 19.75± 2.78 | 8.92± 3.28 ● |
| bolts | 0.69± 0.55 | -0.12± 2.37 | 95.50±10.88 | 84.00±22.90 | 79.29±13.09 | 41.66±14.30 | ● |
| breastTumor | -0.03± 0.09 | -0.11± 0.13 | 95.52± 4.31 | 89.36± 7.13 | ○ | 83.83± 3.21 | 72.08±15.20 ● |
| cholesterol | -0.01± 0.07 | 0.15± 0.93 | 95.09± 4.45 | 97.36± 4.08 | 46.19± 6.28 | 61.88± 4.89 | ○ |
| cleveland | 0.87± 0.40 | -0.45± 0.24 | ○ | 96.60± 4.09 | 95.75± 5.61 | 63.73± 5.95 | 99.20±10.24 ○ |
| cloud | 0.83± 0.40 | 0.61± 1.78 | 93.53± 7.00 | 92.63± 8.46 | 35.17± 7.15 | 26.48± 5.99 | ● |
| cpu | 2.93± 0.67 | 2.76± 0.91 | 95.97± 4.94 | 99.32± 2.97 | 9.89± 4.18 | 7.88± 2.31 | |
| detroit | 0.87± 1.84 | 0.26± 2.31 | 91.50±26.64 | 81.50±38.04 | 70.43±27.58 | 54.88±23.71 | |
| echoMonths | 0.51± 0.30 | 0.22± 0.17 | ○ | 97.15± 4.85 | 93.69±10.10 | 80.39± 6.33 | 91.69±16.24 |
| elusage | 0.50± 0.52 | 0.74± 0.57 | 93.13±11.80 | 90.37±13.15 | 75.63±10.53 | 47.80± 6.23 | ● |
| fishcatch | 2.67± 0.44 | 2.59± 0.33 | 95.57± 4.72 | 91.29± 6.82 | 16.84± 3.52 | 12.42± 2.23 | ● |
| fruitfly | -0.04± 0.17 | -0.42± 0.31 | ○ | 92.75± 7.42 | 94.12± 8.00 | 72.57± 5.18 | 88.36±16.62 ○ |
| gascons | 1.59± 0.82 | 3.60± 0.89 | ● | 95.17±13.67 | 94.50±14.03 | 46.84±11.89 | 15.95±16.79 ● |
| housing | 1.42± 0.28 | 1.37± 0.30 | 93.76± 3.50 | 94.84± 2.76 | 28.48± 2.47 | 30.54± 3.91 | |
| hungarian | 1.12± 0.24 | -1.38± 0.27 | ○ | 99.69± 1.10 | 89.02± 8.07 | ○ | 53.80± 5.95 128.91±21.98 ○ |
| longley | 0.95± 1.57 | 2.63± 1.90 | 95.00±19.46 | 90.50±27.24 | 71.34±20.75 | 17.22± 4.90 | ● |
| lowbwt | 0.59± 0.25 | 0.70± 0.18 | 95.07± 5.18 | 95.86± 4.87 | 48.05± 2.79 | 40.97± 0.82 | ● |
| mbagrade | 0.05± 0.25 | 0.10± 0.42 | 93.76± 8.69 | 95.36±11.11 | 95.62± 4.25 | 99.57± 8.31 | |
| meta | 18.8±124.0 | 17.5±142.1 | 98.24± 1.92 | 99.45± 1.02 | ● | 5.03± 1.33 | 54.84±51.38 ○ |
| pbc | 0.27± 0.17 | 0.22± 0.17 | 95.36± 3.48 | 91.89± 4.41 | ○ | 76.51± 2.59 | 64.72± 1.85 ● |
| pharynx | 0.27± 0.25 | 0.16± 0.23 | 93.34± 5.68 | 94.62± 5.63 | 68.76± 3.78 | 70.25± 4.17 | |
| pollution | 0.14± 0.48 | 0.65± 0.60 | ● | 91.33±12.64 | 91.00±12.40 | 65.44± 7.37 | 55.93±13.61 ● |
| pwLinear | 0.93± 0.37 | 1.21± 0.30 | ● | 92.45± 6.49 | 92.15± 6.64 | 38.50± 3.47 | 32.32± 5.72 ● |
| quake | -0.03± 0.03 | -0.87± 0.06 | ○ | 97.81± 1.32 | 94.12± 1.49 | ○ | 57.97± 0.74 62.42± 0.16 ○ |
| schlvote | 0.15± 9.74 | 0.65± 5.81 | 89.00±17.88 | 96.92± 9.81 | 50.30±18.12 | 130.58±37.14 ○ | |
| sensory | -0.04± 0.14 | -0.10± 0.13 | 94.49± 3.36 | 95.90± 6.35 | 62.55± 3.00 | 63.96± 9.09 | |
| servo | 1.65± 0.36 | -0.07± 0.78 | ○ | 95.09± 5.38 | 94.97± 6.48 | 24.34± 6.18 | 45.41± 4.18 ○ |
| sleep | 0.36± 0.49 | 0.42± 0.36 | 96.92± 7.79 | 88.21±13.08 | ○ | 85.20± 6.68 | 70.77±11.89 ● |
| strike | 0.51± 0.97 | -1.09± 0.69 | ○ | 95.22± 3.03 | 98.24± 2.14 | ● | 16.52± 2.55 35.67± 8.41 ○ |
| veteran | 0.25± 0.28 | -0.52± 0.90 | ○ | 95.04± 5.83 | 95.62± 5.97 | 49.17± 5.12 | 64.79± 5.69 ○ |
| vineyard | 0.22± 0.91 | 0.89± 0.97 | 92.90±11.79 | 91.80±12.82 | 57.98±10.18 | 47.93± 8.98 | ● |

○, ● statistically significant improvement or degradation

GPR performs significantly better. It also performs significantly better on cpu and sensory, and significantly worse on servo.

We now consider the task of quantile estimation. Table 7 shows the pinball loss of our discretization-based method when used for this task (50% quantile), compared to two existing quantile regression techniques, including the nonparametric quantile estimation method NPQR, based on the results in [21]. For details of the pinball loss and the datasets used, see [21]. Note that the results from [21] are based on a single 10-fold cross-validation.

The approximate quantiles for our method were obtained from the weighted kernel density estimates in the same fashion as the interval boundaries used previously (see Section 2.3). The pinball loss was used as the performance estimate for the internal cross-validation employed to choose parameters for the SVMs.

**Table 7.** Comparison of pinball loss (50%-quantile) for proposed method (SVMs with RBF kernels, kernel density estimator, 20 bins), linear quantile regression (Linear QR), and nonparametric quantile regression (NPQR). Results for the latter two methods are reproduced from [21], with standard errors converted to standard deviations.

| Dataset | RBF SVMs (20 bins) | Linear QR | NPQR |
|---|---|---|---|
| caution | 23.71± 9.29 | 32.40± 8.73 | 22.56± 8.04 |
| ftcollinssnow | 42.87± 9.48 | 40.82± 8.85 | 39.08± 9.27 |
| highway | 23.42±10.42 | 45.39±21.12 | 25.33±10.86 |
| heights | 34.85± 2.27 | 34.50± 2.16 | 34.53± 2.16 |
| sniffer | 10.83± 2.55 | 12.78± 3.33 | 9.92± 2.82 |
| snowgeese | 14.23±10.57 | 13.85±10.38 | 18.50±14.88 |
| ufc | 21.91± 2.41 | 23.20± 2.85 | 21.22± 2.70 |
| birthwt | 37.75± 6.73 | 38.15± 5.88 | 37.19± 5.88 |
| crabs | 4.39± 1.08 | 2.24± 0.39 | 2.14± 0.36 |
| GAGurine | 16.06± 3.62 | 27.87± 4.38 | 14.57± 3.33 |
| geyser | 30.37± 5.43 | 32.50± 3.69 | 30.75± 4.20 |
| gilgais | 10.79± 1.94 | 16.12± 3.03 | 12.40± 1.98 |
| topo | 17.17± 6.73 | 26.51± 8.13 | 14.39± 4.95 |
| BostonHousing | 11.80± 2.14 | 17.50± 2.85 | 10.76± 1.83 |
| CobarOre | 40.88±15.84 | 41.93±15.60 | 39.29±20.07 |
| engel | 13.83± 3.21 | 13.72± 3.42 | 13.01± 2.55 |
| mcycle | 18.90± 4.52 | 37.88± 8.28 | 17.06± 4.26 |
| BigMac2003 | 19.46±10.82 | 21.75± 8.55 | 17.89± 9.15 |
| UN3 | 23.00± 5.50 | 26.32± 5.10 | 23.96± 5.52 |
| cpus | 6.13± 5.31 | 5.73± 3.12 | 1.06± 0.51 |

The results show that, in spite of the fact that NPQR optimizes the pinball loss directly using quadratic optimization, discretization-based estimation generally yields highly competitive results. There are only two datasets where the pinball loss is clearly significantly worse: crabs and cpu. In both cases, given the small loss, a possible explanation for the relatively poor performance is the coarseness of the discretization: running the method using 40 discretization intervals yields a reduced pinball loss of 3.84 and 3.89 respectively.

## 4 Conclusions

This paper considered conditional density estimates based on combining class probability estimators with univariate density estimators. Our results indicate that kernel estimators are generally superior to normal estimators in this context, and that both are preferable to histogram-based estimation. We have presented results for both support vector machines and random forests as underlying class probability estimators. The proposed techniques yield useful prediction intervals as well as competitive point estimates. They also show promise when compared to Gaussian process regression and nonparametric quantile estimation.

The generic aspect of the approach discussed here is valuable because of the availability of a large number of class probability estimation schemes that are potentially applicable. An avenue for future work is to investigate the performance of other univariate density estimators in the context considered in this paper. For example, rather than using a kernel density estimator or a normal estimator, one could apply a mixture model.

## References


1. Hyndman, R.J., Bashtannyk, D.M., Grunwald, G.K.: Estimating and visualizing conditional densities. Journal of Computational and Graphical Statistics **5**(4) (1996) 315–336
2. Holmes, M., Gray, A., Isbell, C.: Fast nonparametric conditional density estimation. In: Proc 23rd Conf on Uncertainty in AI, AUAI Press (2007)
3. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press (2006)
4. Tay, A.S., Wallis, K.F.: Density forecasting: A survey. Journal of Forecasting **19** (2000) 235–254
5. Neuneier, R., Ferdinand Hergert, W.F., Ormoneit, D.: Estimation of conditional densities: A comparison of neural network approaches. In: Proc Int Conf on Artificial Neural Networks, Springer (1994) 689–692
6. Bishop, C.M.: Neural networks for pattern recognition. Oxford UP (1995)
7. Williams, P.M.: Using neural networks to model conditional multivariate densities. Neural Computation **8**(4) (1996) 843–854
8. Papadopoulos, G., Edwards, P., Murray, A.: Confidence estimation methods for neural networks: a practical comparison. IEEE Transactions on Neural Networks **12**(6) (2001)
9. Michael Carney, Pádraig Cunningham, J.D., Lee, C.: Predicting probability distributions for surf height using an ensemble of mixture density networks. In: Proc 22nd Int Conf on Machine learning, ACM Press (2005) 113–120
10. Stützle, E., Hrycej, T.: Numerical method for estimating multivariate conditional distributions. Computational Statistics **20**(1) (2005) 151–176
11. Carney, M., Cunningham, P.: Making good probability estimates for regression. In: Proc 17th Europ Conf on Machine Learning, Springer (2006) 582–589
12. Carney, M., Cunningham, P.: Calibrating probability density forecasts with multi-objective search. In: Proc 17th Europ Conf on AI, IOS Press (2006) 791–792
13. Davies, S., Moore, A.: Interpolating conditional density trees. In: Proc 18th Annual Conf on Uncertainty in AI, Morgan Kaufmann (2002) 119–12
14. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, San Francisco, 2nd edition (2005)
15. Schapire, R.E., Peter Stone, David A. McAllester, M.L.L., Csirik, J.: Modeling auction price uncertainty using boosting-based conditional density estimation. In: Proc 19th Int Conf on Machine Learning, Morgan Kaufmann (2002) 546–553
16. Hjort, N.L., Walker, S.G.: A note on kernel density estimators with optimal bandwidths. Statistics & Probability Letters **54** (2001) 153–159
17. Platt, J.C.: Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In: Advances in Large Margin Classifiers. MIT Press (1999) 61–74
18. Nadeau, C., Bengio, Y.: Inference for the generalization error. Machine Learning **52** (2003) 239–281
19. Frank, E., Hall, M.: A simple approach to ordinal classification. In: Proc 12th Europ Conf on Machine Learning, Springer (2001) 145–156
20. Frank, E., Trigg, L.E., Holmes, G., Witten, I.H.: Naive Bayes for regression (technical note). Machine Learning **41**(1) (2000) 5–25
21. Takeuchi, I., Le, Q.V., Sears, T.D., Smola, A.J.: Nonparametric quantile estimation. Journal of Machine Learning Research (2006) 1231–1264