Iterative Quantiles Nearest-Neighbors

Karsten Maurer

Miami University

July 30, 2018

Introduction

Motivation

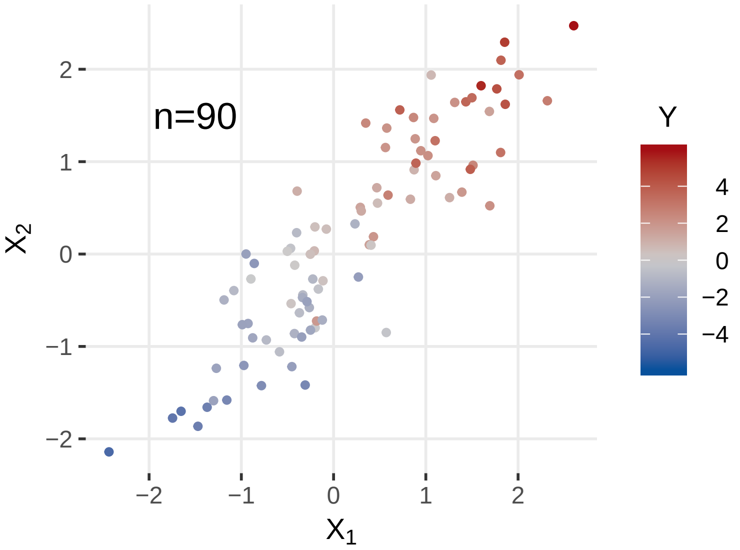
k Nearest-Neighbors (KNN):

- Provides localized non-parametric estimation over feature space
- Computationally expensive distance calculations and sorting
- Efficient algorithms for approximate nearest neighborhoods (AKNN)
- kd-tree AKNN (Arya et al., 1998)
- cover-tree AKNN (Beygelzimer et al., 2006)

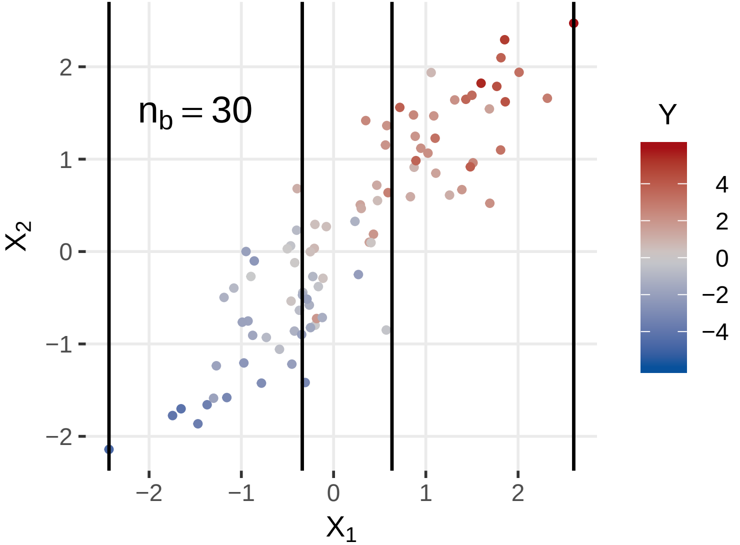
Iterative Quantile Nearest-Neighbors (IQNN):

- Can we make neighborhoods with binned-partitions of feature space?
- Checking for points in intervals fast
- Partition with k training points per partition
- Use iterative algorithm of quantile-based univariate partitions

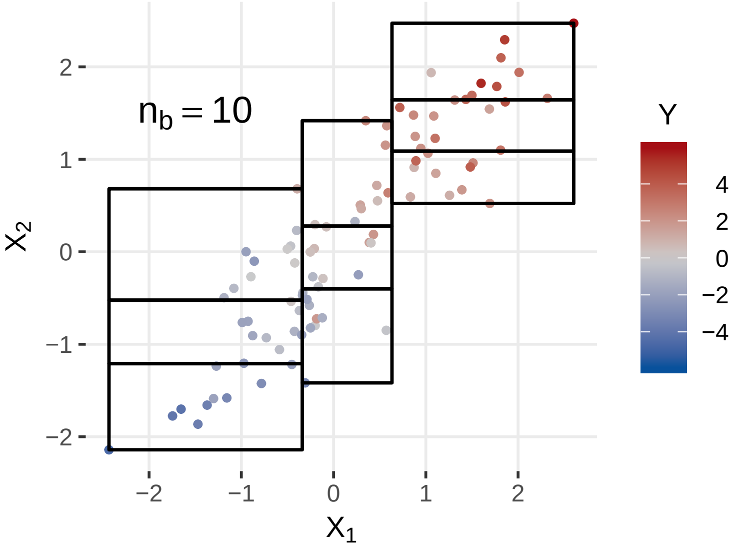
IQNN - Simple Demonstration



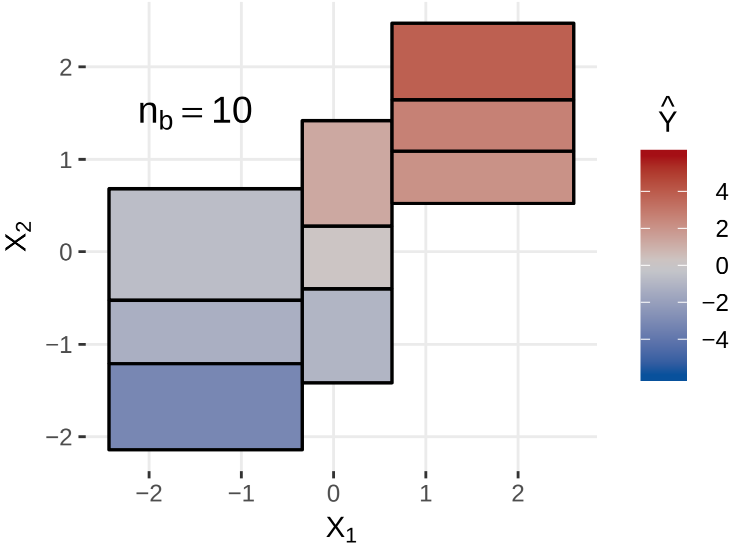
IQNN - Simple Demonstration



IQNN - Simple Demonstration



IQNN - Simple Demonstration



IQNN Query Structure

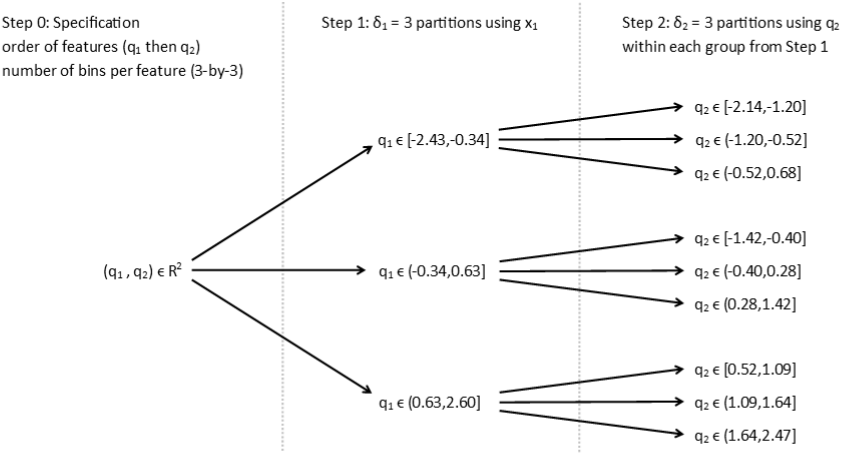


Figure 1: Interval R-tree structure generated by iterative quantile binning in simulated feature data example from above.

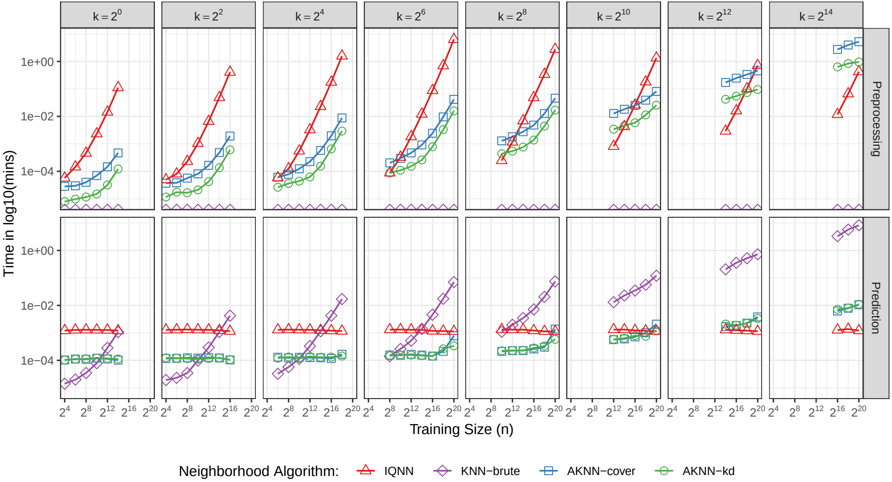
Evaluation

Evaluating IQNN Performance

Computational Efficiency: *Timing study*

- Test with simulated data sets of varying sizes: n=2^4,2^6,...,2^{20}
- Test with various neighborhood sizes: k=2^0,2^4,...,2^{14}
- Speed of pre-processing with IQNN vs AKNN methods
- Speed of identifying neighboring points with IQNN vs AKNN methods

Timing Study

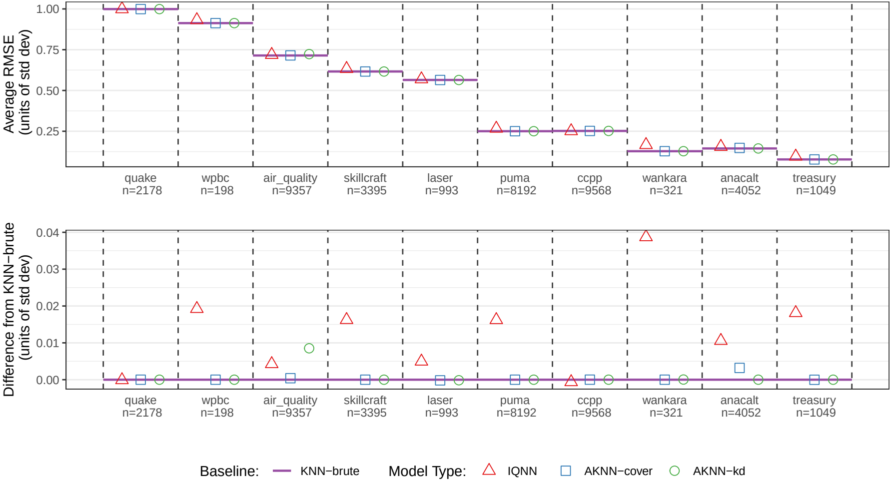


Evaluating IQNN Performance

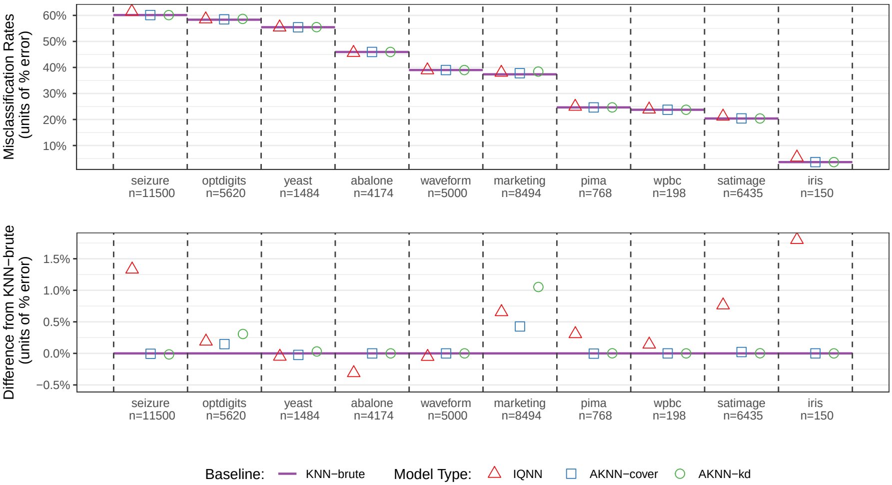
Predictive Accuracy: *Empirical Comparison*

- Test with real data sets: 10 regression problems, 10 classification problems
- Data Repos: UCI (archive.ics.uci.edu) and KEEL (sci2s.ugr.es/keel)
- Accuracy assessed using 10-fold CV with tuned models from each case

Regression Accuracy



Classifier Accuracy



Discussion

Results

Timing Study:

- Requires considerable pre-processing similar to other AKNN methods
- Queries on R-tree structure depends only on number of bins
- Advantage for large n, large k applications

Predictive Accuracy:

- Weak accuracy relative to KNN for regression - less fine control on tuning parameters
- Comparable accuracy relative to KNN for classification - neighborhood voting robust

Thanks!

Thank you for listening!

Any Questions?

contact: maurerkt@miamioh.edu

Algorithm (detail)

Specification: Define order of features $\{X_1, X_2, ..., X_p\}$ to match desired iterative binning order and number of bins $\{\delta_1, \delta_2, ..., \delta_p\}$ for partitioning in each dimension

Binning:

1. Partition all points into δ_1 quantile bins on feature X_1 with index sets $\{B_1, ...B_{\delta_1}\}$ such that $B_\ell = \{i \mid b^q_{X_1}(x_{i1}) = \ell\} \ \forall \ \ell = 1, ..., \delta_1$
2. Repeat the following for $j = 2, ..., p$:
   1. Define $C_{st} = \{i \mid i \in B_s \text{ and } b^q_{X_j}(x_{ij}) = t\} \ \forall \ s = 1, ..., \prod_{d=1}^{j-1} \delta_d$ and $t = 1, ..., \delta_j$ to subdivide each B_s from previous step with δ_j quantile bins on feature X_j
   2. Redefine index sets $\{B_1, ..., B_L\}$ such that $B_\ell = C_{st}$, where $\ell = t(s-1) + t$ to combine parent and child indices of sets into unique indices

Outputs:

1. Bin neighbor sets $\vec{\mathbf{x}}_\ell = \{\vec{x}_i \mid i \in B_\ell\} \ \forall \ \ell = 1, ..., L$, where $L = \prod_{j=1}^{p} \delta_j$
2. Hyper-rectangular bins $\ell = 1, ..., L$ containing points $x_{ij} \in (\beta_{j\ell 1} \ , \ \beta_{j\ell 2}] \ \forall \ j = 1, ..., p$