The Thirty-Second International Florida
Artificial Intelligence Research Society Conference (FLAIRS-32)

Balanced k-Nearest Neighbors

Brian Cook, Manfred Huber
Dept. of Computer Science and Engineering
University of Texas at Arlington
brian.cook@mavs.uta.edu, huber@cse.uta.edu

Abstract

Classic k-Nearest Neighbor (kNN) algorithms approximate a function at a query point based on the k-nearest training observations. In real-world datasets, however, the set of k neighbors is frequently not uniformly distributed around a given query point. This can result in a locally biased estimate and thus in degraded results.

This paper presents two new kNN algorithms that adjust the weight of the k-nearest neighbors to achieve a more balanced distribution. Experiments on real-world and synthetic datasets identify conditions under which the algorithms can improve accuracy with minimal increase in computation time.

Introduction

k-Nearest Neighbor (kNN) algorithms have long been used for regression and classification. They are simple, fast and often perform as well or better than more complex methods (Wu et al. 2008).

kNN methods estimate a function based on the k-nearest neighbors of a query point in a set of training samples. Typically the estimate is a simple average of the neighbor values, or an inversely-weighted average based on distance from the query point. More generally, the estimator may use locally-weighted regression to fit a function to the neighbors.

These approaches only consider the distance of the neighbors, not their spatial distribution, and implicitly assume a uniform distribution around the query point. Yet local variation in sample density is common, arising naturally when training samples are randomly distributed in the input space, as well as from bias in the training data collection process. As a result, kNN estimates are more heavily influenced by regions of higher sample density in the local neighborhood, since those regions are over-represented in the set of k-nearest neighbors. In general, this *local neighborhood bias* is not desirable since there is no a priori reason to prefer samples in any particular region.

Figure 1a illustrates this using a random set of samples from the function $y = sin(x)$. In the highlighted regions, 5-NN overestimates or underestimates the function because most of the neighbors are on one side of the query point. In regions where the nearest neighbors are more equally distributed, the estimate is much closer.

Motivated by this observation, we can adjust the neighbor weights to approximate an equal distribution along each feature axis in the input space. In this example, if 4 neighbors are in the negative x direction, and 1 neighbor is in the positive x direction, we can adjust the relative weight of the positive neighbor by a factor of 4. Intuitively, that neighbor is likely to provide more information since it is in a region that is under-represented among the neighborhood samples. Figure 1b shows results of the Axis-balanced 5-NN algorithm described below using the same sample points as 1a. The mean-squared error (MSE) is reduced by 11%.

A further observation is that if we have noise-free samples and multiple neighbors are in the same direction from the query point, the nearest neighbor in that direction will be the most informative. Additional neighbors further away in the same direction may reduce accuracy rather than improve it. This motivates us to consider at most one neighbor in each axis direction from the set of nearest neighbors, using a subset of neighbors that form an axis-aligned bounding box around the point. Figure 1c shows the effect of this Box 5-NN algorithm. The MSE is reduced by 58%.

Approach

For real-valued outputs $y \in \mathbb{R}$, regression approximates a function $f : X \mapsto \mathbb{R}$ where $X \in \mathbb{R}^d$ is a metric space with distance function $\delta : X \times X \mapsto \mathbb{R}$. Similarly, for categorical outputs $y \in C$, classification approximates a function $f : X \mapsto C$.

Let S be a set of N training samples $\{x^{(i)}, y^{(i)}\}_{i=1}^{N}$ where $x^{(i)} \in X$ is an input variable and $y^{(i)}$ is the corresponding output variable. For query point $q \in X$ and distance function δ, define the ordered set $A \subset S$ of *k-nearest neighbors* $\{a^{(i)}\}_{i=1}^{k}$ such that:

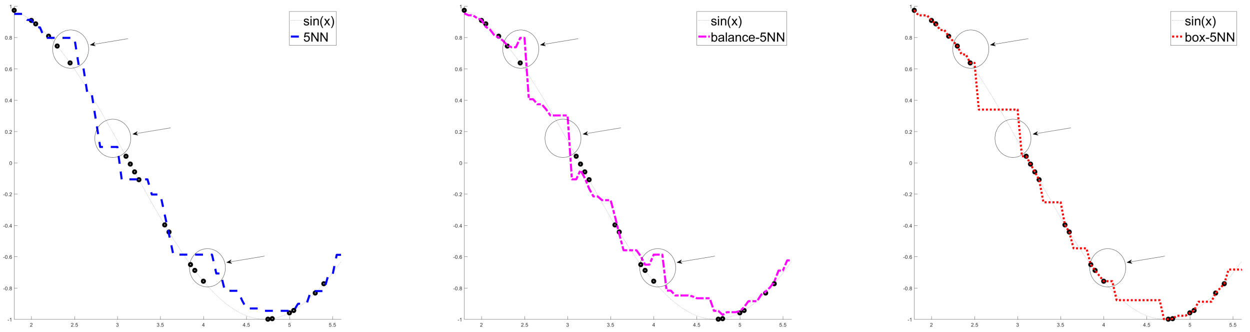
$$|A| = k$$

$$\forall a \in A, b \in S - A, \delta(a.x, q) \leq \delta(b.x, q)$$

$$\forall a^{(i)}, a^{(j)} \in A, i < j, \delta(a.x^{(i)}, q) \leq \delta(a.x^{(j)}, q)$$

Let $w^{(i)}$ be the weight associated with neighbor $a^{(i)}$.

Let ω be an initial weighting function that calculates weights $\omega^{(i)}$ from A, $\omega : \mathbb{R}^d \times \mathbb{R}^{d \times k} \mapsto \mathbb{R}^k$.



(a) 5-NN (b) Axis-balanced 5-NN (c) Box 5-NN

Figure 1: Local Neighborhood Bias

Standard kNN regression uses a weighted average of the k-nearest neighbors. For query point $q \in X$, the true function $f(q)$ is estimated by:

$$\hat{f}(q) = \frac{1}{Z_q} \sum_{i=1}^{k} w^{(i)} y^{(i)} \tag{1}$$

where $y^{(i)}$ is the output value of the i-th nearest neighbor, $w^{(i)}$ is the corresponding weight from the used weighting function, and $Z_q = \sum_{i=1}^{k} w^{(i)}$ is a normalization factor.

Similarly, kNN classification predicts discrete class labels $y \in C$ as:

$$\hat{f}(q) = \operatorname*{argmax}_{c \in C} \sum_{i=1}^{k} \begin{cases} w^{(i)} & \text{if } y^{(i)} = c \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Basic kNN weights all k-nearest neighbors equally with $w^{(i)} = 1/k$. Effectively it assumes a uniform function value in the local neighborhood spanned by the k neighbors.

Distance-weighted kNN (w-kNN) assigns greater influence to neighbors closer to the query point than neighbors farther away. A wide variety of weighting functions based on distance have been investigated (Titterington 1987). A simple choice is to use the inverse distance, $w^{(i)} = 1/\delta(x^{(i)}, q)$.

Axis-balanced kNN

The Axis-balanced kNN algorithm uses Equations (1) and (2) but modifies the weighting factors $w^{(i)}$ as follows.

For query point q, find the k-nearest neighbors A. For each $a^{(i)} \in A$, assign weight $w^{(i)}$ using some initial weighting function ω. Now for each dimension $j \in [1..d]$, partition the neighbors in each direction along axis j.

$$\begin{aligned} L_j &= \{a^{(i)} : a^{(i)} \in A, a.x_j^{(i)} < q_j\} \\ R_j &= \{a^{(i)} : a^{(i)} \in A, a.x_j^{(i)} > q_j\} \\ E_j &= \{a^{(i)} : a^{(i)} \in A, a.x_j^{(i)} = q_j\} \end{aligned} \tag{3}$$

If $|L_j| > 0$ and $|R_j| > 0$ then adjust weights $w^{(i)}$ using:

$$w^{(i)} \leftarrow \begin{cases} w^{(i)}(|L_j| + |R_j|)/|L_j| & \text{if } a^{(i)} \in L_j \\ w^{(i)}(|L_j| + |R_j|)/|R_j| & \text{if } a^{(i)} \in R_j \\ w^{(i)} & \text{if } a^{(i)} \in E_j \end{cases} \tag{4}$$

Compared to standard kNN the additional computation time is $O(dk)$.

Box kNN

The Box kNN algorithm also uses Equations (1) and (2) but modifies the weighting factors $w^{(i)}$ as follows.

For query point q, find the k-nearest neighbors A. For each $a^{(i)} \in A$, assign weight $w^{(i)}$ using some initial weighting function ω. Partition the neighbors along each axis j as in Equation (3) and identify nearest neighbors in each dimension as:

$$\tilde{L}_j = \{l \in L_j : l.x_j = \max_{a \in L_j} a.x_j\}$$

$$\tilde{R}_j = \{r \in R_j : r.x_j = \min_{a \in R_j} a.x_j\}$$

Using this, adjust weights $w^{(i)}$ using:

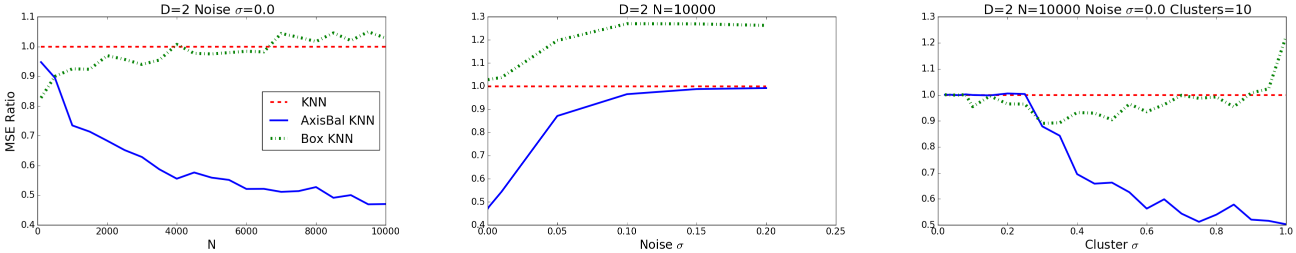
$$w^{(i)} \leftarrow w^{(i)} \sum_{j=1}^{d} \begin{cases} 2 & \text{if } a^{(i)} \in E_j \\ 1 & \text{if } a^{(i)} \in \tilde{L}_j \cup \tilde{R}_j \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

As with Axis-balanced kNN, the additional computation time is $O(dk)$.

Synthetic Experiments

To study the effects of different dataset properties on the estimation accuracy of the kNN algorithms, synthetic datasets were generated using the radial sin function $f(x) = sin(2\pi\|x\|)$ for sample points generated in a unit hypercube. The standard and balanced kNN algorithms were all distance-weighted using inverse Euclidean distance as the initial weighting function $\omega^{(i)} = 1/\|q, x^{(i)}\|$.

Figure 2 shows results for 2 dimensions. Plots show the ratio of MSE vs. standard kNN to make comparison easier.



(a) Effect of Sample Density (b) Effect of Noise (c) Effect of Cluster Density

Figure 2: Synthetic Datasets: Ratio of MSE vs. kNN

Effect of Uniform Sample Density

Figure 2a compares the algorithms over a range of uniform sample densities. In areas with very low density and low noise, Box kNN works best. As density increases, Axis-balanced kNN performs increasingly better.

Effect of Noise

Figure 2b compares the algorithms over a range of noise levels. Box kNN performs relatively poorly for noisy data. This is not surprising since it considers only a subset of its nearest neighbors, resulting in higher variance in its estimates. Axis-balanced kNN performance is more tolerant of noise, but also deteriorates relative to kNN as noise increases.

Effect of Non-uniform Sample Density

To observe the effect of a non-uniformly generated training set, training samples were drawn from 10 normally distributed clusters $x \sim N(\mu_i, \sigma_{cluster})$ and the density of the clusters was varied by varying the standard deviation $\sigma_{cluster}$. Test sample points were drawn from a quasirandom uniform distribution over the unit hypercube using a Hammersley sequence.

Figure 2c compares the algorithms over a range of $\sigma_{cluster}$ densities.

For small dense clusters, Box kNN works best. As cluster density decreases and the training distribution becomes more uniform, Axis-balanced kNN performs increasingly better. Note that one of the balanced kNN algorithms always outperformed standard kNN on these synthetic datasets.

Real-World Experiments

Experiments were performed on regression and classification datasets from the UCI machine learning repository (Dheeru and Karra Taniskidou 2017).

Each experiment performed 5-fold cross-validation to measure performance and was repeated until reaching confidence level $P > 0.99$ that relative error is less than 0.01.

Simple forward and backward stepwise feature selection was performed for each combination of dataset and algorithm. In addition, each experiment was performed using original unscaled features and with features scaled to unit standard deviation, and results are shown using the best scaling option.

Each algorithm was further tested using initial weighting functions $\omega^{(i)} = 1/k$ and $\omega^{(i)} = 1/\delta(x^{(i)}, q)$ and results are shown using the best weighting.

Each experiment varied k from 1 to 40 and selected the k value with the best accuracy. For a fair comparison, it is necessary to allow different k values for each algorithm since each one utilizes the k nearest neighbors differently.

Tables 1 and 2 show summary results. For each dataset the total number of samples and dimensions is shown. Each algorithm shows the number of selected features d and the value of k with the best accuracy.

In the majority of datasets, balancing using one of the two algorithms can improve performance with only two datasets leading to standard kNN having slightly better performance and two sets where it is tied.

Higher values of k suggest increasing noise since better accuracy is obtained by averaging more samples from a larger neighborhood. The results show standard kNN is likely the best choice in this case. Balancing in the presence of high noise increases variance and is less effective.

The results indicate that the balancing algorithms work best on datasets in which the intrinsic dimensionality is less than around 10 dimensions. Beyond this level the sample density is likely insufficient to perform effective balancing.

Related Work

The choice of distance metric is crucial for nearest-neighbor algorithms and much effort has focused on learning distance metrics for classification. Global methods described in (Weinberger and Saul 2009) apply global linear transformations to the input data. Locally discriminative transformations are applied in (Hastie and Tibshirani 1996) and (Mu, Ding, and Tao 2013). In (Wang, Neskovic, and Cooper 2007) the distance to each training point is scaled inversely with its distance to the class boundary, resulting in fewer neighbors from high-variance regions.

Other approaches vary the neighborhood size. In (Garcia-Pedrajas, Romero Del Castillo, and Cerruela-Garcia 2017) locally-adaptive k values are determined for each training sample. In (Pan, Wang, and Ku 2017) the local k-neighborhood is extended to include training samples whose k-neighborhood includes the query point, increasing the influence of neighbors from lower-density regions.

Yet to our knowledge, no existing methods directly take

Dataset	Samples	Dim	kNN			Axis-balanced kNN			Box kNN		
			d	Best K	MSE	d	Best K	MSE	d	Best K	MSE
3D Roads	434874	2	2	2	1.070	2	2	1.059	**2**	**3**	**0.925**
Airfoil Self Noise	1503	5	5	2	5.10	**4**	**4**	**4.99**	**3**	**30**	**4.95**
Concrete	1030	8	5	4	48.2	**6**	**4**	**47.3**	5	10	52.4
Energy Cool	768	8	7	4	3.47	**7**	**4**	**3.20**	4	16	3.88
Energy Heat	768	8	5	1	0.355	6	4	0.379	**6**	**2**	**0.345**
Parkinson's Motor	5876	16	**16**	**13**	**44.1**	16	2	56.1	16	13	44.9
Power	9568	4	4	7	13.5	4	5	15.1	**4**	**18**	**11.8**
Protein	45731	9	9	6	13.7	9	2	15.1	**9**	**10**	**12.9**
Red Wine Quality	1599	11	**4**	**31**	**0.326**	4	37	0.346	8	31	0.341
White Wine Quality	4898	11	**8**	**20**	**0.369**	4	50	0.392	**10**	**31**	**0.369**
Yacht	308	6	**2**	**8**	**1.95**	**2**	**8**	**1.96**	2	3	2.05

Table 1: Real-World Regression Datasets

Dataset	Samples	Dim	kNN			Axis-balanced kNN			Box kNN		
			d	Best K	Error %	d	Best K	Error %	d	Best K	Error %
Balance	625	4	4	17	10.06	**4**	**13**	**9.44**	4	11	19.01
Iris	150	4	4	13	3.15	**4**	**20**	**1.90**	4	1	4.24
Wine	178	13	**8**	**32**	**0.742**	6	12	3.621	9	11	1.040

Table 2: Real-World Classification Datasets

into account asymmetric distribution of samples in the local neighborhood.

Conclusions and Further Work

This paper introduces two modifications of standard kNN. Axis-balanced kNN adjusts the weights of the k-nearest neighbors to approximate a balanced distribution along each feature axis. Box kNN adjusts the weights of the k-nearest neighbors to include only the nearest neighbors in each feature axis direction. Neither method requires additional parameters or tuning beyond that required by kNN.

Experiments using synthetic and real-world data demonstrate balanced kNN approaches can outperform standard kNN when correctly selected, and often show more stability for varying values of k.

Axis-balanced kNN performs better when training data is more dense and there is little to moderate noise. Box-kNN performs better when training data is more sparse and there is little noise. When there is a high level of noise, or the intrinsic dimensionality of the data is 10 or higher, standard kNN is likely the best choice.

Balancing here is performed along feature axes of the original sample data. Balancing in other directions may yield further improvement, such as using whitened data, or by first globally transforming the sample data using distance metric learning methods.

References

Dheeru, D., and Karra Taniskidou, E. 2017. UCI Machine Learning Repository.

Garcia-Pedrajas, N.; Romero Del Castillo, J. A.; and Cerruela-Garcia, G. 2017. A Proposal for Local k Values for k-Nearest Neighbor Rule. *IEEE Trans on Neural Networks and Learning Systems* 28(2):470–475.

Hastie, T., and Tibshirani, R. 1996. Discriminant adaptive nearest neighbor classification. *IEEE Trans on Pattern Analysis and Machine Intelligence* 18(6):607–616.

Mu, Y.; Ding, W.; and Tao, D. 2013. Local discriminative distance metrics ensemble learning. *Pattern Recognition* 46(8):2337–2349.

Pan, Z.; Wang, Y.; and Ku, W. 2017. A new general nearest neighbor classification based on the mutual neighborhood information. *Knowledge-Based Systems* 121:142–152.

Titterington, D. M. 1987. A Re-Examination of the Distance-Weighted k-Nearest Neighbor Classification Rule. *IEEE Trans on Systems, Man, and Cybernetics* 17(4):689–696.

Wang, J.; Neskovic, P.; and Cooper, L. N. 2007. Improving nearest neighbor rule with a simple adaptive distance measure. *Pattern Recognition Letters* 28(2):207–213.

Weinberger, K. Q., and Saul, L. K. 2009. Distance Metric Learning for Large Margin Nearest Neighbor Classification. *Journal of Machine Learning Research* 10:207–244.

Wu, X.; Kumar, V.; Ross, Q. J.; Ghosh, J.; Yang, Q.; Motoda, H.; McLachlan, G. J.; Ng, A.; Liu, B.; Yu, P. S.; Zhou, Z. H.; Steinbach, M.; Hand, D. J.; and Steinberg, D. 2008. Top 10 Algorithms in Data Mining. *Knowledge and Information Systems* 14(1):1–37.