# The labeled cell classifier: a fast approximation to $k$ nearest neighbors [*]

Alessandro M. Palau and Robert R. Snapp
Department of Computer Science
University of Vermont
Burlington, VT 05405 U.S.A.
(palau@cs.uvm.edu and snapp@cs.uvm.edu)

December 10, 1997

**Abstract**

A $k$-nearest-neighbor classifier is approximated by a labeled cell classifier that recursively labels the nodes of a hierarchically organized reference sample (*e.g.*, a $k$-d tree) if a local estimate of the conditional Bayes risk is sufficiently small. Simulations suggest that the labeled cell classifier is significantly faster than $k$-d tree implementations for problems with small Bayes risk; and may be more accurate as a larger reference sample can be examined in a fixed amount of time.

**Key words:** $k$ nearest neighbor classifier, $k$-d tree.

## 1 INTRODUCTION

The $k$-nearest neighbor classifier (Fix and Hodges, [5]) continues to serve as a solution to practical recognition problems (Smith et al. [12]), as a basis for developing new pattern classifiers (Hastie and Tibshirani [11]), and as a framework for illuminating fundamental aspects of pattern recognition. This algorithm has simple implementations, is analytically tractable, and is nearly optimal in the large sample limit (Cover and Hart [3]). The main disadvantage of this algorithm is that, as a nonparametric algorithm, it needs to consult a

[*]Submitted to the 14th ICPR'98 (Algorithms and Techniques). Please direct all correspondence to the second author.

reference sample during each classification. Consequently, the average classification time increases with the size of the reference sample.

Many studies have produced efficient implementations of the $k$-nearest-neighbor algorithm. Exact implementations organize the reference sample into hierarchical data structures that can be quickly accessed (Friedman, Baskett, and Shustek [6], Fukunaga and Narenda [8], Kim and Park [10]). An implementation using a multidimensional binary search tree [1] (Friedman, Bentley, and Finkel [7]), for example, can classify an $n$-dimensional input pattern, using a reference sample of $m$ classified patterns, within an average of $O((\sqrt[n]{k}+1)^n \log m)$ distance computations. Approximate implementations accelerate classification by editing the reference sample (Hart [9], Wilson [13]), or by truncating the search algorithm (Ayra and Mount [1]). The present study, which began as an evaluation of the computational efficiency of existing methods, introduces a fast and simple approximation of the $k$-nearest-neighbor classifier. Simulations suggest that this implementation, which we call the *labeled cell classifier*, is useful for problems that provide an abundant supply of classified patterns, are described by smooth probability distributions, and have a small Bayes risk (*e.g.*, pixel classification of satellite images).

The labeled cell classifier is loosely based on Friedman, Bentley, and Finkel's implementation, in that a classified reference sample is organized into a $k$-d tree.[2] Each node in the tree, which corresponds to a hierarchical cell (or volume) in the feature space, is assigned a class label (during preprocessing) if a recursive local estimate of the conditional Bayes risk is sufficiently small. If during classification, an input pattern falls within a labeled cell, then the pattern is immediately assigned to the indicated class. Otherwise, the class is determined from the exact set of $k$ nearest neighbors in the $k$-d tree.

The classes assigned to patterns that fall within the labeled cells may differ occasionally from the results of the $k$-nearest neighbor algorithm. Thus, like Hart's condensed nearest

[1]This is commonly called a $k$-d tree, where $k$ in this context denotes the dimensionality of the search space, not the number of nearest neighbors[2].

[2]Note that analogous classifiers can be built on top of other hierarchical implementations, such as the ordered-partition classifier of Kim and Park [10]

neighbor rule [9], the labeled cell classifier only approximates the classic algorithm. However, computer experiments suggest that if a classification needs to be performed in a fixed amount of time, then the new algorithm may attain greater accuracy than other implementations of the $k$-nearest-neighbor classifier, as the computation saved in the labeled cells allows this new algorithm to process a larger reference sample.

Section 2 reviews the $k$-nearest-neighbor classifier, and its implementation using $k$-d trees. Section 3 describes the labeled cell classifier, including the method used to label the cells. Section 4 describes our computer experiments, both with artificial data, and data extracted from a multispectral thematic mapper image. Concluding remarks appear in Section 5.

## 1.1 Notation

Following popular convention (cf., Duda and Hart [4]), we assume that patterns are represented as *feature vectors* in an $n$-dimensional Euclidean metric space. Each dimension of the feature space corresponds to a measurable attribute of a pattern. Furthermore, we assume that each pattern is generated from a unique state of nature, or pattern class. Feature vectors are denoted by bold roman letters, as in $\mathbf{x} \in \mathbb{R}^n$; class labels, by script letters, as in $\ell \in \{1, \ldots, C\}$, where $C > 1$ denotes the number of pattern classes. A reference sample of $m$ labeled feature vectors is thus represented as,

$$\mathcal{X}_m = \{(\mathbf{x}^i, \ell^i) \in \mathbb{R}^n \times \{1, \ldots, C\} \mid i = 1, 2, \ldots, m\}.$$

We use superscripts to distinguish different feature vectors, and subscripts to distinguish their components. Thus $x_j^i$ denotes the $j$-th component of $\mathbf{x}^i$.

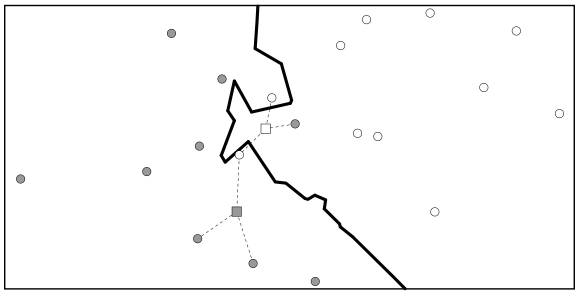

**Figure 1**: The jagged curve represents the decision boundary induced by a $k$-nearest neighbor classifier, assuming a Euclidean metric in $\mathbb{R}^2$, $k = 3$, and the reference sample indicated by the small gray and white circles, the interior color represents the class label. The two squares represent two hypothetical input patterns, each connected to its three nearest neighbors with a dashed line. The left square is thus assigned to the gray class, and the right square, to the white class.

## 2 THE *k*-NEAREST-NEIGHBOR CLASSIFIER

The $k$-nearest-neighbor classifier, as described by Fix and Hodges [5], requires a metric $d$, a positive integer $k$, and a reference sample $\mathcal{X}_m$ of $m$ labeled patterns. A new input vector $\mathbf{x} \in \mathbb{R}^n$ is classified using the subset of $k$-feature vectors of $\mathcal{X}_m$ that are closest to $\mathbf{x}$ with respect to the given metric $d$. Pattern $\mathbf{x}$ is then assigned to the class that appears most frequently within the $k$-subset. (Ties are resolved by an arbitrary procedure.) A simple illustration using a reference sample of twenty reference vectors appears in Fig. 1.

### 2.1 The *k*-d tree implementation

The labeled cell algorithm, described in Section 3, is based on the implementation of Friedman, Bentley, and Finkel[7] that organizes the reference sample $\mathcal{X}_m$ into an $n$-dimensional binary tree, such that the root node represents the entire feature space, and each node in the tree represents an isothetic cell that contains a subset of $\mathcal{X}_m$. The two descendants of each nonterminal node divide the parent cell along one coordinate, called the *key*, such that the number of reference patterns in each child cell differs at most by one. The key may be the coordinate of greatest variation of the reference vectors in the parent cell, and the threshold may be the median of their projections along the chosen coordinate. Pairs of descendants

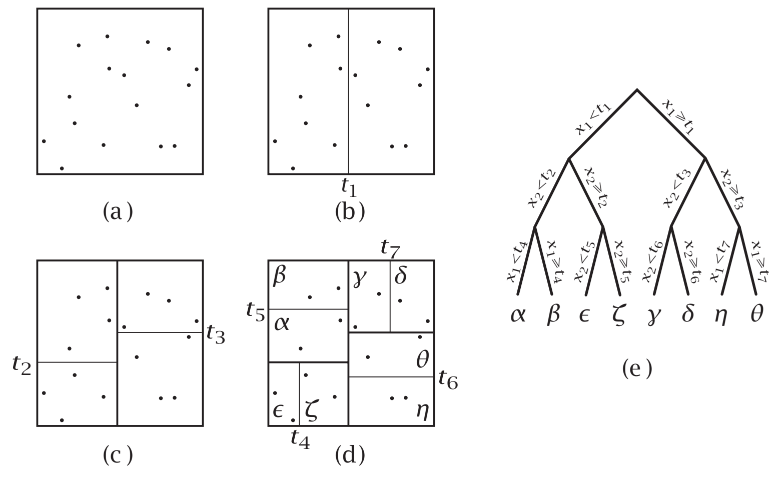

**Figure 2**: (*a*) A *k*-d tree of depth three is constructed from this set of sixteen feature vectors in $\mathbb{R}^2$ that forms the root of the tree. (*b*) The set is bisected into left and right portions, forming the two descendent nodes of the root, as the largest variation appears along the horizontal coordinate. (*c*) Each resulting subset is further divided into two equal partitions along the vertical coordinate, forming the four nodes at depth two in the tree. (*d*) Each resulting subset is then divided along the coordinate of greatest variation. Each resulting cell, labeled with a greek letter, contains two feature vectors, and forms a leaf node of the *k*-d tree (*e*).

are added recursively until the number of vectors in a cell does not exceed a bucket size $b$. Note that the nodes at a constant depth represent a partition of the feature space, as do the leaf nodes. Fig. 2 displays a $k$-d tree constructed from a reference sample of 16 points in $\mathbb{R}^2$, with $b = 2$.

After the tree is completed, the $k$ feature vectors in the tree that are nearest to a given input pattern $\mathbf{x}$ can be identified. A priority queue is used to maintain the $k$ feature vectors encountered so far that are closest to $\mathbf{x}$. Beginning with the root, nodes in the tree are examined recursively until it is certain that the $k$ nearest neighbors have been found. If the current node is a leaf node, then the priority queue is updated after examining its $b$ or fewer feature vectors. Otherwise the key $i$ and threshold value $t$ of the node are examined, and the recursive procedure is applied first to the descendant that falls on the same side of $t$ as $x_i$, and then to its sibling. For efficiency, nodes are only examined if their cell boundaries are closer to $\mathbf{x}$ than the $k$-th nearest neighbor found so far (the bounds-overlap-ball test); and the search is stopped as soon as the $k$-th nearest neighbor is closer to $\mathbf{x}$ than the boundaries of every unexamined cell (the ball-within-bounds test).

The average number of leaves (or buckets) examined is bounded above by $(\sqrt[n]{G(n)k/b} + 1)^n$, where $G(n)$ is a geometric factor that represents the ratio of the volume of an $n$-dimensional hypercube to that of the largest ball it encloses[7]. (For a Euclidean metric, $G(n) = (4/\pi)^{n/2}\Gamma(n/2 + 1)$.) Thus the average number of feature vectors examined is less than $b(\sqrt[n]{G(n)k/b} + 1)^n$. In practice, it is often desirable for $k$ to be an increasing function of $m$.

## 3 THE LABELED CELL CLASSIFIER

The labeled cell algorithm is designed to reduce the number of feature vectors examined during each classification. As in the previous implementation, the reference sample is organized into a multidimensional binary search tree using the coordinates of the feature vectors as keys. An integer $k' \geq k$ and a fraction $\alpha > 0$ are selected. A central test vector from each leaf cell is then classified with an exact $k'$-nearest-neighbor classifier (e.g., the previous implementation). This test vector could be the centroid of the leaf cell (assuming it is compact), or the sample mean of its reference vectors. If the number of $k'$-nearest-neighbors that belong to the most frequent class exceeds $\lfloor \alpha k' \rfloor$, then the leaf cell is given the label of that class.[3] (Otherwise, it remains unlabeled.) Nonterminal nodes are examined recursively: if two siblings share a common class label, then their parent is assigned the same label.

Input patterns are classified by the $k$-d tree algorithm, with one important exception: if an input pattern belongs to a cell that is labeled, then it is immediately assigned to the indicated class. Thus no reference vectors are examined if an input falls within a labeled cell. For different values of $\alpha$, $k'$, and $k$, the labeled cell algorithm implements a variety of classifiers: $\alpha = 1$ yields an exact $k$-nearest neighbor classifier, and $\alpha \leq 1/C$, a pure cell

[3]For simplicity, we assume a zero-one loss matrix, so cells are labeled if their local estimate of the conditional risk is less than $1 - \alpha$. It is straightforward to generalize the algorithm to a asymmetric, multiclass, risk function.

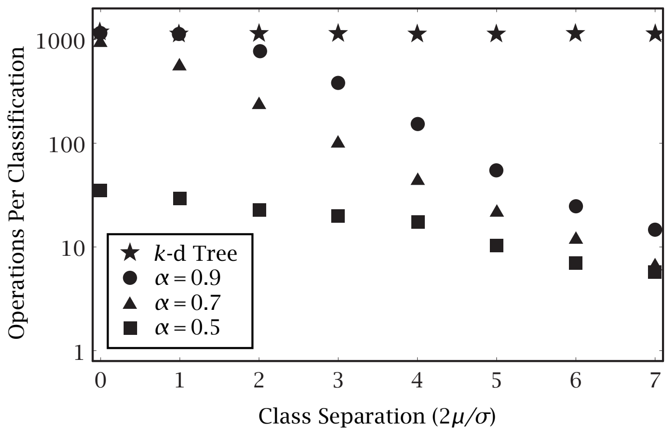

**Figure 3**: A semilogarithmic plot obtained from a classification problem with two normally distributed classes in $\mathbb{R}^3$. The circular, triangular, and square markers describe the average performance of hundreds of labeled cell classifiers with $\alpha$ equal to 0.9, 0.7, and 0.5 (a pure cell classifier) respectively. In all cases $k = k' = 11$. The five-pointed stars describe the performance of an $k$-d tree implementation of an 11-nearest-neighbor classifier. Vertical error bars all lie within each marker.

classifier.

## 4 EMPIRICAL ANALYSIS

Two problems illustrate the differences in performance and accuracy between labeled cell and exact $k$-d tree implementations of the $k$-nearest neighbor classifier. The first, assumes two equally probable, normally distributed classes in $\mathbb{R}^3$. Thus the class-conditional probability densities are

$$f_\ell(\mathbf{x}) = \frac{1}{(2\pi\sigma^2)^{3/2}} e^{-((x_1+(-1)^\ell\mu)^2+x_2^2+x_3^2)/2\sigma^2},$$

for $\ell \in \{1, 2\}$. The classification accuracy (i.e., the expected probability of error), and the expected number of operations per classification are empirically estimated from a sequence of independent trials. For each trial a random reference sample of $m = 10,000$ patterns is used to classify several thousand independent input vectors. The number of operations is estimated heuristically: each comparison and addition count as one operation, and each multiplication as two. (Qualitatively similar results are obtained with a variety of weighting factors.) Results for $k = 11$, a Euclidean metric, and eight values of $2\mu/\sigma$ are displayed in

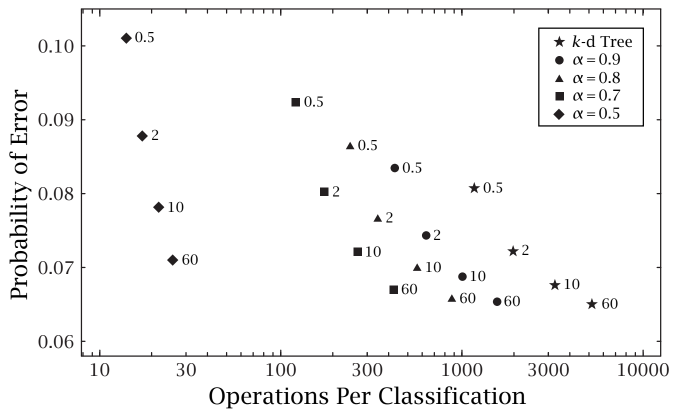

**Figure 4**: Results of the second experiment in which six-dimensional pixels, belonging to two different classes were classified by three different labeled cell classifiers ($k' = k = 7$), and a $k$-d tree implementation of a 7-nearest-neighbor classifier. The reference sample size appears to the right of each marker in thousands. The horizontal axis is logarithmic.

---

Fig. 3. In this example, the greatest absolute deviation in accuracy between two implementations occurs at $2\mu/\sigma = 6$ and $\alpha = 0.5$, where the labeled cell classifier misclassifies 0.15% of the independent test patterns, and the $k$-d tree implementation misclassifies 0.14%. Note in particular, how the recursive labeling scheme accelerates the performance as the class separation is increased, with little degradation in accuracy.

The second problem, uses data extracted from a seven-band digital image. We let each pixel define an independent pattern. The first band is quantized about the median to obtain a binary class label. A six-dimensional feature vector is formed with the remaining spectral bands. Reference and test patterns are selected independently from the image. Fig. 4 displays the trade-off between the classification accuracy and the computational cost for four different reference sample sizes as well as four different values of $\alpha$. These results suggest that the recursive labeling scheme accelerates classification with only a small reduction in accuracy. Note that by increasing the size of the reference sample, it is possible to obtain a labeled cell classifier that is both significantly faster and more accurate than a $k$-d tree classifier. Thus the new algorithm may be useful for real-time applications that provide an abundant supply of classified data. The estimates, redisplayed in Fig. 5, validate that the average classification time of labeled cell classifiers is also $O(\log m)$, but with smaller con-

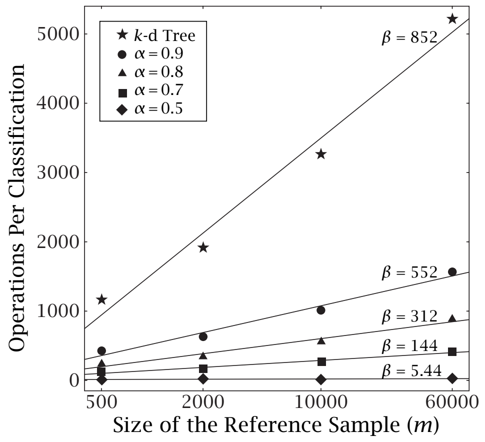

**Figure 5**:Empirical estimates of the average number of operations required for each classification as a function of the sample size $m$ for the second experiment. The linear graphs represent least-square fits of the form $\beta \log_{10} m + \gamma$. (Note that the horizontal axis is logarithmic.)

stants of proportionality $\beta$. Preliminary comparative experiments suggest that the labeled cell classifier is competitive with other approximations of the $k$ nearest neighbor algorithm. Moreover, recursive labeling can be combined with early truncation (Arya and Mount [1]) to yield even faster implementations.

## 5 CONCLUSION

The amount of effort that we employ in real-time decisions usually depends upon our perception of the inherent risk. Decisions of high risk, such as critical medical diagnoses, are fraught with more deliberation and information gathering than those of lesser consequence. The labeled cell classifier applies this principle using a local estimate of the conditional risk, resulting in an especially efficient approximation to the $k$-nearest-neighbor algorithm. This study also illustrates how a classic pattern classifier can be modified to achieve greater efficiency for problems of low Bayes risk. Since many perceptual problems are solved with minimal ambiguity, this approach might be useful for other pattern recognition algorithms.

**Acknowledgments**

This research was sponsored by Rome Laboratory, Air Force Material Command, USAF, under grant number F30602–94–1–0010.

## References

[1] S. Arya and D. M. Mount, “Approximate nearest neighbor queries in fixed dimensions,” in *Proceedings of the 4th ACM-SIAM Symposium on Discrete Algorithms*, 1993, pp. 271–280.

[2] J. L. Bentley, “Multidimensional binary search trees used for associative searching,” *Communications of the ACM*, vol. 18, 1975, pp. 509–517.

[3] T. M. Cover and P. E. Hart, “Nearest neighbor pattern classification,” *IEEE Trans. Inform. Theory*, vol. IT–13, 1967, pp. 21–27.

[4] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York, New York: John Wiley & Sons, 1973.

[5] E. Fix and J. L. Hodges, Jr., “Discriminatory Analysis — Nonparametric Discrimination: Consistency Properties,” *Project 21–49–004, Report No. 4*, USAF School of Aviation Medicine, Randolf Field, TX, 1951, pp. 261–279.

[6] J. H. Friedman, F. Baskett, and L. J. Shustek, “An Algorithm for Finding Nearest Neighbors,” *IEEE Trans. Comput.*, vol. C–24, 1975, pp. 1000–1006.

[7] J. H. Friedman, J. L. Bentley, and R. A. Finkel, “An algorithm for finding best matches in logarithmic expected time,” *ACM Transactions on Mathematical Software*, vol. 3, 1977, pp. 209–226.

[8] K. Fukunaga and P. M. Narendra, “A Branch and Bound Algorithm for Computing $k$-Nearest Neighbors,” *IEEE Trans. Comput.*, vol. C–24, 1975, pp. 750–753.

[9] P. E. Hart, “The condensed nearest neighbor rule,” *IEEE Trans. Inform. Theory*, vol. IT–1, 1968, pp. 515–516.

[10] B. S. Kim and S. B. Park, “A fast $k$ nearest neighbor finding algorithm based on the ordered partition,” *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. PAMI–8, 1986, pp. 761–766.

[11] T. Hastie and R. Tibshirani, “Discriminant Adaptive Nearest Neighbor Classification and Regression,” in D. S. Touretzky et al., *Advances in Neural Information Processing Systems 8*, MIT Press: Cambridge, MA, 1996, pp. 409–415.

[12] S. J. Smith, M. O. Bourgoin, K. Sims, H. L. Voorhees, “Handwritten character classification using nearest neighbor in large databases,” *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. PAMI–16, 1994, pp. 915–919.

[13] D. L. Wilson, “Asymptotic properties of nearest neighbor rules using edited data,” *IEEE Trans. Systems, Man, and Cybernetics*, vol. SMC–2, 1972, pp. 408–421.