NEAREST NEIGHBOR RULE

Jeff Robble, Brian Renzenbrink, Doug Roberts

Nearest Neighbor Rule

Consider a test point x.

x’ is the closest point to x out of the rest of the test points.

Nearest Neighbor Rule selects the class for x with the assumption that: $\omega(x) = \omega(x')$

Is this reasonable?

Yes, if x’ is sufficiently close to x.

If x’ and x were overlapping (at the same point), they would share the same class.

As the number of test points increases, the closer x’ will probably be to x.

Nearest Neighbor Estimation

Possible solution for the unknown "best" window problem

- The best window problem involves deciding how to partition the available data
- Let the cell volume be a function of the training data
- Instead of an arbitrary function of the overall number of samples

To estimate *$p(x)$* from *n* training samples

- Center a cell about a point *x*
- Let the cell grow until k_n samples are captured
- k_n is a specified function of *n*
- Samples are the k_n nearest neighbors of the point *x*

Nearest Neighbor Estimation

- Eq. 1 is the probability of choosing point x given n samples in cell volume V_n
- k_n goes to infinity as n goes to infinity
  - Assures eq. 2 is a good estimate of the probability that a point falls in V_n
- A good estimate of the probability that a point will fall in a cell of volume V_n is eq. 2
- k_n must grow slowly in order for the size of the cell needed to capture k_n samples to shrink to zero
- Thus eq. 2 must go to zero
- These conditions are necessary for $p_n(x)$ to converge to $p(x)$ at all points where $p(x)$ is continuous

$$p_n(x) = \frac{k_n / n}{V_n} \qquad (1)$$

$$\lim_{n\to\infty} k_n = \infty \qquad (3)$$

$$k_n / n \qquad (2)$$

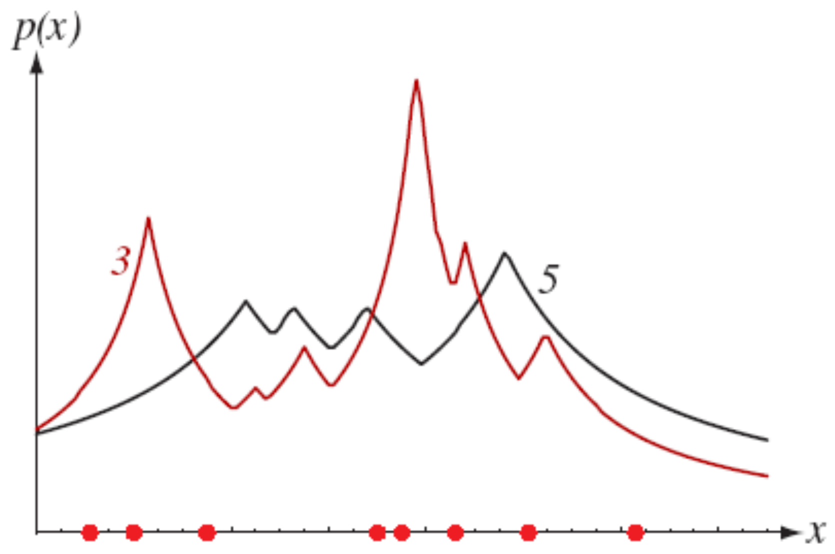
$$\lim_{n\to\infty} k_n / n = \infty \qquad (4)$$

Nearest Neighbor Estimation

The diagram is a 2D representation of Nearest Neighbor applied of a feature space of 1 dimension

The nearest neighbors for k = 3 and k = 5

The slope discontinuities lie away from the prototype points

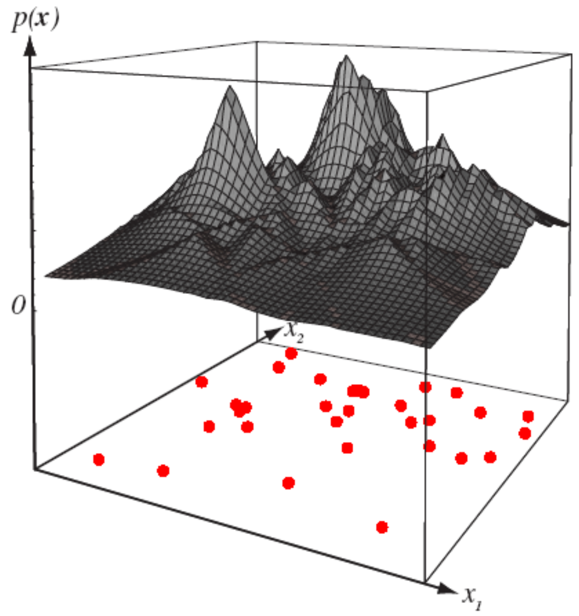


Nearest Neighbor Estimation

The diagram is a 3D representation of Nearest Neighbor applied of a feature space of 2 dimensions

The high peaks show the cluster centers

The red dots are the data points



Nearest Neighbor Estimation

Posterior probabilities can be estimated using a set of *n* labeled samples to estimate densities

Eq. 5 is used to estimate the posterior probabilities

Eq. 5 basically states that ω_i is the fraction of samples within a cell that are labeled ω_i

$$P_n(\omega_i | x) = \frac{p_n(x, \omega_i)}{\sum_{j=1}^{c} p_n(x, \omega_j)} = \frac{k_i}{k} \qquad (5)$$

Choosing the Size of a Cell

Parzen-window approach

- V_n is a specified function of n

k_n-nearest neighbor

- V_n is expanded until a specified number of samples are captured

Either way an infinite number of samples will fall within an infinitely small cell as n goes to infinity

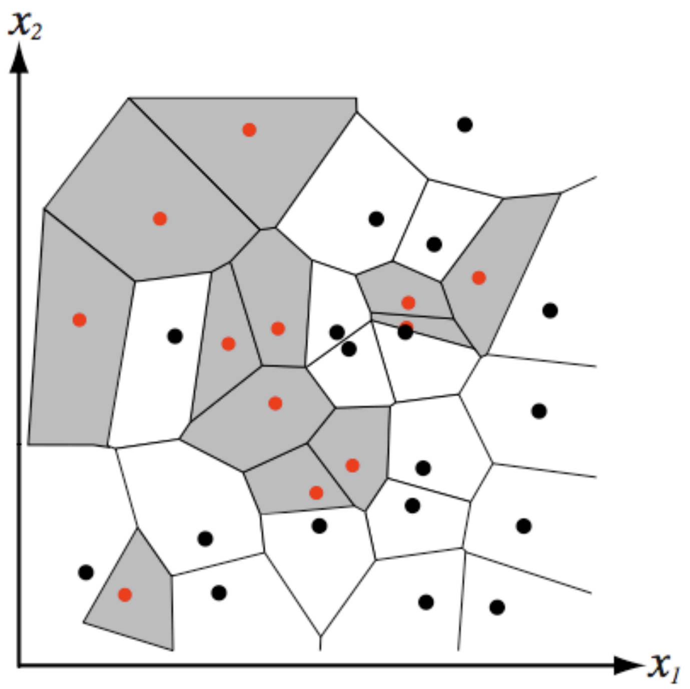
Voronoi Tesselation

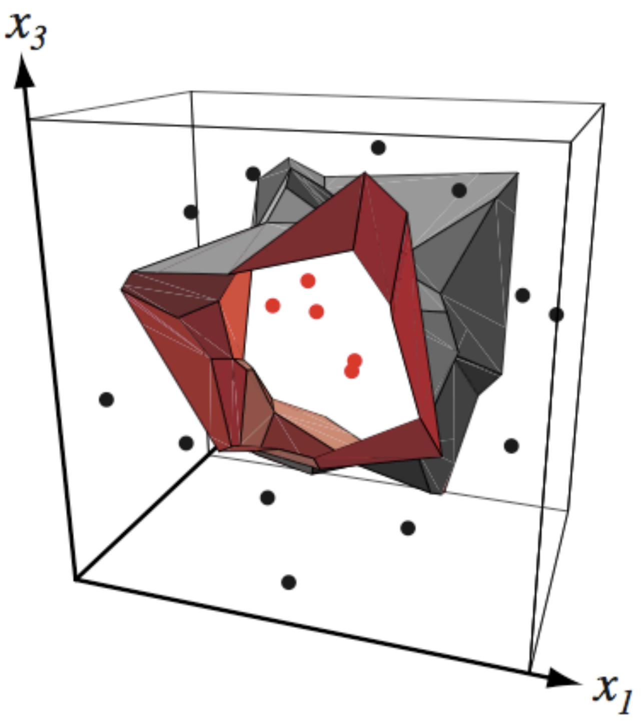
Partition the feature space into cells.

Boundary lines lie half-way between any two points.

Label each cell based on the class of enclosed point.

2 classes: red, **black**





Notation

ω_m is class with maximum probability given a point

$$P(\omega_m|x) = \max_i P(\omega_i|x)$$

Bayes Decision Rule always selects class which results in minimum risk (i.e. highest probability), which is ω_m

P* is the minimum probability of error, which is Bayes Rate.

Minimum error probability for a given x: $P^*(e|x) = 1 - P(\omega_m|x)$ (46)

Minimum average error probability for x: $P^* = \int P^*(e|x)p(x)dx$ (37)

Nearest Neighbor Error

We will show:

- The average probability of error is not concerned with the exact placement of the nearest neighbor.
- The exact conditional probability of error is:

$$P = \int \left[1 - \sum_{i=1}^{c} P^2(\omega_i | x) \right] p(x) dx$$

- The above error rate is never worse than 2x the Bayes Rate: $P^* \le P \le 2P^*$

Approximate probability of error when all classes, c, have equal probability: $1 - (1/c)$

Convergence: Average Probability of Error

Error depends on choosing the a nearest neighbor that shares that same class as x:

$$P(e|x) = \int P(e|x,x')\,p(x'|x)\,dx' \quad (40)$$

As n goes to infinity, we expect p(x'|x) to approach a delta function (i.e. get indefinitely large as x' nearly overlaps x).

Thus, the integral of p(x'|x) will evaluate to 0 everywhere but at x where it will evaluate to 1, so:

$$P(e|x) = P(e|x,x')$$

$$P(e|x) = P(e|x)$$

Thus, the average probability of error is not concerned with the nearest neighbor, x'.

Convergence: Average Probability of Error

Let's use intuition to explain the delta function.

At x, assume probability is continuous and not 0.

There is a hypersphere, S, (with as many dimensions as x has features) centered around point x:

Probability a point falls inside S: $P_s = \int_{x' \in S} p(x')dx'$

Probability that all n test samples drawn fall outside S:

$$(1 - P_s)^n$$

(1-P$_s$) will produce a fraction.

A fraction taken to a large power will decrease.

Thus, as n approaches infinity, the above eq. approaches zero.

Error Rate: Conditional Probability of Error

For each of n test samples, there is an error whenever the chosen class for that sample is not the actual class.

For the Nearest Neighbor Rule:

- Each test sample is a random (x,θ) pairing, where θ is the actual class of x.
- For each x we choose x'. x' has class θ'.
- There is an error if θ ≠ θ'.

sum over classes being the same for x and x'

$$P_n(e|x,x'_n) = 1 - \sum_{i=1}^{c} P(\omega_i|x)P(\omega_i|x'_n)$$

Plugging this into eq. 40 and taking the limit:

$$\lim_{n\to\infty} P_n(e|x) = \int \left[1 - \sum_{i=1}^{c} P(\omega_i|x)P(\omega_i|x'_n)\right] \delta(x_n - x)\, dx'_n$$

$$= 1 - \sum_{i=1}^{c} P^2(\omega_i|x) \qquad (44)$$

delta function: x ≈ x'

Error Rate: Conditional Probability of Error

Error as number of samples go to infinity: $P = \lim_{n \to \infty} \boxed{P_n(e)}$

Plugging in eq. 37:

$$P = \lim_{n \to \infty} \boxed{\int \boxed{P_n(e|x)}\, p(x)dx}$$

Plugging in eq. 44:

$$P = \int \boxed{\left[1 - \sum_{i=1}^{c} P^2(\omega_i|x)\right]} p(x)dx \quad \Rightarrow \tag{45}$$

What does eq. 45 mean?

Notice the squared term.

The lower the probability of correctly identifying a class given point x, the greater impact it has on increasing the overall error rate for identifying that point's class.

It's an exact result. How does it compare to Bayes Rate, P*?

Error Bounds

Exact Conditional Probability of Error:

$\Rightarrow$ $$P = \int \left[1 - \boxed{\sum_{i=1}^{c} P^2(\omega_i|x)} \right] p(x)dx$$

How low can this get?
How high can the error rate get?

Expand: $$\boxed{\sum_{i=1}^{c} P^2(\omega_i|x)} = P^2(\omega_m|x) + \boxed{\sum_{i \neq m}^{c} P^2(\omega_i|x)}$$ $\Rightarrow$

Constraint 1: $P(\omega_i|x) \geq 0$

Constraint 2: $$\boxed{\sum_{i \neq m} P(\omega_i|x)} = \overbrace{1 - P(\omega_m|x) = P^*(e|x)}^{\text{eq. 46}}$$

The summed term is minimized when all the posterior probabilities but the mth are equal:

$$\boxed{P(\omega_i|x) = \begin{cases} \dfrac{P^*(e|x)}{c-1}, & i \neq m \\ 1 - P^*(e|x), & i = m \end{cases}}$$

Non-m Posterior Probabilities have equal likelihood. Thus, divide by c-1.

Error Bounds

Finding the Error Bounds:

$$\sum_{i=1}^{c} P^2(\omega_i|x) = P^2(\omega_m|x) + \sum_{i\neq m}^{c} P^2(\omega_i|x)$$

$$\sum_{i=1}^{c} P^2(\omega_i|x) \geq \boxed{(1-P^*(e|x))^2 + \frac{P^{*2}(e|x)}{c-1}}$$

Plug in minimizing conditions and make inequality

$$\sum_{i=1}^{c} P^2(\omega_i|x) \geq \left[1-2P^*(e|x)+P^{*2}(e|x)\right] + \frac{P^{*2}(e|x)}{c-1}$$

Factor

$$\sum_{i=1}^{c} P^2(\omega_i|x) \geq \left[1-2P^*(e|x)\right] + P^{*2}(e|x)\left(\frac{c-1}{c-1}\right) + \frac{P^{*2}(e|x)}{c-1}$$

Combine terms

$$\sum_{i=1}^{c} P^2(\omega_i|x) \geq \left[1-2P^*(e|x)\right] + \frac{c}{c-1}P^{*2}(e|x)$$

Simplify

$$2P^*(e|x) - \frac{c}{c-1}P^{*2}(e|x) \geq 1\sum_{i=1}^{c} P^2(\omega_i|x)$$

Rearrange expression

$$1-\sum_{i=1}^{c} P^2(\omega_i|x) \leq 2P^*(e|x) - \frac{c}{c-1}P^{*2}(e|x)$$

Error Bounds

Finding the Error Bounds:

$$1-\sum_{i=1}^{c} P^2(\omega_i|x) \le 2P^*(e|x) - \cancel{\frac{c}{c-1}P^{*2}(e|x)}$$

$$P = \int\left[1-\sum_{i=1}^{c} P^2(\omega_i|x)\right]p(x)dx \quad (45)$$

$$P^* = \int P^*(e|x)p(x)dx \quad (37)$$

Integrate both sides with respect to choosing x and plug in the highlighted terms.

$$P \le 2P^*$$

Thus, the error rate is less than twice the Bayes Rate.

Variance: $\sigma^2 = \int (x-\mu)^2 p(x)dx$

$$Var[P^*(e|x)] = \int\left[P^*(e|x) - P^*\right]^2 p(x)dx$$

$$= \int P^{*2}(e|x)p(x)dx - P^{*2} \ge 0$$

$$= \int P^{*2}(e|x)p(x)dx \ge P^{*2}$$

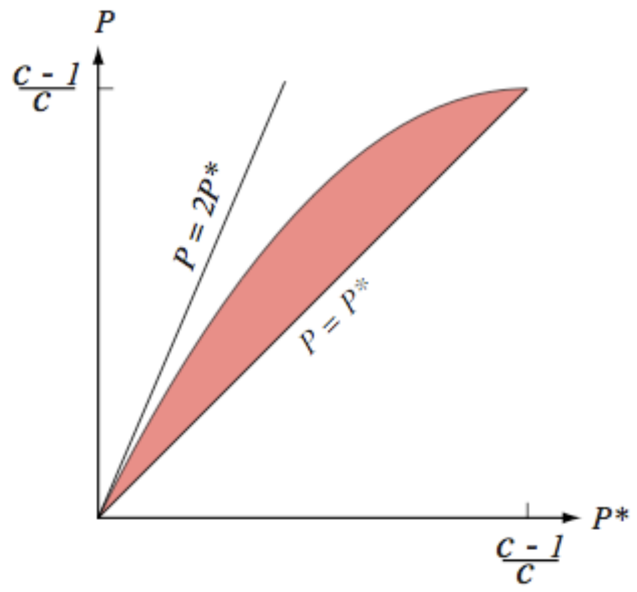
Tightest upper bounds:

$$P^* \le P \le P^*\left(2 - \frac{c}{c-1}P^*\right)$$

Found by keeping the right term.

Error Bounds

Bounds on the Nearest Neighbor error rate.



$$\Rightarrow \quad P^* \le P \le P^* \left(2 - \frac{c}{c-1} P^* \right)$$

Take P* = 0 and P* = 1 to get bounds for P*

$0 \le P^* \le (c-1)/c$

With infinite data, and a complex decision rule, you can at most cut the error rate in half.

When Bayes Rate, P*, is small, the upper bound is approx. 2x Bayes Rate.

Difficult to show Nearest Neighbor performance convergence to asymptotic value.

k-Nearest Neighbor Rule

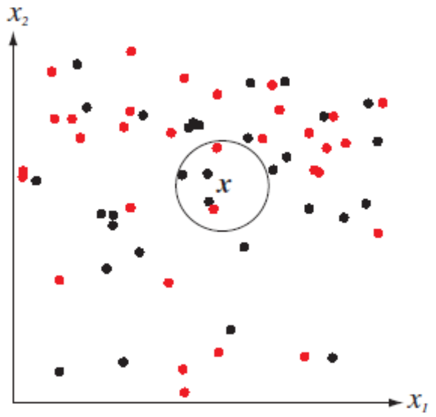
Consider a test point **x**.

v_x is the vector of the ***k*** nearest points to **x**

The *k*-Nearest Neighbor Rule assigns the most frequent class of the points within v_x.

We will study the two-class case.

Therefore, ***k*** must be an odd number (to prevent ties).



k-Nearest Neighbor Rule

The *k*-nearest neighbor rule attempts to match probabilities with nature.

As with the single neighbor case, the labels of each of the *k*-nearest neighbor are random variables.

Bayes decision rule always selects W_m . Recall that the single nearest neighbor case assumes W_m with the probability $P(W_m \mid x)$.

The *k*-nearest neighbor rule selects W_m with the probability of :

$$\sum_{i=(k+1)/2}^{k} \binom{k}{i} P(W_m \mid x)^{i} [1 - P(W_m \mid x)^{k-i}]$$

Error Bounds

We can prove that if *k* is odd, the two-class error rate for the *k*-nearest neighbor rule has an upper bound of the function $C_k(P^*)$ where $C_k(P^*)$ is the smallest concave function of P^* greater than:

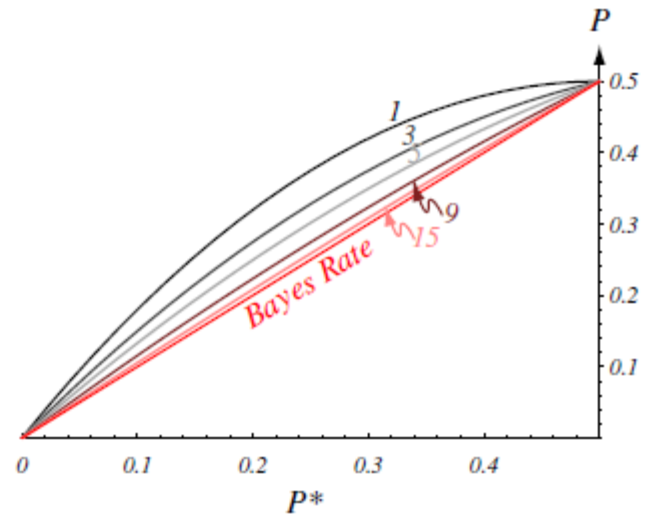
$$\sum_{i=0}^{(k-1)/2} \binom{k}{i} \left[(P^*)^{i+1}(1-P^*)^{k-i} + (P^*)^{k-i}(1-P^*)^{i+1} \right]$$

Note that the first bracketed term [blue] represents the probability of error due to *i* points coming from the category having the minimum real probability and *k-i>i* points from the other category

The second bracketed term [green] represents the probability that *k-i* points are from the minimum-real probability category and *i+1<k-i* from the higher probability category.

Error Bounds

Bounds on the *k*-Nearest Neighbors error rate.



Note that as *k* increases, the upper bounds of the error rate get progressively closer to the lower bound. At infinity, the *k*-nearest neighbors error rate = the Bayes rate

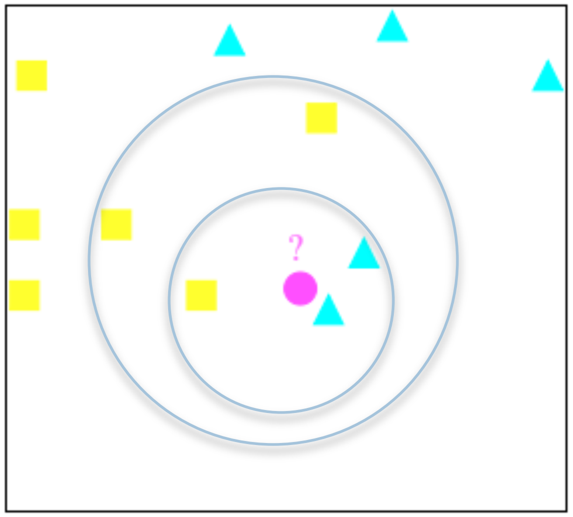
The tradeoff for increasing the value of *k* is that larger values of *k* increase the computational complexity of the problem.

Example

Here is a basic example of the *k*-nearest neighbor algorithm for:

~~*k*=3~~

k=5



Computational Complexity

The computational complexity of the *k*-nearest neighbor rule has received a great deal of attention. We will focus on cases involving an arbitrary *d* dimensions. The complexity of the base case where we examine every single node's distance is O(dn).

There are three general algorithmic techniques for reducing the computational cost of our search:

- Partial Distance calculation
- Prestructuring
- Editing.

Partial Distance

In the partial distance algorithm, we calculate the distance using some subset ***r*** of the full ***d*** dimensions. If this partial distance is too great, we stop computing.

The partial distance based on ***r*** selected dimensions is:

$$D_r(\mathbf{a},\mathbf{b}) = \left(\sum_{k=1}^{r} (a_k - b_k) \right)^{1/2}$$

Where ***r*** < ***d***.

The partial distance method assumes that the dimensional subspace we define is indicative of the full data space.

The partial distance method is strictly non-decreasing as we add dimensions.

Prestructuring

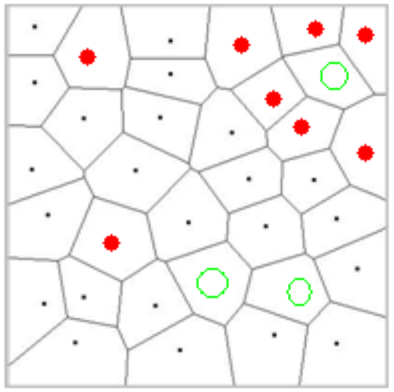
In prestructuring we create a search tree in which all points are linked. During classification, we compute the distance of the test point to one or a few stored root points and consider only the tree associated with that root.

This method requires proper structuring to successfully reduce cost.

Note that the method is NOT guaranteed to find the closest prototype.

Editing

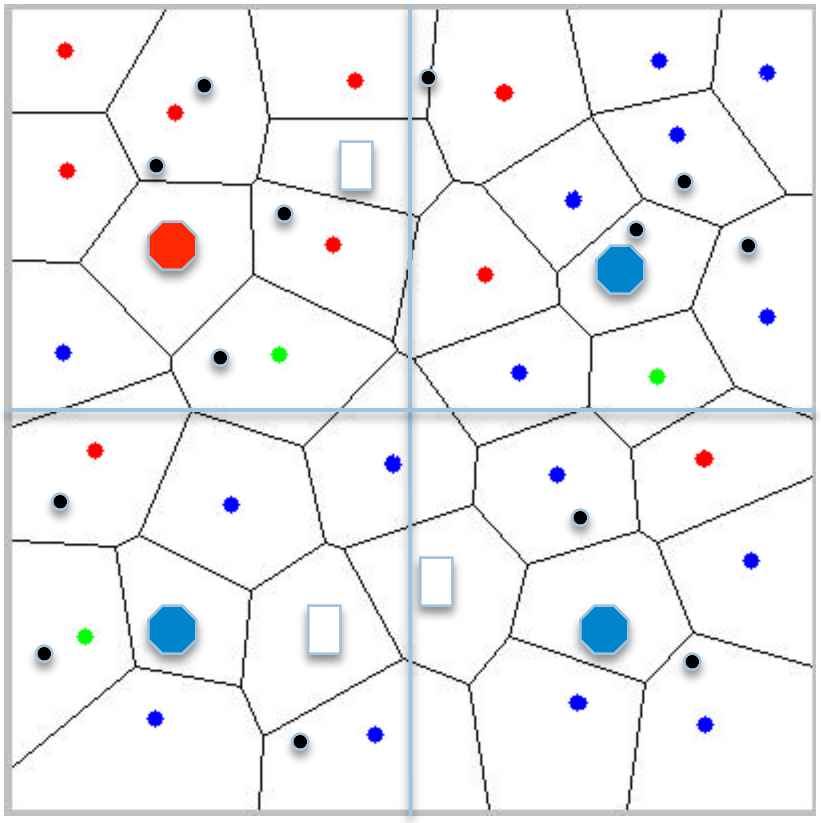
The third method we will look at is Editing. Here, we prune ‘useless’ points during training. A simple method of pruning is to remove any point that has identical classes for all of its *k* nearest neighbors. This leaves the decision boundaries and error unchanged while also allowing for Voronoi Tessellation to still work.



Complexity can be reduced without reducing accuracy, making editing very useful.
This algorithm does not guarantee a minimal set of points is found.
This algorithm prevents the addition of training data later on, as that would invalidate the earlier data pruning.

Example

It is possible to use multiple algorithms together to reduce complexity even further.



k-Nearest Neighbor Using MaxNearestDist

Samet's work expands depth-first search and best-first search algorithms.

His algorithms using the Max Nearest Distance as an upper bound can be shown to be no worse than the basic current algorithms while potentially being much faster.

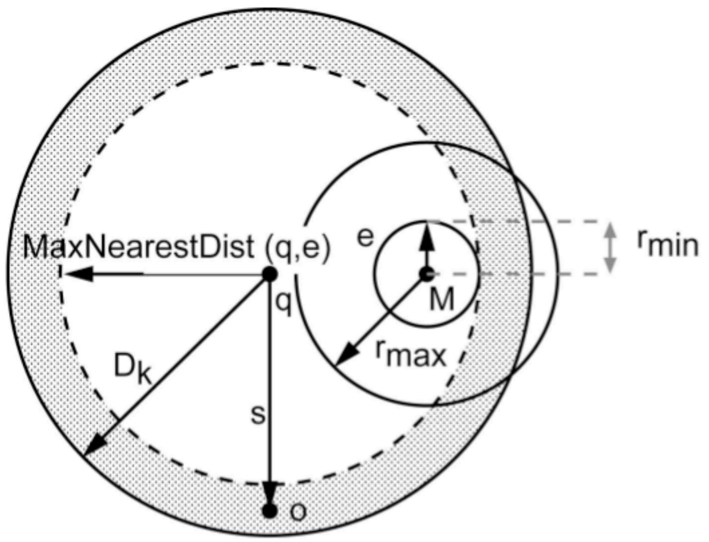


Fig. 2. Example showing that we cannot simply reset D_k to MaxNearestDist (q, e) whenever MaxNearestDist(q,e) < D_k for nonobject element e which is a spherical shell so that MaxNearestDist(q, e) = d(q, M) + r_{min}, assuming a euclidean distance metric.

k-Nearest Neighbor Using MaxNearestDist

Q is our query point

Ma, Mb, and Mc are non-object children of Mp

Rmin is the distance from the cluster center to the closest object in the cluster

Rmax is the distance from the cluster center to the furthest object in the cluster

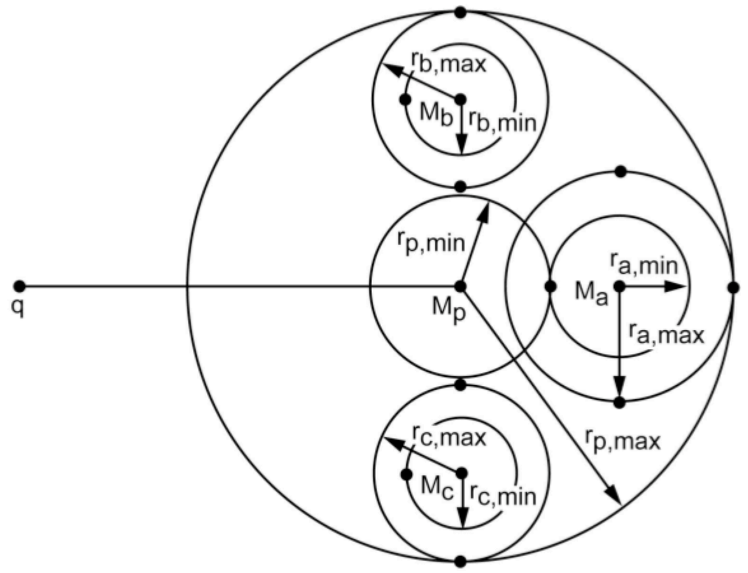


Fig. 3. Example illustrating how the number of elements in L can decrease as a result of the situation that arises when the MaxNearestDist values of the three nonobject child elements e_a, e_b, and e_c of e_p are greater than the current value of D_k which is the MaxNearestDist value of e_p, assuming a euclidean distance metric.

k-Nearest Neighbor Using MaxNearestDist

```
1  procedure MAXNEARESTDISTINSERTL($e, s$)
2  /* Insert element (object or nonobject) $e$ at distance $s$
      from query object $q$ into the priority queue $L$ using
      ENQUEUE. Assume that objects have precedence over
      nonobjects when they are at the same distance $D_k$
      (that is, the $k$th-nearest neighbor) from the query
      object $q$.*/
3  if SIZE($L$) $= k$ then
4     $h \leftarrow$ DEQUEUE($L$)
5     if not ISOBJECT($E(h)$) then
6        while not ISEMPTY($L$)
7              and not ISOBJECT($E$(MAXL($L$)))
8              and $D$(MAXL($L$)) $= D(h)$ do
9           DEQUEUE($L$)
10       enddo
11    endif
12 endif
13 ENQUEUE($e, s, L$)
14 if SIZE($L$) $= k$ then
15    if $D$(MAXL($L$)) $< D_k$ then
16       $D_k \leftarrow D$(MAXL($L$))
17    endif
18 endif
```

References

Duda, R., Hart, P., Stork, D., 2001. *Pattern Classification, 2nd ed. John Wiley & Sons.*

Samet, H., 2008. K-Nearest Neighbor Finding Using MaxNearestDist. *IEEE Trans. Pattern Anal. Mach. Intell. 30, 2 (Feb. 2008), 243-252.*

Yu-Long Qiao, Jeng-Shyang Pan, Sheng-He Sun .Improved partial distance search for k nearest-neighbor classification. IEEE International Conference on Multimedia and Expo, 2004. June 2004, 1275 - 1278