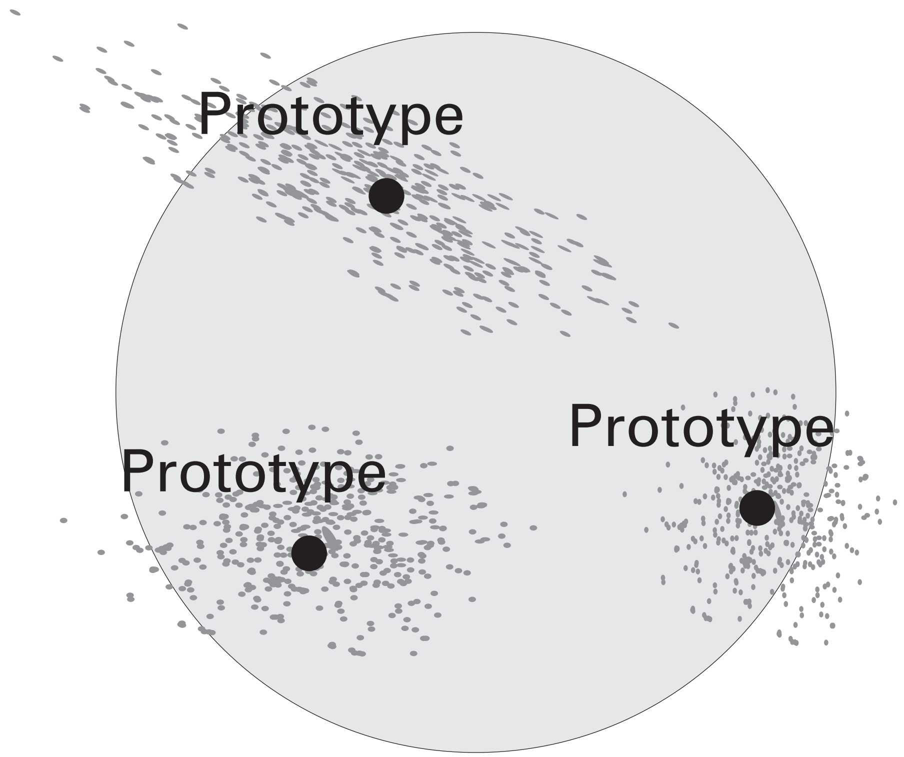
A Distributed Clustering Algorithm



Lund University Cognitive Studies, 74
1998

S–222 22 Lund
Sweden
Department of Mathematics
Uppsala University

nils.hulth@fil.lu.se
grenholm@math.uu.se

Abstract: A new algorithm for clustering is presented — the *Distributed Clustering Algorithm* (DCA). It is designed to be incremental and to work in a real-time situation, thus making it suitable for robotics and in models of concept formation. The DCA starts with one cluster (or rather *prototype* at the center of the cluster), successively adding prototypes and distributing them according to data density until a certain criteria is fulfilled. This criteria is that new prototypes do not add enough extra precision in the representation of the data. A local measure called *roundness* is used to predict how much extra precision a new prototype will add.

1 Introduction

In this paper a new algorithm for *clustering* will be described. It is called the *Distributed Clustering Algorithm* (DCA). Clustering is the process of grouping similar data together. Clustering techniques are being used in many different areas, such as robotics, vision, data mining, statistics, cognitive science and machine learning. The focus of this paper will be on cognition and robotics. The prime use for clustering in these areas is for unsupervised concept formation (for a cognitive approach to concept formation, see Gärdenfors, 1997 and his conceptual spaces). Take as an example a robot being fed with data from 38 sensors getting such information as weight, colour, brightness, smell and conductivity about an object. Suppose the task of the robot is to learn to separate and identify different objects that it encounters. For each object the robot gets a 38-dimensional vector of sensor data, which can be seen as a point in a 38-dimensional space. If the robot looks at several different objects several times, *clusters*, or “clouds”, of points will be created, i.e. each object corresponds to one “cloud”. If the objects are sufficiently different the clusters will be separated, and it is possible to identify objects based on their cluster belongings.

To form these clusters a clustering algorithm is used. There exist a large number of such algorithms with different characteristics. To be used in a real-time and incremental context as the example above certain features are necessary. The algorithm must be able to handle new data without the need to reprocess all the previous data. It must also be able to automatically determine the number of clusters needed. Preferably it should not make any assumptions about the distribution of the data either. Finally it should be possible to implement the algorithm in an efficient way.

The DCA is an attempt to create such an algorithm. This attempt is made since there are very few clustering algorithms that can handle all of the above requirements (see the comparison of the DCA and other common clustering algorithms in section 4).

The approach used by the DCA is similar in some respects to *Vector Quantization* (VQ) Linde et al., 1980; Hassoun, 1995, pp109-110. Like the VQ the DCA uses a number of prototype vectors in the same space as the training data. An example is said to belong to the prototype that is closest to it. This means that the method can generalise, i.e. categorise examples that have not been encountered before.

How then is the number of prototypes decided? If the average distance between examples and their corresponding prototypes is small then the information loss ing is also small. A small information loss means that the prototype is a good representative of its examples. Quite obviously each extra prototype that is added will decrease the average distance. But is also true that sometimes an extra prototype decreases the distance substantially more than average. To understand this, consider the examples shown in fig 1a and fig 1b.

The first example demonstrates the advantage of an extra prototype, i.e. two prototypes clearly reduce the average distance between a data point and its prototype. In the second example this is not the case. The division decreases the distance, but not nearly as much as in the first example. The prototype in the first example is said to have a higher *split gain* (see appendix A).

This is the feature that is used by the DCA to decide on the number of prototypes. The DCA starts out with only one prototype, successively adding prototypes as long as they decrease the average distance between examples and prototypes more than would be expected for a homogenous distribution. Note that this is decided *locally*, i.e. only using the statistics gathered for the prototype about to split.

As stated above the DCA should be able to operate in a real-time and incremental context. This makes it desirable to be able to decide beforehand if an addition of a prototype will decrease the average distance discussed in the last paragraph. To do this a measure of *roundness* is used. Let all examples that belong to a prototype be called the cluster of that prototype. Roundness can be said to measure the shape of the cluster. The closer the shape is to a ball the higher its roundness value is. A shape that is stretched out in some directions, or is hollow, will get a low roundness value. Low roundness values indicates a potentially large decrease in average distance if a new prototype is added. When all prototypes have clusters that are round enough no more clusters are added.

If there are several differently shaped groups of data — some spherical, some not — this method will tend to allocate more prototypes to those that are not spherical. Implicitly, this means that there will not necessarily be more prototypes in areas with higher example density, or at least the prototype density will not have the same indication of when to split a prototype are given in appendix A.

2 The Algorithm

A cluster (i.e. a collection of examples) is described by a prototype vector $\mathbf{p}$ at the centre of the cluster. The set of all N prototypes $\{\mathbf{p}_1,\ldots,\mathbf{p}_N\}$is $\mathbf{P}$. An example vector $\mathbf{e}$ is considered an instance of a cluster i, described by the prototype $\mathbf{p}_i$, if the Euclidian distance[1] between $\mathbf{e}$ and all prototypes is smallest for $\mathbf{p}_i$, i.e. $|\mathbf{p}_i - \mathbf{e}| < |\mathbf{p}_j - \mathbf{e}|$ for all $j \neq i$. This will be written as $\mathbf{e} \in \mathbf{p}_i$. If two or more prototypes are equally close one of the closest is randomly selected.

Without loss of generality it is assumed that the elements of the example vectors are scaled to lie between 0 and 1.

First a rough outline of the algorithm will be given, followed by a more complete description.

When an example is presented the prototype that is closest is selected. The selected prototype is then moved slightly in the direction of the example. This will, after enough example presentations, lead to a situation where the prototypes are in the approximate centre of their respective examples. To control the number of prototypes two mechanisms are utilised — prototype split and prototype removal. When a prototype fulfils a certain criterion (e.g. its roundness is too low, or it is selected more often than the average prototype) it is divided, i.e. a new prototype is inserted at the same position. This criterion is represented by the Boolean function $split(i)$, which, if true, causes a prototype division. Because of the mechanism used to select and move prototypes these two “siblings” will start to move away from each other, defining two new (sub)clusters. Similarly, when another criterion is fulfilled (e.g. less than average selection of the prototype) a prototype can be removed. As with splitting, there is a Boolean function $remove(i)$ to decide if a prototype should be removed.

[1]It is assumed that the examples and the prototypes are vectors in an Euclidian space.

$\mathbf{p}_1$ is set equal to the first example $\mathbf{e}_0$, which under some circumstances speed up the process.

1. Take the next example $\mathbf{e}_t$ and find the prototype $\mathbf{p}_i$ that is closest.

2. Change $\mathbf{p}_i$ according to:

$$\mathbf{p}_i(t+1) = (1-k)\mathbf{p}_i(t) + k\mathbf{e}_t \tag{1}$$

   where k is positive and close to zero (not bigger than one). The result is to move $\mathbf{p}_i$ closer to $\mathbf{e}_t$. A typical value is $k = 0.01$. The same value of k is used for all prototypes. For a discussion about alternative ways of defining k, see section 5.

3. For all n prototypes adjust their respective selection rates $s_1, \ldots, s_n$. This is done using:

$$\mathbf{s}_j(t+1) = (1-k)\mathbf{s}_j(t) + k\delta(i,j) \tag{2}$$

   where $\delta(i,j) = 1$ iff $j = i$, else $\delta(i,j) = 0$.

4. If the Boolean function $split(i)$ evaluates to true insert a copy of prototype $\mathbf{p}_i$ at the same position into the set of prototypes. The selection rate of $\mathbf{p}_i$ and the new prototype is set to half of $\mathbf{p}_i$'s selection rate before the division.

5. If the Boolean function $remove(i)$ evaluates to true prototype $\mathbf{p}_i$ is removed from the prototype set.

6. Repeat from step 1.

2.2 Defining *split*

There are several strategies for defining the $split(i)$ and $remove(i)$ functions. Each set of functions will of course determine the behaviour of the algorithm. The interesting thing is that the utilisation of the $split(i)$ and $remove(i)$ functions make the algorithm adaptive to changes in the data set.

[2]Sliding average is an approximation of the mean, using only the current value and the sliding average at the last iteration. For example, the sliding average $\tilde{x}$ of x is defined as $\tilde{x}(t+1) = (1-c)\tilde{x}(t) + cx(t+1)$, where c is a small positive constant.

termines what to count as stable.

$roundness(i)$ is a measure of how ball-like the body of examples that belong to cluster i is. $R(c_r, d)$ is a dimension dependant function that sets the limit for when a cluster is considered round or not. The parameter c_r is used for tuning the behaviour of equation 3. $roundness(i)$ is defined as:

$$roundness(i) = \frac{\sqrt[2-d]{E|\mathbf{e}_t - \mathbf{e}_{t-1}|^{2-d}}}{\sqrt{E|\mathbf{e}_t - \mathbf{e}_{t-1}|^2}} \tag{4}$$

where $E(\cdot)$ is the sliding average. For $d \neq 2$ the theoretical maximum (i.e. a hyperdimensional ball) is given by:

$$roundness_{\max}(d) = \left(\frac{2d}{d+2}\right)^{-\frac{1}{2}-\frac{1}{d-2}} \tag{5}$$

For $d = 2$ an alternate set of formulas are used:

$$roundness(i) = \frac{e^{E(log|\mathbf{e}_t - \mathbf{e}_{t-1}|)}}{\sqrt{E|\mathbf{e}_t - \mathbf{e}_{t-1}|^2}} \tag{6}$$

and

$$roundness_{\max}(2) = e^{-1/4} \tag{7}$$

The reason for using the *stability* function is easily understood. The statistics for *roundness* and *hollowness* rest on the assumption that the prototype is at a relatively fixed position. If this is not the case, e.g. if the prototype was just split and on it is way to a new position, it is very likely that the statistics collected will be misguiding. Therefore it is necessary to make sure the prototype is sufficiently stable before it is split.

2.3 Defining *remove*

A useful definition of $remove(i)$ is:

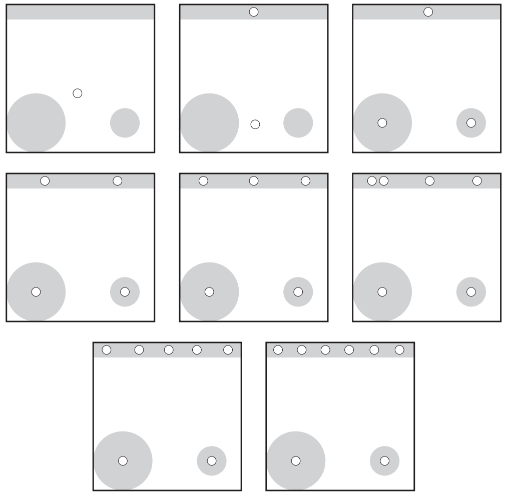
$$remove(i) = true \quad \text{iff} \quad \mathbf{s}_i(t) < \frac{1}{\gamma N} \tag{8}$$

where N is the current number of prototypes and γ is a constant indicating how dependent a prototype is on

In this section a series of examples is presented to demonstrate the behaviour of the DCA. The first three examples are two-dimensional to make visualisation easier, whereas the fourth is of higher dimensionality.

3.1 A Clustering Sequence

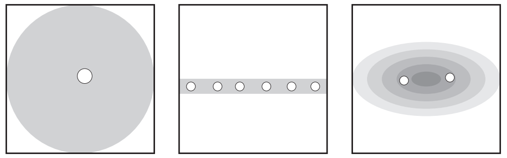
In the figure below a typical sequence is shown. The two axes represent the two dimensions. The gray areas represent clusters from where examples are drawn with uniform probability. As can be seen there are three separate clusters, two circular and one rectangular. The small white circles are prototypes. Each snapshot is taken just before one of the prototypes splits in two (except the last one which is taken after the number of prototypes is stable).



As can be seen in the first box the first prototype moves to stay at the centre of all three clusters. The stability criterion then becomes high enough so that the *split* function can proceed to check for *roundness*. With the distributions given in this example the roundness criterion is low enough to cause a split because of the hollowness around the first prototype (after it has stabilized). The two prototypes then divide the distributions between them — one takes the top cluster, and the other

3.2 Stable States

In this section four different examples are given. Each example shows the result of the clustering process after it has reached a stable state, i.e. no more divisions occur.



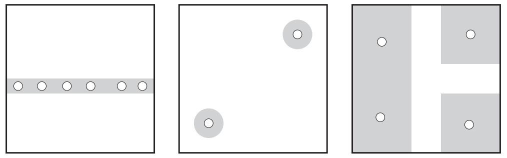
The first case simply demonstrates how a perfectly round cluster results in only one prototype.

The second case shows again how the roundness criterion works to eventually stop the divisions.

In the third case the examples are drawn from a Gaussian distribution, twice as wide as tall, showing that the DCA can handle this common distribution too.

3.3 Adaptivity to Changes in the Distributions

To show the incremental capabilities of the DCA a sequence of stable states is used. The distribution in the first box is used until the DCA is stable and no more divisions occur. After that the distribution is changed to that of the second box. The prototypes therefore need to rearrange themselves, and four of them are also removed because they do not get any examples. But after that has happened the state is again stable, and the distribution is now changed to the one in the third and final box. Both the prototypes in box two reposition themselves and then get low enough values of roundness to cause two splits, which gives the result shown in box three.



15.

Before being used in this example the data was rescaled to a range from 0 to 1. The parameter value used where: $S = 0.4$, $R(c_r, 16) = 0.65$.

Each prototype keeps a count of which letters it has seen. When asked about which letter it represents it answers with the letter it has seen most times. When a cluster divides the count is reset. Note that the information about the letters is not used in any way to influence the clustering behaviour.

As can be seen in the figure 2 the DCA reaches a level of correct classification of about 70%. Also note that the number of clusters reach approximately 300 and then starts oscillating. This is because some prototypes do not match enough examples and therefore have a high risk of being removed. David J. Slate reports in Slate and Frey, 1991 getting about 80% accuracy using a Holland-style adaptive classifier system. This accuracy is higher, but their system was designed explicitly for classification. The DCA has, in the form presented here, no way of using the class information of the examples.

4 Comparison with Other Algorithms

Most other algorithms differ in one or several aspects from the DCA (such as those mentioned above) which make them hard to compare with the DCA. However, below a number of algorithms are listed and contrasted with the DCA.

SOFM Kohonen's *Self-Organizing Feature Maps* Kohonen, 1989 are topologically arranged nodes on a grid (of some dimensionality). A prototype in the DCA is similar to a node in a SOFM, apart from that changes in a node influences its neighbours too. This causes a topological mapping from the data space to the grid, i.e. two neighbouring nodes will respond to similar examples. SOFM requires a fixed number of nodes and is not incremental because the size of the neighbourhood decreases with time.

and ART-2 also differ in the number of examples that need to be shown. ART-2 only needs one example to form a new category, whereas the DCA needs hundreds. This reflects the abstraction levels of the two algorithms. ART-2 is modelling a high level categorisation process, whereas the DCA is thought to work on a lower level of abstraction.

***k*-means clustering** Everitt, 1974 involves k clusters. It starts with the centres of the clusters being k randomly selected data points (examples); thereafter the remaining data points are successively assigned to the nearest (in an Euclidian sense) cluster. After each assignment the centre of the cluster is recomputed to be the centroid of all data points assigned to it. The DCA is similar to the k-means algorithm, with the additional feature that the number of clusters can change to better represent the data. Also, the DCA does not need to store all the data points assigned to each cluster, which makes it computationally more efficient.

Vector Quantization (VQ) The VQ algorithm utilises a fixed number of prototypes of the same dimensionality as the data. As with the DCA the prototypes are moved to the centre of the clusters they correspond to. VQ is also normally incremental. To overcome the problem to decide on the number of prototypes the VQ algorithm sometimes is modified to initially start with an excess of prototypes. After presenting the data those prototypes that were not used are removed. However, this modification only partially solves the problem because there is no way to get new nodes if data from new clusters are presented.

Hierarchical clustering Everitt, 1974 is a collective name of algorithms that build similarity trees of the data. They work by joining together groups that are similar, until there is only one group left (initially the groups consist of single data points). This method creates a tree where the data points are the leaves of the tree, and the last group is the root. Hierarchical algorithms differ from the DCA in that they require the whole data set (i.e. they

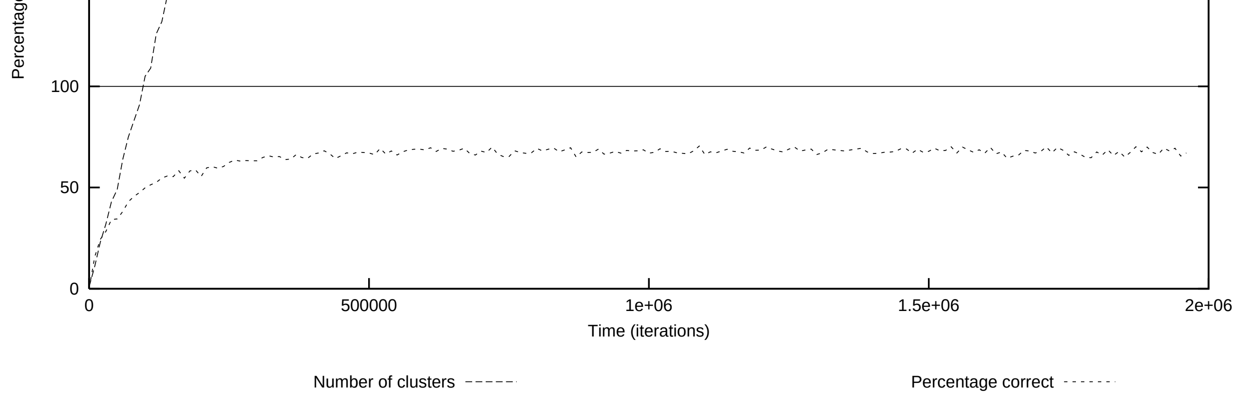


Figure 2: The result of using the DCA on high dimensional data. See text for explanation.

are not incremental) and also need to calculate the distance between all points (there are shortcuts though). Hierarchical methods lend themselves to data that in itself is naturally represented by a tree (such as taxonomy).

5 Discussion

The reason that the DCA is called *distributed* is that the actions of each prototype (i.e. movement, division and removal) are decided *locally*. This is very useful if the algorithm is to be implemented on a parallel architecture.

However, locallity has some drawbacks. It can never be guaranteed that the solution obtained is the optimal solution. The prototypes that DCA calculates might as well be a local optimum.

Another drawback/feature of the DCA is that it is history dependent. The final result depends a lot on the previous distributions. This is because two different mechanisms are used for adding and deleting prototypes, i.e. the split mechanism isn't the inverse of the remove mechanism.

It is important to notice that without proper preprocessing an iterative clustering method may have a very hard job on badly scaled data. Take as an example if one of two sensors is amplified so that its amplitude is tripled. A previously spherical cluster will now be stretched out in one of the dimensions. It might be the case that points within the cluster are now actually further away from each other than they are from points in a neighbouring cluster. This is a problem of having several dimensions and not knowing how they should be scaled. The DCA currently does not deal with this problem.

multidimensional scaling, which the DCA does not deal with in its current version.

There are several extensions/modifications that can be made to the DCA. Some of them have been tried, others have not. What is important to remember though, is that the changes should be made so the properties of the DCA are kept — incremental operation, local decisions, no assumptions about distributions and that the number of prototypes depend on the data. All these properties are very useful for robotic and cognitive applications. Below a number of changes are suggested:

Density matching requires that the density of the prototypes is proportional to the probability density of the examples, i.e. if two clusters are of the same shape and size, but examples are twice as likely to be coming from one of them, then there should also be twice as many prototypes representing that cluster. Using the DCA algorithm, the easiest way to accomplish this is to allow split because of high object clustering (and possibly classification) the value could correspond to innate values (c.f. Friston et al., 1994).

Positioning of new prototypes directly at the example that caused the split. When a new example is shown and a split occurs the new prototype is given the same position as the example (c.f. ART networks and new categories). This could possibly speed up the process slightly.

Adaptive k values is a common way to guarantee the convergence of the prototypes in case of stable example distributions. With this extension each prototype would have its individual k_i, defined as $k_i = k_0 n_i^{\varepsilon - 0.5}$ where k_0 is a constant giving the initial value of k_i, n_i is the number of examples seen by that prototype and ε is a number between 0 and 0.5 determining how fast the prototype converges. Note that $\varepsilon = 0.5$ gives $k_i = k_0$, i.e. no convergence.

Local Principle Component Analysis (PCA) performed at each prototype. This is because not all clusters need to have the same actual dimensionality. PCA could be used at each prototype (each prototype having their own covariance matrix) to determine the most important axes.

6 Acknowledgements

Many thanks to Christian Balkenius for criticism and constructive ideas. This work was supported NUTEK, the Swedish National Board for Industrial and Technical Development, within the ”Complex Systems” research program, project P10522-1; Software Principles for Complex Systems Interconnections and also partly supported by the Swedish Foundation for Strategic Research.

References

Carpenter, G. A. and Grossberg, S. (1991). Art2: Self-organization of stable category recognition. In

Differential Operators I. Springer, 2nd edition.

Kohonen, T. (1989). *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, 3 edition.

Linde, Y., Buzo, A., and Gray, R. M. (1980). An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28:84–95.

Slate, D. J. and Frey, P. W. (1991). Letter recognition using holland-style adaptive classifiers. *Machine Learning*, 6(2).

A The Roundness Condition

A crucial step in the DCA is the decision whether or not to split a prototype in two. If we follow the rule that a prototype should be split whenever its example space is not round enough, we need to find and explain a good measure of roundness.

In the following, μ will denote the probability distribution of examples associated to a prototype at the origin. We want the algorithm to be local, and therefore require that roundness be a function of μ, which is not influenced by the distribution of examples in other parts of the space.

A natural idea is then to measure the mean distance d_c between example and prototype. If we split the prototype at the origin in two, d_c will decrease, and it is reasonable to split if the change in d_c is substantially greater than what we obtain for uniform distribution in a ball.

Assume therefore that μ is absolutely continuous at the origin, and that we split one prototype at the orign in two, located at $t\xi$ and $-t\xi$, where t is small. Then the decrease of d_c will, to the first order, be $t\Delta$, where

$$\Delta = \Delta_\mu(\xi) = E\frac{|\xi \cdot x|}{|x|}, \qquad (9)$$

and x denotes a sample from μ. It is therefore reasonable to define the *split gain* of μ as the supremum of $\Delta_\mu(\xi)$ for ξ on the unit sphere.

The integral of Δ_μ over the unit sphere will always have the value $2\omega_{d-2}/(d-1)$, where d is the dimension and ω_k is the area of the unit sphere in R^{k+1}. Hence the split gain is minimal iff Δ_μ is constant.

where $d \neq 2$ is the dimension, $|\cdot|$ is Euclidean distance and x and y denote independent samples from the distribution μ. When $d = 2$ we instead use the expression

$$\frac{\exp(\log|x-y|)}{(E|x-y|^2)^{1/2}}. \tag{11}$$

Some good properties of this definition is that it gives a stronger condition than split gain (Theorem 1), that it is maximal on balls (Theorem 2), and that this maximal value is easily computable (Theorem 3).

A weakness of the definition is its numerical instability in high dimensions. Furthermore, it is singular when the support μ is not full-dimensional. We can avoid these problems by replacing the exponent $(2-d)$ by 1. The resulting roundness condition still gives good practical results.

B Maximum Roundness

Theorem 1 *For each real number $s \leq$ roundness$_{max}$, there is another number $T_s \geq$ gain$_{min}$, such that whenever the roundness $\geq s$, the split gain $\leq T_s$. We can choose T_s as a continuous and strictly decreasing function of s, and such that T_s approaches gain$_{min}$, as s approaches roundness$_{max}$.*

Proof 1 We assume that the dimension $d \geq 3$. The remaining cases are analogous. Then define T_s as the supremal split gain, given that the roundness is at least s. To calculate these values seems to be difficult, but numerical approximations should be possible to obtain.

It is immediate that T_s is a decreasing function of s. In order to prove the remaining statements, we consider the infimum of $E|x-y|^{2-d}$ over all distributions μ with $E|x-y|^2 = 1$ and split gain $\geq \delta$. Here, as before, x and y denote independent samples from the distribution μ, and $|\cdot|$ is Euclidean distance. By a small modification of the proof of Theorem 2 below, we can show that this infimum is attained. In fact, we need only modify ν as little as we please near the origin, in such a way that μ and ν have the same split gain.

But since this infimum is attained, T_s is continuous from the left. On the other hand, any distribution μ clearly sufficient to prove that $E_\mu|x-y|^{2-d}$, for μ in M, is minimized by a ball with center at the origin.

Let μ belong to M. As the barycenter is at the origin, the variance of distance equals twice the central moment, i.e. $E|x-y|^2 = 2E|x|^2$. The mass outside a ball of radius R is therefore less than $1/2R^2$. Let ν be the measure obtained from μ by moving all mass outside this ball to the origin. By dividing R^d into spherical shells, we can show that the roundness of μ is greater than that of ν, if R is chosen larger than c_d, where c_d only depends d.

Let m be the infimum of $E|x-y|^{2-d}$ for μ in M. By the preceding paragraph, m is strictly positive, and it is possible to find a sequence of distributions μ_k such that all μ_k have support in a ball of radius c_d, and such $E_{\mu_k}|x-y|^{2-d} \to m$. By functional analysis, there is a weakly convergent subsequence of μ_k. If we call the limit distribution μ, it will be a minimizing distribution. Hence there is at least one distribution that maximizes roundness.

We now turn to the question of what a minimizing distribution μ looks like. By the calculus of variation, we must have

$$\int |x-y|^{2-d} d\mu(x) \geq A + B|y|^2, \tag{12}$$

with equality on the support of μ. Here A and B are some constants. We now apply the Laplace operator to both sides of this equation. Some properties of the fundamental solution of the Laplace operator can be found in Hörmander, 1990. Using these, we conclude that μ equals $-\frac{2dB}{(d-2)\omega_{d-1}}$ times the volume measure on the support of μ, where ω_{d-1} is the area of the unit sphere in R^d. Hence a maximizing distribution is uniformly distributed on its support.

Now, for a minimizing distribution, we can compute $A = E|y|^{2-d}$, $B = -\frac{(d-2)\omega_{d-1}}{2ds}$, where s is the volume of the support of μ. This gives

$$E|x-y|^{2-d} = A + BE|y|^2 = E|y|^{2-d} - \frac{(d-2)\omega_{d-1}}{2ds}). \tag{13}$$

Out of all uniform distributions in M, uniform distribution on the ball makes both the first and second term

$$\begin{aligned} B &= -\frac{}{2} \\ E|x-y|^2 &= 3D2E|y|^2 = \frac{2d}{d+2} \\ E|x-y|^{2-d} &= 3DA+BE|y|^2 = \frac{2d}{d+2}, \qquad (14) \end{aligned}$$

and the estimate follows from the definition of roundness. □