Prototype Classifiers and the Big Fish. The Case of Prototype (Instance) Selection

Kuncheva, Ludmila

IEEE Systems, Man, and Cybernetics Magazine

DOI:
10.1109/MSMC.2019.2950534

Published: 20/04/2020

Peer reviewed version

Prototype Classifiers and the Big Fish

The Case of Prototype (Instance) Selection

Ludmila I Kuncheva

I. History, context, and sentiment

ONCE Jim told me "Write just the same way you talk!" And this is my excuse for the unashamedly colloquial text to follow.

Many, many years ago, sometime during the rock'n'roll-1980s, when ladies wore shoulder pads and IBM 80-column punched cards were in high fashion, everybody in our little research team had heard of Jim Bezdek. We were bewitched by fuzzy pattern recognition, and Jim Bezdek – the author of the famous book 'Pattern Recognition with Fuzzy Objective Function Algorithms' [1] – was our hero, alongside Lotfi Zadeh. During those times, none of us could afford to buy a copy of this coveted book but we were devouring all Jim's research articles we could get our hands on.

Years later, in 1993, I had the fortune to attend one of Jim's plenary talks at a conference in Aachen, Germany. In walked Jim Bezdek, in the most colourful Hawaiian shirt, blue shorts, the statutory baseball cap, and a smile brighter than Florida sunshine. And his talk is magic! Lo and behold, in 1996/1997, thanks to a generous grant from the COBASE program of NSF, I spent six months in Pensacola, working with Jim. I treasure those six months as the most valuable and enlightening experience in my career.

By that time, I was gradually losing faith in the 'fuzzy' side of fuzzy pattern recognition. You might ask what I was doing, then, visiting the current Editor-in-Chief of the IEEE Transactions of Fuzzy Systems? (Founding EiC, come to that!) Good question! Turned out, both Jim and I had a soft spot for the nearest neighbour classifier and its variants, and this is what our COBASE grant was about. This little paper tells the story of our collaboration on prototype selection, and what happened since.

II. Prototype Classifiers

A. Definition

In prototype classification, the data live in some metric space $\mathbb{R}^n$ equipped with a distance. Depending on which discipline or school (or continent) you come from, you may have a different name for the elements of $\mathbb{R}^n$. In pattern recognition, we call those 'objects', 'data points' or 'patterns' even. In machine learning, you are more likely to call them 'instances' or 'examples'. In statistics, we talk about 'observations' and (the dubious singular-plural) 'samples'. These are all the same thing: $\mathbf{x} \in \mathbb{R}^n$!

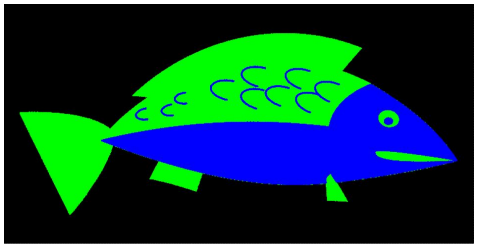
We have a labelled *reference set* of *prototypes*, $X \subset \mathbb{R}^n$. Each prototype is an element of $\mathbb{R}^n$ and is labelled in one of c classes. A new data point $\mathbf{x}^*$ is labelled as its nearest prototype from the reference set X. This is indeed the good old and wonderful nearest neighbour classifier (1-nn) [2], [3] where the points in the reference set are called prototypes.

L. Kuncheva is with the School of Computer Science and Electronic Engineering, Bangor University, UK, e-mail: mas00a@bangor.ac.uk

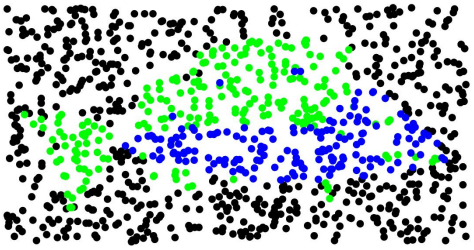
B. My example data

I can hear Jim saying "Pictures, Lucy! I like pictures! Where are the pictures?" Yes, yes, I like pictures too! Here they come.

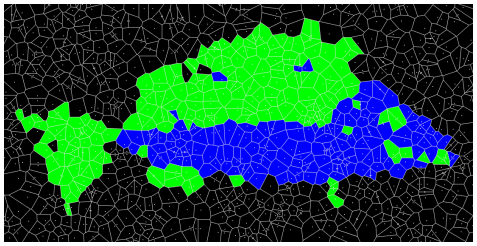
Have you ever seen a picture of Jim Bezdek *not* clutching or cuddling or dangling a 5+ kg fish? You have? Really? Take another look; I bet there is a piranha printed on his cap or T-shirt. Jim and fish... this is something else. So, for my examples here, I am choosing a 2-dimensional data set to suit the theme (Figure 1 (a)). Unusual, eh? Wherever did the good old Gaussians go? Just for fun, I will call the fish George.



(a) Full George data set



(b) Sampled data X (the prototypes)



(c) Classification regions for 1-nn using X as the reference set; error rate 6.68%

Fig. 1. The George data set.

There are three classes in this data set: 1 Background (black), 2 Top-and-tail (green), and 3 Face-and-bottom (blue). The only two features of a data point $\mathbf{x}$ are its x_1 and x_2 coordinates. The Bayes error for this data set is 0, because there are no overlapping points with different class labels. But the configuration of the classes is beautifully bizarre.

Figure 1 (b) shows the data set sampled randomly from the full data. The classification regions of the nearest neighbour classifier using the sampled data as the reference set are shown in Figure 1 (c). George looks a bit dishevelled there but, actually, over 93% of the labels match the original ones.

The classes are imbalanced as class 1 contains about 64% of the points, class 2 contains 20%, and class 3 contains 16%. While the imbalance is not as acute as in many real-life data sets, this still poses a challenge for many classification algorithms.

The full data was prepared from an image of size 391 rows $\times$ 769 columns of pixels, and contains 300,679 points altogether. Our randomly sampled data contains 1,000 points.

The question in this little paper is: can we use a reference set of fewer than 1,000 points and achieve similar (or better!) accuracy in recognising George?

C. Who cares about prototype classifiers?

Who cares about prototype classifiers today? Hello, we have deep learning neural networks! I hear this old question repeated over and over... In the 1980s we were ready to dismiss the decision tree classifier as we were building expert systems. But shortly after, we didn't care much about expert systems either, because the almighty Multi-Layer Perceptron (MLP) came to power. Come the 1990s, and we hailed the new king: the SVM! And today? Today we have deep learning neural networks and nothing else will do.

Delgado et al. [4] carried out a massive experimental comparison of classifiers in an attempt to answer the provocative question: Do we need hundreds of classifiers to solve real-world classification problems? A staggering 179 classifiers from 17 families were compared on 121 data sets. And the authors' answer is no! We don't need hundreds of classifiers. The current favourites are Random Forest [5] and the SVM [6]. Long live the winners! The message from the conclusion of the paper is clear: "The remaining families of classifiers, including other neural networks (radial basis functions, learning vector quantization and cascade correlation), discriminant analysis, decision trees other than C5.0, rule-based classifiers, other bagging and boosting ensembles, nearest neighbors, Bayesian, GLM, PLSR, MARS, etc., *are not competitive at all*." (italic mine). Ah, wait, but there is a new kid on the block! Rotation Forest [7] beats them all according to a more recent study by Bagnall et al. [8].[1] I kind of feel for all those non-competitive classifiers.

But we all know that there is no one single tool for every job! If that was the case, your car, your computer, and your smartphone would all be repaired with a hammer. It all depends on the data, of course. And, hey, not all is lost. In 2008, the nearest neighbour family was included (by experts!) among the top 10 algorithms in data mining [9].

Figure 2 shows a bibliometric snapshot of the publication rate for the nearest neighbour classifier.

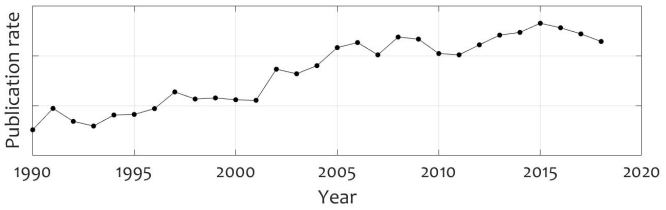


Fig. 2. Publication rate for the nearest neighbour classifier. (Data sourced from Web-of-Science, on the 10th June 2019.)

The data for this graph were sources from Web-of-Science[2] on the 10th June 2019 and cover the period 1990–2018. The number of publications grows every year. Therefore we plotted only the *rate* rather than the number of publications on the subject. To calculate the rate, two searches were carried out for each year. The first search used keyword 'classif*' to account for 'classifier', 'classifying', 'classification', etc. (1,099,295 returns). The second search was done among the results from the first search using keywords ('nearest neighbor' or 'nearest neighbour'), returning 11,918 articles. Suppose we retrieved a papers from Search 1 and b papers from Search 2 for a given year. The rate was calculated as b/a. The trend of the graph in Figure 2 shows that the interest in the nearest neighbour family of classifiers increases steadily over the years.

III. Prototype (instance) selection

Observing that the 1-nn philosophy underpins many seemingly unrelated classifiers, Jim and I set off to unite these classifiers under the same umbrella. We called it the Generalised Nearest Prototype Classifier (GNPC) [10], [11]. We were hoping to pull a rabbit out of the hat, that is, identify niches which were not explored thus far, and propose new alternative versions of the prototype classifier. Alas, Floppy (the rabbit) did not materialise at the time but, instead, Jim and I got properly sucked into one of the sidelines of 1-nn: instance selection (also known as: prototype selection/extraction/generation/replacement, data editing for the 1-nn classifier, data condensing, data reduction, and more).

Here we will take *prototype selection* to mean that we select a subset S of the reference set X which satisfies some criteria related to the classification accuracy of 1-nn using S as the reference set. Requiring a zero error on X (resubstitution error) gives rise to the so-called 'condensing' methods, the classic instance of which is Hart's Condensed Nearest Neighbour (CNN) [12]. A reference set with a zero resubstitution error is called a 'consistent' subset of X. This approach preserves boundary objects which are likely to be misclassified if missing from the prototype set. The alternative approach, called 'editing', is to select prototypes by removing noise. This approach aims at a better generalisation accuracy with S compared to that when using the whole of X as the

[1] I am quite proud of this, actually, as I have a little contribution myself to the Rotation Forest ensemble method.

[2] https://wok.mimas.ac.uk/

reference set. The pioneering method in this category is due to Wilson (1972) [13]. In this approach, border objects that may be misclassified are discarded. A third category, called 'hybrid', includes methods which combine the two ideas. A myriad of methods for prototype selection have been proposed in all three categories since those early years [14], [15], [16], [17], [18], [19].

Jim and I were curious about the hybrid approach. But instead of explicitly combining strategies for keeping and discarding prototypes, we chose a random, criterion-driven approach [20]. Our study was meant as a proof-of-concept, and we only played around with the famous iris data set. We discovered that a random criterion-driven approach (brute-force random search and a basic genetic algorithm) offered the best compromise between classification accuracy and reduction rate compared to the classical examples of editing and condensing. We subsequently carried out experimental comparisons [21] and included also methods which belonged in the group of prototype *replacement*. In other words, S is no longer a subset of X but a subset of $\mathbb{R}^n$, with a cardinality restriction $|S| \leq |X|$. Our random, criterion-driven methods were doing OK but were not as good as the prototype-replacement competitors. In those pre-Google times, we were not even aware that our brute-force random search actually had a name: Monte Carlo 1 (MC1) [22]. Much as we wanted to, we could not afford to run a large experiment then. Intel was yet to release the first Xeon processor, the Pentium II Xeon 400 (1 M cache, 400 MHz).

A funny story unfolded shortly after the publication of our 'apotheosis' of random/GA prototype selection [20]. Our experiments gave a 14-element consistent set for the iris data. The previous record was a 15-element consistent set, so we beat it by 1 element. Before we published the paper Jim said "You know what? I want to be double and treble sure that we have not made a mistake. Delete your 1-nn code, write it again from scratch and verify the result. The first thing people will do is stick our winning prototype set in their 1-nn classifier!" So I did; no mistake. The paper came out. Almost instantly the author of the previous winner (the 15-element consistent prototype set) wrote an indignant e-mail to Jim and me claiming that our supposedly-consistent set mislabels one object in the iris data! They suggested that we write a retractor note and apologise for misleading the journal's readership. This e-mail exchange didn't do my blood pressure any good but I knew that there was no mistake. It transpired that we have been using slightly different versions of the iris data! Jim and I then sourced the original paper by Fisher [23] where Anderson's data [24] were published, and it turned out that the 'real' data set matched Jim's and my version. Could have easily been the other way around. Jim was so amused by the whole story that he indeed wrote a note, but not to apologise. The title was: 'Will the *real* iris data please stand up?' [25]. The note includes a table with the 'real' (original) iris data set and warns about the non-matching variants floating around.

Years passed, technology prospered, and big data descended upon us. And not only big data! In addition to the problem of scalability [26], [27], [28], prototype selection was facing new challenges [29]: streaming data [30], unlabelled data, data with concept drift [31], and more. How did random prototype selection methods fare in the new world order? Quite well, apparently, especially the evolutionary algorithms [32], [33], [34]. Shall we check some of the recent favourites then? Let's see how our random methods manage to reconstruct George to a recognisable character with as few prototypes as possible.

IV. The fun part: recognising George

A. Methods

García et al. (2012) [18] and Triguero et al. (2012) [19] reviewed between them over 75 prototype selection and replacement methods, and ran experimental comparisons. As we have our hearts set on prototype *selection*, here are the methods which were identified in the conclusion of García et al.'s survey as the best ones (the first-mentioned method in each group).

To simplify the algorithms, here we introduce some common concepts and notations:

- All algorithms take as input a labelled data set X with N objects.
- We will denote by M the desired number of prototypes which will be an input parameter for some of the algorithms.
- T denotes the number of iterations, K denotes the population size, W denotes the number of generations.
- We will need two sets of indices $A \leftarrow \{1, 2, \ldots, N\}$ and $B \leftarrow \{1, 2, \ldots, M\}$.
- We will also need two functions: $e(S, X)$ returning the 1-nn classification error for X when S is used as the reference set; and $choose(Q, m)$ returning a random subset of cardinality m sampled without replacement from set Q.

  Note that if I is a set of indices of elements of X, we use $X(I)$ to denote the subset of X containing the indexed elements.

• **RMHC** (1994). From the hybrid family (editing and condensing), Random Mutation Hill Climbing [22] achieved an excellent trade-off between reduction and classifier success in the experiments. RMHC is one of the beautifully-elegant random, criterion-driven methods. (Score!)

While the original algorithm is in a binary form, for the little experiment with George, we can indulge in a less efficient but more straightforward implementation (Algorithm 1).

In our implementation, we evolved several solutions and picked the best S among them.

• **RNGE** (1997). Relative Neighbourhood Graph Editing is an editing prototype selection algorithm [35] classed as the best in its group [18].

Relative Neighbourhood Graph (RNG) is an undirected graph defined on X. There is an edge between $p \in X$ and $q \in X$ if there is no other point $r \in X$ which is closer to both p and q than they are to each other. The editing algorithm works by first building the RNG of X and then removing all points which are misclassified by their immediate neighbours in the graph (Algorithm 2).

Algorithm 1: RMHC prototype selection algorithm

Input: X, M, T

Output: Reference set S, $S \subset X$, $|S| = M$

$C \leftarrow choose(A, M)$ *// the chromosome to mutate*
$e \leftarrow e(X(C), X)$ *// stored error*
for $i = 1 : T$ **do**
  $C_{temp} \leftarrow C$.
  $k \leftarrow choose(B, 1)$ *// index to mutate*
  $C_{temp}(k) \leftarrow choose(A \setminus C, 1)$ *// replace*
  $e_{temp} \leftarrow e(X(C_{temp}), X)$.
  if $e_{temp} \leq e$ **then**
    $C \leftarrow C_{temp}$; $e \leftarrow e_{temp}$
Return $S = X(C)$.

Algorithm 2: RNGE prototype selection algorithm

Input: X

Output: Reference set S, $S \subseteq X$

Build G, the RNG for X.
Remove all points which are misclassified by their immediate neighbours in G.
Return the remaining points as S.

• **RNN** (1972). The Relative Nearest Neighbour rule [36] was singled out as one of the best two methods in the condensing group [18]. RNN starts with a consistent reference set S (zero errors of 1-nn on X) and reduces it further by removing one element at a time and checking whether the set is still consistent. If not, return the element in the set. If the set is consistent, remove the element permanently. Continue until all elements have been checked and there has not been any change of S (Algorithm 3).

Algorithm 3: RNN prototype selection algorithm

Input: X

Output: Reference set S, $S \subseteq X$

Run Hart's algorithm [12] on X to obtain an initial C.
$flag \leftarrow true$.
while $flag$ **do**
  $flag \leftarrow false$.
  for *each element* i *of* C **do**
    $C' \leftarrow C \setminus \{i\}$. *// remove* i *temporarily*
    if C' *is consistent* **then**
      $C \leftarrow C'$. *// remove* i *permanently*
      $flag \leftarrow true$.
Return $S = X(C)$.

Did you notice? All three winning algorithms are *old* and *simple*. My kind of algorithms! We add to this collection our own baselines and competitors as explained below.

• **H** (1968) Hart's CNN [12] returns a consistent set, usually with a very good reduction rate. This is the archetypal condensing algorithm which gave rise to the whole condensing branch.

• **W** (1972) Wilson's algorithm [13] is the forefather of the editing branch of prototype selection algorithms. It marks for deletion all objects of X which are misclassified by their k nearest neighbours (typically, $k = 3$). Then the marked objects are removed, and the remaining set is returned as S.

• **MC1** (1994) The Monte Carlo method for prototype selection [22] is the same as our Random Search [20]. This is a brute-force random search whereby we generate T prototype sets and pick the best among them. The value of T is chosen in advance.

• **GA** (1995) Genetic Algorithms (GA) are a perfect fit for prototype selection [37], [20], [32], [33], [34]. The chromosome can encode $S \subseteq X$ storing 0 at position i if the ith element of X is not in S, and 1, otherwise. The version of a GA which we used here is shown in Algorithm 4. It allows for pre-specifying the number of prototypes. However, due to the cross-over, there may be repeated prototypes within a chromosome. This means that M is an upper limit on the number of prototypes for the GA.

The George data set and the MATLAB code for this illustration are available at https://github.com/LucyKuncheva/instance_selection.

B. Experimental set-up

We can hardly glorify this little illustration with an 'experiment' status but we still need to explain how we made the comparisons as fair as possible.

We used the George data set (Figure 1 (b)) sampled from the 'full' George data set (Figure 1 (a)).[3] The prototype selection methods which we used are listed in Table I together with the results.

In addition to the methods listed in the previous subsection, we included W+H, which is an application of Wilson's method (W) followed by Hart's method (H). This combined method (hybrid type) often leads to a small and accurate references set.

We took care that all our random methods carry out exactly the same number of evaluations of the criterion function (1-nn error rate on the sampled George data). The parameters in this experiment were as follows:

- MC1: Number of iterations $T = 12,000$.
- GA: Population size $K = 40$, number of generations $W = 300$.

[3]Both data sets are included in the GitHub library https://github.com/LucyKuncheva/instance_selection.

Algorithm 4: GA prototype selection algorithm

Input: X,M,K,W

Output: Reference set S, $S \subset X$, $|S| = M$.

for $i = 1 : K$ **do**
  $P(i) \leftarrow choose(A, M)$ *// random chromosome*
  $f_p(i) \leftarrow 1 - e(X(P(i)), X)$ *// population fitness*

for $gen = 1 : W$ **do**
  $O \leftarrow \emptyset$. *// offspring set*
  for $par = 1 : K/2$ **do**
    $p_1 \leftarrow choose(P, 1)$ *// parent 1*
    $p_2 \leftarrow choose(P, 1)$ *// parent 2*
    $k \leftarrow choose(B, 1)$ *// crossover point*
    Swap the tail parts of p_1 and p_2 to create offspring o_1 and o_2. (Tail is from $k+1$ to M. If $k = M$, no crossover occurs and the offspring are the parents themselves.)
    $O \leftarrow O \cup \{o_1, o_2\}$
  for $j = 1 : K$ **do**
    $C \leftarrow O(j)$
    $m \leftarrow choose(B, 1)$ *// index to mutate*
    $C(k) \leftarrow choose(A \setminus C, 1)$ *// replace*
    $f_o(j) \leftarrow 1 - e(X(C), X)$ *// offspring fitness*
    $O(j) \leftarrow C$
    Pool f_p and f_o and sort in descending order. Keep the best K chromosomes from $P \cup O$ to be the new population P and store the respective fitnesses.
Return $S = X(P(1))$ *// the best chromosome*

TABLE I
RESULTS FROM THE EXPERIMENT WITH THE NOISE-FREE GEORGE DATA.

Method	Type	Error rate (%)	Number of prototypes	Time (s)
1-nn	–	6.68	1000	0.18
H	C	7.90	211	24.93
W	E	7.10	913	0.50
W + H	H	8.44	101	22.71
RNGE	E	6.59	921	3.92
RNN	C	8.20	160	27.81
RMHC	H	15.83	10	81.99
RMHC	H	13.31	20	79.27
RMHC	H	10.47	100	83.42
RMHC	H	9.49	200	84.04
MC1	H	20.21	10	75.45
MC1	H	15.62	20	78.80
MC1	H	11.16	100	81.23
MC1	H	9.83	200	83.81
GA	H	**12.68**	**10**	76.63
GA	H	**8.47**	**20**	77.20
GA	H	**6.52**	**98**	84.71
GA	H	6.96	195	82.49

TABLE II
RESULTS FROM THE EXPERIMENT WITH THE NOISY GEORGE DATA.

Method	Type	Error rate (%)	Number of prototypes	Time (s)
1-nn	–	16.24	1000	0.49
H	C	20.00	415	27.87
W	E	8.43	806	0.48
W + H	H	**9.68**	**91**	19.30
RNGE	E	**8.17**	**794**	3.97
RNN	C	21.18	355	33.94
RMHC	H	24.70	10	76.70
RMHC	H	15.65	20	77.74
RMHC	H	14.38	100	80.67
RMHC	H	15.61	200	82.14
MC1	H	21.25	10	74.18
MC1	H	15.62	20	74.91
MC1	H	15.88	100	76.50
MC1	H	14.28	200	79.76
GA	H	**15.85**	**10**	75.64
GA	H	**11.08**	**20**	77.30
GA	H	9.92	100	80.82
GA	H	12.70	197	83.03

- RMHC: Number of chromosomes evolved (separately) $K = 40$, number of mutations $W = 300$.

In the second leg of the experiment, we contaminated George with label noise by flipping the labels of 10% of the sampled data to a different class. Figure 3 shows the classification regions of 1-nn with the contaminated set. Ouch! George looks a little exploded here...

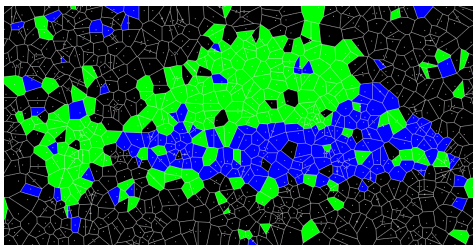


Fig. 3. Classification regions of 1-nn with 10% label-noise contamination; error rate 18.49% (or a nice pyjama pattern).

C. Results

Table I shows the results with the clean data, and Table II, the results with the noisy data.

To make more sense of the numbers, we will use a scatterplot. The x-axis is the logarithm of the number of retained prototypes out of the initial 1,000. We chose the logarithmic scale for the sole purpose of making the graphs less crowded at the smaller cardinalities. The y-axis is the 1-nn classification error on the full data (the whole of George). An ideal point would sit at $(\ln(3) = 1.0986, 0)$ where we have one prototype of each class and zero error. The closer the point is to the origin, the better the method. The results are shown in Figure 4 for the noise-free George and in Figure 5 for the noisy George.

In both figures, each prototype selection method is shown with a yellow marker. Circles represent hybrid methods, triangles represent condensing, and squares represent editing. The

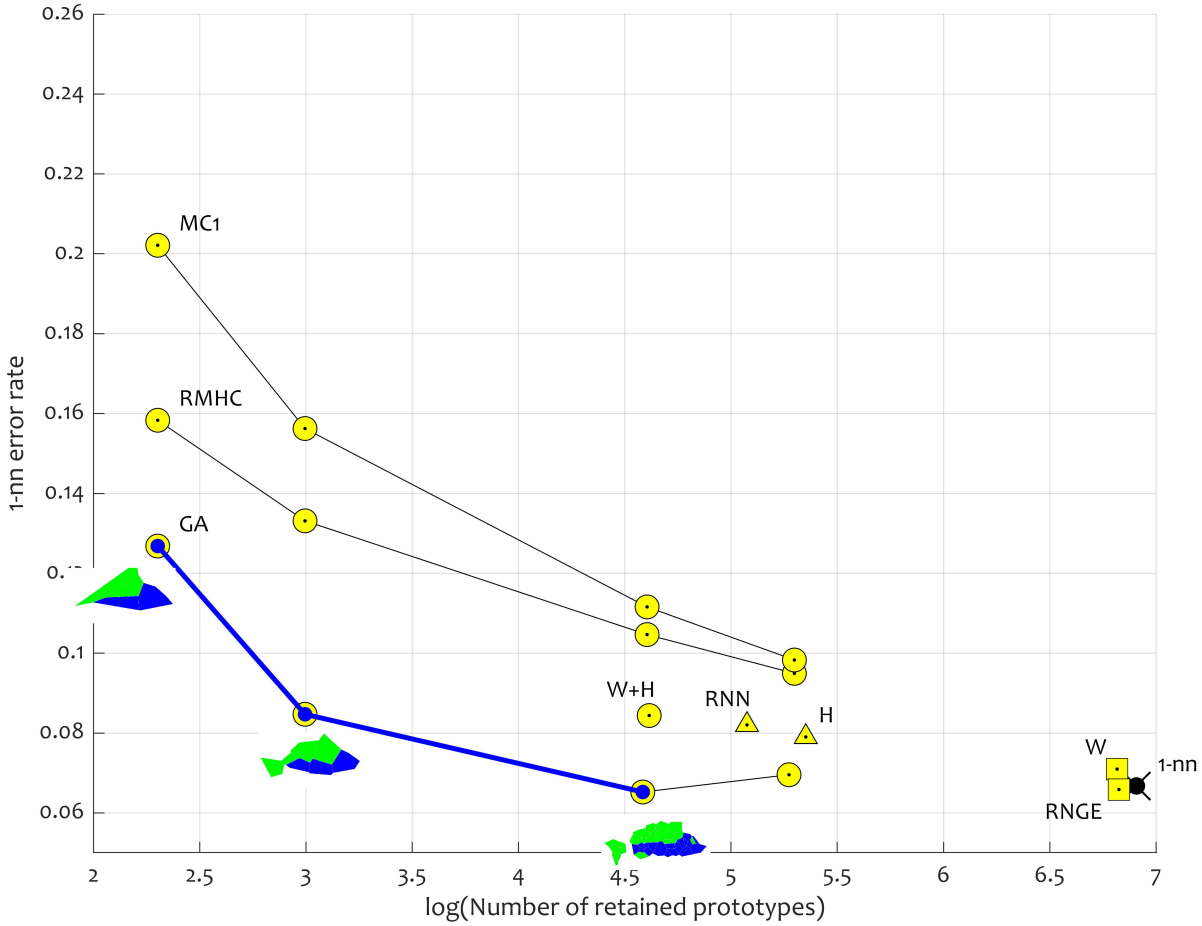


Fig. 4. Scatterplot of the results for the noise-free George. The blue line represents the Pareto front.

thick blue line is the Pareto front, that is, the collection of non-dominated methods. These methods are highlighted in boldface in the respective tables. Note that we can choose the number of prototypes for MC1, RMHC, and GA. The versions of a method for different number of prototypes are shown as line graphs. In addition, next to each method on the Pareto front, we show a cute little portrait of George which resulted from that method (the classification regions, leaving the background white).

D. Discussion

Well, what is trivial is trivial: when there is noise in the data, all points are higher up, indicating higher error. Without noise, the editing methods (RNN and H) were good, and if we hadn't chosen serendipitous parameters of our GA, these methods would have been on the Pareto front. When there is noise, however, the condensing methods learn that noise to perfection, and the generalisation error shoots up (top right corner of Figure 5. The editing competitors (W and RNGE) are unfazed by noise. They consistently return good but large reference sets. The type of random noise that we included is filtered quite well by them, and RNGE found its place in the Pareto front for the noisy George, beating W by a whisker.

The clear winners are the hybrid methods, which echoes the findings of other authors. We don't need to explicitly enforce the strategy ('keep the noise' or 'clean the noise') within the method; criterion-driven methods fare a lot better.

What is the situation with our beloved MC1 and GA? In our 1998 paper [20], we found that random, criterion-driven methods such as MC1 and GA were simple and effective, something that was also mentioned as a surprising observation by Skalak [22] in relation to RMHC. In our later paper [21], however, we could not confirm this result. My GA implementation (I will blame that) kicked the GA towards the bottom of the league table. The difference with Algorithm 4 here is that previously we used a criterion which sought a compromise between the 1-nn error and the number of prototypes in the form of a weighted sum. Here, on the other hand, we specify a limit on the number of prototypes. GA turned out to be the best among the competitors, which were chosen among the most successful prototype selection methods [18]. Fluke as it may be, our little experiment with GA (and George) showed that this strategy can handle noise. However, in both

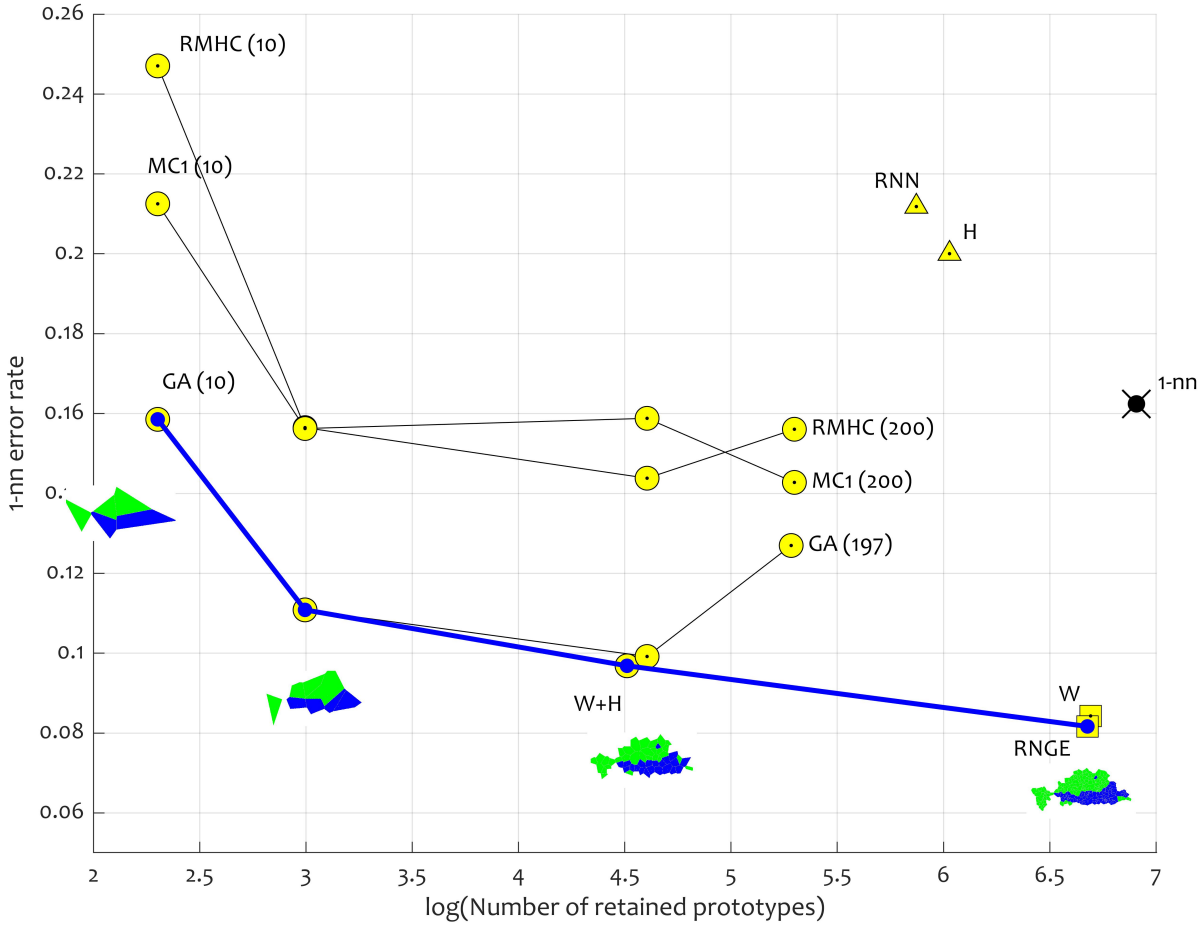


Fig. 5. Scatterplot of the results for the noisy George. The blue line represents the Pareto front.

experiments, the generalisation error for $M = 200$ prototypes increases (possibly due to overfitting), leaving the last point on the GA line graph out of the Pareto front for the clean data. For the noisy George, the GA with 100 prototypes is marginally worse than W+H, another classical hybrid prototype selection method.

The random search (MC1) did not work well here and nor did RMHC. The likely reason is that the class configuration was chosen deliberately to be challenging unlike many experimental studies where the classes are sampled as good old Gaussians.

Yes, yes, we evaluate our criterion on the training data! This is what we have been using all the way here (condensing methods don't have a choice as they are meant to guarantee zero resubstitution error). The scatterplots, however, show the error on the full data, which consists of the 1,000 sampled points (0.33%) and the remaining 299,679 points (99.67%). Don't get me started on the limitations of this example/illustration. Indeed, the list is as long as this journal has pages. But the moral of the story is that if we do need a very small subset of the data with an acceptable error rate, we may have to resort to those random, criterion-driven approaches which seem to offer a good compromise between the cardinality of the reference set and the 1-nn error rate. Long live random search!

V. Conclusion

Guess what? That *was* the conclusion! "Long live random search".

Back in 1997, when Jim and I were writing papers together, I would come to him with a draft of a paper, and he would invariably return a comment "What kind of conclusion is this?!? You have run out of steam, Lucy." And then he writes the conclusion himself. I wish, one day, I could match Jim's astute, eloquent, and endlessly entertaining writing. A girl can dream...

References

[1] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. N.Y.: Plenum Press, 1981.

[2] E. Fix and J. L. Hodges, "Discriminatory analysis : Non parametric discrimination : Small sample performance," USAF School of Aviation Medicine, Randolph Field,Texas, Tech. Rep. Project 21 - 49 - 004 (11), 1952.

[3] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. NY: John Wiley & Sons, 2001.

[4] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *Journal of Machine Learning Research*, vol. 15, pp. 3133–3181, 2014. [Online]. Available: http://jmlr.org/papers/v15/delgado14a.html

[5] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[6] C. Cortes and V. N. Vapnik, "Support-vector networks," *Machine Learning.*, vol. 20, no. 3, pp. 273–297, 1995.

[7] J. J. Rodríguez, L. I. Kuncheva, and C. J. Alonso, "Rotation forest: A new classifier ensemble method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1619–1630, Oct 2006.

[8] A. J. Bagnall, A. Bostrom, G. C. Cawley, M. Flynn, J. Large, and J. Lines, "Is Rotation Forest the best classifier for problems with continuous features?" *CoRR*, vol. abs/1809.06705, 2018. [Online]. Available: http://arxiv.org/abs/1809.06705

[9] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 algorithms in data mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1–37, 2008.

[10] L. Kuncheva and J. Bezdek, "An integrated framework for generalized nearest prototype classifier design," vol. 6, no. 5, pp. 437–457, 1998.

[11] L. I. Kuncheva and J. C. Bezdek, "Presupervised and postsupervised prototype classifier design," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1142–1152, 1999.

[12] P. E. Hart, "The condensed nearest neighbor rule," *IEEE Transactions on Information Theory*, vol. 16, pp. 515–516, 1968.

[13] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," vol. SMC-2, pp. 408–421, 1972.

[14] B. V. Dasarathy, *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. Los Alamitos, California: IEEE Computer Society Press, 1991.

[15] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Machine Learning*, vol. 38, no. 3, pp. 257–286, 2000.

[16] H. Brighton and C. Mellish, "Advances in instance selection for instance-based learning algorithms," *Data Mining and Knowledge Discovery*, vol. 6, no. 2, pp. 153–172, 2002.

[17] H. E. Liu, *Instance Selection and Construction for Data Mining*. Berlin, Heidelberg: Springer-Verlag, 2010.

[18] S. Garcia, J. Derrac, J. R. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: Taxonomy and empirical study," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 417–435, March 2012.

[19] I. Triguero, J. Derrac, S. Garcia, and F. Herrera, "A taxonomy and experimental study on prototype generation for nearest neighbor classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 1, pp. 86–100, Jan 2012.

[20] L. I. Kuncheva and J. C. Bezdek, "On prototype selection: Genetic algorithms or random search?" *IEEE Transactions on Systems, Man and Cybernetics*, vol. C28, no. 1, pp. 160–164, 1998.

[21] J. Bezdek and L. Kuncheva, "Nearest prototype classifier designs: An experimental study," *International Journal of Intelligent Systems*, vol. 16, no. 12, pp. 1445–1473, 2001.

[22] D. B. Skalak, "Prototype and feature selection by sampling and random mutation hill climbing algorithms," in *Proc. 11th International Conference on Machine Learning*. Morgan Kaufmann, 1994, pp. 293–301.

[23] R. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, pp. 179–188, 1936.

[24] E. Anderson, "The irises of the gaspe peninsula," *Bulletin of the American Iris Society*, vol. 59, pp. 2–5, 1935.

[25] J. C. Bezdek, J. M. Keller, R. Krishnapuram, L. I. Kuncheva, and N. R. Pal, "Will the real iris data please stand up?" *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 3, pp. 368–369, 1999.

[26] J. R. Cano, F. Herrera, and M. Lozano, "Stratification for scaling up evolutionary prototype selection," *Pattern Recognition Letters*, vol. 26, no. 7, pp. 953 – 963, 2005.

[27] A. de Haro-García and N. García-Pedrajas, "A divide-and-conquer recursive approach for scaling up instance selection algorithms," *Data Mining and Knowledge Discovery*, vol. 18, no. 3, pp. 392–418, 2009.

[28] I. Triguero, D. Peralta, J. Bacardit, S. García, and F. Herrera, "Mrpr: A mapreduce solution for prototype reduction in big data classification," *Neurocomputing*, vol. 150, pp. 331–345, 2015.

[29] L. I. Kuncheva and I. A. D. Gunn, "A concept-drift perspective on prototype selection and generation," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2016)*, Vancouver, Canada, 2016, pp. 16–23.

[30] C. Alippi, G. Boracchi, and M. Roveri, "A just-in-time adaptive classification system based on the intersection of confidence intervals rule," *Neural Networks*, vol. 24, no. 8, pp. 791 – 800, 2011.

[31] N. Lu, J. Lu, G. Zhang, and R. L. de Mantaras, "A concept drift-tolerant case-base editing technique," *Artificial Intelligence*, vol. 230, pp. 108–133, 2016.

[32] J. R. Cano, F. Herrera, and M. Lozano, "Using evolutionary algorithms as instance selection for data reduction in kdd: an experimental study," *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 6, pp. 561–575, 2003.

[33] N. García-Pedrajas, J. A. Romero del Castillo, and D. Ortiz-Boyer, "A cooperative coevolutionary algorithm for instance selection for instance-based learning," *Machine Learning*, vol. 78, no. 3, pp. 381–420, 2010.

[34] N. García-Pedrajas, A. de Haro-García, and J. Pérez-Rodríguez, "A scalable approach to simultaneous evolutionary instance and feature selection," *Information Sciences*, vol. 228, pp. 150–174, 2013.

[35] J. S. Sánchez, F. Pla, and F. J. Ferri, "Prototype selection for the nearest neighbour rule through proximity graphs," *Pattern Recognition Letters*, vol. 18, no. 6, pp. 507–513, 1997.

[36] G. W. Gates, "The reduced nearest neighbor rule," 1972.

[37] L. I. Kuncheva, "Editing for the k-nearest neighbors rule by a genetic algorithm," *Pattern Recognition Letters*, vol. 16, pp. 809–814, 1995.