Prototype Classifiers and the Big Fish

The Case of Prototype (Instance) Selection

by Ludmila I. Kuncheva

Jim Bezdek once told me, "Write just the same way you talk!" That 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 research team had heard of Jim Bezdek. We were bewitched by fuzzy-pattern recognition, and Jim—the author of the famous book *Pattern Recognition With Fuzzy Objective Function Algorithms* [1]—was our hero, alongside Lotfi Zadeh.

Years later, in 1993, I had the good fortune to attend one of Jim's plenary talks at a conference in Aachen, Germany. He walked in wearing the most colorful Hawaiian shirt, blue shorts, a baseball cap, and a smile brighter than Florida sunshine. His talk was magic. In 1996–1997, thanks to a generous grant from the National Science Foundation's Collaboration in Basic Science and Engineering (COBASE) program, I spent six months in Pensacola, Florida, working with him. I treasure that time as the most valuable and enlightening experience of my career.

Digital Object Identifier 10.1109/MSMC.2019.2950534
Date of current version: 20 April 2020

I was gradually losing faith in the fuzzy side of fuzzy-pattern recognition. You might ask what I was doing, then, visiting the editor-in-chief (EiC) of *IEEE Transactions on Fuzzy Systems* (the founding EiC, at that). Good question. It turned out that Jim and I had a soft spot for the nearest-neighbor (1-NN) classifier and its variants, which was the subject of our COBASE grant. This article tells the story of our collaboration on prototype selection and what has happened since.

I treasure that time as the most valuable and enlightening experience of my career.

Prototype Classifiers

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 even *patterns*. 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 names for the same thing: $\mathbf{x} \in \mathbb{R}^n$.

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

My Example Data

I can hear Jim saying, "Pictures, Lucy. I like pictures! Where are the pictures?" I like pictures, too, and here they come. Have you ever seen a picture of Jim when he wasn't clutching, cuddling, or dangling a 5-kg-plus fish? You have? Really? Take another look; I bet there is a piranha printed on his cap or T-shirt. Jim and fish … it's something else. So, for my examples, I am choosing a 2D 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.

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 data point $\mathbf{x}$ are its x_1 and x_2 coordinates. The Bayes error for this data set is zero 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 1-NN classifier using the sampled data as the reference set are shown in Figure 1(c). George looks a bit disheveled but, actually, more than 93% of the labels match the original ones.

The data were prepared from an image of 391 rows × 769 columns of pixels and contain 300,679 points. Our randomly sampled data contain 1,000 points. The question in this article is: Can we use a reference set of fewer than 1,000 points and achieve similar (or better) accuracy in recognizing George?

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. During the 1980s, we were ready to dismiss the decision-tree classifier since we were building expert systems. Soon, we didn't care much about expert systems, either, because the almighty multilayer perceptron came to power. By the 1990s, we hailed the new king: the support-vector machine (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 was no. We don't need hundreds of classifiers. The current favorites are the random forest [5] and 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

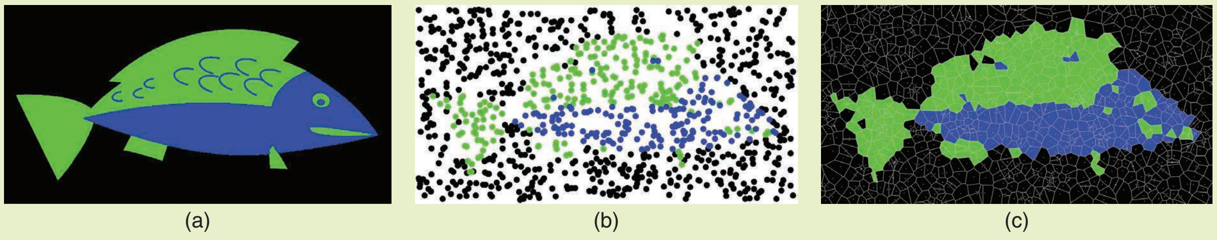

Figure 1. The George data set. The (a) full set, (b) sampled data, X (the prototypes), and (c) 1-NN regions using X (6.68% error).

(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, 1-NNs, Bayesian, generalized linear model, partial least-squares regression, multivariate adaptive regression splines, etc., *are not competitive at all*" (my emphasis). But wait, there is a new kid on the block. The rotation forest [7] beats them all, according to a more recent study by Bagnall et al. [8]. (I am quite proud of this, actually, as I have a little contribution to the rotation-forest ensemble method.)

I sympathize with all those uncompetitive classifiers. But we all know that there is no single tool for every job. If that were the case, your car, computer, and smartphone could all be repaired with a hammer. The tool selection depends on the data, of course. And not all is lost. In 2008, the 1-NN family was included (by experts) among the top 10 algorithms in data mining [9].

Do we need hundreds of classifiers to solve real-world classification problems?

Prototype (Instance) Selection

Observing that the 1-NN philosophy underpins many seemingly unrelated classifiers, Jim and I set off to unite them under the same umbrella. We called it the *generalized nearest-prototype classifier* [10], [11]. We were hoping to pull a rabbit out of the hat; that is, identify niches that had not been explored and propose alternative versions of the prototype classifier. Alas, Floppy (the rabbit) did not materialize, and instead, Jim and I got properly sucked into one of the side issues of the 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).

We will take prototype selection to mean that we choose a subset, S, of the reference set, X, which satisfies some criteria related to the classification accuracy of the 1-NN using S as the reference set. Requiring a zero (resubstitution) error on X gives rise to the so-called condensing methods, of which Hart's condensed 1-NN (CNN) [12] is the classic instance. A reference set with a zero resubstitution error is called a *consistent subset* of X. This approach preserves boundary objects that are likely to be misclassified if they are missing from the prototype set. The alternative approach, called *editing*, is to select prototypes by removing noise. It aims for better generalization accuracy with S compared to the result when the whole of X is used as the reference set. The pioneering method in this category is due to Wilson (1972) [13]. Through this approach, border objects that may be misclassified are discarded. A third category, called *hybrid*, includes methods that combine the two ideas. Myriad methods for prototype selection have been proposed in all three categories since those early years [14]–[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 technique [20]. Our study was meant to be 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 (GA)] offered the best compromise between classification accuracy and the reduction rate compared with the classical examples of editing and condensing. We subsequently carried out experimental comparisons [21] and included methods that belonged in the group of prototype replacement. In other words, S is no longer a subset of X but of $\mathbb{R}^n$, with a cardinality restriction $|S| \leq |X|$. Our random, criterion-driven methods were doing okay but not as well as the prototype-replacement competitors. During 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. Intel was yet to release the first Xeon processor, the Pentium II Xeon 400 (1-MB 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 one. Before we published the paper, Jim said, "You know what? I want to be double and triple 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, and there was no mistake. The paper came out. Almost instantly, the author of the previous winner (the 15-element consistent prototype set) wrote an indignant email to Jim and me claiming that our supposedly consistent set mislabeled one object in the iris data. The author suggested that we write a retraction and apologize for misleading the journal's readership.

That email exchange didn't do my blood pressure any good, but I knew there was no mistake. It transpired that we had 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. It could easily have been the other way around. Jim was so amused by the situation that he wrote a note but not to apologize. The title was: "Will the *Real* Iris Data Please Stand Up?" [25]. The note included a table with the original iris-data set and warned about the unmatching 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]–[28], prototype selection was facing new challenges [29]: streaming data [30], unlabeled 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]–[34]. Shall we check some of the recent favorites? Let's see how our random methods manage to reconstruct George by using as few prototypes as possible.

Algorithm 1. The RMHC prototype-selection algorithm.

```
Input: $X, M, T$
Output: Reference set $S$, $S \subset X$, $|S| = M$
1  $C \leftarrow choose(A, M)$ // the chromosome to mutate
2  $e \leftarrow e(X(C), X)$ // stored error
3  for $i = 1{:}T$ do
4      $C_{temp} \leftarrow C$.
5      $k \leftarrow choose(B, 1)$ // index to mutate
6      $C_{temp}(k) \leftarrow choose(A \backslash C, 1)$ // replace
7      $e_{temp} \leftarrow e(X(C_{temp}), X)$.
8      if $e_{temp} \le e$ then
9          $C \leftarrow C_{temp}$; $e \leftarrow e_{temp}$
10 Return $S = X(C)$.
```

Algorithm 2. The RNGE prototype-selection algorithm.

```
Input: $X$
Output: Reference set $S$, $S \subseteq X$
1  Build $G$, the RNG for $X$.
2  Remove all points which are misclassified by their
     immediate neighbors in $G$.
3  Return the remaining points as $S$.
```

Algorithm 3. The RNN prototype-selection algorithm.

```
Input: $X$
Output: Reference set $S$, $S \subseteq X$
1  Run Hart's algorithm [12] on $X$ to obtain an initial $C$.
2  $flag \leftarrow true$.
3  while $flag$ do
4      $flag \leftarrow false$.
5      for each element $i$ of $C$ do
6          $C' \leftarrow C \backslash \{i\}$. // remove $i$ temporarily
7          if $C'$ is consistent then
8              $C \leftarrow C'$. // remove $i$ permanently
9              $flag \leftarrow true$.
10 Return $S = X(C)$.
```

The Fun Part: Recognizing George

Methods

García et al. [18] and Triguero et al. [19] reviewed, between them, more than 75 prototype selection and replacement methods and ran experimental comparisons. Since we have our hearts set on prototype selection, here are the methods that García et al. identified as the best (the first-mentioned method in each group). To simplify the algorithms, I introduce some common concepts and notations:

- All algorithms take labeled data set X with N objects as input.
- We will denote by M the desired number of prototypes that will be an input parameter for some of the algorithms.
- T denotes the number of iterations, K represents the population size, and W designates the number of generations.
- We will need two sets of indices $A \leftarrow \{1, 2, \dots, N\}$ and $B \leftarrow \{1, 2, \dots, M\}$.
- 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 that contains the indexed elements.

- *Random-mutation hill climbing (RMHC) (1994)*: From the hybrid family (editing and condensing), RMHC [22] achieved an excellent trade-off between reduction and classifier success in the experiments. It is one of the beautifully elegant random, criterion-driven methods. (Score!) While the original algorithm's search space is binary, for the 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 one (subset S).

- *Relative neighborhood-graph editing (RNGE) (1997)*: RNGE is an editing-prototype-selection algorithm [35] classed as the best in its group [18]. The 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 $r \in X$ that is closer to p and q than they are to each other. The editing algorithm works by building the RNG of X and removing all points that are misclassified by their immediate neighbors (Algorithm 2).
- *Relative nearest neighbor (RNN) (1972)*: The RNN rule [36] was singled out as one of the best two methods in the condensing group [18]. The RNN starts with

Algorithm 4. The GA prototype-selection algorithm.

```
Input: $X$, $M$, $K$, $W$
Output: Reference set $S$, $S \subset X$, $|S| = M$.

1  for $i = 1:K$ do
2      $P(i) \leftarrow choose(A, M)$ // random chromosome
3      $f_p(i) \leftarrow 1 - e(X(P(i)), X)$ // population fitness
4  for $gen = 1:W$ do
5      $O \leftarrow \emptyset$. // offspring set
6      for $par = 1:K/2$ do
7          $p_1 \leftarrow choose(P, 1)$ // parent 1
8          $p_2 \leftarrow choose(P, 1)$ // parent 2
9          $k \leftarrow choose(B, 1)$ // crossover point
10         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.)
11         $O \leftarrow O \cup \{o_1, o_2\}$
12     for $j = 1:K$ do
13         $C \leftarrow O(j)$
14         $m \leftarrow choose(B, 1)$ // index to mutate
15         $C(k) \leftarrow choose(A \setminus C, 1)$ // replace
16         $f_o(j) \leftarrow 1 - e(X(C), X)$ // offspring fitness
17         $O(j) \leftarrow C$
18         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 respec-
           tive fitnesses as the new $f_p$.
19 Return $S = X(P(1))$ // the best chromosome
```

a consistent reference set S (zero errors of 1-NN on X) and further reduces it by removing one element at a time and checking whether the set is still consistent. If not, the element is returned to the set. If the set is consistent, the element is permanently removed. The process continues until all the elements have been checked, and there has been no change to S (Algorithm 3).

Did you notice? All three winning algorithms are old and simple. That's my kind of algorithm. We add to this collection our own baselines and competitors, as explained next.

- *Hart (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.
- *Wilson (1972)*: Wilson's algorithm [13] is the forefather of the editing branch of prototype-selection. It marks for deletion all objects of X that are misclassified by their k nearest neighbors (typically, $k = 3$). Then, the marked objects are removed, and the remaining set is returned as S.
- *MC1 (1994)*: The MC1 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)*: GAs are a perfect fit for prototype selection [20], [32]–[34], [37]. The chromosome can encode $S \subseteq X$ storing zero at position i if the ith element of X is not in S, and one, otherwise. The GA version that we used here is shown in Algorithm 4. It enables

Table 1. The results from the experiment with noise-free George data.

Method	Type	Error Rate (%)	Number of Prototypes	Time (s)
1-NN	—	6.68	1,000	0.18
Hart	C	7.9	211	24.93
Wilson	E	7.1	913	0.5
Wilson + Hart	H	8.44	101	22.71
RNGE	E	6.59	921	3.92
RNN	C	8.2	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	78.8
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**	77.2
GA	H	**6.52**	**98**	84.71
GA	H	6.96	195	82.49

Boldface indicates that the result is in the Pareto front in terms of error-rate/reference-set size. C: condensing; E: editing; H: hybrid.

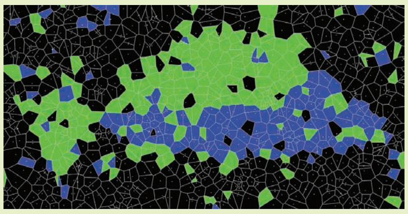

Figure 2. The classification regions of the 1-NN with 10% label-noise contamination; the error rate is 18.49% (or a nice pajama pattern).

prespecifying the number of prototypes. However, due to the crossover, 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 MATLAB code for this illustration are available at https://github.com/LucyKuncheva/instance_selection.

Experimental Setup

We can hardly glorify this little illustration by labeling it as an experiment, but we still need to explain how we made the comparisons as fair as possible. We sampled the George data [Figure 1 (b)] from the full set [Figure 1 (a)]. The prototype-selection methods that we used are listed in Table 1 with the results. In addition to the techniques listed in the previous section, we included Wilson's method followed by Hart's. This combined approach (hybrid type) often leads to a small and accurate reference set.

We took care that all our random methods carried out exactly the same number of evaluations of the criterion function (a 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: $W = 300$
- *RMHC*: number of chromosomes evolved (separately): $K = 40$; number of mutations: $W = 300$.

During 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 2 shows the classification regions of the 1-NN with the contaminated set. George looks exploded here.

Table 2. The results from the experiment with noisy George data.

Method	Type	Error Rate (%)	Number of Prototypes	Time (s)
1-NN	—	16.24	1,000	0.49
Hart	C	20	415	27.87
Wilson	E	8.43	806	0.48
Wilson and Hart	H	**9.68**	**91**	19.3
RNGE	E	**8.17**	**794**	3.97
RNN	C	21.18	355	33.94
RMHC	H	24.7	10	76.7
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	76.5
MC1	H	14.28	200	79.76
GA	H	**15.85**	**10**	75.64
GA	H	**11.08**	**20**	77.3
GA	H	9.92	100	80.82
GA	H	12.7	197	83.03

Boldface indicates that the result is in the Pareto front in terms of error-rate/reference-set size.

Results

Table 1 presents the results with the clean data, and Table 2 gives 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 outcomes are shown in Figure 3 for the noise-free George and in Figure 4 for the noisy George.

In both figures, each prototype-selection method is shown with a yellow marker. Circles represent hybrid methods, triangles indicate condensing, and squares signal editing. The thick blue line is the Pareto front; that is, the collection of nondominated methods, which 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 numbers of prototypes are shown as line graphs. In addition, next to each

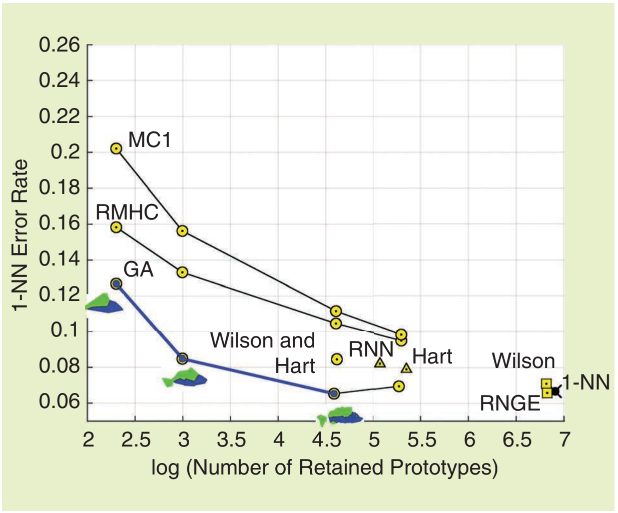

Figure 3. The scatterplot of the results for the noise-free George data. The blue line represents the Pareto front.

method on the Pareto front, we show the cute little portrait of George (the classification regions, leaving the background white).

We were hoping to pull a rabbit out of the hat; that is, identify niches that had not been explored and propose alternative versions of the prototype classifier.

Discussion

What is trivial is trivial: When there is noise in the data, all points are higher up, indicating greater error. Without noise, the editing methods (RNN and Hart) 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 generalization error shoots up (top-right corner of Figure 4). The editing competitors (Wilson and RNGE) are unfazed by noise. They consistently return good but large reference sets. They filter the type of random noise quite well, and the RNGE found its place in the Pareto front for the noisy George, beating Wilson by a whisker. The clear winners are the hybrid methods, a fact that 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 happened with Jim's and my MC1 and GA? In our 1998 paper [20], we found that random, criterion-driven methods, such as the MC1 and GA, were simple and effective, something that was also mentioned as a surprising observation by Skalak [22] in relation to the RMHC. In a later paper [21], however, we could not confirm this result. My implementation (I will blame that) kicked the GA toward the bottom of the league table. The difference from Algorithm 4 here is that, previously, we used a criterion that sought a compromise between the 1-NN error and number of prototypes in the form of a weighted sum. Here, we specify a limit on the number of prototypes. The GA turned out to be the best among the competitors, which were chosen from the most successful prototype-selection methods [18].

It may be a fluke, but our experiment with the GA (and George) showed that this strategy can handle noise. However, in both experiments, the generalization 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 the Wilson + Hart, another classical hybrid prototype-selection method. The random search (MC1) did not work well here, nor did the 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 Gaussians.

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, since 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; the list is as long as this magazine has pages. But the moral of the story is that if we need a very small subset of the data with an acceptable error rate, we may have to resort to those random, criterion-driven approaches that seem to offer a good compromise between the cardinality of the reference set and the 1-NN error rate. Long live random search!

Where next? Instance selection from big, semi-supervised, streaming, nonstationary, and non-independent and identically distributed data. Instance selection could be invaluable in that area if we find a smart and successful way of addressing these challenges.

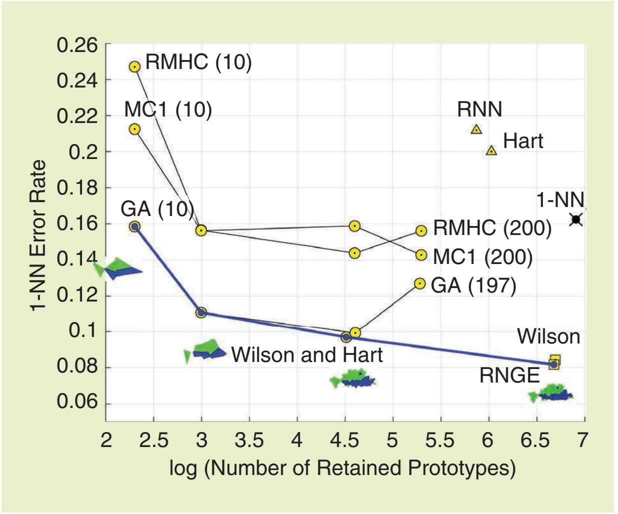

Figure 4. The scatterplot of the results for the noisy George data. The blue line represents the Pareto front.

Conclusion

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

About the Author

Ludmila I. Kuncheva (mas00a@bangor.ac.uk) is with the School of Computer Science and Electronic Engineering, Bangor University, United Kingdom.

References

[1] J. C. Bezdek, *Pattern Recognition With Fuzzy Objective Function Algorithms*. New York: 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, TX, Tech. Rep. 4., 1952.

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

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

[5] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001. doi: 10.1023/A:1010933404324.

[6] C. Cortes and V. N. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995. doi: 10.1007/BF00994018.

[7] J. J. Rodríguez, L. I. Kuncheva, and C. J. Alonso, "Rotation forest: A new classifier ensemble method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 10, pp. 1619–1630, Oct. 2006. doi: 10.1109/TPAMI.2006.211.

[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? 2018. [Online]. Available: https://arXiv:abs/1809.06705

[9] X. Wu et al., "Top 10 algorithms in data mining," *Knowl. Inform. Syst.*, vol. 14, no. 1, pp. 1–37, 2008. doi: 10.1007/s10115-007-0114-2.

[10] L. Kuncheva and J. Bezdek, "An integrated framework for generalized nearest prototype classifier design," *Int. J. Uncertain. Fuzz. Knowl.-Based Syst.*, vol. 6, no. 5, pp. 437–457, 1998. doi: 10.1142/S0218488598000355.

[11] L. I. Kuncheva and J. C. Bezdek, "Presupervised and postsupervised prototype classifier design," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1142–1152, 1999. doi: 10.1109/72.788653.

[12] P. E. Hart, "The condensed nearest neighbor rule (Corresp.)," *IEEE Trans. Inf. Theory*, vol. 14, no. 3, pp. 515–516, 1968. doi: 10.1109/TIT.1968.1054155.

[13] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," vol. SMC-2, no. 3, pp. 408–421, 1972. doi: 10.1109/TSMC.1972.4309137.

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

[15] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Mach. Learn.*, vol. 38, no. 3, pp. 257–286, 2000. doi: 10.1023/A:1007626913721.

[16] H. Brighton and C. Mellish, "Advances in instance selection for instance-based learning algorithms," *Data Min. Knowl. Discov.*, vol. 6, no. 2, pp. 153–172, 2002. doi: 10.1023/A:1014043630878.

[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 Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 417–435, Mar. 2012. doi: 10.1109/TPAMI.2011.142.

[19] I. Triguero, J. Derrac, S. Garcia, and F. Herrera, "A taxonomy and experimental study on prototype generation for nearest neighbor classification," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 1, pp. 86–100, Jan. 2012. doi: 10.1109/TSMCC.2010.2103939.

[20] L. I. Kuncheva and J. C. Bezdek, "On prototype selection: Genetic algorithms or random search?" *IEEE Trans. Syst., Man Cybern.*, vol. 28, no. 1, pp. 160–164, 1998. doi: 10.1109/5326.661099.

[21] J. Bezdek and L. Kuncheva, "Nearest prototype classifier designs: An experimental study," *Int. J. Intell. Syst.*, vol. 16, no. 12, pp. 1445–1473, 2001. doi: 10.1002/int.1068.

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

[23] R. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugen.*, vol. 7, no. 2, pp. 179–188, 1936. doi: 10.1111/j.1469-1809.1936.tb02137.x.

[24] E. Anderson, "The irises of the Gaspe Peninsula," *Bull. Amer. Iris Soc.*, 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 Trans. Fuzzy Syst.*, vol. 7, no. 3, pp. 368–369, 1999. doi: 10.1109/91.771092.

[26] J. R. Cano, F. Herrera, and M. Lozano, "Stratification for scaling up evolutionary prototype selection," *Pattern Recog. Lett.*, vol. 26, no. 7, pp. 953–963, 2005. doi: 10.1016/j.patrec.2004.09.043.

[27] A. de Haro-García and N. García-Pedrajas, "A divide-and-conquer recursive approach for scaling up instance selection algorithms," *Data Min. Knowl. Discov.*, vol. 18, no. 3, pp. 392–418, 2009. doi: 10.1007/s10618-008-0121-2.

[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, Feb. 2015. doi: 10.1016/j.neucom.2014.04.078.

[29] L. I. Kuncheva and I. A. D. Gunn, "A concept-drift perspective on prototype selection and generation," in *Proc. Int. Joint Conf. Neural Networks (IJCNN)*, Vancouver, BC, 2016, pp. 16–23. doi: 10.1109/IJCNN.2016.7727175.

[30] C. Alippi, G. Boracchi, and M. Roveri, "A just-in-time adaptive classification system based on the intersection of confidence intervals rule," *Neural Netw.*, vol. 24, no. 8, pp. 791–800, 2011. doi: 10.1016/j.neunet.2011.05.012.

[31] N. Lu, J. Lu, G. Zhang, and R. L. de Mantaras, "A concept drift-tolerant case-base editing technique," *Artif. Intell.*, vol. 230, pp. 108–133, Jan. 2016. doi: 10.1016/j.artint.2015.09.009.

[32] J. R. Cano, F. Herrera, and M. Lozano, "Using evolutionary algorithms as instance selection for data reduction in KDD: An experimental study," *IEEE Trans. Evol. Comput.*, vol. 7, no. 6, pp. 561–575, 2003. doi: 10.1109/TEVC.2003.819265.

[33] N. García-Pedrajas, J. A. Romero del Castillo, and D. Ortiz-Boyer, "A cooperative coevolutionary algorithm for instance selection for instance-based learning," *Mach. Learn.*, vol. 78, no. 3, pp. 381–420, 2010. doi: 10.1007/s10994-009-5161-3.

[34] N. García-Pedrajas, A. de Haro-García, and J. Pérez-Rodríguez, "A scalable approach to simultaneous evolutionary instance and feature selection," *Inf. Sci.*, vol. 228, pp. 150–174, 2013. doi: 10.1016/j.ins.2012.10.006.

[35] J. S. Sánchez, F. Pla, and F. J. Ferri, "Prototype selection for the nearest neighbour rule through proximity graphs," *Pattern Recog. Lett.*, vol. 18, no. 6, pp. 507–513, 1997. doi: 10.1016/S0167-8655(97)00035-4.

[36] G. W. Gates, "The reduced nearest neighbor rule (Corresp.)," *IEEE Trans. Inform. Theory*, vol. 18, no. 3, 1972. doi: 10.1109/TIT.1972.1054809.

[37] L. I. Kuncheva, "Editing for the k-nearest neighbors rule by a genetic algorithm," *Pattern Recog. Lett.*, vol. 16, no. 8, pp. 809–814, 1995. doi: 10.1016/0167-8655(95)00047-K.

SMC