

A deterministic biologically plausible classifier

T. Viéville* and Sylvie Crahay, INRIA, Sophia, France

February 19, 2003

In preparation for Journal of Computational Neuroscience

Abstract

Regarding biological visual classification, recent series of experiments have enlighten that data classification can be realized in the human visual cortex with latencies of about 100 ms, which, considering the visual pathways latencies, is only compatible with a very specific processing architecture, described by the so-called Thorpe model.

Surprisingly enough, this experimental evidence is in coherence with algorithms derived from the statistical learning theory, following the work of Vapnik. More precisely, there is a double link: on one hand, the Vapnik theory offers tools to evaluate and analyze the Thorpe model performances and on the other hand, this model is an interesting front-end for algorithms derived from the Vapnik theory.

The present contribution develops this idea and experiments its performances using a tiny sign language recognition experiment.

Key-words: Neuronal classifier, Supervised learning, Vapnik dimension, Biological model.

Foreword: This paper is a “long version” with many footnotes in order to help the reviewers either from life-science or from computer-science to work with a self-contents document. The final draft will be enlighten.

1 Introduction: biological classification is a fact

Biological visual classification is a well-known and very common, but still intriguing fact. As illustrated in Fig. 1, an “object” is recognized in very extreme situations: here, even if there is no explicit visual cues (edge, texture, etc..). For subjects not aware of this picture, the recognition usually takes about a second. A step further, it appears that this classification is made “before” any visual bottom-up analysis: here, we seem to recognize the dog itself *before* being able to perceive some of the dog silhouettes edge or texture. In other words, data classification seems to *lead to* visual analysis, contrary to earlier views of visual data processing [39]. In this case, it is clear that a bottom-up processing, with extraction of some “symbolic” information is not likely to be the front-end of this data classification mechanism.

In order to avoid any ambiguity, let us state that, in the present work, by data classification, we simply means being able to put a *unique label* on a given data input (e.g. “oh, there is a dog”). This differs from *categorization* (e.g. [1]) where not only a *label* but a more complex “semantic structure” is extracted from a given data input. The fact this label is *unique* is illustrated in Fig. 2, where an ambiguous picture is presented, with two possible classifications. At a given time, only one alternative is perceived.

The ability to group stimuli into such categories is a fundamental well-established cortical cognitive process (e.g [24]).

*<http://www.inria.fr/Thierry.Vieville>



Figure 1: The recognition of the dog in this picture (image devised by R.C. James) seems not obtained from visual cues analysis. The dog seems in fact recognized “before” its silhouette or shape is perceived. This picture is well known because computer vision scientists confessed not being able to analyze it [39].



Figure 2: An example of “multi-stability” picture: the young women’s chin is the old woman’s nose. (originally published by W.E. Hill in 1915 as “My wife and my mother-in-law”). The very well known fact, here, is that, at a given time, *either* the young *xor* the old woman is perceived, but neither both of them, nor any intermediate shape.

Recent series of experiments have enlighten this biological mechanism: data classification can be realized in the human visual cortex with latencies of about 100 ms [49] which, considering the visual pathways latencies [41], may only be compatible with a very specific processing architecture and mechanism, the so-called *Thorpe model*. Even “high level” visual data classification such as face recognition [17] as pointed out recently by [47] can be realized at such a very fast rate.

We consider this result as crucial because it is an experimental evidence that data classification *leads to* visual bottom-up analysis and not the other way round, for the simple reason that there is enough time only to do it in the former way. This may be viewed as a truism, but when considering many data classification biological models (see e.g. [33] for a review) or statistical artificial algorithms (see [36] for an overview) it appears that it is not the case.

Surprisingly enough, this experimental evidence is in coherence with some algorithms derived from the statistical learning theory, following the work of Vapnik, because the architecture of the derived algorithms have the same architecture as what has been proposed by [48]. More precisely, there is a double link: on one hand, the Vapnik theory offers tools to evaluate and analyze the Thorpe model and on the other hand, the Thorpe model is an interesting front-end for algorithms derived from the Vapnik theory.

The goal of this work is to develop this double link.

In a 1st section we review the Thorpe model and describe its computational properties. In the next section we revisit the well known nearest-neighbor classifiers because this may be one of the simplest way to explain the concepts of the Vapnik theory we have to considered here. Applying this piece of theory, we propose in the third section a notion of “optimized nearest-neighbor classifier” which: (i) statistical dimension is minimal and which (ii) seems to be the best computational representation of the Thorpe model. Experimentation of this mechanism is reported in the last section.

2 Computational properties of the Thorpe model

As pointed out in the introduction, data classification can be realized in the human visual cortex with latencies of about 100 ms [49] . Indeed, not all visual information can be analyzed on the basis of this first wave of information processing (e.g. facial expression or identity, or complex stimulus recognition as proposed in Fig. 2 are only available later) but object “labelization” is already performed when such a cognition occurs.

More precisely, such latencies [41] are only compatible with a specific feed-forward and straight-forward neuronal architecture [48], as schematized in Fig. 3, with two pathways: (fast) *data classification* and *category learning*.

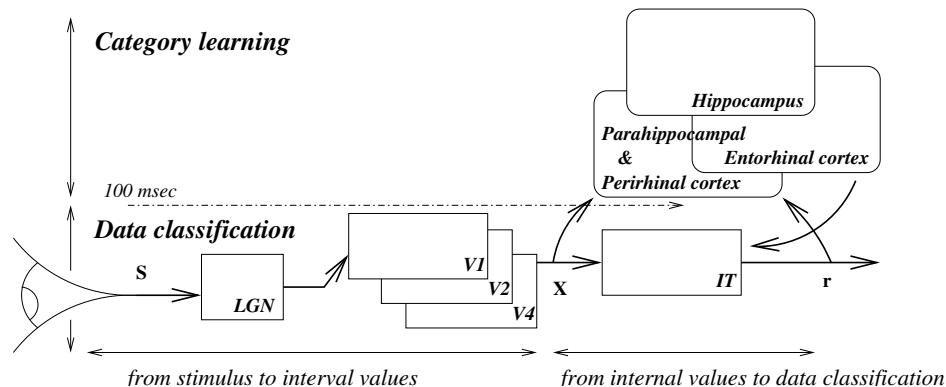


Figure 3: An abstract simplified view of the Thorpe model and its potential relations with hippocampal areas, see text for details.

Data classification: from input to “grand-ma” neurons

The underlying architecture of the *data classification* pathway of the Thorpe model may be decomposed in two blocks:

1. from the stimulus input s a very high dimensional array of “internal values” values $\mathbf{x} = \phi(s)$ is computed;
2. from a subset of this vector of values \mathbf{x} the data classification, i.e. the detection of a “label” $r = \mathcal{C}(\mathbf{x})$, is performed in “one step”.

From stimulus input to internal values.

The complex cellular characteristics of the parietal and ventral¹ division of the visual system make it especially suitable to provide various combinations of the data input. Such combinations

¹About the “ventral” visual pathway. Prior to the inferior temporal cortical area, is the so called parietal/ventral pathway (sometimes improperly called “parvocellular” pathway), neurons in the inter-blobs of V1

should help for data classification, as discussed in the sequel.

In the case of fast data classification, the magnocellular² pathway of the parietal/ventral division of the visual system is involved, as asserted by [16] where it is demonstrated that rapid categorization of natural scenes is color blind.

Here we consider the detection of static visual information, while data classification of dynamic visual events involves other cortical tracks³ where motion detection occurs⁴.

From internal values to data classification.

A step further, neurons found in the inferior temporal (IT) cortex respond to very complex stimulus features (e.g. [8]), regardless of size or position on the retina. For instance, some neurons in this region respond selectively to faces of particular overall feature characteristics. Damage in this area induce disorders⁵ of object recognition. There are many neuro-physiological evidences (e.g. [20, 43]) about the fact that the visual temporal areas⁶ function is related to data classification.

Analyzing the neuronal data classification process.

Regarding fast data classification, the Thorpe model [47] is based on the following property:

- the very short observed latencies are only compatible with an information flow related to the emission of the first spikes (when not a unique spike by neuron) of each neuronal layer.

This is not only due to process short latencies. As developed in [47], this assumption is based on the experimental evidence that, most of the time, a second spike arriving via a synapse

project to the pale stripes of V2. This pale stripes of V2 project to the inferior temporal cortex. Other feed-forward pathways include the V4 visual area, see [6] for general review. This pathway is composed of feature detectors (simple, complex and hyper-complex cells) (e.g. [35] for an introduction). Neurons in this pathway show a low sensitivity to contrast, high spatial resolution, and low temporal resolution or sustained responses to visual stimuli. See for instance [20], Chap 2 for a discussion.

²*About (magno/parvo)cellular streams.* There are two classes of cells from the retina and LGN: magnocellular, and parvocellular. These two cell types are contained in different parts of the LGN, and they have different response properties: (i) magnocellular cell receptive fields are 2-3 times larger than parvocellular cell receptive, fields parvocellular have better acuity, resolution magnocellular have better sensitivity, magnocellular cells respond well to moving stimuli, whereas parvocellular cells do not parvocellular cells respond well to color stimuli, whereas magnocellular cells do not.

The magnocellular pathway (older than the parvocellular in phylogenetics) continues the processing of visual detail leading to the perception of shape in area V3 and movement in areas V5 and MST. It has less synaptic relays than the parvocellular pathway, but is faster.

³*About the “dorsal visual pathway.* Here a second visual cortical pathway, parietal and dorsal, in which the neurons in layer 4B of V1 project to the thick stripes of V2 is considered. Area V2 then projects to V3, V5 (or MT, middle-temporal cortex), and MST (median superior temporal cortex). This pathway is mainly an extension of the magnocellular pathway, but not only.

⁴*About V5.* Simplifying the situation: cells in V5 are particularly sensitive to small moving objects or the moving edge of large objects; cells in dorsal MST respond to the movement of large scenes such as is caused with head movements; cells in ventral MST respond to the movement of small objects against their background. See for instance [20], Chap 10 for a discussion.

⁵Common examples of such disorders include visual agnosia, or the inability to identify objects in the visual world, and prosopagnosia, a subtype of visual agnosia that affects specifically the recognition of once familiar faces.

⁶*About IT.* The inferior temporal cortex is thought to consist of three parts: The TEO (the occipital division of the intra-temporal cortex), the TE (the median division), and the STS (superior temporal sulcus). The TEO is used for making discriminations between 2-D patterns which differ in form, color, size, orientation, or brightness. The TE is used for recognition of 3-D objects. Both the TE and STS are thought to be used in facial recognition and in the recognition of familiar objects. The STS may be the place in which the feature maps of objects (which contain separate information about each primitive of an object, such as color, orientation, or form) become object files.

is significantly attenuated, when not inhibited, for 50-100 ms (e.g. for some thalamic excitatory inputs to cortical pyramidal cells [45]).

This has two consequences:

1. the transmitted information is much more related to the *occurrence* of a neuronal signal than to the *quantitative* value transmitted (e.g. via the spike frequency, see [27] for a development about rate versus temporal order coding),
2. since the latency of the first spike of a given neuron is a direct decreasing function of the neuron input value (a quantitative model is proposed in [27]), only neurons with the *highest values* generate spikes fast enough to be taken into account.

A step further,

- (i) let us consider that the temporal discrimination of a neuron (see e.g. [9] for an extensive study) is of about $\tau = 1 \text{ ms}$, considering that two spikes arriving within the same 1 ms are viewed as simultaneous input by the next neural input and
- (ii) let us assume that at each stage, the neural inputs received during a temporal window of $T = 10..20 \text{ ms}$ (e.g. [47]) is taken into account, as schematized in Fig. 4

With these assumption, the temporal discrimination is simply made during $n = T/t$ consecutive intervals⁷ (like when building a temporal “histogram” of the spike occurrences) and we easily compute the number of possible combinations, i.e. $C = [T/\tau]^N$, where N is the number of neural inputs, active during the 1st T ms.

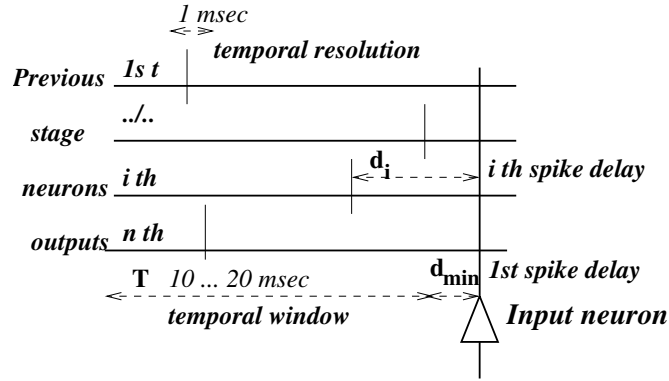


Figure 4: Schematic representation of the spike temporal combinations: only the spikes received during a temporal window, i.e. during the first $T = 10..20 \text{ ms}$ are taken into account. Spikes within a temporal resolution of about 1 ms are considered as simultaneous (in this example the 1st spikes of the 1st an n th neurons). The 1st spike neuronal delay is considered as the most relevant neuron information, in this “fast brain” neuronal behavior. Furthermore, neurons outputs are treated by the input neuron using a progressive inhibition mechanism, as detailed in the text.

⁷ *About the measurable number of permutation.* There are $N!$ possible permutations for a given set of N data. However, in practice, at a given temporal resolution τ and during a time window T it is not possible to observe all these permutations. What happens is that when, at a given time t , an input i has been detected, all inputs in the $I_t = [t..t + \tau]$ interval are viewed as “synchronous with i ”. This means that the I_t interval is not fixed but triggered by the 1st occurring input. However, we are in a situation where N is very large, so that at the end of each I_t interval yet another spike very likely (almost) immediately occurs, starting a new $I_{t+\tau}$ interval and finally allowing to consider consecutive intervals of duration τ .

Yet another step further, the Thorpe model assumes a progressive inhibition mechanism⁸, the 1st spikes increasing the inhibition for other input spikes emitted by other neurons. This shunting inhibition mechanism (see e.g. [15] for a biologically plausible simulation) may be related to fast-spiking inhibitory inter-neurons in contact with the soma of pyramidal cells (e.g. [4]). As a consequence, not all neuronal inputs but only the firsts, are taken into account.

These facts have two consequences, regarding the computational process:

Quantification of the neuronal information: the quantitative value measured from a given neuron by the next neuronal input is thus directly related to its delay d as schematized in Fig. 4. As discussed previously, this delay is thus a bounded value (with *min* and *max* values) with a *finite precision*.

The ratio between these two quantities, i.e. $T/\tau \simeq 10 - 20$ is the *number of steps* of the data value. This number will be precious in order to evaluate the statistical performances of the model.

Sparseness of the neuronal information: among the rather huge number of input neurons (typically the dimension n of the neuronal “vector” has an order of magnitude of 10^5) only a rather small number is taken into account. This is a key point in the Thorpe model and it has been experimented that a simple “threshold” mechanism (canceling information small enough to have a delay higher than the time window) is already a plausible model of the neuronal mechanism.

In spike-net (the computer implementation of the Thorpe model, e.g. [14]), the *number of degrees of freedom* is of about 10^2 , it is expected to be higher in the brain, around 10^3 as estimated in [47]: it is bounded by both the short time window and by the progressive inhibition mechanism.

Here we consider that (a) during the *learning phase*, for a given set of prototypes, i.e. for a given “labelization”, a subset of significant components, has been selected, defining the number of degrees of freedom, while during the *classification phase* only these components are considered.

This differs from (b) thresholding the highest values. In this second case a variable number of degrees of freedom is taken into account.

Solution (a) indeed reduces the number of degrees of freedom, solution (b) simply reduces the quantification steps, as discussed previously.

This notion of *sparse classifiers*⁹ [21, 22] has been developed in computer data classification, designing an approach where non-mandatory components of the input data are

⁸*Quantification of progressive inhibition mechanisms.* It has been quantified considering the previous stage neuronal inputs $(\dots \nu_i, \dots)$ in the order of their occurrence weighted by “inhibition coefficients” (ω_i, \dots) and limited by a threshold θ , obtaining for the neuron output σ an expression of the form :

$$\sigma = Sg \left(\sum_i \alpha^i \omega_i \nu_i - \theta \right)$$

$0 < \alpha < 1$ is the decreasing ratio (typically around $1/2$ for simulations)

while $Sg(u) = \text{if } u < 0 \text{ then } -1 \text{ else if } u > 0 \text{ then } 1 \text{ else } 0$ is the sign function.

⁹*About sparse classifiers.* It is clear that not all n components of the input data are likely relevant for a given classification process. It is thus suggested to “forget about” non-mandatory coefficients (i.e. set them to zero) [21, 22] in order to reduce the number of degrees of freedom of the estimation process. This approach is formalized as the minimization a \mathcal{L}^1 criterion (i.e. minimizing the absolute value of the classifier parameters) because it is well known, as extensively used in the Lasso estimation method [50] that minimizing such a norm allow variables to vanish. This is simply due to the fact with these piece-wise linear constraints, finding the minimum of such a criterion corresponds to a linear-programming problem which minimum is on the vertices, defined by the fact that some of the absolute values vanish (e.g. [28]). Although experimental performances

canceled. In other words, a very small dimensional linear sub-space of the input data space (defined by equations stating that all other components are equal to zero) is considered.

Discussing the “grand-ma” neurons behavior.

As reviewed here⁶, the neuronal outputs of the TE and STS transform feature’s maps of objects into object files (e.g. [24]). With this kind of behavior, such neurons are sometimes called “banana” or “grand-ma” neurons [20] because they selectively respond during object recognition.

This must be understood with the following properties:

- there is not a unique “grand-ma” neuron for a given “grand-ma”: indeed, several neurons code different configurations of a given object, for instance depending on its visual aspect (e.g. its 3D orientation). In other words, we must have several *prototypes* for a given object;
- we must have not only “grand-ma” neurons but also “no-grand-ma” neurons: indeed, the detection of the occurrence of an object must be learned not only with examples where the object is present, but also with examples where the object is absent, in order to *discriminate* between these two situations. As a consequence the “no-object” case must be also coded and the object recognition is a relative comparison of both cases;
- “grand-ma” neurons also partially respond to objects which are not “grand-ma”: since we may have coded examples where the object occurs with other objects, i.e. we may have coded “grand-ma with a banana”, “grand-ma with john-smith” etc.. in order to be able to detect “grand-ma” in several situations, it is clear that these neurons also respond to other objects. In such a case post-processing¹⁰ is required.

This has been discussed in recent studies about the IT neuronal behavior (e.g. [43] Chap 8.)

In spike-net (the computer implementation of the Thorpe model, e.g. [14]) such behavior is simulated by a “nearest-neighbor” mechanism: i.e. the object which *proximity to a prototype* is maximal is selected. The underlying “distance” is a general semi-distance because it is based on a finite precision quantification and sparse representation, as discussed in this section.

With these considerations, the selection mechanism of the data category may be viewed as a simple nearest-neighbor mechanism.

seem rather good, there is no guaranty that this process indeed cancel a maximal number of redundant data, while the complexity of the minimization process is also not easily bounded.

¹⁰*From nearest-neighbor classification to categorization:* This description does not exclude the fact that “post-processing” mechanisms occur, the final *categorization* of a complex object being related to a combination of *labels* extracted by the *classification* mechanism. In order to discuss this aspect, let us consider “fancy” examples:

- (i) “centaurs” may be detected as something which proximities w.r.t. “horse” and “human” prototypes are equivalent; this is roughly equivalent to add a new prototype in between;
- (ii) frogs exist within two forms: tadpoles or frog itself, detecting the category frog requires to detect something which proximities w.r.t. tadpoles or to frog-it-self prototypes; this is not equivalent to add a new prototype, but to detect *either one or another* category;
- (iii) similarly, categories such as “fish” defined, says, as “animals with slippers and tail except dolphins or whale” .. require to post-process the result of different data-classification.

In other words, the prototype-proximity mechanism intrinsically define “conjunction” of prototypes but not “disjunction”; a category defined by a logical expression seems to be put in a disjunctive normal form in order to be implemented by combinations of such nearest-neighbor classifiers.

An extension, allowing to combine the influence of several prototypes is the notion of *K*-Nearest-Neighbor Estimation (see e.g. [19], Chap 4 for an introduction).

Category learning: the role of the hippocampus

It is commonly admitted (e.g. [20] Chap. 7 and 8) that the hippocampus plays a predominant role of memorization in data classification. More precisely, it acts as an episodic memory which stores the “raw data” until it has been integrated in long-term memory areas.

This integration is interpreted as the category learning process (e.g. [43], Chap. 6) i.e. the tuning of the different input contributions used to detect a given object.

The hippocampus receives its inputs via the parahippocampal gurus and the entorhinal cortex; efferent connections pass through the entorhinal cortex back to the visual association areas ([43], Chap 6). It receives input from virtually all association areas in the neocortex.

The modelization of this biological learning mechanisms in this structure is dominated by “Hebbian-like” classifiers (including the well-known Perceptron). As summarized in [29], this kind of mechanism may correspond either to long-term potentiation or long term depression synaptic mechanisms. It may also correspond to allosteric properties of cell bodies. This is the case for hippocampal cells [23]. The precise role of these pathways is however still only conjectured and not really well established [20].

The architecture of what is called Hebbian learning corresponds to a conjunction of pre and post “synaptic” influences which tends to modify the weights of the classification layer.

If \mathbf{x}_l is the pre-synaptic vectorial value of the data input and \mathbf{a}_h the post-synaptic vectorial representation of a given category, the linear Hebbian-like correction rule writes $\mathbf{a}_h + = \delta_{hl} \mathbf{x}_l$ for some small δ_{hl} (e.g. [20]). Here, we point out the fact that this adaptation rule is not restricted to standard neural nets à-la Hopfield. It is also used in other architectures: alternatives are based on competitive models (e.g. following the pioneer work of [37]) or “anti”-Hebbian (i.e. Hebbian rule with a negative sign in order to de-correlate signals) rules.

Unfortunately, as pointed out in [20] and confirmed by a recent discussion [29], such model still describes the “neuronal architecture of cortical maps in a rather crude way”.

Nevertheless, this is a standard point of view and as a first step, in the sequel, we are going to demonstrate that such a Hebbian-like rule is sufficient to tune and optimized sophisticated classifiers (i.e. related to the Vapnik theory), showing that the best data classifiers derived from the statistical learning theory are biologically plausible.

3 Revisiting deterministic artificial classifiers

Defining the problem

Here, the “data classification” formalization problem is addressed. Roughly speaking, such a mechanism allows to “label” quantitative data. This mechanism is typically used in object recognition paradigms. In such a case, data often corresponds to a set of “features” measured from inputs related to the observed object. See [46] for a recent comprehensive and introductory treatise on the subject and [40] for a formal introduction.

Given :

- a *data* vector $\mathbf{x} \in \mathcal{R}^n$
(i.e. data is represented by an array of numerical values) and
- *categories* $r \in \{1..R\}$
(i.e. the class or category is numbered from 1 to R)

a *classifier* \mathcal{C} is a function :

$$\mathcal{C} : \mathcal{R}^n \rightarrow \{1..R\}$$

which associates a category to each data.

For a given *input data* \mathbf{x} belonging to a category r we use the classifier to estimate this category, i.e. $r = \mathcal{C}(\mathbf{x})$.

Such a classifier is *trained* (i.e. calibrated) by a *calibration set* i.e. a set of M pairs $\{\dots, (\mathbf{x}_i, r_i), \dots\}$ if $\forall i, r_i = \mathcal{C}(\mathbf{x}_i)$. As such, the present paradigm corresponds to *supervised learning*.

In our context, this calibration set contains *typical features* allowing to characterize the categories. It is thus *exact* data. In other words the calibration set is not “sampled” but “chosen”. We refer to such a paradigm as *deterministic learning*. This differs from usual paradigms used in statistical learning [46], where a training set which may contain mistakes is randomly sampled.

Nearest-neighbor (NN) classifiers

If we consider the calibration samples as *prototypes*, it is a natural idea to say that a *data* belongs to a *category* iff it is “close” to one of the *prototype*.

For N prototypes, the classifier is thus formally¹¹ defined by:

$$\mathcal{C}(\mathbf{x}) = \arg \max_{r_i, i=1..N} c_i(\mathbf{x}) \quad (1)$$

where the “proximity” $c_i()$ to the i th sub-category related to the prototype of index i is to be defined now. Such a mechanism is illustrated in Fig. 5.

In order to develop our ideas, we consider, in the Euclidean space \mathcal{R}^n , a quadratic form $\|\mathbf{v}\|_{\Lambda}^2 = \mathbf{v}^T \mathbf{\Lambda} \mathbf{v}$ defined by a positive definite symmetric matrix $\mathbf{\Lambda}$.

This matrix $\mathbf{\Lambda}$ is “application dependent” in the sense that it weights the relative importance of each data component, including inter-relations between these components. It is thus provided as *a priori* information. This will be used in the sequel.

The present technical restriction to a constant Euclidean metric is only used to simplify our developments.

We also have in mind that some more general metric, considering not simply the Euclidean space \mathcal{R}^n but a general Riemannian manifold as data space. Although out of the scope of this paper, this is a direct extension of this work.

Combining these ideas, but in the simple “Euclidean case”, we consider, for some threshold θ_i , the following so called *minimal thresholded linear squared proximity to a prototype*, i.e. :

$$\begin{aligned} c_i(\mathbf{x}) &= -\|\mathbf{x} - \mathbf{x}_i\|_{\Lambda}^2 + \theta_i + \|\mathbf{x}\|_{\Lambda}^2 \\ &= -\|\mathbf{x}\|_{\Lambda}^2 + 2(\mathbf{x}_i^T \mathbf{\Lambda} \mathbf{x}) - \|\mathbf{x}_i\|_{\Lambda}^2 + \theta_i + \|\mathbf{x}\|_{\Lambda}^2 \\ &= 2\mathbf{x}_i^T \mathbf{\Lambda} \mathbf{x} - \|\mathbf{x}_i\|_{\Lambda}^2 + \theta_i \\ &= \mathbf{a}_i^T \mathbf{x} - b_i \text{ with } \begin{cases} b_i = \|\mathbf{x}_i\|_{\Lambda}^2 - \theta_i \\ \mathbf{a}_i = 2\mathbf{\Lambda} \mathbf{x}_i \end{cases} \end{aligned} \quad (2)$$

Here we:

- consider the opposite of the squared distance $\|\mathbf{x} - \mathbf{x}_i\|_{\Lambda}^2$, say the *proximity*, to the prototype \mathbf{x}_i for the chosen metric,

¹¹ *General categories and proximity definition.* Let us consider that each sub-category related to the prototype of index i corresponds to a “region” of the data space, defining a partition of this space. Let us define the border of each region using a general equation $c_i(\mathbf{x}) = 0$. This defines an hyper-surface which, according to the Jordan theorem, delimits what is inside (say when $c_i(\mathbf{x}) > 0$) and outside (when $c_i(\mathbf{x}) < 0$) this region. In this general context, equation (1) precisely determines the region of a given data. There is thus no loss of generality to define classification from (1).

- add the same quantity $\|\mathbf{x}\|_{\Lambda}^2$ to every $c_i(\mathbf{x})$ so that:
 - the comparison in (1) is not modified, while:
 - this allows to cancel quadratic terms in (2) and obtain a linear function,
- add a threshold θ_i in order to :
 - + control the relative influence of each prototype (the higher θ_i the higher the proximity to the i th prototype) and also to :
 - + obtain a one to one correspondence:

$$\{\mathbf{a}_i = 2 \Lambda \mathbf{x}_i, b_i = \|\mathbf{x}_i\|_{\Lambda}^2 - \theta_i\} \Leftrightarrow \{\mathbf{x}_i = \Lambda^{-1} \mathbf{a}_i/2, \theta_i = \|\mathbf{a}_i\|_{\Lambda^{-1}}^2/4 - b_i\}$$
 between the linear function and the prototype data and threshold.

As a consequence we thus obtain a linear classifier and frontiers between categories are represented by piece-wise planar hyper-surfaces, as illustrated in Fig. 5. Let us called such classifier a [thresholded] nearest-neighbor (NN) classifiers.

The biological plausibility of this mechanism reduces to the biological plausibility¹² of (1), since other operations are linear calculations. The biological plausibility of “maximum” operators has been extensively studied in [56]. Such architecture is known as “maxnet” or Hamming networks and weights of such initialized networks may be trained (e.g. [55]) to obtain a a correct data classification.

In the statistical interpretation of NN classifiers (e.g. [46]), under “reasonable” assumptions (i.e. normal distribution of the data in each category with similar covariances equal to Λ^{-1}), this corresponds to a Bayesian classifier, writing $\theta_i = 2 \log(p(r_i))$ where $p(r_i)$ is the a-priori probability for a given data to belong the i th sub-category.

Beside being conceptually extremely simple and also obvious to implement in practice, this well-known classifier [19] has reasonable performances. More precisely, the probability of error for the nearest neighbor rule (i.e. with $\theta_i = 0$ in (2)) given enough members in the training set is sufficiently close to the Bayes (optimal) probability of error. It has been shown [11] that as the size of the training set goes to infinity, the asymptotic nearest neighbor error is never two times worse than the Bayes (optimal) error.

However, calibration or training set sizes never go to infinity! The real problem is to understand the performances of the classifier for a *limited* calibration set.

Furthermore, the present approach does not provide, as it, any “modelization” of the calibration set. As a consequence, since no prediction/inference is possible with this method, the quality of the training is highly dependent upon the calibration set itself. It may not be very “accurate” with respect to data which are not calibration data, i.e. generalization performances are expected to be poor.

¹²*Biological plausibility of NN classifiers.* Several neuro-scientists consider [artificial] *neuronal networks* as plausible models of [biological] *networks of neurons* (e.g. [29] for a recent review). Following this track, let us consider “formal neurons” with inputs $(\dots \nu_i, \dots)$, output σ , weights $(\dots \omega_i, \dots)$ and threshold θ , defined by an expression of the form :

$$\sigma(\dots \nu_i, \dots) = Sg(\sum_i \omega_i \nu_i - \theta)$$

where $Sg(u) = \text{if } u < 0 \text{ then } -1 \text{ else if } u > 0 \text{ then } 1 \text{ else } 0$. This is the simplest model (see e.g. [55] for alternatives and [33] for a discussion about their plausibility).

With a few algebra, the arg-max operator in (1) is easily implemented using a “neuronal network” design, i.e.:

$$C(\mathbf{x}) = \sum_{r=1..N} r \sigma_r$$

$$\text{with } \sigma_r = Sg\left(\sum_{s=1..N, s \neq r} Sg(c_r(\mathbf{x}) - c_s(\mathbf{x})) - \theta_r\right) \text{ with } N-3 < \theta_r < N-1$$

the computational complexity being of $o(N^2)$ since comparisons $Sg(c_s(\mathbf{x}) - c_r(\mathbf{x}))$ are made for all $r \neq s$.

Here, as the reader may easily verify, $\sigma_r = 1$ iff the data belongs to the r th category else $\sigma_r = 0$. In other words the output is equivalently: (i) a number r , index of a category, (ii) a set of exclusive boolean signals σ_r , true for a given category.

Indeed, the point is not to claim that this *is* the way it is done in the brain, but simply to state that this computation is simple enough to be performed by “standard” biologically plausible computational elements (see [5] for a development).

The Vapnik theory [51] allows to formalize and analyze this idea in the case of a *binary* classification (i.e. when, in our case, there is only *two* prototypes). This has recently been generalized to multi-categories, as discussed now.

Training capability and learning performances

The Vapnik learning theory [51] allows to formalize the idea that efficient models have a limited complexity. As such, it is a formalization and in fact an improvement of the well-known Occam’s Razor principle¹³.

Let us review this piece of theory following the recent works of Guermeur [30, 31]. For a given classifier \mathcal{C} , it relates:

- the *expected risk* $R(\mathcal{C})$ (i.e. the “average” probability for the classifier to provide a wrong answer) for a set of inputs, randomly chosen according to an unknown probability distribution with
- the *empirical risk* $R_{emp}^M(\mathcal{C})$ (i.e. the “average” probability for the classifier to provide a wrong answer) for the calibration set of size M ;

this quantity being equal to zero for a deterministic classifier, as discussed¹⁴ previously.

More precisely, for a chosen probability δ , the *expected risk* can be bounded with a probability at least $1 - \delta$ as follows:

$$R(\mathcal{C}) \leq R_{emp}^M(\mathcal{C}) + \underbrace{\epsilon(M, \delta)}_{\text{bias}} \quad (3)$$

“guaranteed risk”

where the *bias* (also called confidence bound) $\epsilon(M, \delta)$ is a function of the chosen probability δ , the calibration set size M and the *class of the classifiers*, i.e. the set of classifiers which is used during the learning phase.

Indeed, we expect this bias to decrease with the calibration set size M . The learning mechanism is *consistent* iff $\lim_{M \rightarrow \infty} \epsilon(M, \delta) = 0$. Better than that, if the classifier functions are bounded, at the convergence, the smallest/optimal value of the expected risk [51] is obtained.

It appears that if the classifiers class is too large, the process is *not* consistent: with a very large class of classifiers, we can classify everything, but what does everything does anything. Here, inconsistency means that the bound does not decrease to zero when M increases.

Binary Vapnik-Chervonenkis classifiers.

For the case of binary classifiers which *shatters* the data into two categories (i.e. performs a dichotomy), following the presentation of [46], we can quantify this number of dichotomies by

¹³*The Ockham razor.* William of Ockham (may be the most influential philosopher of the 14th century) stated: *one should not increase, beyond what is necessary, the number of entities required to explain anything* (see for instance <http://pespmc1.vub.ac.be/OCCAMRAZ.html> for details).

¹⁴*Deterministic selection of calibration sample.* In this statistical approach, it is assumed that the calibration samples are chosen by M independent draws from the same probability distribution as the other inputs. In the present deterministic paradigm, we however assume that calibration samples are chosen by an “expert”. On one hand, this means that it is a “very lucky” set of draws, without mistake. On the other hand this means that the expert randomly choose a “representative” set of draws, which is not necessarily the best strategy: for instance a “discriminative” set of draws (in which examples close to the limit between two categories are chosen in order to help building this border, or in which “exceptional examples” are highlighted because not easily detected otherwise) is a interesting alternative. A perspective of the present work is thus to extend this formulation.

a *shatter* coefficient σ_M which is the maximal number of dichotomies of M points that can be formed by 2-categories classifiers in the considered class. Obviously $\sigma_M \leq 2^M$.

A class of classifiers which indeed can realize any shattering does not model anything. On the contrary, a classifier in a class for which the amount of shattering is limited will indeed induce *choices / decisions*, thus introduce some additional knowledge in the process.

Formalizing this idea, from this shatter coefficient, Vapnik defines *the V_c dimension as the largest integer M for which the maximal number of dichotomies σ_M of M points that can be formed by 2-categories classifiers is not bounded, i.e. $\sigma_M = 2^M$ and by convention, $V_c = +\infty$ if $\forall M, \sigma_M = 2^M$.*

These two definitions are indeed related since [51]:

$$\begin{aligned} \sigma_M &\leq M^{V_c} + 1 & \text{if } V_c < +\infty \\ \sigma_M &= 2^M & \text{if } V_c = +\infty \end{aligned}$$

and they allow to derive, for classifiers with a finite V_c , a bound for the previous bias, i.e.

$$\epsilon(M, \delta) \leq \sqrt{\frac{32}{M} (\log(8 \sigma_M) - \log(\delta))} \leq \sqrt{\frac{32}{M} (V_c \log(9 M) - \log(\delta))}$$

(obtained from [46] with a few algebra. Similar bounds, usually a bit better but more complicated to write, are available (e.g. [51, 31]).

Vapnik-Chervonenkis classifiers have a finite V_c dimension and are thus consistent. This result has been completed by several authors: for instance [18] shows that *classifiers with small “complexity” (i.e. the V_c dimension) have better generalization performances*.

For binary linear classifiers (i.e. a NN classifier with two prototypes) the V_c dimension is bounded by the number of degrees of freedom¹⁵ of the classifier parameters.

If the V_c dimension is finite but very large, the number of training samples must be even larger to correctly bound the classifier bias. The following numeric order of magnitude is easily derived from the previous formula:

	$V_c = 10$	$V_c = 100$	$V_c = 1000$
$\delta \simeq \epsilon \simeq 10^{-1}$	$M \simeq 10^6$	$M \simeq 10^7$	$M \simeq 10^8$
$\delta \simeq \epsilon \simeq 10^{-2}$	$M \simeq 10^8$	$M \simeq 10^9$	$M \simeq 10^{10}$
$\delta \simeq \epsilon \simeq 10^{-3}$	$M \simeq 10^{10}$	$M \simeq 10^{11}$	$M \simeq 10^{12}$

choosing $\delta \simeq \epsilon$ (i.e. a similar risk and bias) and solving numerically the previous equation with respect to M .

This is clearly a problem for neuronal networks used as classifiers, because their V_c dimension is higher than the order of magnitude of the number of neurons. More precisely [2], for an arbitrary feedforward neuronal network with a binary activation function the V_c dimension is of $o(W \log(W))$ where W is the number of weights free parameters in the network, while [38] for a multilayer feedforward neuronal network with a sigmoid activation function, the V_c dimension is of $o(W^2)$.

Other asymptotic results have been obtained using different formalisms (e.g. [42] for linear and quadratic classifiers). They all show that the number of parameters must be bounded and

¹⁵The V_c dimension of linear binary classifiers. In the case of a linear classifier, based on affine functions, in a space of dimension n , the number of linear dichotomies of a set of M points in general position is (e.g. [40]):

$$\sigma_M = \begin{cases} \text{if } n < M \text{ then } 2 \sum_{k=0}^n \binom{M-1}{k} \\ \text{else } 2^M \end{cases}$$

where $\binom{n}{p} = \frac{n!}{p!(n-p)!}$ is the binomial coefficient.

As a consequence, since $\sigma_M = 2^M$ iff $M \leq n + 1$, the V_c dimension of a 2-categories classifier is $n + 1$. As obtained in [51], it equals the number of parameters. This is not the case for non-linear classifiers, where V_c can be either lower or higher than the number of parameters.

a number of times smaller than the calibration data set size. From the previous formula, it appears that we must have:

$$V_c \ll \frac{M}{\log(M)}$$

Learning performances of classifiers with margins.

We stated previously that a *linear binary classifier* has its V_c dimension bounded by its number of degrees of freedom and this result is true considering an unconstrained linear classifiers. A step further, in the statistical learning theory [51, 52] it is shown that the V_c dimension is also bounded¹⁶ by:

$$V_c \leq \min \left(\left\lceil \frac{D^2}{\rho^2} \right\rceil, n \right) + 1 \quad (4)$$

where D is the radius of a sphere containing all calibration data and ρ the *margin* of the separating hyper-plane, i.e. the smaller distance between the calibration sample and the frontier of the region of its class.

This improved bound is related to the fact that a structure is imposed on the mechanism of the classification, constraining the margin to be higher than a minimal value. This is true even if separating hyper-plane do not verify the optimal property of support-vector-machines [51]. The important fact, here, is that this bound does not depend on the number of calibration data M used in the computation of the separating hyper-plane.

As pointed out by e.g. [32], the definition of this margin is valid only if it is an *à-priori* value which does *not* depends on the data set itself. Surprisingly enough, this is *not* the case for standard SVM so that there is a small caveat at this point in the Vapnik construction. This notion of margin has thus been generalized to the case where data dependency occurs by [44] and then [31].

Another interpretation of the D/ρ ratio is that D is related to maximal variation of the parameters values: their “*bounds*”, while ρ is related to the minimal “non-negligible” values: their *precision*. This ratio is thus related to the “number of steps” which define the numerical values of the problem. This does not depends on the input data and the previous caveat is avoided.

Considering the Thorpe model discussed in a previous section we notice two important facts: as being based on nearest-neighbor mechanism it *is a [equivalent to a] linear classifier* as, for instance a Perceptron (e.g. [46]). A step further, we have evaluated that the order of magnitude of the number of degrees of freedom is 10^3 , while there are about 10 quantification steps. From (4):

The V_c dimension of the Thorpe model is of about 100, does not depends on the neuronal network size and is likely bounded because of the quantification steps (in relation with the temporal resolution) rather than by its number of degrees of freedom.

Such a low V_c dimension is an impressive result, if we compare to usual neural-nets, as reviewed here. It is clear that this is a very positive theoretical justification of the Thorpe approach.

Introducing the so-called, Guermeur (Y_g) dimension, let us now discuss how to generalize this idea to multi-category classifiers.

¹⁶We consider the smallest integer higher than $\lceil D^2/\rho^2 \rceil$ in this formula.

Learning performances for multi-category NN classifiers

Several attempts have been made to generalize the previous formalism from binary classifiers to multi-category classifiers: either combining several binary classifiers (see [34] for a review and a quantitative comparison of existing solutions) or using enhanced formalisms (e.g. [44] or [12] for a clean mathematical formulation). In [30] a lighter result is proposed, recently improved in [31] and linked to other approaches in the field.

With this approach, a complexity bound using the notion of “covering number” (i.e. the smallest cardinality of a cover of the classifiers class by neighborhoods of a given size) is obtained.

In the particular case of a *linear classifier with N categories* (in our case: a NN classifier with N prototypes) it has been shown that, in coherence with the previous V_c dimension¹⁷, $\epsilon(M, \delta)$ is an increasing function of the, say, Y'_g dimension¹⁸ :

$$\frac{Y'_g}{D^2} = \frac{N(N-1)}{2} \sum_{i < j}^N \|\mathbf{a}_i - \mathbf{a}_j\|^2 \text{ minimal iff } \frac{Y_g}{D^2} = \frac{N^2(N-1)}{2} \sum_{i=1}^N \|\mathbf{a}_i\|^2 \text{ is minimal} \quad (5)$$

where D is the radius of a ball containing all calibration samples, other parameters being defined previously. Since D depends on the calibration samples and not the classifier itself, this quantity is fixed and not to be optimized. The advantage of using Y_g instead of Y'_g as a criterion to minimize is that coupling between prototypes is avoided.

This theoretical work also point out that minimizing the Y_g dimension is equivalent to minimizing the original heuristic criterion proposed by Vapnik [52], with now a well-founded basis. This may be interpreted as *maximizing the margin between the prototypes, and as a consequence between the classifier frontier*.

In the case of NN classifier where several prototypes may correspond to a single category we may consider: $\frac{Y''_g}{D^2} = \frac{N(N-1)}{2} \sum_{i < j, r_i \neq r_j}^N \|\mathbf{a}_i - \mathbf{a}_j\|^2$ i.e. only minimize the margin between prototypes of different categories, but not between prototypes of the same category. This is an alternative, but we have experimented that it is not very interesting because minimizing Y''_g tends to maintain, for a given category, similar thus redundant prototypes whereas maximizing the margin between prototypes of the same category is a positive factor to minimize the number of prototypes.

In relation with this approach, following [40], an equivalence¹⁹ between a N-category NN classifier and a binary classifier can be derived. More precisely, a NN-classifier is equivalent to

¹⁷Relation between V_c and Y_g dimensions. In the particular case of a *linear binary classifier* using the notation of (5) the V_c dimension given in (4) writes:

$$V_c \leq \min(n+1, 1 + D^2 \|\mathbf{a}_1 - \mathbf{a}_2\|^2)$$

the 1st term being the number of independent parameters (i.e. degrees of freedom) of the linear classifier, while the second term is exactly equal to the $1 + Y_g$.

¹⁸Derivation of the Y_g dimension. More precisely, Theorem 1, 2 and 6 of [30] establish that $\epsilon(M, \delta)$ is an increasing function of

$$D^2 \frac{N(N-1)}{2} \sum_{i < j}^N \|\mathbf{a}_i - \mathbf{a}_j\|^2$$

while appendix A.3 of this paper reviews that at the optimum

$$\sum_{i < j}^N \|\mathbf{a}_i - \mathbf{a}_j\|^2 = N \sum_{i=1}^N \|\mathbf{a}_i\|^2$$

eq. (5) being the combination of both.

¹⁹The equivalent V_c dimension of N-category linear classifier. Let us consider a linear space of dimension $N(n+1)$ and:

(1) The hyper-plane of parameter:

$$\underline{\mathbf{a}} = (\mathbf{a}_1, b_1, \dots, \mathbf{a}_i, b_i, \dots, \mathbf{a}_N, b_N)$$

is considered, this vector being build by “concatenation” of N blocks of dimension $n+1$ containing the piece-wise linear parameters.

Since the linear transformation $(\mathbf{a}_i \leftarrow \mathbf{a}_i - \bar{\mathbf{a}}, b_i \leftarrow b_i - \bar{b})$ yields an equivalent classifier, we do not have $N(n+1)$ but $(N-1)(n+1)$ linear independent parameters.

(2) For each calibration data \mathbf{x}_k with $r_i = k$, we consider the $N-1$ vectors, for $i \neq j$:

a binary classifier of V_c :

$$V_c \leq \min \left(\left[\frac{2D^2 + 1}{\rho^4} \sum_i \|\mathbf{a}_i\|^2 \right] + 1, (N-1)(n+1) \right) \quad (6)$$

It appears that minimizing the V_c of the equivalent binary linear classifier is, up to a scale factor, equivalent to the minimization of Y_g .

4 Designing optimized nearest-neighbor classifier

Let us now discuss how to apply the previous piece of theory to nearest-neighbor classifiers.

Fixed margin in NN classifier

As noticed, e.g. in [26], a “physical” parameterized object is always represented though a vector of bounded parameters, with a finite precision. More precisely, it is proposed that, at the *specification*²⁰ level, each numerical value of a parameter component x^i

(i) is *bounded* i.e. $x_{min}^i \leq x^i \leq x_{max}^i$ and (ii) has a *finite precision* x_ϵ^i

so that there is a finite set of significant values, which size is $[x_{max}^i - x_{min}^i]/x_\epsilon^i$.

It has been observed that there is a real gain²¹ to take this experimental specification into account.

$$\underline{\mathbf{x}}_{ij} = (0.. \underbrace{\mathbf{x}_k, 1}_{\leftarrow i} .. \underbrace{-\mathbf{x}_k, -1}_{\leftarrow j} .. 0)$$

i.e. with the block corresponding to $[\mathbf{a}_i, b_i]$ equal to $[\mathbf{x}_k, 1]$ and the block corresponding to $[\mathbf{a}_j, b_j]$ equal to $[-\mathbf{x}_k, -1]$, the other components being 0.

Obviously, since $\underline{\mathbf{a}}^T \underline{\mathbf{x}}_{ij} = (\mathbf{a}_i^T \mathbf{x}_k - b_i) - (\mathbf{a}_j^T \mathbf{x}_k - b_j)$ we obtain:

$$r_k = \arg \max_{r_i} \mathbf{a}_i^T \mathbf{x} - b_i \Leftrightarrow \forall i \neq k, \underline{\mathbf{a}}^T \underline{\mathbf{x}}_{ij} > 0$$

which defines the equivalence between both classifiers, as detailed in [40], Chap 5.

The *margin* ρ of this classifier is the minimal distance from $\underline{\mathbf{x}}_{ij}$ to the hyper-plane of parameter $\underline{\mathbf{a}}$. In the linear sub-space of dimension N defined by $(\mathbf{0}, b_1, \dots, \mathbf{0}, b_i, \dots, \mathbf{0}, b_N)$ this minimal distance is zero since the corresponding components of $\underline{\mathbf{x}}_{ij}$ are constant values. As a consequence, considering the linear sub-space defined by $(\mathbf{a}_1, 0, \dots, \mathbf{a}_i, 0, \dots, \mathbf{a}_N, 0)$ we obtain:

$$\rho = \min_{ij} \underline{\mathbf{a}}^T \underline{\mathbf{x}}_{ij} / \sqrt{\sum_i \|\mathbf{a}_i\|^2} = \rho^2 / \sqrt{\sum_i \|\mathbf{a}_i\|^2}$$

where ρ is the margin in the original data space as defined in (9).

The *radius* \underline{D} is the minimal value verifying for a *center* $\underline{\mathbf{c}} = (\mathbf{c}_1, d_1, \dots, \mathbf{c}_i, d_i, \dots, \mathbf{c}_{M_\bullet}, d_{M_\bullet})$, $\forall \underline{\mathbf{x}}_{ij}$:

$$\begin{aligned} \underline{D}^2 &\geq \|\underline{\mathbf{x}}_{ij} - \underline{\mathbf{c}}\|^2 \\ &= \|\mathbf{x}_k - \mathbf{c}_k\|^2 + \|\mathbf{x}_k + \mathbf{c}_i\|^2 + (d_k - 1)^2 + (d_i + 1)^2 \\ &= 2\|\mathbf{x}_k\|^2 + 2(\mathbf{c}_i - \mathbf{c}_k)^T \mathbf{x}_k + \|\mathbf{c}_k\|^2 + \|\mathbf{c}_i\|^2 + (d_k - 1)^2 + (d_i + 1)^2 \end{aligned}$$

For $\underline{\mathbf{c}} = \mathbf{0}$ this reduces to $\underline{D}^2 \geq 2\|\mathbf{x}_k\|^2 + 2$. Furthermore, for small variations $(\epsilon_{c_h}, \epsilon_{d_h})$ around $\underline{\mathbf{c}} = \mathbf{0}$ we obtain:

$$\begin{aligned} \text{if } h = p \quad \|\underline{\mathbf{x}}_{ij} - \underline{\mathbf{c}}\|^2 &= 2\|\mathbf{x}_k\|^2 + 2 + 2\mathbf{x}_k^T \epsilon_{c_h} + 2\epsilon_{d_h} + o(\|\epsilon_{c_h}\|) + o(|\epsilon_{d_h}|) \\ \text{if } h = i \quad \|\underline{\mathbf{x}}_{ij} - \underline{\mathbf{c}}\|^2 &= 2\|\mathbf{x}_k\|^2 + 2 - 2\mathbf{x}_k^T \epsilon_{c_h} - 2\epsilon_{d_h} + o(\|\epsilon_{c_h}\|) + o(|\epsilon_{d_h}|) \\ \text{else} \quad \|\underline{\mathbf{x}}_{ij} - \underline{\mathbf{c}}\|^2 &= 2\|\mathbf{x}_k\|^2 + 2 + \|\epsilon_{c_h}\|^2 + |\epsilon_{d_h}|^2 \end{aligned}$$

so that $\|\underline{\mathbf{x}}_{ij} - \underline{\mathbf{c}}\|^2$ increases for all linear combination of such variations. It thus a local minimum, and in fact the global minimum, since this quadratic function is convex. This quantity is thus minimal for $\underline{\mathbf{c}} = \mathbf{0}$. A step further, in the calibration data linear space, under the transformation $\mathbf{x}_i \rightarrow \mathbf{x}_i - \mathbf{c}_\bullet$, $\mathbf{a}_i \rightarrow \mathbf{a}_i$ and $b_i \rightarrow b_i - \mathbf{c}_\bullet^T \mathbf{a}_i$ we obtain an equivalent classifier for which the corresponding linear classifier margin ρ is unchanged. Let us choose for \mathbf{c}_\bullet the center of the sphere minimal radius D containing all calibration data \mathbf{x}_k , yielding to a minimal value of $\max_k \|\mathbf{x}_k - \mathbf{c}_\bullet\|^2$. This allows to finally consider a classifier with a radius $\underline{D}^2 = 2D^2 + 1$.

Finally, combining these results, from (4), piece-wise linear classifiers are equivalent to a linear classifier of V_c bounded, according to (6).

²⁰ *Physical parameter specification.* We also attach to such a “physical” parameter: an “initial”, “a-priori” or (iii) *default value* x_0^i (e.g. $(x_{min}^i + x_{max}^i)/2$) plus a (iv) *name* and a (v) *physical unit* (second, pixel, ...).

²¹ *Local quasi-static estimation.* With such specification, “quasi-static” estimation methods (e.g. [53]), with step by step variations from an initial estimate towards the problem solution, are powerful strategies for local estimations (adaptations to limited range variations from a default value, interactive estimation where a user

At step further, two numerical values of a parameter x^i and x'^i can be considered *distinct* only if:

$$|x^i - x'^i| > 2x_\epsilon^i$$

Here, we have to double the value of the bound because each value may vary in a $\pm x_\epsilon^i$ range thus their difference may vary in twice this range. If, on the contrary, $|x^i - x'^i| \leq 2x_\epsilon^i$ we cannot decide whether (i) these values are the same or (ii) differ by a quantity too small to be measurable. In the latter case, we can not say that they are equal, but *indistinguishable*.

Such a precision is in practice very easy to estimate (e.g. 1 mm for a pupil ruler, 1 deg for a protractor, 1 pixel in an image, etc...).

In our context, this specification also allows to determine if two data of the calibration set with different categories are distinguishable. If not, this means that the corresponding categories are not distinguishable and the problem is ill-posed.

Following this track, two vectorial parameters are *indistinguishable* if and only if all components are indistinguishable, i.e.:

$$\mathbf{x} \equiv \mathbf{x}' \Leftrightarrow \underbrace{\max_i \left[\frac{|x^i - x'^i|}{2x_\epsilon^i} \right]}_{\|\mathbf{x} - \mathbf{x}'\|_{\Lambda_\epsilon \infty}} \leq 1 \Rightarrow \underbrace{\sqrt{\sum_{i=1}^n \left[\frac{x^i - x'^i}{2x_\epsilon^i} \right]^2}}_{\|\mathbf{x} - \mathbf{x}'\|_{\Lambda_\epsilon}} \leq \rho = \sqrt{n} \quad (7)$$

with $\Lambda_\epsilon = \begin{pmatrix} 1/(2x_\epsilon^1) & 0 & \dots \\ 0 & 1/(2x_\epsilon^2) & \dots \\ \dots & \dots & \dots \end{pmatrix}$ defining a diagonal metric, in coherence with (2).

Here, we make the choice to consider not $\|\mathbf{x} - \mathbf{x}'\|_{\Lambda_\epsilon \infty}$ but $\|\mathbf{x} - \mathbf{x}'\|_{\Lambda_\epsilon}$ because it is reasonable²² to consider that if many components are almost indistinguishable, the vector is likely indistinguishable. These quantities are anyway only order of magnitudes.

The diagonal metric Λ_ϵ has an obvious statistical interpretation in terms of the inverse of a covariance or “quadratic information” (e.g. [54]). In this case, the precision between two components may be “coupled” (i.e. correlated), the metric not being diagonal any more. It is however obvious to *diagonalize* such a covariance matrix, i.e. consider linear combinations of these components. As a result, the precision is decoupled. There is thus no lack of generality with the present “diagonal” approach.

A step further, these specifications allow to bound the parameter variation, i.e.:

$$\|\mathbf{x} - \mathbf{x}'\|_{\Lambda_\epsilon} \leq D = \sqrt{\sum_{i=1}^n \left[\frac{x_{max}^i - x_{min}^i}{2x_\epsilon^i} \right]^2} \quad (8)$$

as easily obtained with a few algebra.

With these specifications, the complexity of the classifier is bounded, in relation with the notion of *margin* [30], discussed in the previous section. Here the margin is a fixed quantity and does not depend on the input. This, for instance, provides a bound of the V_c dimension,

given initial estimate is to be refined, efficiency in tracking tasks ...), experimentally more efficient than standard usual methods, because the stability of the estimation process is easy to control in this case. Furthermore, the estimation is stopped as soon as the required precision is obtained, whereas for standard methods, convergence to a non-negligible precision only is not so easy to obtain, so that overhead occurs.

²²On *bounded versus average precision*. A less conservative choice would have been to consider $\|\mathbf{x} - \mathbf{x}'\|_{\Lambda_\epsilon} \leq 1$ since:

$$\|\mathbf{x} - \mathbf{x}'\|_{\Lambda_\epsilon} / \sqrt{n} \leq \|\mathbf{x} - \mathbf{x}'\|_{\Lambda_\epsilon \infty} \leq \|\mathbf{x} - \mathbf{x}'\|_{\Lambda_\epsilon}$$

but, regarding statistical inference, such a choice could induce erroneous decisions (i.e. consider as distinct indistinguishable values), whereas the drawback of our choice is only to delay the decision (because with indistinguishable values, no decision is made). For the classifier, the consequence of our choice is to increase the required data precision.

combining (7) and (8) to bound D/ρ , in the case of a linear binary classifier, as defined in eq. (4).

For a thresholded nearest-neighbor classifier, according to these specifications, from (2) and (7), we can decide that the category of an input data \mathbf{x} is r if and only if

$$\begin{aligned} \min_{i, r_i=r} \|\mathbf{x} - \mathbf{x}_i\|_{\Lambda_\epsilon}^2 - \theta_i &< \min_{j, r_j \neq r} \|\mathbf{x} - \mathbf{x}_j\|_{\Lambda_\epsilon}^2 - \theta_j - \rho^2 \\ \Leftrightarrow \max_{i, r_i=r} c_i(\mathbf{x}) &> \max_{j, r_j \neq r} c_j(\mathbf{x}) + \rho^2 \\ \Leftrightarrow \max_{i, j, r_i=r, r_j \neq r} (\mathbf{a}_i - \mathbf{a}_j)^T \mathbf{x} &- (b_i - b_j) > \rho^2 \end{aligned}$$

also written

$$\mu(\mathbf{x}, r) = c_{i_\bullet}(\mathbf{x}) - c_{j_\bullet}(\mathbf{x}) > \rho^2 \text{ with } \begin{cases} i_\bullet = \arg \max_{i, r_i=r} c_i(\mathbf{x}) \\ j_\bullet = \arg \max_{j, r_j \neq r} c_j(\mathbf{x}) \end{cases} \quad (9)$$

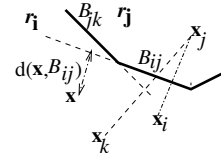
where $\mu(\mathbf{x}, r)$ defines “how much” \mathbf{x} corresponds to the r th category.

This quantity is positive if and only if r is the category of \mathbf{x} . Furthermore, this decision is numerically significant only if this quantity is higher than ρ^2 . Numerically, the higher the quantity, the better the decision. More formally:

$$r = \mathcal{C}(\mathbf{x}) \Leftrightarrow \mu(\mathbf{x}, r) > \rho^2 \quad (10)$$

This corresponds to what has been defined in [30], for the definition of a classifier *margin*. Intuitively, if we tune the classifier with large margins, the category will be correctly estimated for data close to the calibration set but with some variants.

This quantity has also the following geometrical interpretation: since the frontier between two categories is a piece-wise planar surface, this quantity is in direct relation with the distance to the hyper-plane containing the planar patch between category r_{i_\bullet} and r_{j_\bullet} i.e.

$$\mu(\mathbf{x}, r) = \|\mathbf{a}_{i_\bullet} - \mathbf{a}_{j_\bullet}\|_{\Lambda_\epsilon^{-1}} d(\mathbf{x}, B_{i_\bullet j_\bullet})$$


since²³ for an hyper-plane P of equation $\mathbf{a}^T \mathbf{x} - b = 0$: $d(\mathbf{x}, P) = |\mathbf{a}^T \mathbf{x} - b| / \|\mathbf{a}\|_{\Lambda^{-1}}$.

In order to simplify the algebra in the sequel, from now on, we consider the following transformation of our quantities, now *normalized with respect to their precisions*:

$$x^i \leftarrow x^i / (2x_\epsilon^i) \text{ and } a^i \leftarrow a^i (2x_\epsilon^i) \Rightarrow \|\mathbf{x}\| \leftarrow \|\mathbf{x}\|_{\Lambda} \text{ and } \|\mathbf{a}\| \leftarrow \|\mathbf{a}\|_{\Lambda^{-1}} \text{ with } \mathbf{a}^T \mathbf{x} \leftarrow \mathbf{a}^T \mathbf{x}$$

This will lighten the notations without loss of generality.

Centered normalized NN classifier

A step further, we observe that if we consider the linear functions $c_i(\mathbf{x}) = \mathbf{a}_i^T \mathbf{x} - b_i$ defined in (2), it appears that we can add or subtract a constant term \bar{b} , an identical linear factor $\bar{\mathbf{a}}$, and multiply by a positive constant $\bar{c} > 0$, i.e.:

$$c_i(\mathbf{x}) = \bar{c} [\mathbf{a}_i^T \mathbf{x} - b_i] - (\bar{\mathbf{a}}^T \mathbf{x} - \bar{b})$$

²³Distance to an hyper-plane. The distance $d(\mathbf{x}, P)$ is the minimal value $d(\mathbf{x}, \mathbf{z}) = \|\mathbf{x} - \mathbf{z}\|_{\Lambda}$ for a point $\mathbf{z} \in P$, i.e. with $\mathbf{a}^T \mathbf{z} - b = 0$. From the related Lagrangian:

$$\min_{\mathbf{z}} \max_{\lambda} \frac{1}{2} \|\mathbf{x} - \mathbf{z}\|_{\Lambda}^2 + \lambda (\mathbf{a}^T \mathbf{z} - b)$$

and the normal linear equations yield the formula for $d(\mathbf{x}, P)$.

without modifying the comparison in (1). This is the most general transformation²⁴ we can apply if we want to preserve the fact that $c_i(\mathbf{x})$ are linear functions.

In order to cancel this indetermination, we apply the following transformation to the (\mathbf{a}_i, b_i) parameters:

$$\begin{cases} \mathbf{a}_i \leftarrow \bar{c} \mathbf{a}_i - \bar{\mathbf{a}} \\ b_i \leftarrow \bar{c} b_i - \bar{b} \end{cases} \text{ with } \begin{cases} \bar{\mathbf{a}} = \bar{c} \sum_{i=1}^N \mathbf{a}_i / N \\ \bar{b} = \bar{c} \sum_{i=1}^N b_i / N \\ \bar{c} = 1/\rho^2 \end{cases} \quad (11)$$

which allows to *center* and normalize the classifier parameters so that, after this transformation:

$$\sum_{i=1}^N \mathbf{a}_i = \mathbf{0}, \sum_{i=1}^N b_i = 0, \rho = 1 \text{ and } \|\mathcal{C}\|^2 = \sum_{i=1}^N \|\mathbf{a}_i\|^2 \text{ is minimal} \quad (12)$$

(easily verified²⁵ with a few algebra) and we obtain a unique parameterization for this classifier with $N(n+1)$ components.

We also observe that this classifier has $(N-1)(n+1)-1$ independent parameter components, i.e. degrees of freedom, because $n+2$ parameters are constrained via the choice of $\bar{\mathbf{a}}$, \bar{b} and \bar{c} .

Defining a Y_g minimal NN classifier

Minimizing, from (5), the dimension $Y_g/D^2 = N^2(N-1)/2 \sum_{i=1}^N \|\mathbf{a}_i\|^2$, for a fixed number N of prototypes, considering a centered normalized classifier, while preserving the margin of the calibration set $(\dots, \mathbf{x}_k, \dots)$ corresponds to the following optimization problem:

$$\|\mathcal{C}\|^2 = \sum_i \|\mathbf{a}_i\|^2 \text{ with } \begin{cases} \sum_i \mathbf{a}_i = \mathbf{0}, \sum_i b_i = 0 \\ \forall k \quad \max_{r_i=r_k, r_j \neq r_k} (\mathbf{a}_i - \mathbf{a}_j)^T \mathbf{x}_k - (b_i - b_j) > 1 \end{cases} \quad (13)$$

as easily obtained, combining (5) with (9) and (12).

As being a convex quadratic criterion with linear constraints, it has a unique minimum and the local minimization of this criterion leads to the global minimum.

In order to solve this minimization problem, we can easily write the Lagrangian of this constrained criterion:

$$\mathcal{L} = \frac{1}{2} \sum_i \|\mathbf{a}_i\|^2 + \sum_{i,j,k} \alpha_{ijk} [(\mathbf{a}_i - \mathbf{a}_j)^T \mathbf{x}_k - (b_i - b_j) - 1] + \alpha \sum_i b_i + \beta^T \sum_i \mathbf{a}_i$$

with the related Kuhn-Tucker²⁶ conditions:

$$\alpha_{ijk} > 0 \Leftrightarrow \begin{cases} r_i = \arg \max_{r_i=r_k} \mathbf{a}_i^T \mathbf{x}_k + b_i \\ \text{and } r_j = \arg \max_{r_j \neq r_k} \mathbf{a}_j^T \mathbf{x}_k + b_j \\ \text{and } (\mathbf{a}_i - \mathbf{a}_j)^T \mathbf{x}_k - (b_i - b_j) = 1 \end{cases}$$

²⁴*Invariance in the arg-max equation:* In equation (1), the reader can easily verify that any strictly increasing transformation $t: \mathcal{R} \rightarrow \mathcal{R}$ of the proximities $c_i(\mathbf{x})$, i.e. $c_i(\mathbf{x}) \rightarrow t(c_i(\mathbf{x}))$ will not change the comparisons. On the reverse, if $t(\cdot)$ is not a strictly increasing transformation comparisons may be modified for some $c_i(\mathbf{x})$. The most general transformation is thus a composition with a strictly increasing function.

Now, if we want to preserve the linearity, i.e. that $c_i(\mathbf{x})$ being linear, $t(c_i(\mathbf{x}))$ is still linear, this transformation must be linear, the only solution being of the form $t(c_i(\mathbf{x})) = \bar{c} c_i(\mathbf{x}) - (\bar{\mathbf{a}}^T \mathbf{x} - \bar{b})$ with $\bar{c} > 0$.

²⁵*Derivation of centered/normalized linear classifiers.*

The constant \bar{c} is simply chosen in order to have $\rho^2 = 1$ in (9).

The quantity $\bar{\mathbf{a}}$ and \bar{b} are obtained solving $\sum_{i=1}^N [\bar{c} \mathbf{a}_i - \bar{\mathbf{a}}] = \mathbf{0}$, $\sum_{i=1}^N [\bar{c} b_i - \bar{b}] = 0$.

Finally, if we consider the criterion $\min_{\bar{\mathbf{a}}} \|\bar{c} \mathbf{a}_i - \bar{\mathbf{a}}\|^2$ the related normal equation is precisely $\sum_{i=1}^N [\bar{c} \mathbf{a}_i - \bar{\mathbf{a}}] = \mathbf{0}$.

²⁶*On Kuhn-Tucker conditions.* We consider for the purpose of this derivation, weak inequalities (\geq) instead of strict inequalities ($>$). The Kuhn-Tucker conditions state that the Lagrangian multiplier α_{ijk} (i) vanishes iff the inequality is strictly verified and (ii) is positive if the inequality is verified as an equality. In practice, this bound is numerically never attained.

and the related normal equations:

$$\begin{cases} 0 &= \frac{\partial \mathcal{L}}{\partial b_h} &= \sum_{i,k} \alpha_{ihk} - \sum_{j,k} \alpha_{hjk} + \alpha &= \sum_k \alpha_{hk} + \alpha \\ 0 &= \frac{\partial \mathcal{L}}{\partial \mathbf{a}_h} T &= \mathbf{a}_h + \sum_{j,k} \alpha_{hjk} \mathbf{x}_k - \sum_{i,k} \alpha_{ihk} \mathbf{x}_k + \beta &= \mathbf{a}_h - \sum_k \alpha_{hk} \mathbf{x}_k + \beta \end{cases}$$

with $\alpha_{hk} = \sum_i \alpha_{ihk} - \sum_j \alpha_{hjk}$.

The optimal solution of (13) thus writes:

$$\mathbf{a}_h = \sum_k \alpha_{hk} \mathbf{x}_k - \beta \text{ with } \sum_k \alpha_{hk} + \alpha = 0 \quad (14)$$

these equations being used in the sequel to implement the minimization process.

Edited nearest-neighbor classifier.

Let us now consider, not a fixed set of N prototypes, but discuss how to edit it.

A traditional criticism about NN classifiers pointed at large space requirement to store the entire calibration set and the seeming necessity to query the entire calibration set in order to make a single membership classification. There has been considerable interest in *editing* the training or calibration set in order to reduce its size (e.g.: proximity graphs, Delaunay triangulation) eliminating “redundant” data (see [19] for a review). Such editing mechanisms only delete redundant but do not optimize the calibration data used as prototypes.

Eliminating redundant data prototypes has another important consequence: it is expected to bound the classifier complexity. More precisely, from (5): $Y_g/D^2 = \frac{N^2(N-1)}{2} \sum_{i=1}^N \|\mathbf{a}_i\|^2$, it appears that the classifier complexity is an increasing function of the cube of the number N of prototypes. Eliminating redundant prototypes indeed reduces this complexity. However, eliminating too much prototypes may increase each $\|\mathbf{a}_i\|^2$ so that the complexity is not minimum. These two effects must be balanced.

A step further, it is also expected that when the calibration set size M increases, the number N of required prototypes does not increase with the same order of magnitude. This conjecture can be qualitatively explained as follows.

Geometrically, NN classifiers approximate the frontier between two categories by piece-wise linear elements, as discussed previously.

If these borders are regular, they are correctly approximated by a finite and bounded number of planar patches. For instance, if the classifier border is piece-wise linear or very close to a piece-wise linear hyper-surface, a finite number of prototype will correctly represent this border.

As a consequence, an increase of the number of calibration data, will generate only redundant prototypes as soon the classifier border is correctly represented.

In order to apply this idea, we not only need to eliminate redundant prototypes.

In our context, considering a margin as discussed before and using the notation introduced in (9), a prototype of index h is *redundant with respect²⁷ to the calibration set* if, for all calibration data indexed by l :

$$\mu_h(\mathbf{x}_l, \mathbf{r}_l) > \rho^2$$

where the $\mu_h(\mathbf{x}_l, \mathbf{r}_l)$ is the margin defined for a classifier \mathcal{C}_h builded with all prototypes except the h th prototype.

At the algorithmic level, editing the prototypes is a complex process: when eliminating a redundant prototype of index h , other redundant prototypes of the same category may not be

²⁷In comparison, a prototype of index h is *completely redundant* if, for all input data :

$$\arg \max_{r_i} c_l(\mathbf{x}) = \arg \max_{r_k, k \neq h} c_k(\mathbf{x})$$

redundant anymore (since they might be redundant because of the action of the h th prototype), while non-redundant prototypes of another category may become redundant (because the h th prototype may not interfere anymore with these prototypes). We thus must edit the prototypes one by one to properly detect redundancy. An exponentially complex search space is thus to be explored and several heuristics have been proposed to deal with this algorithmic complexity (e.g. [19]). Two new strategies are introduced here.

The margin strategy. In order to deal with this algorithmic complexity, considering both the existing methods in the literature and some preliminary experimentations on our side, we propose the following heuristic:

1. edit the redundant prototypes one by one, if any and
2. consider always the prototype which *preserves a maximal subset of minimal margins*. If we consider the margins in increasing order:

$$\mu_h = \{\mu_h(\mathbf{x}_{l_1}, \mathbf{r}_{l_1}) < \dots < \mu_h(\mathbf{x}_{l_M}, \mathbf{r}_{l_M})\}$$

compared as follows:

$$\mu_h < \mu_{h'} \Leftrightarrow \exists i \in \{1..M\}, \quad \begin{array}{l} 1 \leq j < i \quad \mu_h(\mathbf{x}_{l_j}, \mathbf{r}_{l_j}) = \mu_{h'}(\mathbf{x}_{l_j}, \mathbf{r}_{l_j}) \\ \text{and} \quad \mu_h(\mathbf{x}_{l_i}, \mathbf{r}_{l_i}) < \mu_{h'}(\mathbf{x}_{l_i}, \mathbf{r}_{l_i}) \end{array}$$

this means that we eliminate the redundant prototype with a maximal μ_h

This mechanism is rather general, since it is not related to fact that $c_i(\mathbf{x})$ are linear functions. It attempts to preserve the minimal margin of the classifier and the next minimal margin, etc.. As such, it allows to obtain the “safest” set of non-redundant prototypes.

This mechanism has a polynomial complexity. More precisely: with at most N redundant prototypes, computing margins (in $o(N)$ from its definition) and sorting the result for each prototype (this operation complexity being of $o(M \log(M))$ with a “quick-sort” method) at each step of the editing mechanism, we obtain a complexity lower than $o((N + M \log(M)) N^2)$. This mechanism is indeed *sub-optimal*, i.e. we have no guaranty to obtain an absolute minimal number of non-redundant prototypes. But we obtain a somehow minimal set of prototypes *and* we also preserve the classifier margins. A formal study of this mechanism is a perspective of the present work.

The magnitude strategy. Let us finally point out the following complementary heuristic.

Considering Y_g minimal NN classifiers, i.e. minimizing (13), if a prototype of index h is redundant, it has no influence on the linear inequalities constraining the criterion minimization, the quantity $\|\mathbf{a}_h\|$ is thus minimized without any interaction with other components and its minimum is indeed $\|\mathbf{a}_h\| = 0$. Furthermore, in order to have no influence anymore in the comparison (1) b_h is set to a huge value.

On the reverse, if we can set $\|\mathbf{a}_h\| = 0$ and b_h to a huge value, in coherence with (13), the i th prototype is redundant. As a consequence, *minimizing (13) allows to eliminate redundant prototypes*.

A step further, when there are several redundant prototypes, deleting the prototype with the highest magnitude $\|\mathbf{a}_h\|$ maximizes the local decrease of the Y_g dimension, yielding another *sub-optimal* but reasonable strategy.

Both heuristics have been implemented as discussed in the experimental section.

Stratification of NN classifiers.

In order to discriminate between categories which are not linearly separable a natural idea [51, 44, 31] is to consider also non-linear combinations or functions of the original parameter

components, building an extended parameter vector. For instance, it is known (e.g. [3]) that considering algebraic functions (i.e. polynomials) is sufficient to define classifiers of relatively huge complexity since polynomials approximate any regular curve. Other alternatives exist but, for our purpose polynomials are sufficient.

A polynomial is a linear combination of monomials, such non-linear classifiers thus appear as a linear classifier in the extended parameter space²⁸. A step further the margin between two prototypes increases (perhaps slowly) with the dimension for the simple reason that, being the sum of the squared differences between components, it increases with the number of components.

In order to control the complexity increase²⁹ of the classifier, the principle of *minimal structural risk minimization* has been proposed [52]: in this framework a series of “models” with increasing complexity (e.g. V_c dimension, related to the margin or to the number of degrees of freedom) is defined. The estimation method scans this hierarchy in order to find an estimation which “guaranteed risk” defined in (3) is minimum. A higher complexity model is considered if and only if it significantly increases the performances. This corresponds, in (3), to a trade-off between the model *bias* (increasing with the model complexity) and the *empirical risk* (decreasing with the model complexity). A unique optimum is assumed to occur [44].

In our context, we thus have the double choice to (i) consider more or less prototypes as discussed previously and (ii) to consider classifiers with monomial of degree $d = 1, 2, 3, \dots$. This corresponds to a multi-model³⁰ parametric estimation (e.g. [54]).

Sparse classifiers [21, 22] consider a model of maximal complexity, the algorithm reducing the model complexity during the learning phase. For classifiers à-la-Thorpe [48], the selection of a small subset of components is realized by simply thresholding those with a maximal magnitude. It allows to consider a minimal number of pertinent monomial at a bounded degree d and not only to bound this degree d using all monomial with a degree less than d . At a given degree, we thus suggest to use a sparse subset of monomial, combining both methods.

Considering Y_g minimal NN classifiers, we propose to combine both approaches and to :
- consider classifiers with monomial of degree $d = 1, 2, 3, \dots$, postponing the analysis of classifiers

²⁸*Precision and bounds of monomials.* In relation with our specifications, we easily calculate the *precision* m_ϵ and *bounds* $[m_{min}..m_{max}]$ of a monomial M of degrees $\alpha = (\dots, \alpha_i, \dots)$. To simplify the derivation, let us consider -without loss of generality and up to an affine transform- a *centered normalized* monomial, i.e.:

$$m = \prod_{i=1}^n (u^i)^{\alpha_i} \in [-1..1] \text{ with } u^i = \frac{x^i - (x_{max}^i + x_{min}^i)/2}{(x_{max}^i - x_{min}^i)/2} \in [-1..1]$$

thus obviously bounded by $[-1..1]$. The precision m_ϵ of this monomial is obtain by a simple derivation, up to the 1st order:

$$m_\epsilon \simeq \sum_{j=1, \alpha_j > 0}^n \prod_{i=1, i \neq j}^n (u^i)^{\alpha_i} \alpha_j (u^j)^{\alpha_j - 1} u_\epsilon^j \text{ with } u_\epsilon^j = \frac{x_\epsilon^j}{(x_{max}^j - x_{min}^j)/2}$$

so that a minimal constant bound $|m_\epsilon| \leq \sum_{j=1}^n \alpha_j u_\epsilon^j$ is derived. Combining these formula, we obtain, up to the 1st order:

$$m_\epsilon = \sum_{j=1}^n \alpha_j \frac{x_\epsilon^j}{(x_{max}^j - x_{min}^j)/2}$$

²⁹*On the complexity of non-linear combinations of parameters.* It is in fact demonstrated [7] that polynomial support-vector machines have a V_c dimension which increase is exponential in the polynomial degree, since a polynomial of degree d in n variables has $K = \frac{(n+d)!}{n!d!}$ monomial. Several alternatives exist. e.g. basic radial Gaussian function (RBF). They have similar properties and performances, for instance RBF without any constraints have an infinite V_c [7].

³⁰*Advantages of the multi-modeling strategy.* In [54], for such a strategy it had been highlighted that:

1. estimating a parameter or calibrating a classifier does not only mean calculating numerical values .. but choosing which model (e.g. “shape” equation or “mechanism” law) best represents the calibration data set,
2. general models may be irrelevant / singular / etc ... on some data set (i.e. data set with some specific constraints) and specific models (which make explicit such constraints) must be considered,
3. non-linear (thus iterative) local estimators of general models must be initialized with a relevant initial value. Such a value can be provided by the estimation, even biased, of a simpler model.

having monomial of degree $> d$ after classifiers of degree $\leq d$ have been analyzed. This allows to find a classifier of minimal degree;

- cancel all a_h^i components which are small not enough not to violate the classifier calibration (those with the smallest magnitude according to the Thorpe strategy) in order to limit the number of degrees of freedom.

We finally have all ingredients to obtain *fixed margin, centered and normalized Y_g minimal, edited sparse and stratified nearest-neighbor classifiers*. Let us now discuss how to implement such an optimization.

Implementing optimized nearest-neighbor classifier

Optimized nearest-neighbor classifier initialization and on-line modification

For a given calibration set $(\dots, \mathbf{x}_k, \dots)$ of size M we initialize the set of prototypes $(\dots, (\mathbf{a}_i, b_i), \dots)$ from the calibration set itself, as defined in (2), thus starting with $N = M$ prototypes. This is a standard NN classifier at this stage.

As such, we *already* have an initial solution (likely not a very good one) and have only to *improve* this solution.

At this stage we also detect if the data calibration set is coherent with respect to the precision margin: the occurrence of two indistinguishable calibration data of different categories raises an error.

As far as “real-time” reactivity is concerned, as soon as a new data calibration is given, without any lack of reactivity, the system is able to provide a first-approximation classification, inserting this data as a new prototype (with eventually other data no more calibrated because of this insertion).

This guarantees that the classifier is immediately calibrated with respect to this new calibration data. This new sub-optimal solution is then simply to be improved.

If, on the contrary, a calibration data is deleted, the current solution is simply to be re-optimized, taking benefit of the fact that one constraint is removed.

Optimization is thus an independent process, realized “when time is available”.

With, these simple rules, the learning mechanism is entirely adaptive with respect to calibration data addition / deletion and also with respect to the computation time resources.

Using Hebbian rules for the optimization

When minimizing (13), one usual track is to consider the dual problem of this convex problem, i.e. solve it in function of α_{hk} . Although computational expensive, this usually an efficient strategy (e.g. [32] for a review) used by SVM-like algorithms, requiring a quadratic minimization under multi-dimensional linear inequalities. However, in our context it would have been quite difficult to relate such algorithm to biologically plausible mechanisms.

Following another track, the normal equations (14) tell us that *the optimal solution is an affine combination of the calibration data, with a constant sum of weights*. We thus also can iteratively obtained this affine combination considering a unique calibration data and a unique prototype at each step.

This corresponds to Hebbian-like learning rules as discussed previously: considering a calibration data $\mathbf{x}_l, l = 1..M$ and a classifier parameter $(\mathbf{a}_h, b_h), h = 1..N$ such a rule is of the form:

$$\mathbf{a}_h \leftarrow \mathbf{a}'_h = \mathbf{a}_h - \delta \mathbf{x}_l \text{ and } b_h \leftarrow b'_h = b_h - \nu \quad (15)$$

for some increment (δ, ν) (here we omit the indexes on δ and ν to lighten the notation) which are to be computed in order to decrease the criterion, while preserving the constraints.

As extensively discussed elsewhere [20, 43, 29] and reviewed within this work, this rule corresponds to biologically plausible learning mechanisms.

If there is not such increment, this means that there is no linear combination of the form $\mathbf{a}_h + = \sum_k \delta_{hk} \mathbf{x}_k$ which decreases this convex criterion. There is thus no local linear variation, compatible with the normal equations (14) which may improves this criterion. We thus are at a local minimum. This local minimum is the optimal value, because the criterion is quadratic and thus convex. This very simply demonstrates the convergence of this method.

The application of this rule is to be followed by another Hebbian-like mechanism:

$$\forall k, \mathbf{a}_k \leftarrow \mathbf{a}_k + \delta \mathbf{x}_l / N \text{ and } b_k \leftarrow b_k + \nu / N$$

in order to preserve $\sum_i \mathbf{a}_i = \mathbf{0}$ and $\sum_i b_i = 0$, as the reader can easily verify.

Using (15), let us look for a (δ, ν) increment decreasing the constrained criterion (13).

Writing $\delta = 2 \kappa (\mathbf{x}_l^T \mathbf{a}_h) / \|\mathbf{x}_l\|^2$, and assuming $\mathbf{x}_l \neq \mathbf{0}$ (otherwise the Hebbian rule has no effect) and $\mathbf{x}_l^T \mathbf{a}_h \neq 0$ (otherwise our derivation is singular), the criterion decreases iff:

$$\begin{aligned} & \|\mathbf{a}'_h\|^2 = \delta^2 \|\mathbf{x}_l\|^2 - 2 \delta (\mathbf{x}_l^T \mathbf{a}_h) + \|\mathbf{a}_h\|^2 \leq \|\mathbf{a}_h\|^2 \\ \Leftrightarrow & 4 \kappa^2 (\mathbf{x}_l^T \mathbf{a}_h)^2 / \|\mathbf{x}_l\|^2 - 4 \kappa (\mathbf{x}_l^T \mathbf{a}_h)^2 / \|\mathbf{x}_l\|^2 \leq 0 \\ \Leftrightarrow & \kappa^2 - \kappa \leq 0 \\ \Leftrightarrow & 0 \leq \kappa \leq 1 \end{aligned}$$

the decrease being maximal for $\kappa = 1/2$, while the decrease for values $\kappa > 1/2$ corresponds to a decrease for a value $1 - \kappa$, below $1/2$.

We thus look for a couple (κ, ν) , $0 \leq \kappa \leq 1/2$, with a maximal value of κ while the constraints:

$$\max_{r_i=r_k} [\mathbf{a}_i^T \mathbf{x}_k - b_i] - \max_{r_j \neq r_k} [\mathbf{a}_j^T \mathbf{x}_k - b_j] > 1$$

are verified.

These constraints can be rewritten in a more compact form:

$$\begin{aligned} & \text{if } r_k = r_h \quad \max(u'_k, w_k - c_k \kappa + \nu) > v_k \\ & \text{if } r_k \neq r_h \quad u_k > \max(v'_k, w_k - c_k \kappa + \nu) \end{aligned} \tag{16}$$

with the following notations:

$$\begin{cases} u'_k &= \max_{r_i=r_k, i \neq h} [\mathbf{a}_i^T \mathbf{x}_k - b_i] & v_k &= \max_{r_j \neq r_k} [\mathbf{a}_j^T \mathbf{x}_k - b_j] + 1 \\ u_k &= \max_{r_i=r_k} [\mathbf{a}_i^T \mathbf{x}_k - b_i] - 1 & v'_k &= \max_{r_j \neq r_k, j \neq h} [\mathbf{a}_j^T \mathbf{x}_k - b_j] \\ w_k &= \mathbf{a}_h^T \mathbf{x}_k - b_h & c_k &= 2 (\mathbf{x}_l^T \mathbf{a}_h) (\mathbf{x}_l^T \mathbf{x}_k) / \|\mathbf{x}_l\|^2 \end{cases} \tag{17}$$

A step further, since these constraints were already verified at the previous step, for the previous value of (\mathbf{a}_h, b_h) i.e. for $\kappa = \nu = 0$ we also can write, from (16):

$$\text{if } r_k = r_h \text{ then } \max(u'_k, w_k) > v_k \text{ and if } r_k \neq r_h \text{ then } u_k > \max(v'_k, w_k)$$

As a consequence, (i) if $r_k = r_h$ and $u'_k > v_k$, the corresponding inequality is already verified and (ii) if $r_k \neq r_h$ we already have $u_k > v'_k$.

The inequalities (16) thus reduce to:

$$\begin{aligned} & \text{if } r_k = r_h \quad \text{and } u'_k \leq v_k \quad w_k - c_k \kappa + \nu > v_k \\ & \text{if } r_k \neq r_h \quad u_k > w_k - c_k \kappa + \nu \end{aligned}$$

finally rewritten as:

$$\max_{r_k=r_h, u'_k \leq v_k} (v_k - w_k + c_k \kappa) < \nu < \min_{r_k \neq r_h} (u_k - w_k + c_k \kappa)$$

Summarizing: in order to find a solution to (15) decreasing (13), we have to look for a maximal value of κ compatible with the following inequalities (written using (17)):

$$0 \leq \kappa \leq 1/2 \quad \text{with} \quad \begin{cases} \nu_{\min}(\kappa) &= \max_{r_k=r_h, u'_k \leq v_k} (v_k - w_k + c_k \kappa) \\ \nu_{\min}(\kappa) &< \nu_{\max}(\kappa) &= \min_{r_k \neq r_h} (u_k - w_k + c_k \kappa) \end{cases} \quad (18)$$

while $\kappa = 0$ is a known solution.

In (18), if $\forall k, r_k = r_h, u'_k > v_k$ there is no minimal bound for ν , thus no maximal bound for b_h . As a consequence b_h may have huge values, large enough for the h th not to be used in the comparison process. This simply means that this prototype is redundant and can thus be deleted. We again verify that we can edit the prototype list, within the minimization process.

In both paradigms, since the criterion is convex, we may choose \mathbf{a}_h and \mathbf{x}_l either sequentially or randomly, the choice of a strategy being of no influence on the final result, but only the calculation duration. This means that there is no specific constraint in choosing a $(\mathbf{a}_h, \mathbf{x}_l)$ couple as soon as all couples are finally selected.

Let us finally note³¹ that the same method could have been used to minimize Y'_g instead of Y_g in (5).

Computer simulation. If a computer simulation is to be derived, this specific linear programming problem can be solved, searching a maximal value $\kappa \in [0..1/2]$ such that $\nu_{\min}(\kappa) < \nu_{\max}(\kappa)$ using a dichotomous method for a maximal efficiency.

If any solution, (15) is to be used with:

$$\delta = 2 \kappa (\mathbf{x}_l^T \mathbf{a}_h) / \|\mathbf{x}_l\|^2 \text{ and, say, } \nu = (\nu_{\min}(\kappa) + \nu_{\max}(\kappa)) / 2$$

A step further, in order to speed up the computation, redundant prototypes can be deleted before the optimization, using either the *margin strategy* or the *magnitude strategy*. This may change the final result since we do not simply minimize the convex criterion, but edit it. However, results are expected to be qualitatively similar.

Regarding the minimization itself, the following heuristic:

- *initialization phase*: considering each couple $(\mathbf{a}_h, \mathbf{x}_l)$ once,
 - *iteration phase*: selecting the couple which yielded a maximal decrease of the criterion,
 - *termination phase*: until all couples do not decrease the criterion anymore
- has been implemented in order to maximize the local decrease of the criterion.

Biological plausibility. As far as biological plausibility is concerned, the previous derivation states that any a small $\kappa > 0$ compatible with (18) decreases the criterion.

A biological system may thus simply use “epsilon-values”, (say, $\kappa \simeq 10^{-3}$ since all quantities have been normalized with respect to unity), checking $\nu_{\min}(\kappa) < \nu_{\max}(\kappa)$ in (18).

This test is easily done by a min/max operator, which is biologically plausible as reviewed previously and shown in [56],

A step further, this optimization rule also integrates a “switch” (detecting redundant data, detecting if an increment is valid, etc.). This is related to so called inhibition mechanisms, commonly observed in such biological neuronal layers, [29].

³¹ Using Hebbian rules to minimize the Y'_g dimension. If we consider the minimization of $\sum_{i < j}^N \|\mathbf{a}_i - \mathbf{a}_j\|^2$ instead of $\sum_i^N \|\mathbf{a}_i\|^2$ in (13) using the Hebbian rule defined in (15) we easily derive:

$$\arg \min_{\mathbf{a}_h} \sum_{i < j}^N \|\mathbf{a}_i - \mathbf{a}_j\|^2 = \arg \min_{\mathbf{a}_h} \sum_{j \neq h}^N \|\mathbf{a}_h - \mathbf{a}_j\|^2$$

so that if we replace \mathbf{a}_h with $\mathbf{a}'_h = \mathbf{a}_h - \delta \mathbf{x}_l$ in order to obtain:

$$\sum_{j \neq h}^N \|\mathbf{a}_h - \mathbf{a}_j\|^2 \geq \sum_{j \neq h}^N \|\mathbf{a}'_h - \mathbf{a}_j\|^2 = \sum_{j \neq h}^N \|\mathbf{a}_h - \mathbf{a}_j\|^2 - 2 \delta \sum_{j \neq h}^N (\mathbf{a}_h - \mathbf{a}_j)^T \mathbf{x}_l + \delta^2 \sum_{j \neq h}^N \|\mathbf{x}_l\|^2$$

writing $\delta = 2 \kappa \sum_{j \neq h}^N (\mathbf{a}_h - \mathbf{a}_j)^T \mathbf{x}_l / \sum_{j \neq h}^N 1 / \|\mathbf{x}_l\|^2$ the previous inequality reduces to $0 \leq \kappa \leq 1$ as for the derivation proposed for Y_g and the rest of the derivation is identical, as the reader can easily verify.

A step further, with $\delta = 2 \kappa \sum_{j, r_j \neq r_h}^N (\mathbf{a}_h - \mathbf{a}_j)^T \mathbf{x}_l / \sum_{j, r_j \neq r_h}^N 1 / \|\mathbf{x}_l\|^2$ the same mechanism would have been used to minimize Y''_g .

As a conclusion what is proposed here is fully compatible with what happens in the hippocampus (e.g. [20] Chap. 7 and 8, [43] Chap. 6).

5 Experimental results

Interactive 2D demonstration

In order to validate the previous method, a piece-wise optimized classifier has been implemented. Incremental adaptation to new data is automatic and integrated in the process. Please refer to the on-line³² software documentation for details. The present implementation has been designed so that the “training loop” execution can be eventually controlled (start/stop or suspend/resume it, get a temporary information on the fly, etc...).

Example of results, shown in Fig 6, 7, 8 and 9, illustrate the method behavior and allow to validate the implementation. The module can be experimented on Internet.

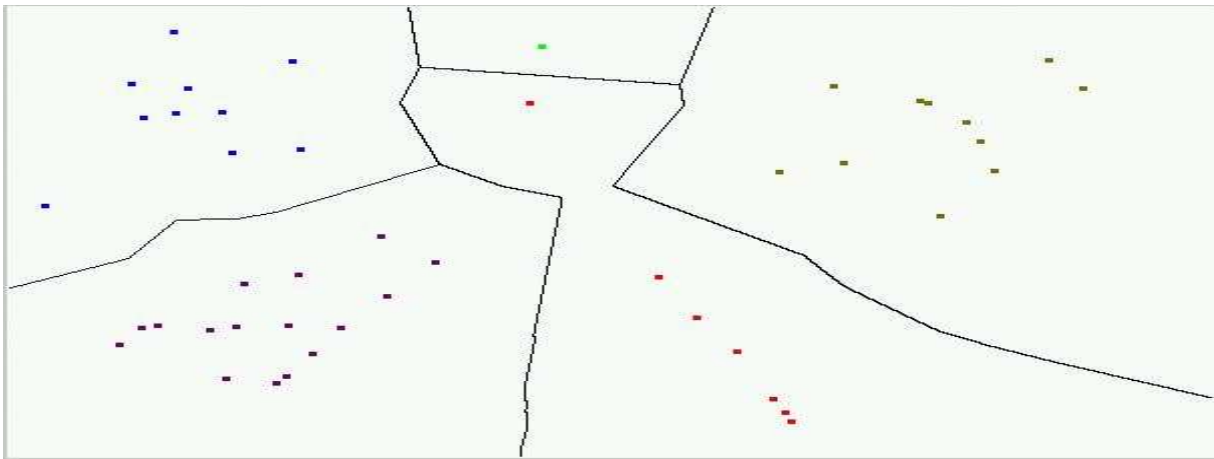


Figure 5: An example of raw nearest-neighbor classification, with 45 prototypes, the equivalent Vapnik dimension as defined in (6) is bounded by the number of degrees of freedom $V_c = 132$. The Guermeur dimension as defined in (5) is huge $Y_g \simeq 1.5 \cdot 10^4$.

In order to qualitatively compare this with a standard SVM method, we have considered the 1-to-1 multi-class C-SVM method (comparing each pairs of category using a standard SVM and choosing the category with the best result), as implemented³³ by Chen and Lin [10]. This is illustrated in Fig. 10.

Experimenting on bench-mark data

In order to validate the method on a real data set, we have considered *Pima indians diabetes* as provided by B.D. Ripley³⁴. The goal is to decide whether a subject has a diabetes or not³⁵.

We have selected this data set since performances of other methods are available for comparison, as reported by [22]. Percentage of test errors on the Pima data set, for a test set of 269 samples are:

³²In <http://www-sop.inria.fr/odyssee/imp> the `imp.math.Classifier` classes.

³³A free-ware library for support vector machines, thanks to [10], is available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>

³⁴ Available at <http://www.stats.ox.ac.uk/pub/PRNN>

³⁵See <http://www.stats.ox.ac.uk/pub/PRNN/README.html> for details.

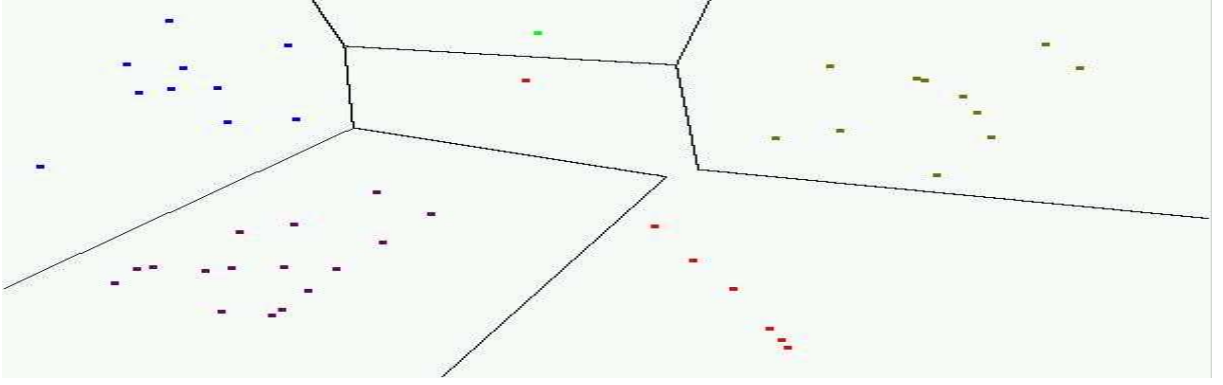


Figure 6: An example of result corresponding to the raw classification given in Fig. 5: here redundant prototypes have been removed but no optimization is performed. A margin of 8 pixels is specified as an input of the optimization process. We obtain more than one prototype for each category and this is a sub-optimal result. We have also observed that both editing strategies (the *margin* strategy and the *magnitude* strategy detailed in this paper) yields the same number of prototypes but not the same prototypes. Comparing to Fig. 5, with 6 prototypes: $V_c = 15$, still bounded by the number of degrees of freedom and $Y_g = 122$.

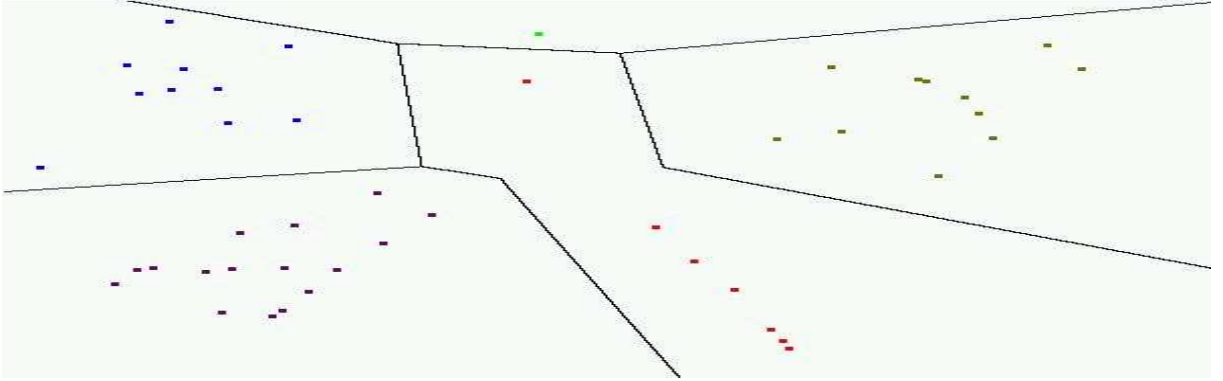


Figure 7: An example of result corresponding to the raw classification given in Fig. 5: now the classifier is optimized after prototypes edition, with 6 prototypes: $Y_g = 44$. Both the margin and the magnitude strategies yields the same result in this case.

SVM method	23.8 %
Sparse classifier	22.7%
Neuronal network	27.9 %
Other methods	24.2-25.3 %
Piece-wise linear classifier	23.81 %

In our experimentation we have used only 200 among the 300 training samples provided because of some missing data. Using an equivalent implementation, we have obtained a performance of 23.81 % of errors on the test data set, to be compared with the error percentage of 23.8 % for a SVM method and 24.2-25.3 % for other methods in the literature. We have obtained an equivalent $V_c = 392$, bounded by the number of degrees of freedom.

Performances are thus similar to existing methods. This was not likely to occur because we are using here a training data set with “mistakes”, i.e. such that our model does not apply. This is also a confirmation that using “simple” Hebbian-like rules leads to relevant results in this case.

Let us now turn to an experiment where a deterministic calibration set is provided.

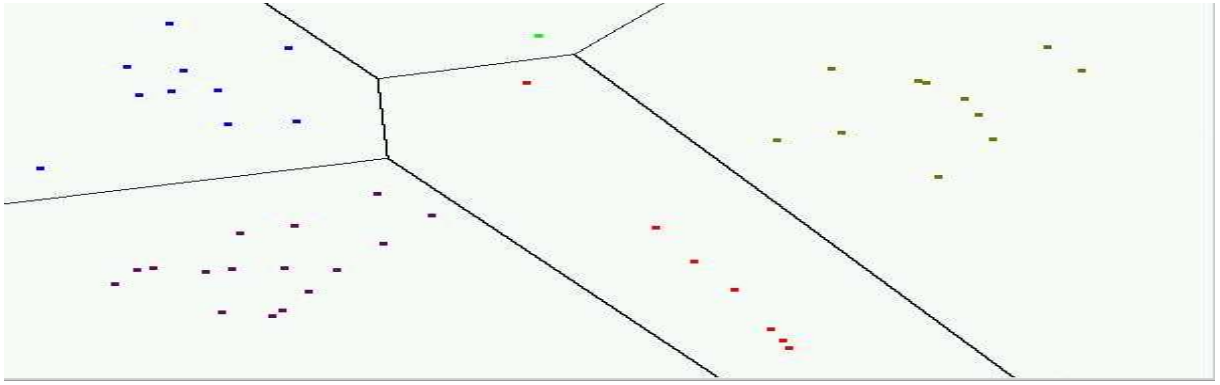


Figure 8: An example of result corresponding to the raw classification given in Fig. 5: now the classifier is optimized *without* prototype edition, with 5 prototypes: $Y_g = 35$. The difference with Fig. 7 illustrates that prototype edition being a non-linear process, different optimal solutions may be output depending on the obtained criterion. Surprisingly, in this case, the best solution is this obtained without prototype edition. This fact varies with the input margin and the data set.

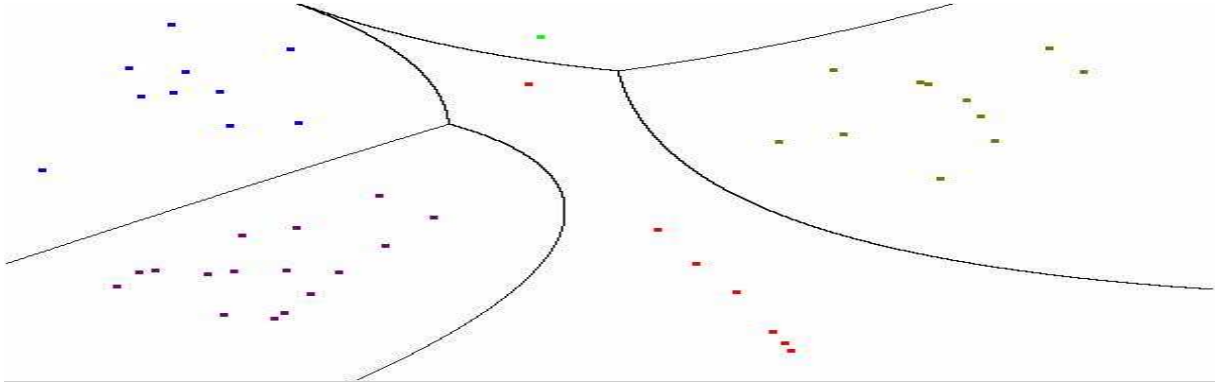


Figure 9: An example of result corresponding to the raw classification given in Fig. 5: now the classifier is optimized with a second order polynomial model, with 5 prototypes: $Y_g = 28$. In this case, we obtain a solution with 5 prototypes regardless the fact that prototypes have been edited or not, before the optimization.

Sign-language recognition

Description of the experiment

We consider a tiny experiment related to the recognition of the Quebecian Sign Language Alphabet. This can not be considered as a real experiment of sign-language recognition, whereas it is only used to evaluate the present method.

The static (one image) spelling of four subjects have been recorded using a standard video system with a resolution of 384×288 , as follows:

<i>Subject</i>		data
Sy	Experimented	2 series of 9 letters (1 particularly good)
Ad	Kid, naive 7	2 series of 9 letters, 1 acceptable, 1 "bad" (used as counter-example)
Th	Beginner	3 series of 9 letters, 2 without shadow, 1 with hand shadow
Li	Beginner	1 series of 9 letters good quality

examples of such images being given in Fig. 11.

The following experimental configurations have been chosen in order to evaluate the method with respect to various combinations of data:



Figure 10: An example of result corresponding to the raw classification given in Fig. 5: here a standard 1-to-1 SVM method is utilized and 10 support vectors are used by the algorithm. Since $N(N-1)/2$ SVM, with $n+1$ degrees of freedom each, are used, although the margin between categories is higher with this method, much more degrees of freedom are involved.

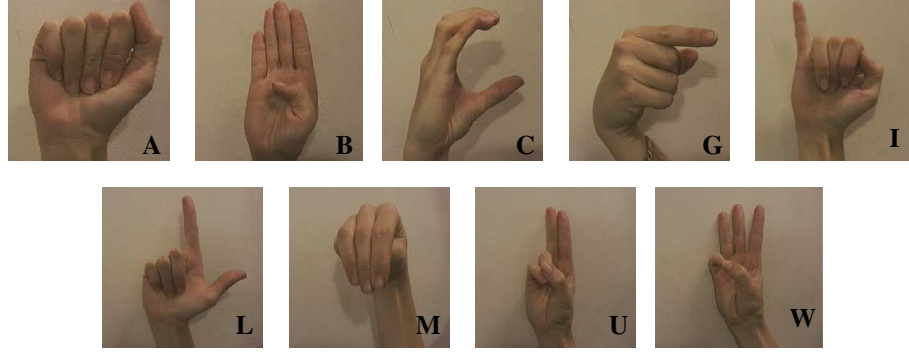


Figure 11: An example of the nine letters taken into account in this experiment, here sub-images containing the hand have been automatically cropped as discussed in the text.

<i>Experiment label</i>	learning set	test set
Experiment 1	Sy, 9 letters from all samples except 1 series	Sy, 9 letters from the last sample
Experiment 11	Th, 9 letters from all samples except 1 series	Th, 9 letters from the last sample
Experiment 2	Sy, all samples of 9 letters	Th, 3 series of 9 letters
Experiment 3	Th, series of 9 letters without shadow	Th, 2 series of 9 letters with shadow
Experiment 4	Sy/Li, all samples of 9 letters	Th, all samples of 9 letters
Experiment 41	Th/Li, all samples of 9 letters	Sy, all samples of 9 letters
Experiment 42	Sy/Th, all samples of 9 letters	Li, the sample of 9 letters

learning and test sets intersection being always empty.

Parameter extraction

In order to extract relevant parameters, using standard image analysis methods, edges are detected and binarized using a fixed threshold. This simple paradigm is sufficient, with the lighting conditions of the data acquisition, to pre-process the image.

The first and second-order momenta (center of gravity, main orientation and lengths) are computed as schematized in Fig. 12. This allows to encapsulate the hand in an ellipse. This also allows to crop a rectangle in the image containing the hand, eliminating a part of the background influence.

The ellipse position (related to the hand position in space) and the ellipse length (related to the hand size and camera proximity) are not relevant to detect the hand sign, whereas the ellipse orientation and the ellipse length ratio (i.e. eccentricity) are. These two parameters are related to the hand relative position (whether it is open/close, tilted, etc..) and are thus used by the classifier with the second-order momenta.

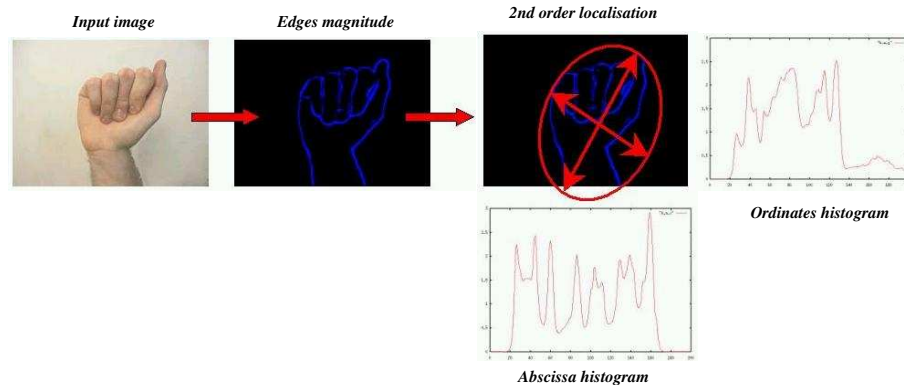


Figure 12: Extracting relevant parameters from raw images, see text for details.

A step further, the histograms of the edge abscissa and ordinates are computed, these histogram being smoothed and normalized, as shown in Fig. 12. It has been experimentally verified, in this context, that this provides a relevant set of parameters to discriminate different hand signs. Therefore, there are combined with the previous parameters in order to parameterize the data input to the classifier.

Performance evaluation

In order to evaluate the performances of the present method we use a standard 1-to-1 SVM method, as discussed previously.

Obtained results are summarized in the following table:

<i>Experiment label</i>	Raw classifier	standard 1-to-1 SVM	optimized NN classifier
Experiment 1	11 %	11 %	11 %
Experiment 11	30 %	22 %	27 %
Experiment 2	37 %	33 %	28 %
Experiment 3	61 %	61 %	61 %
Experiment 4	56 %	41 %	62 %
Experiment 41	30 %	37 %	31 %
Experiment 42	0 %	11 %	0 %

where the percentage of errors have been reported. They clearly demonstrate, that up to the 1st order, both methods have similar performances. This was not entirely obvious since we have implemented the optimization using a biologically plausible minimization method. A step further, we clearly observed that the deterministic method have better performances when the calibration data set has no mistakes, while performances are degraded when it contains errors.

These experimental facts nicely illustrate the advantage of the present methods .. and its drawbacks !

Training time for the standard SVM method (about $50 \text{ ms} \pm 20 \text{ ms}$) and Optimized NN classifier (about $100 \text{ ms} \pm 40 \text{ ms}$) have the same order of magnitude but always significantly longer for the second method since the Hebbian-like optimization method is not optimal, as discussed previously. Testing time is faster for optimized NN classifier (about $0.2 \text{ ms} \pm 0.1$

ms) than 1-to-1 SVM methods (about $0.8 \text{ ms} \pm 0.2 \text{ ms}$), not surprising because the number of comparisons is higher in the latter case. This difference is however too small to be significant.

The real important fact here is that Hebbian-like optimization does not induce huge computation times, but only some overhead.

6 Conclusion

We have proposed an approach which allows to consider deterministic classifiers in the case where categories are not linearly separable. The -somehow very simple- key idea was to consider piece-wise linear classifiers of minimal dimension, as a generalization of support-vector machines. This is an alternative to non-linear combinations of data input usually used as a front-end to linear methods, in SVM. Here, both approaches are easily combined.

As such, this allows to re-interpret basic nearest-neighbor classifiers, based on the notion of prototypes, with respect to the statistical learning theory and obtain an optimized version of this basic mechanism. A key feature is that optimizing the statistical dimension of nearest-neighbor classifiers allows to automatically edit them, i.e. remove some of the redundant prototypes.

Although out of the scope of the present study, the generalization to calibration data set containing mistakes is likely straightforward, following the usual SVM methodology.

A step further, we have proposed to consider the notion of margin as a fixed predefined specification related to the à-priori precision of the input data. Not being related to the data itself, it yields a clearer interpretation of the underlying theory.

We also have made explicit and experimented that SVM like mechanisms can easily be implemented using Hebbian-like correction rules. Such optimization mechanism is not as fast as the standard method, but its biological plausibility is much better, while final performances are similar and seem better when the training set is a deterministic calibration set.

This point of view is in deep relation with fast visual recognition in the brain. It may, for instance, explain why biological classifiers have better generalization performances, although the input data has a huge dimension.

More precisely, it has been possible to estimate a bound ($\simeq 10^2$) of the V_c dimension of the Thorpe model, showing that its generalization performances are expected to be much better than standard models based on neuronal networks.

The main limitation of this approach, with respect to biological learning, is that we have limited the point of view to *supervised learning*. This paradigm is a valid track (e.g. [25]) and was mandatory to relate what is actually known in statistical learning with biological learning. However, it is a limited paradigm and the next step is to consider generative models (e.g. [13]), breaking this gap.

Acknowledgments: *Simon Thorpe*, *Yann Guermeur* and *Olivier Faugeras* are gratefully acknowledged for some powerful ideas at the origin of this work.

This work has been realized within the scope of the French Amaria/Robea project.

References

- [1] R. Bajcsy and F. Solina. Three dimensional object representation revisited. In *Proceedings of the 1st International Conference on Computer Vision*, London, England, June 1987. IEEE Computer Society Press.
- [2] E. Baum and D. Haussler. What size net gives valid generalization. *Neural Computation*, 1:151–160, 1989.
- [3] R. Benedetti and J.-J. Risler. *Real algebraic and semi-algebraic sets*. Hermann, Paris, 1990.
- [4] L. Borg-Graham, C. Monier, and Y. Fregnac. Visual input evokes transient and strong shunting inhibition in visual cortical neurons. *Nature*, 393:369–373, 1998.
- [5] G. Bugmann. Biologically plausible neural computation. *Biosystems*, 40:11–19, 1997.
- [6] J. Bullier. Integrated model of visual processing. *Brain Res. Reviews*, 36:96–107, 2001.

- [7] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.
- [8] Y. Burnod. *An adaptive neural network: the cerebral cortex*. Masson, Paris, 1993. 2nd edition.
- [9] C. E. Carr. Processing of temporal information in the brain. *Annu. Rev. Neurosci.*, 16:223–244, 1993.
- [10] C.-C. Chang and C.-J. Lin. Training nu-support vector classifiers: Theory and algorithms. *Neural Computation*, 13(9):2119–214, 2001.
- [11] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans.on Information Theory*, 13(1), 1967.
- [12] F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39(1):1–49, 2001.
- [13] P. Dayan and L. F. Abbott. *Theoretical Neuroscience : Computational and Mathematical Modeling of Neural Systems*. MIT Press, 2001.
- [14] A. Delorme, J. Gautrais, R. VanRullen, and S. J. Thorpe. Spikenet: A simulator for modeling large networks of integrate and fire neurons. *Neurocomputing*, 26:989–996, 1999.
- [15] A. Delorme, L. Perrinet, and S. Thorpe. Network of integrate-and-fire neurons using rank order coding b: spike timing dependant plasticity and emergence of orientation selectivity. *Neurocomputing*, 38:539–545, 2001.
- [16] A. Delorme, G. Richard, and M.Fabre-Thorpe. Rapid categorisation of natural scenes is colour blind: A study in monkeys and humans. *Vision Research*, 40(16):2187–2200, 2000.
- [17] A. Delorme and S. Thorpe. Face processing using one spike per neuron: resistance to image degradation. *Neural Networks*, 14:795–804, 2001.
- [18] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer Verlag, 1996.
- [19] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification, 2nd edition*. Wiley Interscience, 2000.
- [20] R. Durbin, C. Miall, and G. Mitchinson, editors. *The computing neuron*. Addison-Wesley, 1989.
- [21] M. A. T. Figueiredo. Adaptive sparseness using jeffreys prior. In *Neural Information Processing Systems*, 2001.
- [22] M. A. T. Figueiredo and A. K. Jain. Bayesian learning of sparse classifiers. In *Computer Vision and Pattern Recognition*, 2001.
- [23] R. Fitzsimonds, H. Song, and M. Poo. Propagation of activity dependent synaptic depression in simple neural networks. *Nature*, 388:439–448, 1997.
- [24] D. J. Freedman, M. Riesenhuber, T. Poggio, and E. K. Miller. Categorical representation of visual stimuli in the primate prefrontal cortex. *Science*, 291(5502):312–316, 2002.
- [25] K. Friston. Functional integration and inference in the brain. *Prog Neurobiol*, 68:113–143, 2002.
- [26] F. Gaspard and T. Viéville. Non linear minimization and visual localization of a plane. In *The 6th International Conference on Information Systems, Analysis and Synthesis*, volume VIII, pages 366–371, 2000.
- [27] J. Gautrais and S. Thorpe. Rate coding vs temporal order coding : a theoretical approach. *Biosystems*, 48:57–65, 1998.
- [28] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, 1993.
- [29] T. Gisiger, S. Dehaene, and J. P. Changeux. Computational models of association cortex. *Curr. Opin. Neurobiol.*, 10:250–259, 2000.
- [30] Y. Guermeur. Combining discriminant models with new multi-class svms. *Pattern Analysis and Applications*, 5(2):168–179, 2002.
- [31] Y. Guermeur. A simple unifying theory of multi-class support vector machines. Technical Report 4669, INRIA, 2002.
- [32] Y. Guermeur and H. Paugam-Moisy. Théorie de l'apprentissage de vapnik et svm, support vector machines. *READ*, 12(5):517–571, 1999.
- [33] S. Haykin. *Neural Networks*. Prentice Hall, 1999.
- [34] C.-W. Hsu and C.-J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.
- [35] D. Hubel. *L'oeil, le cerveau et la vision : les étapes cérébrales du traitement visuel*. L'univers des sciences. Pour la science, 1994.
- [36] A. Jain, R. Duin, and J. Mao. Statistical pattern recognition : a review. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 22(1):4–38, 2000.

- [37] T. Kohonen. *Self-organization and associative memory*. Springer-Verlag, 1984.
- [38] P. Koiran and E. Sontag. Neural networks with quadratic vc dimension. *Advances in Neural Information Processing System*, 8:197–203, 1996.
- [39] D. Marr. *Vision*. W.H. Freeman and Co., 1982.
- [40] N. Nilsson. *Learning Machines*. Morgan Kaufmann, Palo Alto, 1990.
- [41] L. Novak and J. Bullier. *The Timing of Information Transfer in the Visual System*, volume 12 of *Cerebral Cortex*, chapter 5, pages 205–241. Plenum Press, New York, 1997.
- [42] S. Raudys and A. Jain. Small size effects in statistical pattern recognition: recommendations for practitioners. *IEEE Transactions on PAMI*, 13(3):252–264, 1991.
- [43] E. T. Rolls and A. Treves. *Neural networks and brain function*. Oxford university press, 1998.
- [44] J. Shawe-Taylor, P. Bartlett, R. Williamson, and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Trans. on Information Theory*, 44(5), 1998.
- [45] K. Stratford, K. Tarczy-Hornoch, K. A. C. Martin, N. Bannister, and J. Jack. Excitatory synaptic inputs to spiny stellate cells in cat visual cortex. *Nature*, 382:258–261, 1996.
- [46] S. Theodoridis and K. Koutroubas. *Pattern Recognition*. Academic Press, 1999.
- [47] S. Thorpe, A. Delorme, and R. VanRullen. Spike based strategies for rapid processing. *Neural Networks*, 14:715–726, 2001.
- [48] S. Thorpe and M. Fabre-Thorpe. Seeking categories in the brain. *Science*, 291:260–263, 2001.
- [49] S. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *Nature*, 381:520–522, 1996.
- [50] R. Tibshirani. Regression shrinkage and selection via the lasso. *J.R. Statist. Soc.*, 58(1):267–288, 1996.
- [51] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.
- [52] V. Vapnik. *Statistical Learning Theory*. John Wiley, 1998.
- [53] T. Vieville. Using markers to compensate displacements in MRI volume sequences. Technical Report 4054, INRIA, Nov. 2000.
- [54] T. Vieville, D. Lingrand, and F. Gaspard. Implementing a multi-model estimation method. *The International Journal of Computer Vision*, 44(1), 2001.
- [55] G. Weisbuch. *Complex systems dynamics : an introduction to automata networks*. Addison-Wesley, 1991.
- [56] A. J. Yu, M. Giese, and T. Poggio. Biophysiologicaly plausible implementations of maximum operation. *Neural Computation*, 14(12), 2003.